DIGITAL SUPERPOWERS

Open-source friendly

Windows
Linux
macOS

A whirlwind tour of readily-available tools that turbocharge **productivity**, ignite **creativity**, and empower **learning**

Nick Touran

Digital Superpowers

Nick Touran

Mar 24, 2024

https://digitalsuperpowers.com

Revision: 2.0

Published on: Sun Mar 24 10:29:07 2024

Contents

1 Introduction

Computers are everywhere. They're in our purses, our offices, our TVs, our hobby-dens, our cars, and our toasters. Many of us spend significant parts of our days operating these machines in the palms of our hands and on our desks or couches. They entertain us and make us laugh. They connect us to family and help us find love. They help us get reports written and make project videos at school. They facilitate our work by crunching numbers, formatting documents, and helping us express our ideas. In a sense, they are extensions of our brains. But many of us could be getting more out of our computers. This book will show you how.

Computers are capable of *being* dozens of appliances, gadgets, and even large facilities. This overwhelming breadth of capability is easy to overlook. I've been keeping a list:

Table1: Several things that computers are capable of *being*

TV	Movie player	Phone/address book	Phone
Drafting table	Diary	Encyclopedia	Photo album
Nintendo	Typewriter	Calculator	Dictionary
Musical tuner	Atlas	Audio studio	Newspaper
Darkroom	Printshop	GPS	Card game
Calendar	Card catalog	Radio	File cabinet

This book posits that some powerful capabilities of the computers many people have on hand are neither well-known nor easy to discover. Getting exposure to some of them will directly improve your life at home, school, work, and play. Seeing these in action should inspire thoughts to blossom about new achievements you can reach, as well as how to do your current activities better and with more pleasure.

Intended audience

This book is written with a few readers specifically in mind:

- The everyday school, home, and office computer user who wants to take it to the next level
- The artist looking for ideas and orientation regarding computers
- The entrepreneur looking for new/modern skills and pathways to success
- The scientist who knows the computer can help, but isn't sure how to get started
- The retiree looking to keep up-to-date on tech
- The executive who wants to be better informed on IT tools

We will go through many hands-on examples that you will be able to follow right away. But because every superpower could have (and likely does have) multiple books written specifically about it, each will not be treated in detail. Thus, you should be ready and willing to dig (sometimes significantly) deeper into anything we talk about here when it makes sense for you to seek mastery. This is not an in-depth tutorial on one topic, but rather a guided tour of many for you to pick up based on your needs and interests.

I assume you have a baseline level of computer knowledge. Ideally, you'll already know how to install a program, copy and paste things, how to make a slide deck, and some basics in a spreadsheet. If you don't, you'll still get something out of this, but don't expect this book to teach you all of those things. This is a solidly intermediate book.

Each chapter will progress from basic to more complicated in an effort to cover the broad scope, and the wide range of experience and interest out there.

About me

I'm a lifelong computer lover who has spent many personal and professional years learning about them. As much as I love computers, I really wanted to work on the energy/climate problem, so I went into nuclear engineering. Conveniently, it's a field that can be computationally heavy. So I applied computer interests to simulating nuclear chain reactions. Guided by many mentors through the years, I enjoyed seeking out and learning new or interesting technologies, and after many years, found myself with a fairly significant body of knowledge on hand.

Along the way, I got into technical communication by founding https://whatisnuclear.com. I have been told that I have a knack for explaining complex things.

Then one day I thought, "Sheesh! A lot of people could use some help looking deeper into computers in their daily lives." and so here we are.

Philosophy

It seems that computers have diminished from exciting world-changers to dull keepers of the spreadsheet and the cat video. People use computers within the bounds of the programs they have without knowing to explore what else is possible or available to make their lives easier and better. I want to illuminate the possibilities of computers to help people lead more exciting, engaged, and fulfilling lives using equipment they already have sitting around.

In the spirit of accessibility, this book highlights many free and open-source tools first while mentioning key proprietary software when appropriate.

2

Fundamentals

This chapter starts with basics and covers a few bits of info and best practices that everyone who uses computers regularly will benefit from knowing. We'll build more superpowers based on this foundation in the later chapters.

Basic parts of computers

Computers bring in, manipulate, and present data with lightning speed and hardly any moving parts. This is why they've had such a massive impact on our lives, and have brought forth the information age. For a bit of context, here are the key components in computers that perform all that wonder:

Component	Purpose
Central Processor Unit	The CPU is the brain and does most of the number crunching. Faster is better, speed is measured in billions of computation cycles per second (gigahertz).
Memory (RAM)	The working short-term fast-acting memory; it holds the data while the CPU crunches it. More is better, measured in billions of octets of ones and zeros (gigabytes).
Hard drive	The long-term but slower memory. It saves your files and photos and spreadsheets. Also measured in gigabytes.
Monitor	Displays information visually. On phones, it's an input device too. Measured in the number of horizontal and vertical picture elements (pixels).
Network card	Sends and receives data to/from other computers for posting cat pictures, etc. Measured in billions of ones and zeros communicated per second (Gbps).

Operating systems

When a computer turns on, a series of simple built-in operations run to check the processor and memory, and to figure out what other hardware is attached. Eventually, enough systems are online to read larger and more complex programs from the hard disk. In a sense, the computer is pulling itself up by its bootstraps (hence the term "booting up"). At this point, the computer will find an Operating System (OS) on the hard disk, which will tell the computer how to run everyday programs like an office suite or a web browser and everything else.

There are many OSs out there, but we're going to mostly just talk about three of them: Microsoft Windows, Apple macOS, and Linux. The first

two are familiar to everyone. The third is an open-source product (meaning it's available for anyone for free and that anyone can look at its underlying source code) developed over the years by a community of volunteers and professionals. Linux is the underlying technology behind Google's Android OS and also runs many of the servers powering the internet. In the past few years, more people started running Linux on their personal laptops and desktops as a powerful, cost-effective, fun, and philosophically-pure alternative. We'll try hard to make sure everything discussed in this book works fine in all three of these OSs.

Files and folders

As you probably know, most computer systems store information in a hierarchical structure of nested digital folders (aka *directories*) on the hard drive. You put your photos in the Photos folder, and so on. You can create folders and files, delete them, rename them, copy them, paste them, etc.

Note: If you are unfamiliar with copy/paste, you are in for a treat. It is a core superpower but you may benefit more from this book after learning about it. The WikiHow[1] page on it is fantastic. I'd also be a little worried that this book is slightly too advanced for you. If you already know about copy/paste but not the keyboard shortcuts (`Ctrl-C`, `Ctrl-V`) just know that those shortcuts are speed-demons.

We've all lost files in folders before. Many of us have opened a document from e-mail and worked on it all day only to close the editor and then never find the file again. This usually happens when it gets saved to some temporary folder where files downloaded from the internet go. I've done this myself many times, and have fielded dozens of calls for help from friends through the years due to this. The solution is the **Save As** button, which is distinct from **Save** in that it lets you choose where to save a file as opposed to just saving it where it was opened from. Click it, navigate to a normal place like **Documents**, and save it there. Problem solved!

[1] https://www.wikihow.com/Copy-and-Paste

Organizing folders is an ongoing battle for many of us as well. We try breaking things down by file type (Photos, Documents, Music) and then by year. Sometimes for multi-year projects, it doesn't go well and you may need a Projects folder. It's deeply personal, so just explore what works best for you. For photos, one nice method is to organize them by year and then name subfolders with the month and then an incrementing integer and an event, like `07.1 Fourth of July` or `12.4 Yodit's Christmas Sweater Party`. Operating system designers are often trying to move away from the hierarchical structure and use tags (so a file can have multiple tags, similar to being in multiple folders at once) but we're not quite there yet.

Backups

Backups are really important, and I don't know a lot of people who take them very seriously besides professional IT staff. I personally am really obsessed with my document, photo, and music collections and would be just heartbroken if I lost them in a fire, robbery, hardware failure, electromagnetic pulse, earthquake, volcano, and/or nuclear war. But even I am not great with backups. The main categories of data loss to protect against are:

- Hardware fails and data can't be recovered
- Hardware is stolen or hacked and can't be recovered
- User (or family member) accidentally deletes data

A reasonable and easy solution for backups is *the cloud* which we will discuss momentarily.

A more-difficult-to-implement but conceptually straightforward method for slow-moving data is to buy two external USB hard drives with a few terabytes of space. Copy all your important files over to them both. Take one to work or a friend's house or some other off-site location. Every month or six months or whenever (depends on your pace of accumulating new stuff that you care about), copy new files onto them and swap them. This will protect your archival data against robbery and fire but not necessarily earthquakes or EMPs. For those, you need more distance.

Of course, there's the challenge of finding out which files are out of date, and having the discipline to regularly update the backups. Figuring this out and quickly synchronizing is the job of the thousands of backup software products you'll hear about out on the market. Sure, you could just copy everything over every day, but that would take a long time and wear out your disks. Most modern operating systems like Windows, Linux, and macOS come with backup utilities. Try those out or look for third-party commercial products.

As your digital superpowers develop, you will become knowledgeable in a few other systems that will lead you to exceedingly slick backup options. `git-annex` is an advanced option worth noting that builds on the `git` tool you'll learn about in *Programming*. It's quite powerful for keeping well-organized and secure backups. For scientific, engineering, and data users, the `datalad` tool builds further upon `git-annex` to add reproducibility on top of backups, so your scientific results can be perfectly replicated by anyone in the future. In particular, `datalad` auto-discovers any output files created by running particular commands and tracks the input, commands, and output.

Network Attached Storage (NAS)

It's not too uncommon these days to have a NAS in the home. These are lightweight computer/appliances containing big hard drives that you can connect to your home computers. Professional photographers and videographers all have these. They offer some benefits in terms of sharing files across the family's computers and in terms of data reliability (they're often resilient against a single failure of one hard drive). You can buy them for a few hundred bucks or, if you're into the hobby of it, you can turn an old computer into a NAS. PC Magazine[2] (bless their hearts) runs a review of the latest and greatest NAS machines every year, so check them out for more advice.

Warning: NAS provides some level of backup, but they do not help in the event of fires, burglary, or sometimes even accidental deletion.

[2] https://www.pcmag.com/article2/0,2817,2401086,00.asp

Some NAS products come with nicely polished backup software that can help with your offsite backups.

File Encryption

If your phone or laptop gets lost or stolen with all your work and online dating on it, no problem, right? Because you have a backup? Great job! But you may still worry because there's a lot of personal and confidential stuff on most devices. It's wise to keep your private matters private no matter if they're just your mom's recipes. You *DO* have something to hide! If someone gets access to your e-mail for instance, they can reset your bank password and cause all sorts of ruckus. So let's just hide your data.

Encryption is the process of scrambling the bits on your hard drive in such a fashion that even if someone steals the computer/phone, they will not be able to access the information within. Without encryption, a bad guy or random teen could just plug a cable in and see all your information. Math to the rescue!

Options for encrypting your hard drives in different systems:

Kind of computer	Encryption options
Windows	Find the BitLocker program and activate full-disk encryption
Linux PC	Choose to encrypt entire drive during installation.
Mac	Find the FileVault utility and set it to encrypt the full disk
Android phone	Search settings for "Encryption" and ensure it is enabled. If not, enable it. Make sure to set a strong PIN or other credential
iPhone	Almost always encrypted by default; set a strong PIN

Warning: If you encrypt something and then forget the password, your data is very nearly impossible to recover (by design, after all that is the point). Hopefully you have a backup! This seems obvious but it is a little scary. Just be careful.

The Cloud

The Cloud is a *nebulous* term that really just means "a large semi-automated collection of nearly identical computers out there on the internet that is operated by some company." A lot of computation is moving to The Cloud because of the efficiencies of scale inherent to mega-operators like Amazon Web Services. The deal The Cloud offers to new business owners is that they will deal with the hassle of procuring, maintaining, and servicing the computers, allowing you to focus on software, marketing, and users. In other words, you do you. If you do really well and need 10,000 new computers right now, they click a button and (bloop-bleep) it's done. If you mess up and everyone starts deleting their account on your service, (bleep-bloop) those computers are no longer reserved for you. What a deal! Services like Netflix and Uber run on cloud-based back ends operated by other companies, like Amazon, Google, or Microsoft.

What does it mean *to you*? Many services offer the ability to automatically synchronize files on your hard drive with their hard drives in the cloud (via the network card). This is nice because when you take a photo on your phone, it shows up on your computer. If you experience a tsunami, anything that was synced in The Cloud far away will survive and re-sync when you get a new computer.

Services like Dropbox, Microsoft's OneDrive, and Apple's iCloud do this. If you're uninterested in running your own server or dealing with your own backups, I have to recommend using these services. The risk, as always, is that you have to trust them, because they can see all of your files. Also, with computer systems being so complex, they always have some new severe vulnerability, so it's probably just a matter of time until these services are breached. At that point, the biggest services will be the most likely target. On the other hand, they also probably have the

best defenses. Just something to think about.

It's the same story with e-mail. If you use a web-based e-mail system in the cloud you can just log on from anyone's computer and have it.

The command line

Vast computer superpowers are controlled by a somewhat-obscure window into your computer: *The Command Line* (or *Terminal*). Go onto your computer and launch it. On Windows, click **Start**, and type `cmd` and click `Command Line` when you see it (or just press enter). On macOS, go to the Finder and then click `Go` → `Utilities` → `Terminal`. In Linux, just hit `Ctrl-Alt-T`. A little box will pop up with an ominous blinking prompt. That's it. That's the command line. You're gaping into a universe of possibility, like in *Men in Black* when they gape into the cat's necklace and it's a whole galaxy.

The idea here is that you type commands in and it responds by doing something. These commands are precise, expressive, and repeatable; you can tell the computer exactly what you want and it will do it. As you'll see in later chapters, this can be significantly faster than pointing and clicking, especially for repeated actions. In other words, *a command is worth a thousand clicks*.

Most importantly, hundreds of grand-slam superpowers are hidden in small and simple command-line utilities, the majority of which are freely available. The elegant design benefit that led to the existence of these rich tools is one of reductionism: many complex operations can be broken down into a series of simple operations. For example, consider a spell-checker from a software developer's perspective. If she writes a word processor and needs spell checking, she can either write an entire spell-checker in addition to the word processor, *OR* pray that someone else has made a spell-checker before and made it available for her to pull in and use. As it turns out there is indeed a spell-checker library and utility that's available in this regard and it has a command-line interface. Thousands of such building blocks exist, and they're all available to you right now through the command line.

There is a reputation that the command line is for "advanced" users

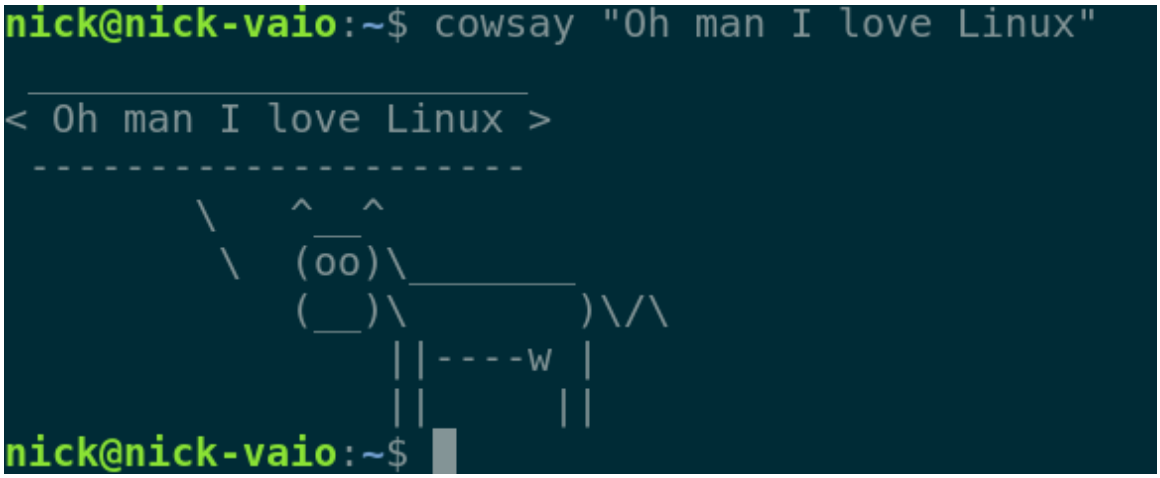
only. I don't think this is true at all, and I hope I can convince you that it's accessible and useful for everyone. We'll leverage the command line quite frequently in this book.

Note: When you read something like "Type this in" from now, and it's unclear where to type it, it's very likely the command line.

Let's type some commands in to see that it's fun and easy.

Operation	Windows	Linux/macOS
Which folder am I in?	`pwd`	`pwd`
What files exist in this folder?	`dir`	`ls`
Move up one folder in the hierarchy	`cd ..`	`cd ..`
Move back to the original folder, `[NAME]`	`cd [NAME]`	`cd [NAME]`
Print the current time and date	`date /T` and/or `time /T`	`date`
Print the computer's uptime	`net statistics server`	`uptime`
Make a folder	`mkdir cat_pictures`	`mkdir cat_pictures`
Remove a folder	`rmdir cat_pictures`	`rmdir cat_pictures`
Convert units	`units "12 cups" mL`	`units "12 cups" mL`
Make a cow say something	N/A :(	`cowsay "Oh man I love Linux"`

That last one is not a joke.



Some commands come with an OS and others have to be installed from a third party. The last two examples above are the latter, and the installation of `units` is discussed in *Unit conversion*, so you'll actually have to wait until then to try out that command.

Here are a few productivity enhancers in the command line:

- If you ever want to get the previous command again (to either rerun it or modify it slightly and then rerun it), press the *up arrow* on your keyboard. Each time you press it goes up in a history of all the commands you've executed. The *down arrow* goes in the other direction.
- If you're typing out a command you can often press `TAB` to auto-complete it. For instance, if there is a folder called `my_very_long_folder_name` you can just type `cd my_very` and then press `TAB` and it will fill in the rest. This is called *tab completion*. I cringe badly and then speak up when people don't use it because it can save hours per day when you're working heavily in the command line.
- You can copy and paste commands into the command line.
- At least in Windows, you can drag a file into the command line instead of typing its full path.
- If you ever typed a command that's taking too long or isn't what you wanted, you can abort it by pressing and holding the `CTRL` key and then pressing `C`.

Programs and package managers

Your computer undoubtedly came with some programs, and you have probably downloaded and installed others. It was once a big pain to go out and find/install the latest versions of various programs and to keep them up to date. In the Linux world, this was solved years ago with special programs called *Package managers*, which are simple little commands that know how to go find, download, install, keep up-to-date, and (if desired) uninstall programs. macOS and Windows have started catching up and there are now good package managers for them as well. Without further ado, it's time for your next superpower: package management!

Note: This is optional but will make it easier to follow along in sections of the book that rely on special programs. In all future uses, you can alternatively search for the program at hand, find its web page, download it, and run its installer.

Warning: These programs change what's on your computer (that's the point). It is unlikely but possible that something will go wrong. Try these out on a computer that you can afford to break if you're just learning.

macOS has one de-facto standard package manager called Homebrew[3] that can be installed from the command line by running the following command (which is also on their webpage if you'd like to copy/paste it):

```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/
↪Homebrew/install/master/install)"
```

A good package manager called Chocolatey[4] is available on Windows. To get it, you have to open the Command Prompt with *Administrative* privileges, meaning commands you type can change system settings. You generally do this by typing `cmd` in the Start Menu and click-

[3] https://brew.sh/
[4] https://chocolatey.org/install

ing the choice that has the word "Administrator" in it (or by right-clicking the normal Command Prompt choice and choosing "Run as Administrator"). Once you're in, copy/paste the command shown at https://chocolatey.org/install into it and press `Enter` to run it (it's simply too long to reasonably put in this book expecting you to type it). You may have to right-click and choose `Paste` to paste, or (in Windows) just press `Control-V`.

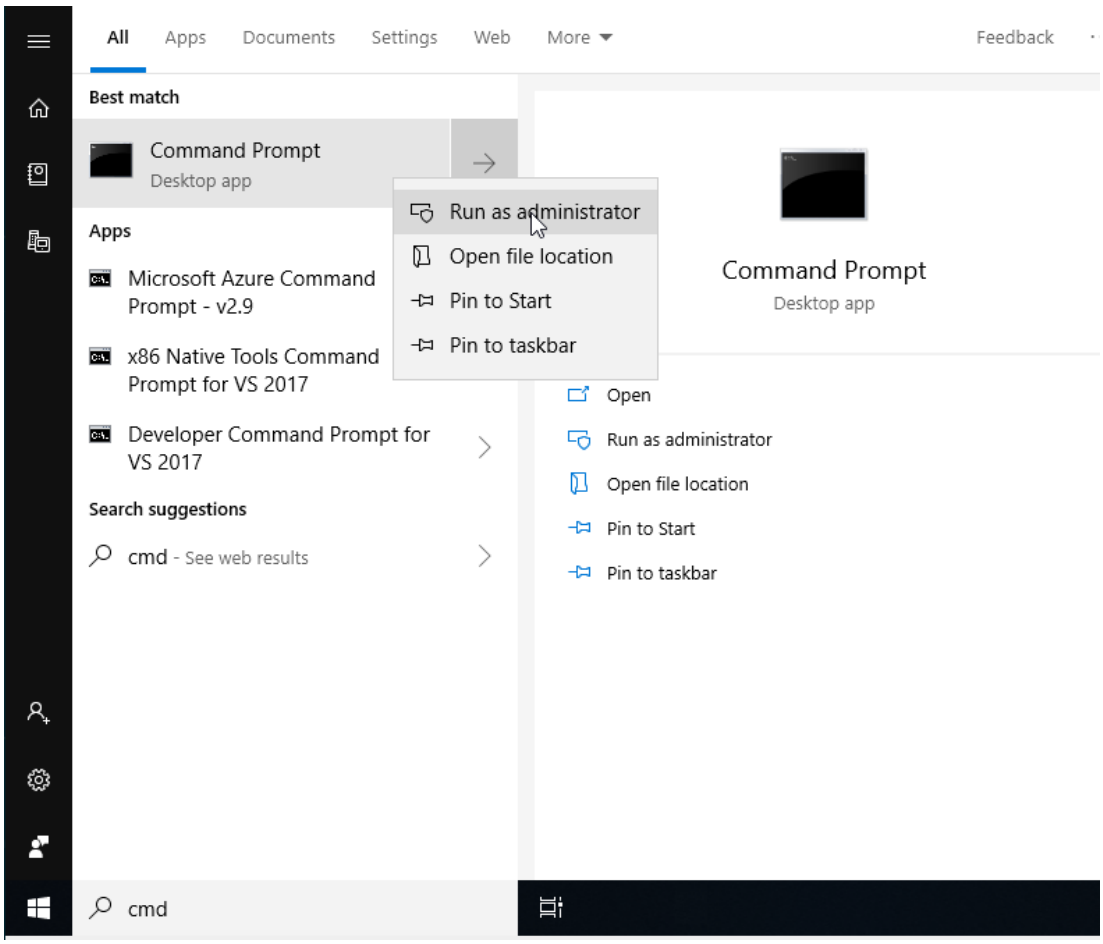


Figure1: Opening the Administrator Command Line in Windows 10 (hover your mouse over and right-click, then left click on the Administrator option). You generally have to use the Administrator option every time you run a `choco` command.

Now you have a package manager and can install common programs

with simple commands! The commands will look similar to what's shown below. We'll provide similar tables frequently.

Table1: Installing programs with package managers

OS	Command
Windows	`choco install [program_name].`
Linux	`apt install [program_name]`
macOS	`brew install --cask [program_name]` or `brew install [program_name]`

After issuing a command like these, there may be some prompts and alerts that scroll by as the packages are downloaded and installed. In some cases, if things look frozen, you may have to just press `Enter` (I had to do this in `choco` a few times). If you install a large package, the process may take a few minutes depending on your internet connection speed.

Note: In Windows with `choco` you have to either run `refreshenv` or open a new command terminal after you install a program before it is available as a new command-line program.

You can just as easily uninstall anything installed through these systems by running the same kind of command but with `uninstall` or `remove` instead of `install`. And you can `update` as well.

Life's getting easier already! Let's put that new skill to work.

Password managers

You know that feeling when you are trying to buy something and the web page *forces* you to make a new account? Many of us just reuse the same old password again and again, maybe with a few modifications based on the site name. This ends up being really dangerous in addition to annoying because the likelihood that Arnold's Pizza Shop has rock-solid secu-

rity practices is low, and after you give (roughly) the same password to a billion such establishments, the probability of some nefarious hacker breaking into them and getting your password approaches 100%.

Annoying *and* dangerous!? Hopefully there's a superpower for that. Indeed, there is: **Password Managers**. Password managers are programs you run on your computer or phone that do two essential things:

- They act as your own personal encrypted vault of passwords so you can look them up easily when you need them
- They have random password generators that will whip up a 25-character password whenever you need a new one for Arnold's Pizza Shop or similar

If you use a unique and random password for everything, the threat discussed above is neutralized. Plus you don't have to type in the password ever, Password Managers allow you to copy it and then paste it right into the web form. In this workflow, you never actually know most of your passwords.

Tip: You will still need to memorize a few key passwords for extra-important accounts. Just keep them unique. See *Making secure but memorizable passwords with Diceware* below for ways to do this well.

There are some great open-source password managers out there like KeePassXC[5] as well as commercial offerings (with better support) such as LastPass[6] or 1Password[7]. A good combo is to put your KeePassXC database in a cloud synchronization program so it synchronizes across all your devices.

To try out KeePassXC with a package manager (see above), run the following in the Command Line:

[5] https://keepassxc.org/
[6] https://www.lastpass.com/
[7] https://1password.com/

Table2: Installing KeePassXC with your package manager

OS	Command
Windows	`choco install keepassxc`
Linux	`apt install keepassxc`
macOS	`brew install --cask keepassxc`

You may also want to store credit cards in your password manager, making it really easy to copy/paste them into web forms when shopping.

Of course there is a new threat now: If someone finds out the one password you use for your password manager and has access to your files, they can get all of your passwords at once. This is less likely to happen, but you should consider it when typing your master password into the password manager. Keep in mind that Edward Snowden puts a blanket over his head whenever he types a password, and he's somebody who knows what's going on out there.

Tip: Does your password at work auto-expire every few months? Companies used to think this was good for security but in 2016, NIST released guidelines saying that unnecessary auto-expiry was leading people to choose weak passwords. They now recommend to only force someone to change a password if there's been evidence of a data breach. Same with password complexity (it's no longer wise to require mixes of uppercase and symbols), just make a long password and check it against a common password list.

Making secure but memorizable passwords with Diceware

For the few very strong passwords you do need to memorize, the Diceware method is here to help. Here are the steps:

- Grab a Diceware word list like one of these[8] from the *EFF*. These have 5-digits between 1 and 6 (for each possible value of a die). Here is an excerpt:

```
46663   reflex
46664   reflux
46665   refocus
46666   refold
51111   reforest
51112   reformat
51113   reformed
51114   reformer
```

- Get 5 dice and roll them. Line them up and then look up the word that corresponds to the values you rolled (if you had 4, 6, 6, 6, 4, your first word would be `reflux`, from the list above).
- Roll 4–7 more times. Now you have 5–8 random words. String them together and that's your super-secure but still memorizable password.

Note: Don't have 5 dice sitting around? You can find online generators of diceware passwords. Just know that they aren't random enough and the Diceware method recommends against using anything but actual dice. Besides, everyone should have some dice around for playing 10,000[9].

Two-factor authentication

Two-factor authentication (2FA) is a login method offered by many web pages and institutions these days. It makes it so you have to type a password *and also* type in a one-time code sent to your phone via text or through a special app. The idea here is to combine something you know (your password) with something you have (your phone). It's similar to

[8] https://www.eff.org/deeplinks/2016/07/new-wordlists-random-passphrases

[9] A surprisingly fun dice game, similar to Farkle. Try it out, you won't regret it: https://en.wikipedia.org/wiki/Dice_10000

typing your zip code at the gas pump when using a credit card[10], or a PIN with a ATM card. For your most important accounts, it's really essential to turn it on to minimize the likelihood that someone can access them. Banks, big shopping sites, and e-mail providers are good first places to start. You can learn how to activate it with a web search along the lines of:

```
activate [web page name] two-factor authentication
```

The new risks here are that it takes longer to log in and, if you lose your phone, it's extra hard to just reset your password. Such are the tradeoffs. Many companies are relaxing the pain by only requiring that you use the 2FA code on new computers so it's not all that bad. Other companies make you obtain and enter a 2FA token twice per transaction, which is really annoying.

Warning: Getting 2FA tokens through text messages is probably better than nothing, but it's significantly less safe than using a dedicated 2FA app. Why? Because it's possible for a bad guy to use social engineering to convince your phone operator to change your phone number to a different SIM card, thereby allowing them to access your 2FA tokens and get into your accounts. This has happened in serious attacks, such as one on CloudFlare in 2012[11]. It's much harder for them to get access to an app on your phone.

Common scams

Billions of dollars are lost each year to fraud, much of which occurs on computers. You should always be a little skeptical if someone you don't know asks you to pay for anything or give you access. Here are a few guidelines for avoiding fraud.

[10] In the USA, ZIP codes are used for credit card authentication at many self-serve gas pumps because the cards themselves don't have PINs like they do in most of the rest of the world.

[11] https://blog.cloudflare.com/the-four-critical-security-flaws-that-resulte/

- **Never give your password out**

  No well-meaning entity will ever ask you for your password on the phone or in e-mail or in any other place than a log-in form. If this ever happens to you, you should recognize it as a huge red flag.

- **Computer companies will never call you**

  There are lots of scams where people pretend to be calling from some big company you've heard of offering to fix something on your computer if you just go to a certain link. This is someone trying to access your PC to find banking information about you. Hang up!

- **E-mails where people know an old password of yours don't have video of you**

  Through the years, various companies have experienced data breaches. If you had an account with any of them with a weak password (i.e. a single dictionary word), they may e-mail you and say they know your password and claim that they hacked into your computer and recorded you doing something naked or private. They threaten they'll send it to all your friends if you don't send them about $1000 via Bitcoin. It's very unlikely they have anything on you since these breaches are widely available. If you used that same password on other accounts, you do need to change it, like, yesterday. See https://haveibeenpwned.com/ for help in checking to see which accounts of yours have been compromised in known breaches.

- **If someone tells you that you're in big trouble, but they can help, it's probably a scam**

  You may get a call saying that you have dozens of unpaid violations at [your county courthouse] but if you go down to Walgreens and get $2000 worth of gift cards and read them one after the other to this person on the phone they'll get you out of it. That's a scam. Gift cards are very hard to trace and are a favorite instrument of scammers. If you're in actual trouble just deal with it through the proper channels and get a lawyer. This happens with people impersonating the IRS as well. They may know a little about you, like where you live and some previous addresses. This is all public record but it makes the scam seem more real. Don't fall for it. If you're really unsure, ask them for the number of their institution,

check it online, and then call them back.

- **Watch out for phishing**

  Phishers will send you an e-mail that looks like it's from a familiar institution. It will have the right logos, the right address, and will have a link you can click to do something important, like reset a password or confirm something. The best defense here is to hover your mouse over the link and take a look in the address bar (usually at the bottom of the screen) to see where the link is actually going. If it's to some weird internet address or some misspelling of the actual site, it's a scam. If you do click the link, you can also check the address bar of the web browser. Be careful, these can be very tricky. This has been such a problem that your employer may hire someone to send phishing-like messages to you and if you click them you'll have to take mandatory training. It's like anti-phishing phishing.

- **Ransomware is real**

  If you get a message that all your files are encrypted and you have to pay to get them decrypted, it may be real; especially if you can't access your files. Ideally, you'll have a backup of your files and can just wipe your computer clean and restore the backup. Otherwise, you have to choose whether you want to give the criminal what they want or kiss your files goodbye. Another good reason to have backups!

Those are just a few. The FBI has hundreds more to watch out for at https://www.fbi.gov/scams-and-safety/common-fraud-schemes.

Well, that sums up chapter one. I hope you've learned something new and useful.

3

Around the House

You browse the web through your phone and computers at home, you print stuff, you e-mail friends, you learn, you play. In this chapter, we'll make these more pleasant, productive, and safe.

Life-changing keyboard shortcuts

You probably use a keyboard, at least a little. A few subtly-sublime keyboard shortcuts can really speed up your work and play. You may have heard that professional typists never click around in a word to go back and make corrections, they just delete the world and hammer it out again in under a second. These shortcuts can help you when changing text in anything (e-mail, word processor, web browser, etc.). Anything you can do to prevent yourself from switching from keyboard to mouse and back is going to turbocharge you. If you think about it, the keyboard is more expressive than a mouse. Try them out; you'll love them!

Table1: Some surprisingly useful keyboard shortcuts while editing text:

Effect	Windows/Linux	macOS
Delete previous entire word	`Control-Backspace`	`Option-Delete`
Delete next entire word	`Control-Delete`	`Function-Delete`
Move cursor to previous or next word	`Control-LEFT`, `Control-RIGHT`	`Option-LEFT`, `Option-RIGHT`
Move cursor to beginning of line	`Home`	`Control-A`
Move cursor to end of line	`End`	`Control-E`
Move cursor to beginning of document	`Control-Home`	`Command-Up`
Move cursor to end of document	`Control-End`	`Command-Down`

Here's the kicker: for all of these "move cursor" shortcuts, you can throw a `SHIFT` in there as well to select everything from where you are now to where the cursor's going to go. Once something is selected you can delete it, copy it, change the formatting… You can do anything!

Exercise

Try these out. Open an e-mail editor and type out a sentence of your choosing. Now don't leave the keyboard! From the end of the sentence press `Shift-Control-Left` (to select the previous word) and then press `Left` three more times (while keeping `Shift-Control` held). This should have selected the previous three words. Copy them with `Control-C`, press `End`, and press `Control-V` a few times to paste them.

Now try `Control-Shift-Home` followed by `Backspace`. You're really hauling now!

(macOS people use the equivalents from the table above)

I didn't learn `Control-backspace` until well into my career and it has

really been useful since then.

Besides editing text, keyboard shortcuts can streamline web-browsing too. The same "leaving the keyboard is slow" philosophy applies online in its own way. The most essential shortcut takes you to the address bar with `Control-L` (`Command-L` on a Mac). Now you can type a new web address or a search and just press enter to proceed.

To turbocharge this, combine it with the `!bangs` feature of the *DuckDuckGo* search engine and you'll really be moving. If you set DuckDuckGo as your default search provider (something you can do in the settings of your web browser), then you can put, for instance, a `!t` anywhere in the search and it will take you *directly* to the thesaurus page for your search term. So `big !t` goes to a thesaurus page for the word `big`. *Thousands* of bang codes are available, and you can even submit your own. Here are a few more good ones:

Table2: Some other awesome DuckDuckGo !bang codes

`!d` Dictionary	`!a` Amazon
`!g` Google	`!gi` Google images
`!gschol` Google Scholar	`!wu` Weather Underground
`!w` Wikipedia	`!cl` Craigslist
`!fb` Facebook	`!yt` YouTube
`!tw` Twitter	`!pub` PubMed

That's right, if you whimsically want to conjure up your ex-boyfriend while you're in the middle of reading a news article, you can just type `Control-L !fb Jacob Olson Enter` and there he will be. Sigh.

Dozens of other keyboard shortcuts are available in various browsers. See if you can find the list of them (usually in *Help → Keyboard shortcuts*) and see if any others seem useful to you.

Avoiding printer dry-out

Printers have a bad reputation for never working right and having really expensive ink. Regarding the ink issue, InkJet printer jets dry out if they're not used frequently. So if you're shocked by a printer never really working or always needing its jets cleaned, that's why. If you print less than once a month but still really need that printer when you need it, consider getting a laser printer instead. They're a bit more expensive (especially for color or combo printer/scanner), and their color for photo-printing is not as good, but their ink reliability is usually worth it for a lot of people.

Get to know your router

Home networks consist of public internet coming in from your internet service provider (e.g. Comcast) and going into your *router* which distributes it around your house via Wi-Fi radio waves and sometimes Ethernet cables. Your router and each device connected to it gets assigned Internet Protocol (IP) addresses, which are like street addresses, but for the internet backbone. There exist a few special blocks of IP addresses for local networks (e.g. the one in your home), and the most common of these is anything that starts with `192.168`.

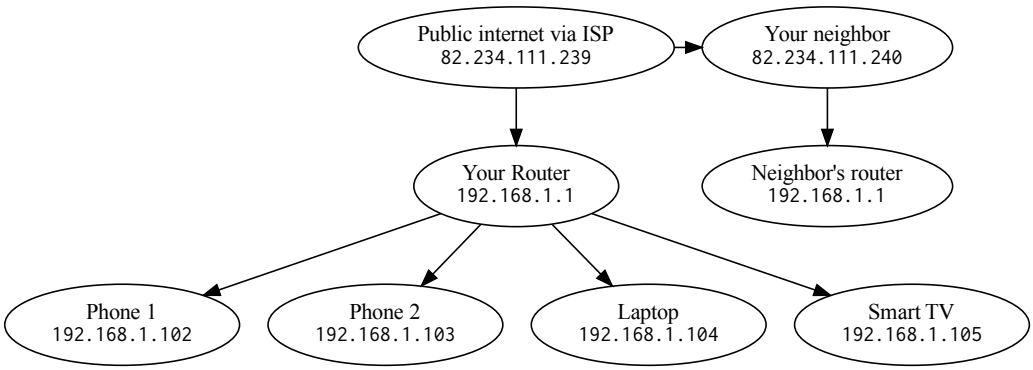


Figure1: Your home network with example IP addresses

Your router is a little computer in itself and it has settings where you can change your Wi-Fi password, set up a guest network, adjust security settings, and other things. It doesn't have a screen or keyboard like your laptop so it presents its user interface through a web browser. To get to it, simply point your web browser to its IP address, which is very often `192.168.1.1` or `192.168.0.1`. If you struggle, check the bottom of your router for a sticker that says what its default address is.

Setting a strong Wi-Fi password

With more smart-this and smart-that in the home, the security of your Wi-Fi router is fairly important. It's easy for people driving around with laptops to get onto unsecured networks or ones with easy-to-crack passwords (they're called war-drivers). It's no problem to tote around terabytes of pre-hashed passwords (called *rainbow tables*) and try to get in. So try to come up with a password that is unlikely ever to have been used before. As always, no single dictionary words.

Note: Did you know? Wi-Fi doesn't really stand for anything. It's just riding upon the good reputation of Hi-Fi stereo equipment, which means

High Fidelity. “Wireless fidelity” would make no sense at all.

You can use your password manager to make a random Wi-Fi password. This might annoy your family if it’s too hard, but you can at least make it easy to get on by generating a 2-D barcode (QR code) in a special format. Websites like qifi.org[1] do this for you in a few seconds. Once it’s done, anyone can use a barcode scanner app and just scan it to get online. Just don’t make it visible from any windows! Also beware that some devices like printers will still require you to type the password in manually.

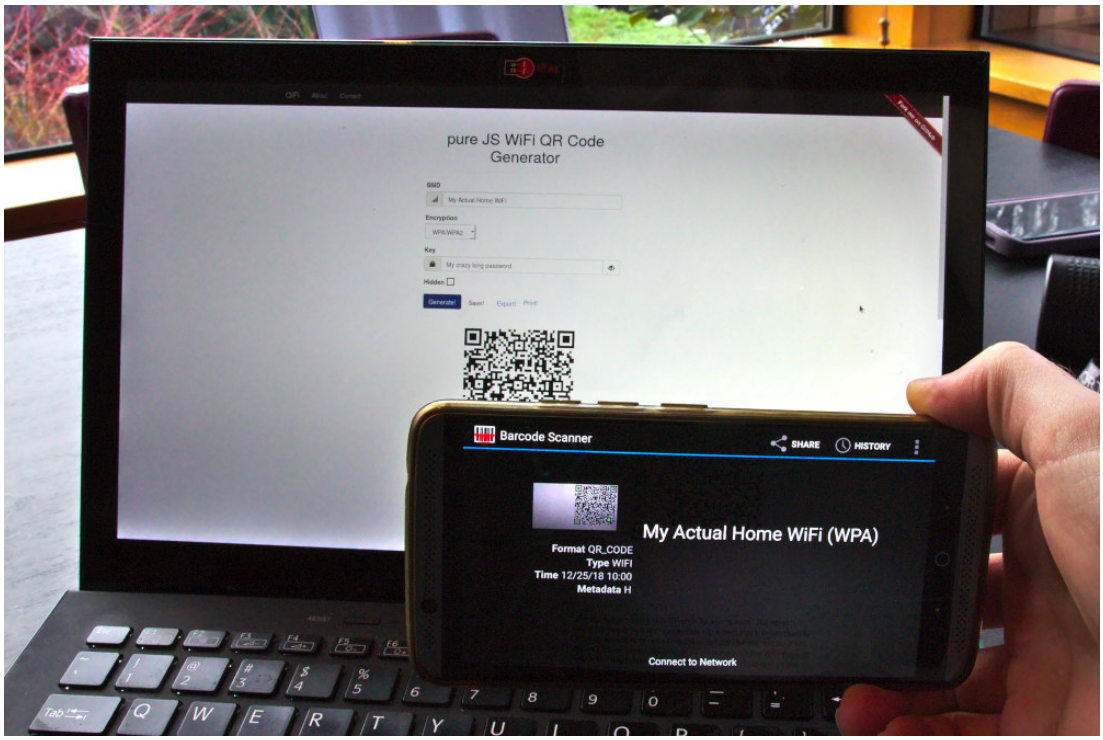


Figure2: Scanning a QR code to connect to Wi-Fi.

[1] https://qifi.org/

Guest networks

Many Wi-Fi routers allow you to set up a separate network for guests. Doing so gives them unfettered internet access without exposing your devices to any viruses or misbehavior they may be dealing with.

It may be wise to connect your Smart TVs and other devices with ulterior motives to the guest network. Some devices have been known to monitor other devices on the network to try to figure out how to send better ads or otherwise track you. If they're on a different network they can't watch your phone and laptop.

Avoiding DNS hijack

Have you ever mistyped a web address only to be greeted by a page with ads from your internet service provider (e.g. Comcast)? That's something they can do because they are providing the internet phone book of lookup tables between web page names and actual server IP addresses (more on this phonebook system, called *DNS*, in *Publishing*). They may also be slower and less secure than some of the other options. You can adjust your DNS settings in your router's setup (near where you set the Wi-Fi password). You're looking for `DNS Server` settings or `Custom DNS`. I recommend using CloudFlare's DNS entries for good security and performance:

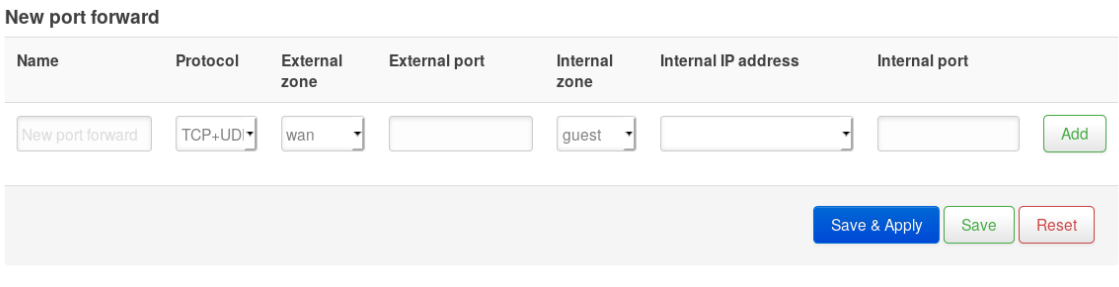
Table3: CloudFlare's DNS settings

Primary (IPv4)	`1.1.1.1`
Secondary (IPv4)	`1.0.0.1`
Primary (IPv6)	`2606:4700:4700::1111`
Primary (IPv6)	`2606:4700:4700::1001`

They have detailed instructions at https://1.1.1.1 if you're interested.

Opening ports

Your router acts as a firewall between the dangerous public internet and your devices in your home. Sometimes you may need to carefully poke holes in this firewall for one reason or another (you want to access your security system from afar, or your video game needs to communicate with peers for multiplayer, or something else). This is done in your router configuration, near the Wi-Fi password setup. Search for *Firewall* settings and/or *Port Forwarding*. Once you find it, you'll see a table similar to this:



In this form you choose which port number to forward and which internal device it should get forwarded to. So if you had a Raspberry Pi at `192.168.1.106` you wanted to remote into on Port 22, you'd enter 22 for both the internal and external port, and the IP address for internal IP address. Make sure that Raspberry Pi is on lock-down though because it will be subject to *Script Kiddies* trying to hack into it almost immediately. Consider using a non-standard port number as the external port to reduce this.

Virtual machines

Virtual Machines (VMs) are programs that emulate a fresh, empty computer. They allow you to run an entire other operating system as a program on your computer. At home, this can be really useful if you have a particularly curious family member who really wants to mess around with computers, but you don't want them to break yours. You can let them mess around all they want in the virtual computer on your computer!

Professionally, they're great for seeing how your products or services work on other kinds of computers. If you're making a website on a Mac and want to make sure it works in Windows, a VM can help[2].

They can also be useful for trying out a program or operating system that you don't feel comfortable with or trust for some reason. If it does have a problem, you can just reset the VM to its initial state. This is a phenomenal way to try out things like Linux that you're just curious about getting a feel for. It's also how security researchers research and monitor known computer viruses.

A few Virtual Machine managers exist, but let's just get started with one called VirtualBox[3].

Table4: Installing the VirtualBox VM manager

OS	Installation method
Windows	`choco install virtualbox`
Linux	`apt install virtualbox-qt`
macOS	`brew install --cask virtualbox`

When you run `virtualbox` it will pop up a window from which you can either import pre-made virtual machine "appliances" or make new ones. Here are the steps to try out Ubuntu Linux in a VM:

- Download the latest Ubuntu image from here[4].
- Open `virtualbox` and choose `Machine → New`. Name it `Ubuntu` and set type to `Linux`, version: `Ubuntu_64`.
- Click through the defaults for Memory size, hard disk, etc. (or adjust as you please)
- Click `Start` to boot the empty machine.

[2] You can grab a functional Windows 11 VM from Microsoft at https://developer.microsoft.com/en-us/windows/downloads/virtual-machines. It took me a while to figure it out, but they're `zip` files that when unzipped can be loaded in VirtualBox.

[3] https://www.virtualbox.org/

[4] https://www.ubuntu.com/download/desktop

- It will prompt you for a start-up disk. Click the browse button and choose the downloaded Ubuntu `iso` image.
- Click Start and the Ubuntu installation process will begin!
- If you haven't been exposed to Linux before, bask in the glory of how nice it is.

Note: Related to VMs, you'll also hear about *containers* which set up isolated "jails" for programs to run in while sharing the same host operating system. Containers require less hard-disk space, are faster to spin up and down, and require fewer system resources, so they're really popular in modern cloud-based data centers and infrastructure. The *Docker* container manager is the most popular. Some software developers/vendors are distributing their programs these days in docker containers to simplify the setup process for users.

Nostalgia alert

It's sometimes fun to re-live your younger years by setting up virtual machines with old versions of software like DOS and Windows 3.1 (or whatever was around when you were a kid). You can get the *images* of the installation discs at places like WinWorld[5] and then mount those images one by one as floppy disks in VirtualBox and have a good old time[6]. Designasaurus, anyone?

Along these lines, the DOSBox[7] project is a better way to actually play old games on your current computer. It will integrate nicely with modern video and audio hardware, which is harder to do with pure virtual machines. It's in your package managers (`dosbox`).

[5] https://winworldpc.com/library/operating-systems

[6] If you want to see exactly how to install old OSs in VirtualBox, check YouTube for tutorials, and you'll find dozens of them doing just this.

[7] https://www.dosbox.com/

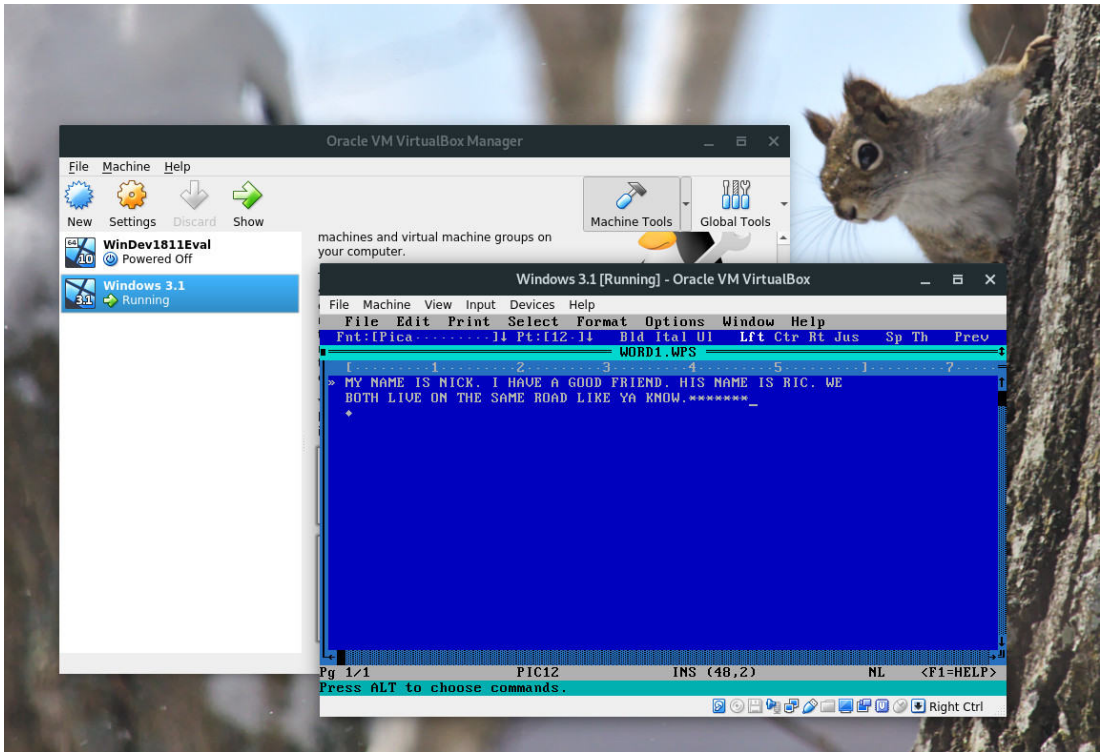


Figure3: MS Works running on DOS 6.22 on a Virtualbox VM.

Ad-blockers

Ads are like a sad knock-off of Santa Claus. They track us and watch over our shoulders as we shop. They try to get us to buy boots for months after we buy some boots. We can't afford to give up precious brain cycles. If they ever sneak onto Netflix, it's game over. You can do a few things to reduce your shopper footprint in the nefarious systems of commercial retail if you so desire. Here are some options:

- Use the Firefox web browser instead of Chrome or Edge, and install the EFF's Privacy Badger[8] plugin and/or the uBlock ad blocker[9].

[8] https://www.eff.org/privacybadger
[9] https://www.ublock.org/

Google at its core makes most of its money off of advertising. Firefox works on your phone as well as on your laptop. Try using your package manager to install `firefox` if you don't have it yet.

- Enable the "DO NOT TRACK" setting in your web browser.
- If you're into Raspberry Pis (small $30 computers) get another one and install Pi-Hole[10] on it. This is slightly advanced (you have to adjust your Wi-Fi router settings as part of the setup) but is really effective in the home. It "hijacks" the internet phonebook system (DNS) that maps human-readable internet names (such as `https://this.com`) to the actual IP addresses that underlie networking (like `10.0.0.1`) and compares them to a crowd-sourced blacklist of advertisers and baddies. If the name is on the blacklist, the request just gets sent to the abyss. Ads on all your devices on your home network simply disappear. It's pretty nice. My Roku tries to talk to a blacklisted site like 5,000 times per day but can't get through. Muahahaha.
- Turn off your Wi-Fi and Bluetooth when you're in malls. They're apparently watching metadata from your phone's modem as you walk around even if you don't connect to the hotspot. Look up "people counters" if you don't believe me.
- Just don't ever shop or carry a cell phone.
- Move to Antarctica.

Using a VPN Service

Virtual Private Network (VPN) providers are useful for peace of mind, especially while traveling or in places where you don't trust the Wi-Fi operators (hotels, coffee shops, on travel with sensitive business information, hacker-friends' houses, etc.). When you get on a network, anyone else on the same access point and especially the people operating it can "sniff" the network to see what websites you're going to and, if there's no green lock in your browser window, what information you are

[10] https://pi-hole.net/

sending and receiving (though more and more websites have the green lock these days as a best practice).

You can pay a VPN provider that will help with this. You can click a button on your computer and it will create a strongly-encrypted connection between you and the VPN server. Then you will tell the VPN server which sites you want to interact with and it will go out and do your bidding, sending the results back to you through the super-encrypted channel. Anyone between you and the VPN is now in the dark as to what you're doing online. They can't see what servers you interact with nor what data is being sent to and fro. Instead of trusting dozens of operators and their staff along the way, you only have to trust the VPN provider. Huzzah!

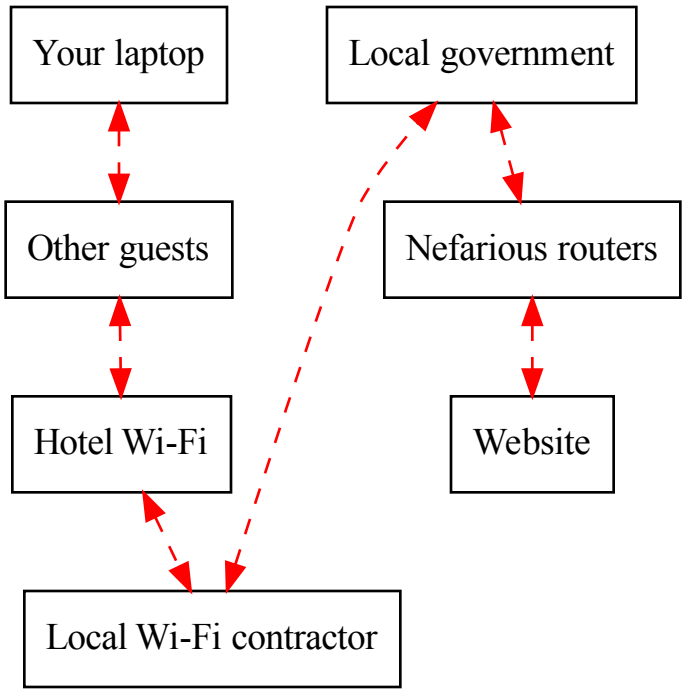


Figure4: Browsing the web from a hotel. Everyone along the way can see any non-TLS (i.e. green lock in browser window) traffic and all metadata, including what sites you're browsing and for how long. This is bad if you don't want people along the way to know what you're doing.

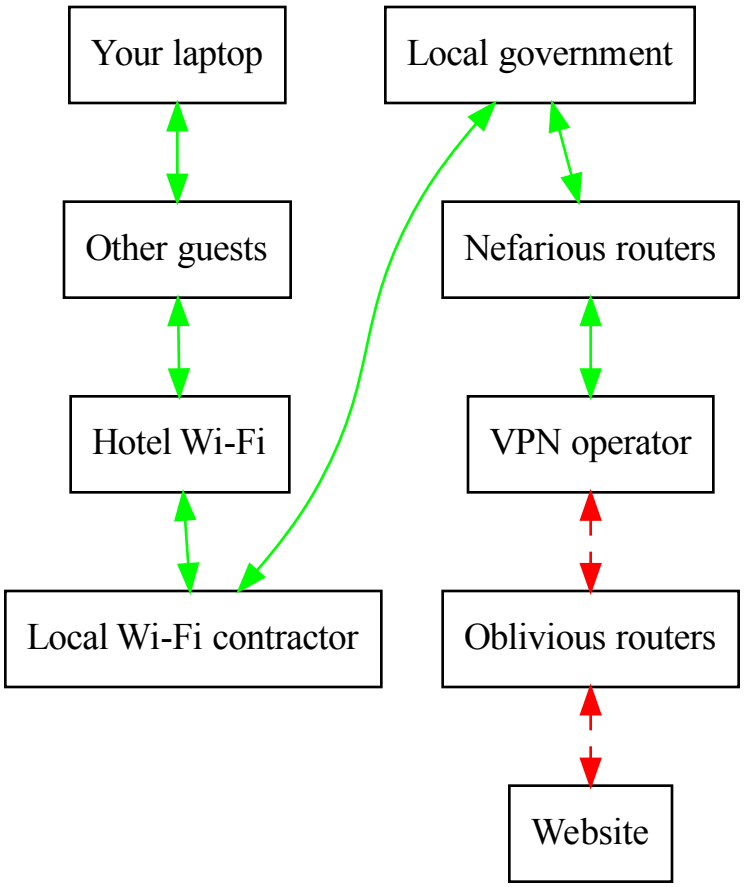


Figure5: Browsing the web through a VPN hides what you're up to from almost everyone. After the VPN service, metadata is available but it's anonymized to everyone except the VPN operator themselves. So you only have to trust the VPN operator in this scenario.

The downside here is that your connection will be at least a little slower since it has to route everything through the VPN service, wherever it may be.

Companies offering VPN services include ExpressVPN, Private Internet

Access VPN, IPVanish, NordVPN, Cyber Ghost, and many others. Prices are borderline $40–$50/year.

Instead of connecting each of your individual devices to the VPN, you can alternatively choose to configure your Wi-Fi router as a VPN client (only certain router models have this feature). I'm not sure this makes too much sense at home unless you really don't trust your ISP. Some people actually take a travel router with them on trips that automatically connects to their VPN, and they only connect through it. That is particularly neat.

Depending on your interests, you can also run your own VPN service right from your home as discussed in *Set up your own VPN Server*.

The Onion Router and the dark web

Speaking of network security, we have to mention the ultimate system in this regard: The Onion Router (Tor)[11] and the so-called *dark web*. The underlying technology was started by the US Naval Research Laboratory and later well funded by DARPA. Its job is to get you as close as possible to truly anonymous internet communication where the user doesn't have to trust anyone along the communication pathway. It prevents people from monitoring what you browse and it prevents the sites you visit from knowing where you are.

Web server admins can put their web page up as an *onion service* and it's supposed to be hard to figure out who's operating it and hard to shut down.

It's used by secret agents, hacktivists, normal people who want to try it out, people who want to look something sketchy up without leaving a trace, and many other legitimate people.

It's also used by criminals. There is a thriving black market for drug sales coordinated through Tor and other illicit stuff.

It's called the *onion* router because it has many layers. The Tor network consists of a bunch of nodes run by volunteers. These nodes pass encrypted onion traffic from node to node. When a user sends a request to

[11] https://www.torproject.org/

browse to `website x`, their system chooses a random pathway based on a network directory server, puts a bunch of layers of encryption over the request, and tosses it into the network. The first Tor server decrypts the first layer and finds the address of another Tor server so it passes the binary blob along. The second server pulls off the second layer, and so on. Eventually, when no layers remain, the website to visit is revealed. Whichever random server in the Tor network gets this goes out, grabs the requested information from the website, and sends the request back through the Tor network to the original user in an equally randomized way.

Tor is not impossible to see through. Governments monitoring networks can know when you're on Tor and if they really want, they can have their hackers use one of the thousands of unpublished major computer bugs (called *zero-days*) to just get on your computer and watch what you're doing directly. Or they can use *end-to-end correlation* to guess what you're doing. If you send a request that causes `website x` to send you some exact number of bytes, those bytes will eventually go to your computer. By monitoring the entire internet, governments can say, "OK well these 3987345 bytes came out of that server and, ope! 3987345 encrypted bytes just went into this guy's computer way over here right after that! I think I know what they're doing!"

Tor can be useful. But don't think you can get away with something horrible.

Using Tor is actually really easy. The Tor Project folks have set it up so you just download their web browser and it auto-connects to Tor and you're off. You can find phone apps that trivially connect to it as well.

From your package manager:

Table5: Installing Tor Browser

OS	Installation method
Windows	`choco install tor-browser`
Linux	`apt install torbrowser-launcher`
macOS	`brew install tor`
Android	Download `Orbot: Proxy with Tor` from Play store or F-Droid

Launch Tor Browser (it's in your Start Menu in Windows) and you're good to go. Don't go logging into your Facebook now though, that would deanonymize you rather quickly. On second thought Facebook did make a Tor Service address (https://facebookcorewwwi.onion/) and that might be a fun first Tor Service to try to access. This would be useful to access your Facebook when you're not supposed to. Again, if Facebook is banned where you are, connecting to Tor might raise some red flags too so just be careful.

Note: Due to the nature of Tor, browsing the web through it is inherently slower.

Planning a night of star-gazing

To wrap up this chapter, let's leave the network and take a step out into the yard or a nearby park. Have you ever wondered which star, planet, constellation, or moon you're looking at in the night sky? Ever need to know exactly when an eclipse is going to peak at a particular location? If so, you're in luck because your computer and phone can easily have all this information on it. Stellarium[12] is an open-source planetarium for your computer, available on all platforms, and Androids and iPhones have apps with some related features.

Table6: Installing Stellarium and other star-map software

OS	Installation method
Windows	`choco install stellarium`
Linux	`apt install stellarium`
macOS	`brew cask install stellarium`
Android	Download `Sky Map` from Play store or F-Droid
iPhone	Download `SkyView Lite`, or similar

[12] http://stellarium.org/

Stellarium allows you to choose your location and date/time (defaults to your here and now) and shows you the sky. You can search for particular planets or constellations, figure out which side of the house to set up for an upcoming eclipse, even zoom way in on deep-space objects like the Horsehead Nebula. You can step through in real-time or in fast-forward to see how the stars and planets will shift over the night or across the seasons. If you have a motorized telescope, Stellarium can hook into it and steer it to (and track) any visible object. It's a whole lot of fun and you or your family members will have a wonderful time searching around on it. It's basically a sky simulator.

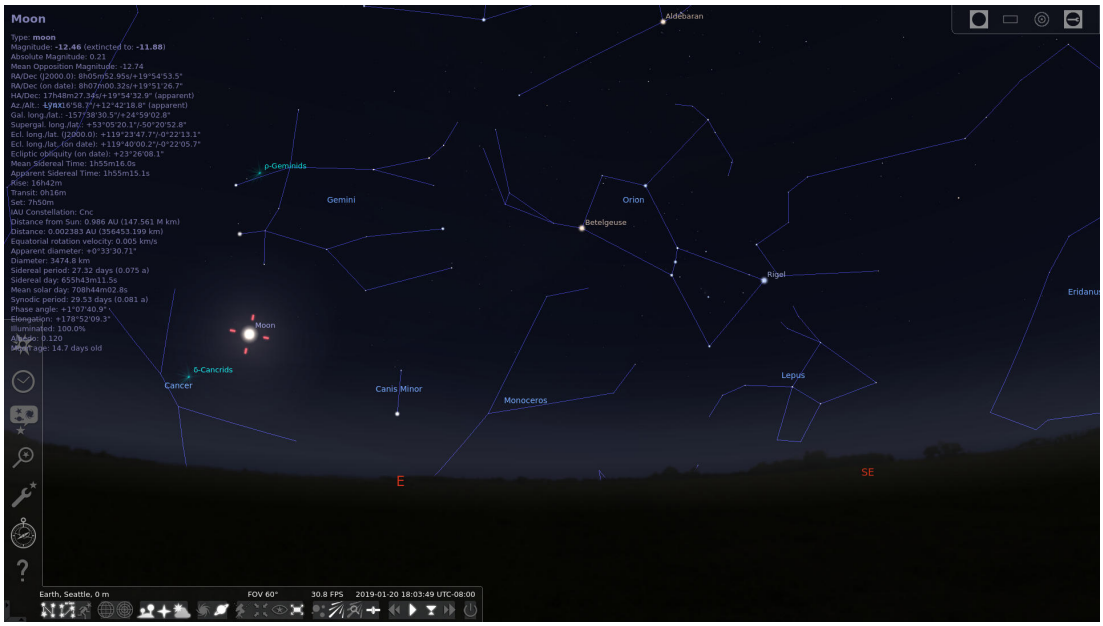


Figure6: Stellarium showing the night of a lunar eclipse.

After you have Stellarium installed, open it and you'll see its best estimate of your current sky in your current location. If you push the mouse to the left edge, some widgets will appear. The first one lets you adjust your location, and the second lets you adjust the time. Change these to see how the sky changes. If it's not night, try spinning the clock until it is night. When you push the mouse to the bottom of the screen, you'll see various toggles to turn on and off constellation labels and other points of interest. Now just click and drag in the screen to turn your view, and scroll with the mouse wheel to zoom in and out. If you click an object

like the Moon, you'll see all sorts of interesting information about it. This is a wonderful program for people of all ages.

That's all we'll cover for the home. I hope you are already enjoying some of your newfound superpowers!

4

Around the Office

Fluorescent lights. Cubicles. Bosses. TPS forms. Computers. Many of us dread these things even though we go into an office full of them every day and get paid to do so. I swore to myself in high-school that I'd never end up in one of those places! But when I actually got a job that involved cubicles, it wasn't nearly as bad as I expected. One thing I didn't realize was just how *mentally big* a computer can make a small cubicle. The computer is an infinite window. Inside a computer, we can make wonderful things happen. We can coordinate a project with people around the world. We have *phenomenal cosmic power* in an itty bitty living space[1].

But we have to make the computer our partner in crime to get into this mindset. The computer, as an extension of our brains, should be seamless and natural. This chapter shows how to make that computer at work a more valuable and cooperative life-comrade.

[1] That's an Aladdin reference...

Note on office suites

A lot of people spend a lot of their time in e-mail programs, presentation software, word processors, and spreadsheets. Most offices have at least a few people who are power-users of these programs and are relied upon for giving tips. The superpowers from the previous chapter should help a lot in these programs. Beyond that, skills in these tools are well-known or discoverable enough that we just aren't going to focus on them much. In many cases, the things you're learning here provide new options beyond your typical office suite.

Concentrating in offices

Offices are loud. Open floor plans are making them even louder. Many victims of such an environment have already discovered how important noise-canceling headphones are (they're worth it and can even drown out baby cries on airplanes). I still struggle sometimes because it's often difficult for me to concentrate when music is playing. I discovered a simple solution in the form of a wonderful web page called MyNoise.net[2]. They feature free, high-quality noise generators with different ambiances (like Café Restaurant, Rain, Temple Bells, etc.). I can concentrate fully again; they're really a lifesaver. Best of all, they recently released an app that can do offline playback which is a godsend on airplanes.

You can also make custom white noise in Audacity, discussed below. If you're on Linux, you also have a nice noise-generator available on the command line. You can pipe the random number generator into your speakers at various frequencies (e.g. 10 kHz) like this:

```
cat /dev/urandom | aplay -r 10000
```

[2] https://mynoise.net/

Note Taking

Organizing thoughts and notes is a time-honored challenge. People come up with all sorts of ways to take notes, ranging from using real or digital sticky notes for everything, to putting everything in one or many word processor or text files, to using fancy note taking applications. Several extremely capable note taking applications have emerged in the past few years that really do make a difference: Obsidian[3] and Logseq[4]. To a degree, these are based on concepts that first showed up in a commercial tool called Roam Research[5].

Table1: Installing logseq for note taking

OS	Installation method
Windows	`choco install logseq`
Linux	`sudo snap install logseq`
macOS	`brew install --cask logseq`

Logseq starts with a daily journal of bullet points. You can write how your day is going. You can put `[[` and `]]` characters around certain key phrases (for recurring meetings, big projects, specific people, specific things) when you mention them, and logseq will set up bidirectional links to them. So for instance if you mention `[[Tom]]` a bunch of times, you can `Shift-Click` on it as a link and a side window will open up and show all the times you ever mentioned **Tom**. It's exhilarating and extremely useful. It's the only note taking system I've used daily for over a year.

[3] https://obsidian.md/
[4] https://logseq.com/
[5] https://roamresearch.com/

Screenshots

Oftentimes you'll need to make a step-by-step tutorial for someone to follow on a computer system you're demonstrating at work. For this and other uses, *screenshots*, or capturing what's on your screen to a file or the clipboard can be useful. You can paste what's captured into a slide deck, document, or image editor. Generally, pressing the `Print Screen` button will capture the entire screen to a file. On Windows, check out `The Snipping Tool` (search for it by typing after clicking the Start button), which is nice because it will let you lasso a rectangle over the region you'd like to snapshot. On macOS, you just press `Command-Shift-4` to get a custom rectangle to snap (it will save to the Desktop unless you also hold `Ctrl`, which will send it to the clipboard). On Linux, the lasso-clipboard is usually `Shift-Print Screen`.

Text editors and extensions

Many digital superpowers we'll soon see involve manipulating files containing only text. There's minimalism and elegance in pure text files that puts them at the forefront of power. Text files are the universal medium upon which many digital power-tools operate. Text files are what web pages and new programs are written in (as you'll see in *Programming*). Some poets and writers prefer writing in pure text to minimize distractions.

"Surprisingly"[6], many computers don't come with overly useful programs for viewing and manipulating text files. We're all used to word processors but those add special whitespace characters and formatting controls in proprietary (or at least widely varied) formats that we want to avoid when dealing with text and data. Many computers come with default text editors (like `Notepad`) that are bad at dealing with text files properly.

Fortunately, this is easy to reconcile. In Windows, an excellent option

[6] Not too surprisingly; many software companies have a profit motive to push you into their more expensive software tools.

called Notepad++[7] can hop into your digital toolbox with a flourish via your package manager (see *Programs and package managers*) using:

```
choco install notepadplusplus
```

On a Mac, sublime[8] is one of many good choices:

```
brew install --cask sublime-text
```

On Linux, the often-included editors (like `gedit`) are likely sufficient for what we'll be doing here.

Tip: For those of you who work in text a lot, whether it be for writing books, programming, making web pages, etc., you may be interested in the next-level text editors like `vim`. It's available on all platforms and has a significant learning curve. Most people's first experience is rebooting their computer because they can't quit it (hint: use `:q`). But it is extraordinarily useful if you dedicate yourself to it. It took me years, but I got to a point where I couldn't live without it. It's basically an advanced magic wand. The best resource to get started is a book called Practical Vim by Drew Neil[9].

We should mention file extensions too. The extension is just the filename suffix; the stuff after the dot, like `txt` in `myfile.txt`. By convention, it determines the type of a file, and therefore which program your computer will pass it to when you click on it. Windows and macOS hide these extensions by default, but you can turn it them back on if you want:

[7] https://notepad-plus-plus.org/
[8] https://www.sublimetext.com
[9] https://pragprog.com/titles/dnvim2/practical-vim-second-edition/

Table2: Showing file extensions by default

OS	Operations
Windows	Start → Type `show or hide file extensions` → uncheck `Hide extensions for known file type`
Linux	Already shown ☺
macOS	Finder → Preferences → Advanced → `Show all file extensions`

In text editors, when you are creating a new file and you press `Save As`, the save dialog will let you specify both a file name and a file type. If the file type section is set to something like `Text files (*.txt)` and you type `myfile.dat` there is a reasonable chance it will actually get written as `myfile.dat.txt`. So if you want to specify a different extension (e.g. for making a `.py` file when making a Python program), then first choose `All files (*.*)` from the file type dialog and then type your desired extension in the file name area.

Column editing

This superpower really blew my mind when I first experienced it. Did you know you can edit an entire column of a text file at the same time? Well, you can, and it's called Column Edit or Block Editing. This can really be a time-saver in certain scenarios. For example, the screenshot below was made by entering Column Select mode and then just typing `testing 123 OK HERE GO` a single time:

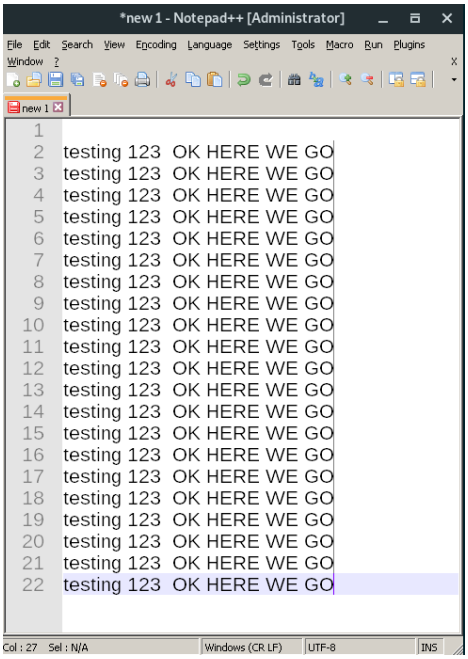


Figure1: An example of Column editing in Notepad++. Note that my cursor is 21 lines tall!

I could just as easily go in there and delete all the 0Ks in another easy swoop. This is remarkably useful for rapidly manipulating columnar data.

Column editing isn't available in every text editor, but here are a few that can do it:

Table3: How to get to Column Edit mode in various editors

Editor	How to activate Column Edit
Notepad++	`[Click starting point]`, then hold `Alt-Shift` and `[Click and drag]`
Word	`Alt + [Click/drag]` to select. Now you can delete or change formatting but not insert text.
Eclipse	`Alt-Shift-A` to toggle on and off
Vim	From starting point, press `Ctrl-v`. Now use navigation to select column. Now edit (i.e. with `I`) and escape.

Note: In macOS `Alt` is often replaced by `Command`.

Slicing and dicing PDFs

Portable Document Format (PDF) files were developed in the 1990s to ensure that documents would look as intended no matter what hardware or software the viewing user was running. Before them, files even from different versions of the same program could experience unintended format changes. The format is defined by an open standard, and as a result, plenty of programs can create and view PDFs, including many well-known office tools.

The task of modifying PDFs is often reserved for Adobe Acrobat Pro. If you only need to do simple modifications, however, you may enjoy exploring an endlessly useful utility called the PDF toolkit, or `pdftk` for short. This tool will allow you to mix and match PDFs, combine pages from different documents, remove pages, encrypt/decrypt, add watermarks, fill out forms, and so on.

Table4: Installing pdftk "server" (command line PDF tool)

OS	Installation method
Windows	`choco install pdftk`
Linux	`apt install pdftk`
macOS	Go to official download page[10] and get Mac version

To try it out, let's add a cover page to a PDF. If you have a cover page called `cover.pdf` and another called `document.pdf` and you'd like to put the cover page on the document, `pdftk` can do it with its `cat` command (short for *concatenate*, which means *slap together in a certain order*):

```
pdftk A=document.pdf  B=cover.pdf cat B A output book.pdf
```

In that command we called `document.pdf` a shorter name, `A`, and we called the cover `B`. Then we invoked the `cat` command and asked it to first put all of `B` and then all of `A` into a new PDF called `book.pdf`. You can also make the cover go in as the *second* page by specifying more complex concatenation sequence, like this:

```
pdftk A=document.pdf  B=cover.pdf cat A1 B A2-end output book.pdf
```

Want to require viewers to type a password of `j4j2jxk4` before they can read a PDF? That's just:

```
pdftk book.pdf output book-encrypted.pdf user_pw j4j2jxk4
```

Many more commands and operations are available, and you can read all about them in the pdftk manual[11].

Note: You'll see a whole world of PDF creation from scratch in *Publishing*

[10] https://www.pdflabs.com/tools/pdftk-server/
[11] https://www.pdflabs.com/docs/pdftk-man-page/

While we're looking at PDFs, you may sometimes happen up on a PDF where you can't highlight or search through the text. Many tools can remedy the situation by scanning over the file, recognizing characters, and embedding the text into the file. My favorite utility for doing this Optical Character Recognition (OCR) is OCRmyPDF[12], which is a simple and powerful interface riding on top of some cutting edge OCR libraries.

Table5: Installing ocrmypdf (command line PDF OCR tool)

OS	Installation method
Windows	A see instructions online[13]
Linux	`apt install ocrmypdf`
macOS	`brew install ocrmypdf`

To use, just run:

```
ocrmypdf input.pdf output.pdf
```

The `output.pdf` file generated will have a OCR layer and be in the PDF/A format. I use this all the time.

Ultimate find and replace (*regular expressions*)

Brace yourself, you're about to learn about *regular expressions* (REs), perhaps the most powerful but poorly-named superpower yet. REs are expressive ways to describe a pattern that you want to find and/or change. They are famous both for looking very opaque and for saving thousands upon thousands of hours of painfully tedious work while causing rapid promotions. There's even a webcomic[14] about REs being a superpower.

[12] https://ocrmypdf.readthedocs.io/en/latest/index.html

[13] https://ocrmypdf.readthedocs.io/en/latest/installation.html#installing-on-windows

[14] https://www.xkcd.com/208/

Imagine you have a thousand phone numbers formatted like so:

```
(509) 123-4567
```

But your boss is trying to be more international and has asked you to convert them all to the format:

```
+15091234567
```

You could go through one by one and do them, but that'd be tedious, inefficient, and frustrating. Worse, what if you had ten thousand to modify? You can't use regular-old find and replace for this because each phone number is slightly different. This is exactly the kind of thing regular expression are made to handle!

REs have special symbols to match different classes of characters. For instance, `\d` represents any single number. So the following regular expression matches all phone numbers in the format above:

```
\(\d\d\d\) \d\d\d-\d\d\d\d
```

We used `\(` and `\)` on the parentheses because parentheses without backslashes are used to *group* things. We can use the + shortcut which means "match one or more" to write the same RE as:

```
\(\d+) \d+-\d+,
```

which will still match the phone numbers, and would also match patterns that have 20-digit long area codes (if that were a thing for some reason). See? They look scary but aren't that bad!

Here are some RE identifiers and their interpretations:

Table6: A small subsection of RE identifiers

Character	Interpretation
\d	Any digit (0-9)
\s	Any whitespace (space, tab, etc.)
\S	Any non-whitespace (letters, numbers, punctuation)
+	One or more of previous pattern
*	Zero or more of previous pattern
\w	Any alphanumeric character, or underscore
.	Any single character of any kind
(...)	Define a new group matching whatever pattern is inside

If you want to match an actual plus sign or an actual parenthesis, you have to precede it with a backslash, like \+. This gets tricky because you can have expressions with both + and \+ which mean totally different things (one or more vs. actual +).

Now for the superpower of RE find/replace. Let's define 3 groups, one for the area code, one for the prefix, and another for the number by using parentheses:

```
\((\d+)\) (\d+)-(\d+)
```

All we did was put non-backslashed parentheses around each group of numbers. In doing this we defined three groups: Group 1, Group 2, and Group 3. Group 1 is the area code, Group 2 is the prefix, and Group 3 is the last four numbers.

When we're telling a RE find/replace system what to replace our matches with, we can refer to those groups by number using \1 for Group 1, \2 for Group 2, and so on. This allows us to mix and match the groups and even change their order.

For our example, we want to replace each match with +1, followed by the area code (\1), then immediately the prefix (\2) and then the rest (\3) so our replace pattern looks like this:

```
+1\1\2\3
```

Note that we used non-backslashed + because in the replace pattern, we don't have to worry about the special meaning; that only matters in the match pattern.

There are lots of programs that can perform RE find/replacements for us. One simple option is to use Perl, which is available in your package managers[15]. Yes, it's an entire programming language, but it's pretty small, and we're only going to use one tiny feature of it. To install, use one of these:

Table7: Installing Perl

OS	Installation method
Windows	`choco install strawberryperl`
Linux	`apt install perl`
macOS	`brew install perl`

The syntax we need to have `perl` swap one pattern for another is weird looking, but straightforward. It starts with the program name (`perl`) then some options `-pe` which tell it to run an in-line program and print the output, then the `s/` command (for substitute), then the `MATCH PATTERN`, then a `/` separator, then the `REPLACEMENT PATTERN`, then a `/` to close out the replacement pattern, then a `g` for *global* (meaning change all occurrences on a line, not just the first), and then finally the filename to operate upon. So it's going to *substitute* the match with the replacement on each line and print the results. Explicitly:

```
perl -pe 's/MATCH PATTERN/REPLACEMENT PATTERN/g' filename
```

Let's try it out. Make a file called `input.txt` in a folder somewhere:

```
(509) 123-4567
(123) 555-4567
```

(continues on next page)

[15] Arguably, `sed` is the simplest program to do line-by-line RE find/replace. However, it uses a more basic (and in my opinion uglier) RE format instead of the Perl-style REs.

(continued from previous page)

```
(000) 509-1234
(555) 551-0000
```

Now run this command in your command line (filling in `MATCH PATTERN` and `REPLACEMENT PATTERN` from above):

```
perl -pe 's/\((\d+)\) (\d+)-(\d+)/+1\1\2\3/g' input.txt
```

This should produce the following output:

```
+15091234567
+11235554567
+10005091234
+15555510000
```

Wow! To further illustrate, let's leave the match pattern the same and make a more complex replacement pattern, changing the order of the groups and everything:

```
perl -pe 's/\((\d+)\) (\d+)-(\d+)/Prefix: \2 Num: \3 Area: \1/g'␣
↪input.txt
```

This gives:

```
Prefix: 123 Num: 4567 Area: 509
Prefix: 555 Num: 4567 Area: 123
Prefix: 509 Num: 1234 Area: 000
Prefix: 551 Num: 0000 Area: 555
```

Tip: If you want to create a new file with that output, redirect it to a file by adding the following to the end of the command:

```
> output.txt
```

This is a fairly universal way to save command line output as a file rather than printing it on the screen.

Or even better, if you want to find and replace inside the file itself, use the in-place option of perl with `perl -i -pe` followed by the rest of the

replacement command.

Let's do one more example with something other than just numbers. Imagine your input looks as follows:

```
Dr. Michelle Obama
Dr. Newt Gingrich
Dr. Peter Pan
Dr. Jane Eyre
Dr. Winston Churchill
Dr. Herodotus Johnson
```

Further imagine that we want to get rid of the `Dr.`, add `Mc` to their last name, and add `M.D., Ph.D.` at the end. The grouped match pattern would be:

```
(Dr\.) (\w+) (\w+)
```

The replacement would be:

```
\2 Mc\3, M.D. Ph.D.
```

Notice that we never referred to Group 1 in the replacement, so the `Dr.` will be dropped entirely. See if you can figure out the command required to get this output:

```
Michelle McObama, M.D. Ph.D.
Newt McGingrich, M.D. Ph.D.
Peter McPan, M.D. Ph.D.
Jane McEyre, M.D. Ph.D.
Winston McChurchill, M.D. Ph.D.
Herodotus McJohnson, M.D. Ph.D.
```

Now that's a superpower!

If you prefer to not use the command line for this, things like Notepad++ can do these kinds of operations with graphics. Even MS Word has some RE capabilities these days. Check out RegExOne[16] for a great resource to interactively learn more RE syntax.

[16] https://regexone.com/

Tip: Useful related searches to explore:

- `regex find replace **program name**`
- `regular expression to match e-mail address`

Encrypted communications

We talked about how to keep the bits on your hard drive safe in *File Encryption* and how to browse the web with encryption in *The Onion Router and the dark web*, so now let's talk about how to securely communicate directly with our friends and business associates over networks.

Encrypted text messaging, photos, and voice

For text messaging and encrypted voice calling over the internet, there's really no good reason to not use Open Whisper System's Signal[17] app on your phone and computer. This is an open-source, user-friendly app (with a Desktop version) that does it all without any complications. The encrypted voice sound quality is far superior to normal, and the messaging is very straightforward. If that doesn't convince you, it was made by a dreadlocked anarchist hacker sailor named Moxie Marlinspike and is funded through privacy-oriented donations to its foundation. There is no profit motive to this organization to snoop on you or build a profile of you. Also, Edward Snowden uses it. Go to https://signal.org/download/ to get it on any phone or computer. I like having it on my computer for longer text conversations because I can type faster there.

[17] https://www.signal.org/

How to send, receive, and verify info securely over the public internet

We've all needed to send some confidential document to our associate in Ecuador, but we often don't trust anyone between here and there. How do we do it? Basic e-mail is a no-go because e-mails bounce around from server to server between users and can easily be intercepted. If we had arranged a shared password in advance, we could have used it to hide the document, at least once, but we didn't (though Signal could help with this).

Your computer has the ability to use an incredible technology called public-key cryptography to solve this problem. It's useful in every-day business between contractors, suppliers, employees, friends, and lovers.

Each individual in an asymmetric cryptography system needs two mathematically-related digital keys (actually they're just large numbers) that can be used to encrypt, decrypt, and sign messages. One will be held secret, and is called the private key. The other is called the public key, and is widely distributed in the open to everyone we want to be able to send secure messages to use or verify that we sent messages they receive from us. Together, they're called a key pair.

The details behind this math have a fairly straightforward analogy. If you think of a lock on a box, the public key turns the lock one way, and the private key turns it the other way. So if my colleague puts a document in the box and locks it with *my public key*, the only possible way anyone can get the document back out is to have my private key, and the only one who has it is me.

As it turns out, we all have immediate and full access to this wonderful and powerful technology. An open-source implementation has emerged as the GNU Privacy Guard[18] (GnuPG). Its most commonly-cited inadequacy is "Poor understanding by the public". The situation is such that people in regulated industries are buying super-expensive secure portals to transfer e-mails and files between business partners. All for want

[18] https://gnupg.org/

of a good user interface. No longer![19]

Let's drop down into the command line and see how easy it really is to send someone a truly secret message. There are GUI tools for this too, but the command line is fun, powerful, and easy.

First, install GnuPG:

Table8: Installing the `gnupg` encryption tool

OS	Installation method
Windows	`choco install gpg4win` (includes GUI) or just `choco install gnupg-modern`
Linux	`apt install gnupg`
macOS	`brew install gnupg`

Now we can do some cryptography.

Generating your own private/public key pair

Run the following command to generate a cryptographic key pair:

```
gpg --gen-key
```

You will encounter a series of prompts that you should fill in with your name and e-mail address. When it prompts you with `Change (N)ame, (E)mail, or (O)kay/(Q)uit?`, choose `O` if all looks good and press `Enter`. Now it will ask for a password that you'll need to type every time you access your *Private* key. If all goes well, there will be a short delay while it generates good random numbers, and you should see text along these lines:

```
gpg: key 999B22A1FD07D5CD marked as ultimately trusted
gpg: revocation certificate stored as '/home/nick/1FD07D5CD.rev'
public and secret key created and signed.
```

(continues on next page)

[19] Granted, some commercial secure portals offer features that plain GnuPG may not have, such as a master key for an IT administrator to unlock secret messages if an employee is fired or forgets their password, which is sometimes a business requirement.

(continued from previous page)

```
pub   rsa3072 2018-12-26 [SC] [expires: 2020-12-25]
      7C32B1F5C7124D4D5607EC04999B22A1FD07D5CD
uid    Nicholas William Touran <encryption-
↪demo@digitalsuperpowers.com>
sub   rsa3072 2018-12-26 [E] [expires: 2020-12-25]
```

You're doing some pretty fancy computer work now!

Encrypting a file

Make or choose a file to try encrypting. I'll make a little text file called `secret.txt`:

```
all my secrets are in this file.
Secret 1
Secret 2
```

To encrypt it, run the encryption command with the `armor` option (to put it in a nice format) and the `recipient` option (`-r`), filling in your e-mail address followed by the file name we wish to encrypt:

```
gpg --encrypt --armor -r encryption-demo@digitalsuperpowers.com␣
↪secret.txt
```

Note: As is typical on the command line, you can type `gpg --help` to see a list of available commands and short descriptions.

In the same folder, you'll now see a file called `secret.txt.asc` which will look something like this:

```
-----BEGIN PGP MESSAGE-----

hQGMA0d8dZFzT0eOAQv/YOigihUbA5ZEkCouh+xnDtPOUpWQctl8nL5kfituq3JH
HxS5Oxjf36N0K/Gh9K67mopZd/bd4/N8inrjXDMQL/pR7oLwRd2YHkCBga25W6oc
34POd0SgvNrhM6l9x/ZIcPPGJzZqhkCXtu6kCby6UbpXfcU4EqI2MRAoTe/06yra
RxcN6p4Vxh2XE/HT/p/eYgJhvx+CFZhFfzemiyiegGFLbD52O1wcGFt8Qojem2+d
Y2nTNAB7FWOJQO6skXhActo3oRgfSJZHEpR5re1XK+AFJTyP+++1MZSa919qlIe/
eGzOhSr4hEPyac1Ds1GZqUsjdqyksh6vrIEgoLE85JBB2p4LlnMMpsgnMYuyx/Nl
9XAWI9RsvehXpnqWIB4oxu3MNZ8Al5VjD68D6VmCNs1sk+M/Fl/DxE0+tWnzRqZ8
DAjLPHugjnJmVMGtbiSOuJeeqZKqZ2e36ZtrTsv7uuTlLZsfBjjlTcNNmFnjHpkQ
FbTJQDufC2GPwMcFEnD70moBKu6XGlnicvTUHg2i/zw/hbjsy/mddz3aMT74cTrn
5GsFBwpb5F5kulK91prsEJl+BA73M/NF8dPrgVq9HXS0K5bzEu68ZyBL2/CJFRrL
PjPyKb54r3LT6b31H5AnH/+O6P+2PwHifWUe
=Kct/
-----END PGP MESSAGE-----
```

Figure2: A very strongly encrypted message that really beats the heck out of Pig Latin.

If you encrypted that with one of your friend's or co-worker's public keys, you would not be able to decrypt it; only they could (that's why public-key encryption is also called *asymmetric*). But in this case, you sent it to yourself so you should be able to decrypt it with your private key.

Decrypting a file encrypted against your public key

You can decrypt any file encrypted against your public key with your private key. If in the previous step you entered your own e-mail as the recipient then you can try this out:

```
gpg --decrypt secret.txt.asc
```

It will prompt you for the password you made while generating your key pair. If you enter it correctly, you will see something like:

```
gpg: encrypted with 3072-bit RSA key, ID 477C7591734F478E,␣
↪created 2018-12-26
      "Nicholas William Touran <encryption-
↪demo@digitalsuperpowers.com>"
all my secrets are in this file.
```

(continues on next page)

(continued from previous page)

```
Secret 1
Secret 2
```

It's working! If you want to make a decrypted file rather than just printing to the screen (useful for any non-text file like a `zip` or `pdf`), just add the `-o output_name` option right after the `--decrypt` flag.

Exchanging public keys with friends

For this to be useful we have to make sure our friends know our public keys so they can encrypt stuff to us, and vice-versa. To write your public key to a file, run (with your e-mail, not mine):

```
gpg --export --armor -o myname-publickey.asc encryption-
↪demo@digitalsuperpowers.com
```

Now e-mail the `myname-publickey.asc` to your friends. When you receive a key from a friend called `friendsname-publickey.asc` you can import it into your system by running:

```
gpg --import friendsname-publickey.asc
```

Easy! Once you have your public key, spread it widely. You can even put it on your Facebook profile in the Contact Information section.

Exercise

Convince a friend also generate a key pair and have them send you their public key (e-mail is reasonable to start). Import their key and encrypt a message to them. See if they can receive it. Have them send you one back.

Warning: You have to make sure your friend's public key is the one you got. Check the *fingerprint* that's printed out upon import and verify it over another channel (like Signal!) or in person. Once you trust that the public key you got is indeed from your friend, the rest

of the system should be rock solid. Don't forget to keep your private key private too.

Signing information with our keys

This technology also enables the incredible operation of electronically *signing* documents such that anyone with your public key can verify that you signed it and that it *hasn't been modified since.* This is much more powerful than just slapping a scan of your physical signature on the bottom of a document; that's easy to forge by everyone who works at a restaurant where you signed the receipt before, and offers almost no protection against people modifying the document after you signed it.

The one-way key analogy from above works here too, as long as you can imagine that the locked box has *two* locked positions and one unlocked position between them (i.e. it's locked in the 9 o'clock and 3 o'clock positions, but unlocked in the 12 o'clock position). To sign a document, I'd put it in the box and then lock it to the 3 o'clock position with my private key (which can only turn the lock clockwise). Anyone and everyone with my public key can open it, thus proving that it was indeed I who put the document in the box.

Let's try signing a document. Make a new document and call it `declaration.txt`:

```
I hereby declare that Little Johnny is sick
with fever and needs the day off. The fact
that there's a spelling test today is merely a
coincidence.
```

Sign it with the following command:

```
gpg --detach-sign --armor declaration.txt
```

You will have to type your password to unlock your private key. If you have more than one private key, you can choose one with the `-u encryption-demo@digitalsuperpowers.com` command (with your e-mail address, of course). Now you'll find a file called `declaration.txt.asc` alongside the original file. This is the signature! Send it along

with the file to your contacts or associates, or post it online. Given the file, the signature, and your public key, anyone (including yourself) can verify the signature on the receiving end. Here's how:

```
gpg --verify declaration.txt.asc
```

You should see a message indicating success that looks like:

```
gpg: assuming signed data in 'declaration.txt'
gpg: Signature made Fri 11 Jan 2019 06:42:32 PM EST
gpg:                using RSA key␣
↪7C32B1F5C7124D4D5607EC04999B22A1FD07D5CD
gpg:                issuer "encryption-demo@digitalsuperpowers.
↪com"
gpg: Good signature from "Nicholas William Touran <encryption-
↪demo@digitalsuperpowers.com>" [ultimate]
```

Exercise

Try modifying the declaration.txt after signing it and see what the verification step does.

Encrypting with GUIs

Admittedly, not everything is best in the command line. There are ways to integrate this into e-mail. For instance, the e-mail client, Thunderbird[20], works great with any kind of e-mail account and has built-in support of GPG encryption. Since most people are using web-based mail, it's also worth highlighting FlowCrypt[21], which is a Chrome extension that puts these features on top of a Gmail account. Some people electronically sign all their outgoing e-mail for good measure automatically with these tools.

For drag-and-drop file encryption/decryption that you can then e-mail around without any special extension, Windows users can look at the

[20] https://www.thunderbird.net/en-US/

[21] https://flowcrypt.com/

GUIs that come with Gpg4win, and other OS users will find similar tools for their system with brief web searches.

Note: Estonia actually went all-in on this kind of thing and put cryptographic key pairs on their national IDs. They had a hiccup in that the system used to generate the keys was weak and then it was hard to replace them, but I think they're absolutely on the right track. This kind of thing can make our lives more efficient and secure in many ways. Our social security numbers are significantly weaker than those Estonian keys. If people learn about these capabilities and realize that they can use them right away on their computers, I hope we can improve process efficiencies and security. Data is important and it's important for us to become more comfortable with encryption.

Making flowcharts

Flowcharts come in handy for communicating process as well as mind-mapping complex situations. Three open-source tools capable of making good flowcharts are Graphviz[22], MermaidJS[23], and dia[24]. `Graphviz` and `Mermaid` are unique in that the nodes and connections are expressed in text, and the system generates the actual flowchart. This is favorable when you aren't interested in the precise layout of the flowchart (it's determined algorithmically). It can be useful for making charts by hand, but especially shines for data manipulators who need to generate flowcharts dynamically, based on some data set or user interaction.

Here is what a `graphviz` flowchart looks like:

[22] http://graphviz.org/
[23] https://mermaid.js.org/intro/
[24] http://dia-installer.de/

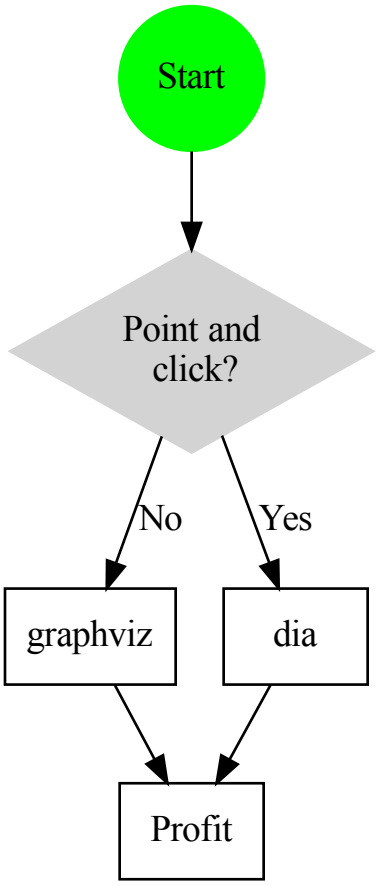


Figure3: Choosing a flowchart

The source to make that flowchart is:

```
digraph fig {
  // First we'll define a few nodes
  Start [shape=circle,style=filled,color=green];  // single node
  node [shape=box]; graphviz, dia, Profit;        // node list
  decision [shape=diamond,style=filled,color=lightgrey, label=
↪"Point and\n click?"];
```

(continues on next page)

(continued from previous page)

```
  // Then we define the connections
  Start -> decision;
  decision -> dia [label="Yes"];
  decision -> graphviz [label="No"];
  {dia, graphviz} -> Profit;
}
```

To do such a thing, you first install graphviz:

Table9: Installing the `graphviz` tool

OS	Installation method
Windows	`choco install graphviz`
Linux	`apt install graphviz`
macOS	`brew install graphviz`

and then build the flowchart with a command like this:

```
dot -Tpng flowchart.dot > flowchart.png
```

(Hey! There's that output redirection that we saw earlier. Don't forget it or else you'll see what a text representation of a binary `png` image file looks like.)

The downside is that it's hard to get the flowchart looking just how you want it. That's where `dia` comes in, allowing you to just draw out your own in detail. If you do want to use declarative text to create flowcharts with precise layouts, the more advanced library that can do this is called Tikz[25].

[25] http://www.texample.net/tikz/

The GNU utilities

An exceedingly useful set of small, free utilities is often packaged together as *The GNU utilities*. A few of them are obscure but absolutely worth trying out.

Table10: Installing the GNU utilities. Many, but not all, of these are pre-installed on Linux/macOS. Here, we show an example of installing one that isn't pre-installed. Others are similar. On Windows, the installation can take 10 minutes.

OS	Installation method
Windows	`choco install gnuwin` (a bit outdated but will work, I also had to press `Spacebar` when this installation hung...)
Linux	`apt install units`
macOS	`brew install gnu-units`

Unit conversion

Perhaps my favorite utility here is `units`, which converts units from one system to another. It can be run interactively or with one-offs. Let's show it interactively first. Type `units` and press `Enter`. Type `10 cups` at the prompt and press `Enter`. Then type `tablespoons` and press `Enter` again. Behold! It tells you that there are 160 tablespoons in 10 cups:

```
You have: 10 cups
You want: tablespoons
    * 160
    / 0.00625
You have:
```

That's not even getting started. Press `Control-C` to quit. Now let's try it in one-liner mode. You just say `units "what you have" what you`

want. For the previous example, that'd be:

```
units "10 cups" tablespoons
```

This thing has over 3000 units in it. Here are some more to try:

- `units "2 dozen feet" meters` gives `7.3152`
- `units "2 bakersdozen feet" meters` gives `7.9248`
- `units "2 dozen bakersdozen feet" meters` gives `95.0976`
- `units "1.7 meters" "feet;in"` gives `5 feet + 6.9291339 in` (Woah! It split those out!)
- `units "tempC(32)" tempF` gives `89.6`[26]
- `units "1 googol femtobarns" "teraparsecs^2"` gives, astoundingly, `1.050265` (that's an obscure nuclear unit of area converted to square astronomical units of distance made famous by the Millennium Falcon's Kessel Run)

Read more about it at https://www.gnu.org/software/units/

Pattern matching with grep

The `grep` utility **g**ets a **r**egular **e**xpression and then **p**rints it. This is endlessly useful when dealing with large datasets, especially when chained together with other utilities. For the next few examples let's assume we're maintaining a website for our company and it involves a deeply nested folder structure with thousands upon thousands of text files in it (HTML, CSS, etc.). At the bottom of each page is a footer that says something like `Copyright © 2025 My Company, LLC`. Say we want to just find and print all of those for any year. We can use `grep` with the `-R(ecursive)` and `-P(erl mode)` options to do this:

```
$ grep -RP 'Copyright © \d\d\d\d My Company' *.html
```

And you'll see:

[26] This special syntax is to differentiate absolute temperatures from the different sizes of the temperature units.

```
Copyright © 2025 My Company, LLC
Copyright © 2025 My Company, LLC
Copyright © 2024 My Company, LLC
Copyright © 2023 My Company, LLC
...
```

Extracting a column with awk

The last command printed out every line. But what if we only want to grab the year out of that because we're going to put it in a spreadsheet and do something with it. The `awk` tool can be chained to `grep` to process each line that `grep` prints. There's a whole world of `awk` commands, but one to know is the `print` command, which can print the first, second, third, or nth column in an output stream. In the example above, the year is the *third* column (space-delimited) so we would use this (`|` can be typed with `Shift - \`):

```
$ grep -RP 'Copyright' *.html | awk '{ print $3 }'
```

Giving:

```
2025
2025
2024
2023
```

Tip: Throw a `| clip` on the end of that last command and the results will be in your system clipboard. Now you can paste into another program, like a spreadsheet.

Multi-file find/replace

You've seen how we can find matches in multiple files and even down-select things in the lines we want to see. Earlier, you saw how to do RE find/replace with `perl`. For the grand slam, we need to find and replace in all those files. In our example, we will update all © years to 2026. Here's the command if all the files were in one directory:

```
perl -i -pe 's/Copyright © \d\d\d\d/Copyright © 2026/g' *.html
```

When files are in multiple directories, we can chain commands together. First, we'll find all the files we want to cover in all directories, then we'll send each one to the above `perl` command to do the actual in-place modifications. In the command-line philosophy alluded to in *The command line*, having small tools that can be chained together is the best way to provide power, simplicity/maintainability, and flexibility. Here, we will use the `find` command to find all files recursively that match a pattern, then we'll use the `xargs` command to translate that list of files into input for the `perl` find/replace command.

We need the names of all the files in our scope. In this example, it's any `.html` file in this or any child folders. To list those, we can use the GNU `find` command:

```
find . -iregex '.+\.html'
```

Try it out and you'll see all `html` files in a folder (put some there if you need to build this example up). That command is looking in the current folder (called `.`) for any files matching the regular expression listed, which, as you learned above, is anything ending in `.html`.

Warning: On Windows, a built-in `find` command will conflict with the GNU one we are using here, so you may have to type the full path, like:

```
C:\GnuWin\bin\find.exe . -iregex '.+\.html'
```

To pass the results of the `find` command on to another command, we use `xargs` as follows (first we'll just re-print the file names):

```
find . -iregex '.+\.html' | xargs echo 'Found files: '
```

As you can see, all files found have been passed as one long chain of arguments to the `echo` command. Now we can substitute our `perl` command in and let the magic happen:

```
find . -iregex ".+\.html" | xargs perl -i -pe "s/Copyright © \d\
↪d\d\d/Copyright © 2026/g"
```

Behold: You now have the multi-file, multi-directory, regular-expression find and replace superpower! Remember, with great power comes great responsibility. Consider testing out a sweeping command like this (here by leaving out the `-i` in-place argument) before actually changing all the files.

5 Art Studio

Computers are integral to the workflows of photographers, cartoonists, videographers, graphics designers, musicians, podcasters, and many other artists. They're also a primary medium through which art is experienced by patrons. In this chapter, we'll explore some superpowers related to art and other creative activities. Even if you're not an artist, you'll find uses of these powers in your work, home, and hobbies.

Basic image manipulation

You will often want to crop an image, rotate it, add some text, touch it up slightly, or make significant modifications. Adobe Photoshop has held the crown for professionals in this regard for many years. One high-quality and open-source program that also excels in doing these things is GIMP[1]. We will go through a few basic examples of how to use it here.

[1] https://www.gimp.org/

Table1: Installing GIMP

OS	Installation method
Windows	`choco install gimp`
Linux	`apt install gimp`
macOS	`brew install --cask gimp`

Once you open an image in GIMP, you will find many options, including these useful and basic ones:

- Image → Scale Image → Change image size by pixels or percentage
- Image → Transform → Rotate → Rotate or flip image

The toolbar on the left has lots of tools for cropping and selecting while the tools on the right deal with layers.

The clone stamp tool

The *clone stamp* tool is a wonderful superpower. This tool effectively lets you erase things from images, or duplicate things. You have to try it to believe it.

Here's a photo of a classic watch. On the left is the original, and on the right is a version that has been clone stamped a lot.

Figure1: The photo on the left is the original, and the one on the right has been clone stamped.

Is it magic? Nope! It just allows you to clone parts of the image onto another location. So it didn't *see* under the digits; that's impossible. Instead, the space between digits was carefully cloned over the digits. Less obviously (and more usefully), the smudge over the CASIO logo has been removed, and some scratches were removed on the band just below the face. Here's how to do it in GIMP.

- Load up your first image and choose the clone stamp tool (hover your mouse over all the tools in the toolbar until you find it; it kind of looks like a stamp). Your cursor will now be a dotted circle. Adjust the size by changing the `Size` slider in the left-hand toolbar.
- Find the area you want to clone from and put the mouse over it.

Press and hold the `Ctrl` key (it may be `Command` in macOS) and click once with the left mouse button to mark the *clone-from* area. In my example above, this was the empty space between digits.

- Zoom in as much as you need (try `Ctrl` with `Mouse Wheel`)
- Now move the mouse to where you want to clone to. Click with the left mouse button and paint over it carefully. You will see an indicator in the *clone-from* area moving in sync with your cursor. If/when you mess up, press `Control-Z` to undo.
- Adjust the *clone-from* area a few times to make it look extra realistic (by repeating the second step).

Exercise

Try the clone stamp tool out on one of your photos. See if you can add or remove someone from a family portrait, or do something else fun. Note that you can load two photos at once and clone from one to the other.

Note: Speaking of panoramics, the `hugin` tool is a very powerful panoramic stitcher. Yes, cell-phones often have this built it these days, but the feature is still pretty useful in certain cases, and gives you as much or as little control as you'd like.

Image manipulations from the terminal

Editing images from the command line may sound odd, but its utility comes up surprisingly often. You can modify images dynamically as part of a bigger system (like an interactive website that applies filters to images that people upload, for instance, hint hint), do something complex without a bunch of clicks, and apply the same modification to a whole set of images.

For example, if you set your camera out all night with an intervalometer and star-tracking camera mount to capture a new astrophotography time-lapse (or any other kind of time-lapse), you may want to crop them all the same, add some text, put a border around them, etc. Or maybe

you have a bunch of still images from a simulation and you want to make them into a movie but need to shrink them all down first. A delightfully powerful suite of tools called ImageMagick[2] can do all sorts of things like this.

Table2: Installing ImageMagick

OS	Installation method
Windows	`choco install imagemagick`
Linux	`apt install imagemagick`
macOS	`brew install imagemagick`

The simplest thing it's good at is converting image formats. To convert a `JPG` to a `PNG`, just run:

```
convert picture.jpg picture.png
```

You can convert them to `pdf` or `bmp` or many other things. If you have a bunch of files, use `mogrify` instead of `convert`:

```
mogrify -format png *.jpg
```

What if you want to make a bunch of thumbnails for all the `JPG`s in a folder, while rotating them 45° and making them black-and-white? We got you:

```
mkdir thumbnail
mogrify -path thumbnail -thumbnail 100x100 -rotate 45 -
↪colorspace Gray *.jpg
```

Imagine having to do that on 100 images, or 1000, by hand!

Another fun one is the `montage` command (also part of ImageMagick). You give it some images and it slaps them together in a montage of whatever shape and size you want. I had some nice pictures of the August 2017 Eclipse, and made a montage with each individual square being reduced to 960x960 pixels and with 10-pixel margins in each direction with this:

[2] https://www.imagemagick.org/

```
montage IMG*.jpg -geometry 960x960+10+10 eclipse.jpg
```

Which resulted in this:

Figure2: A beautiful Eclipse montage from Malheur National Forest, OR. By the way, I highly recommend seeing a total eclipse someday if you get the chance.

You can add the `-tile 4x1` option for a row montage and boost the margins a bit:

```
montage IMG*.jpg -geometry 960x960+20+20 -tile 4x1 eclipse-row.
↪jpg
```

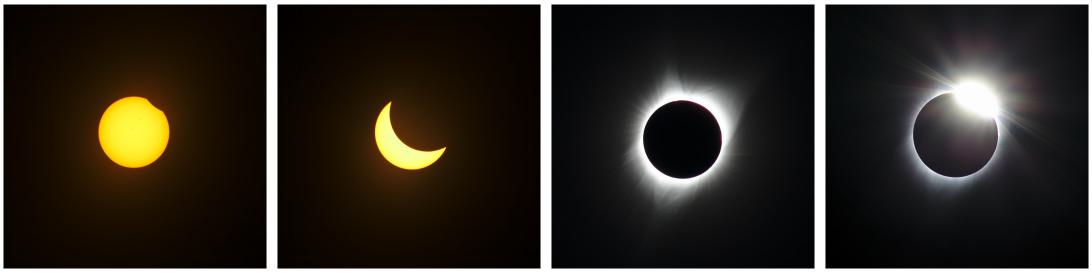


Figure3: The same eclipse photos but montaged as a row.

Exercise

Try it yourself with some images of your own but stack them vertically like a photo booth.

Sure you *could* do that kind of thing by clicking and dragging and perfecting this and that, but that'd take a really long time. This command line option is nearly instantaneous and can be repeated trivially on different images.

This utility is also excellent for adding Copyright © messages to images in bulk.

See hundreds of more examples at the Usage manual[3]. Another must-see resource on this topic is Fred's ImageMagick Scripts[4], which has hundreds of examples of kind of crazy and amazing things you can do with this tool. It's not the prettiest presentation, nor the most discoverable, but if you click through some of those you'll get a real taste of what this can do. The wheels of creation in your head are going to almost red-line.

[3] http://www.imagemagick.org/Usage/

[4] http://www.fmwconcepts.com/imagemagick/index.php

Computer graphics

Making graphics from scratch on a computer for fun and profit is another computer superpower. You can design your own logo, make signs for your neighbors, make diagrams for your research or web page, make your own comic strip, draw, trace, animate, and so on. It's a whole wonderful world. This is another one of those areas where full 4-year+ college programs exist to teach people how to do it well. But that doesn't mean you can't get started right away. I've used these tools in many technical publications as well as for public communications.

Vector graphics

Fundamentally, images can be represented on a computer in two ways:

Bitmaps
: Bitmaps are images made of an explicit list of which colored pixel is at each location in a grid. These are ideal for complex images and photographs. If you zoom too far in, they become pixelated and blurry. File types include `BMP`, `JPG`, `TIFF`, `PNG`, etc.

Vector images
: Vector images are made up of mathematical descriptions of shapes, such as "line from (x1,y1) to (x2,y2)". These require fewer bits for simple line-art graphics, and you can zoom in on them infinitely without losing any sharpness. Files types include `svg`, `pdf`, and others.

Note: Different bitmap formats have pros and cons too. `JPG` uses lossy compression and can give small file sizes, but you have to be careful as the quality can degrade significantly. `PNG` uses lossless compression, so quality remains high, but file sizes can be larger.

GIMP deals with bitmaps. Other programs, such as **Adobe Illustrator** and Inkscape[5] (open-source) deal with vector graphics. Inkscape is absolutely essential, so let's get it.

[5] https://inkscape.org/

Table3: Installing Inkscape

OS	Installation method
Windows	`choco install inkscape`
Linux	`apt install inkscape`
macOS	`brew install inkscape`

Here is a vector graphic made in Inkscape using the Rectangle tool, the Pen tool, and the Circle tool:

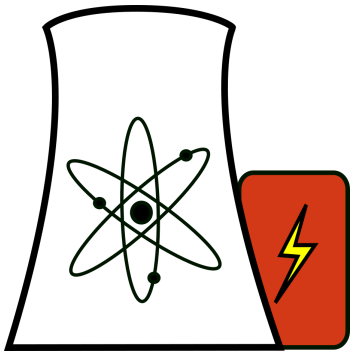


Figure4: An icon of a nuclear reactor

Let's make a cool sailboat logo with it.

Step 1: Making a sail

- After opening Inkscape, hover the mouse over the icons along the left panel until the tooltip tells you that you've reached the `Draw Bézier curves and straight lines` tool. Click it.
- On the canvas, click once somewhere in the middle to start (we're drawing a sail/triangle)
- While holding `Control` but no mouse buttons, move the mouse to the right a bit and click again. Holding the key constrains you to a perfectly horizontal line.

- Now move up and to the left until you're over the first point. Click once again. If you hold `Control` here it will constrain you to one of a few particular angles.
- Now go back to the starting point of this curve. It will turn into a red square (indicating that clicking will close and terminate the path). This time, click **and hold** the mouse button. While holding it, drag down and slightly to the right. This will make the previous line a bit curved, like a sail. Let go when you're happy with it.

It should look like this:

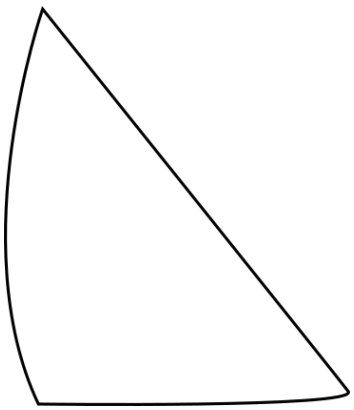


Figure5: My sail

Step 2: More detail

- Repeat the sail process but draw a boat structure under it (be creative)
- Click the sail (you have to hit one of the lines because the inside is currently empty). Now click one of the colors from the bottom (it scrolls) to color it how you like.
- Do the same with the boat.
- Click the Text tool on the left (big letter `A`) and draw a text box with it under the sailboat. Write "SAIL LIFE" or something else in it.
- Click the circle tool and click/drag to make a circle around the boat. Hold `Control` to constrain it to a perfect circle.

- If it's solid, click the `X` in the far left of the color bar to make it clear/empty.
- Find the `Fill and Stroke` dialog (on the right-hand side). It has three tabs in it: `Fill`, `Stroke paint`, and `Stroke style`. In the `Stroke style` tab, increase the `Width`. You'll see the circle getting larger.
- Choose a stroke color from the `Stroke paint` section to color it, (or `Shift-Click` a color in the color selector at the bottom of the screen).
- Save it as an SVG (native vector format).
- Export it as a `png` by going to the `Export PNG Image` option (on the right), choosing a file name in the `Export As` field and then pressing `Export`.

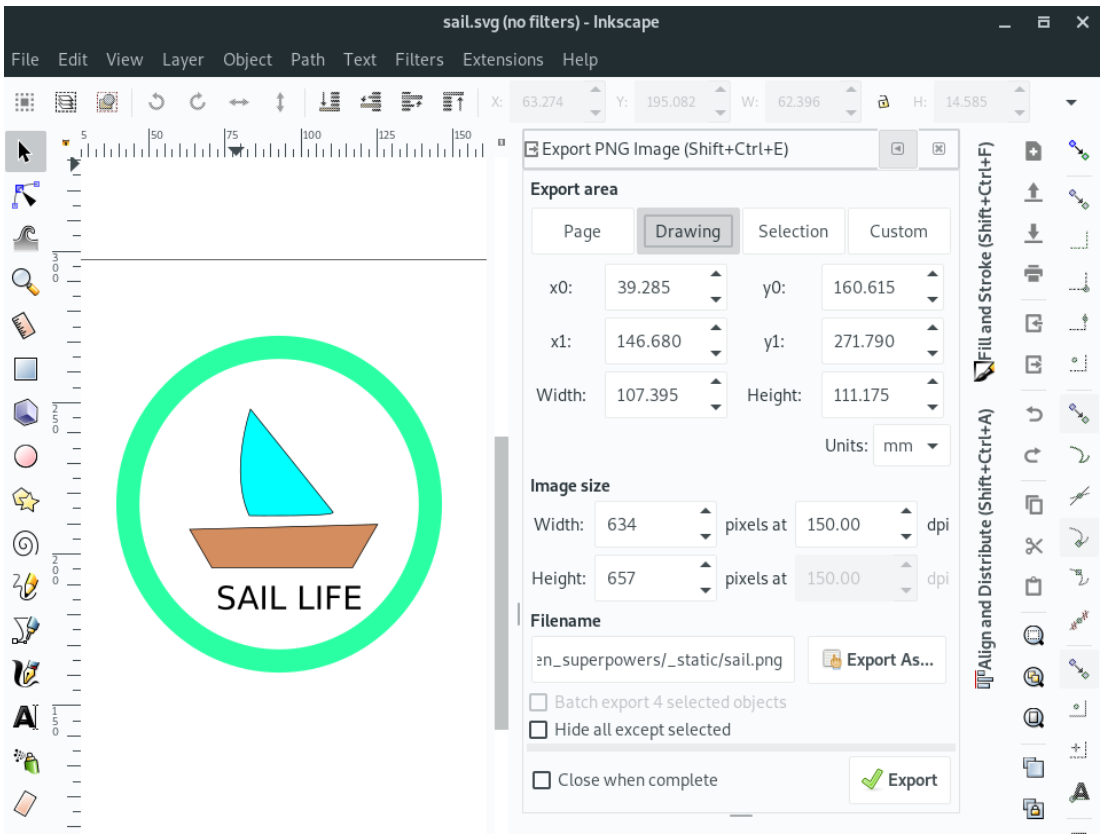


Figure6: My sailboat logo in Inkscape

This just *barely* scratches the surface of this tool and others like it, so explore around and, as usual, check out more serious tutorials for the real magic if this strikes your interest. I suggest looking into the bitmap tracing capability for your next superpower (it converts bitmap graphics into vector graphics).

Another serious open-source graphics tool out there is Krita[6], known mostly for concept art, illustrations, and comics. A lot of people use digital pen hardware with this program like a Wacom tablet.

For current or aspiring cartoonists, a truly professional tool used by Studio Ghibli and others called OpenToonz[7] was made open-source in 2016. It has been used in the production of Futurama, Anastasia, and Balto. An older version is currently available in `chocolatey` for Windows. There is a `snap` for Ubuntu Linux. Mac users can get it directly from the link above.

Color lifehack

One neat way to get a bunch of colors that look borderline good together is to choose them in the Hue-Saturation-Lightness (HSL) space instead of the more typical Red-Green-Blue (RGB) space. Pick one color anywhere in HSL and then choose all others by adjusting the Hue only (keeping saturation and lightness constant). These always end up looking like they belong together. All graphics programs allow you to choose colors in HSL these days. Of course, color wheels can also help you figure out complementary colors.

Here's what a few boxes made with this trick in Inkscape look like:

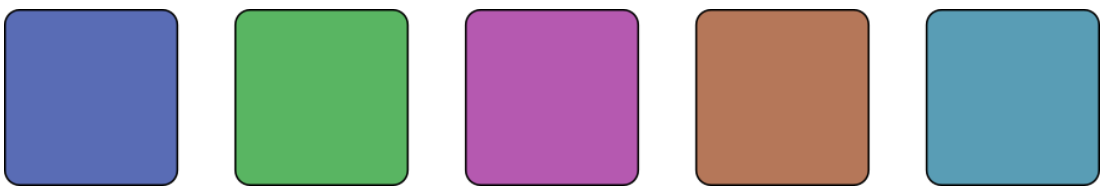


Figure7: Some nice colors that go together with the same S and L values. (Sorry B&W readers, this also looks better when viewed in color)

[6] https://krita.org/en/

[7] https://opentoonz.github.io/e/index.html

Websites like https://coolors.co/ can do this too.

3-D Modeling with Blender

Blender[8] is a 3-D modeling program. It is considered one of the best open-source programs ever made. With it, you can create objects for 3-D printing, animated characters for games, and even entire animated scenes with physics modeling such as a swinging ball-and-chain smashing into a pile of little boxes that go flying all over the place. You can get it the usual way:

Table4: Installing Blender

OS	Installation method
Windows	`choco install blender`
Linux	`apt install blender`
macOS	`brew install --cask blender`

It's a professional-grade program which could make one's entire career. With its graphical nature, the best way to get started is to watch the Blender Fundamentals[9] YouTube playlist, which at press time has 41 introductory video walkthroughs. Once you get used to the mouse and keyboard controls and see how to rapidly sculpt the progeny of your imagination into being, the world will be at your fingertips!

Once a 3-D object is built, the process of rendering it applies textures and lighting to turn it into a realistic-looking object. The technology of *ray-tracing* allows renders to take on a photorealistic quality, but requires serious computation. This method creates computer simulations of rays (kind of like virtual photons), transporting them through the scene. As they experience reflection, refraction, and blockage, they

[8] https://www.blender.org/

[9] https://www.youtube.com/playlist?list=PLa1F2ddGya_8V90Kd5eC5PeBjySbXWGK1

get recorded in a virtual camera looking at the scene[10]. In effect, this allows the creation of scenes that are indiscernible from real life. One of my favorite examples of this method was created way back in 2006 using another great tool called POV-Ray (not Blender). Here it is:

Figure8: A ray-traced render of some glasses created by Gilles Tran with POV-Ray 3.6. You can find the POV-Ray code for this, remove a glass, and then re-render it to convince yourself that it's totally computer-generated.

[10] Ray tracing actually transports virtual photons backwards from the camera to the light source. This ensures that the computer only spends time on light that will actually be seen rather than wasting computer cycles on light that goes way off into the distance.

Computer-Aided Design (CAD)

While Blender can be used to design physical objects, it's more specialized for making visual scenes intended to be viewed on a screen. The field of Computer-aided Design (CAD) covers programs that are specialized for making real objects like machine parts, structures, skyscrapers, sculptures, and so on. Many mechanical and structural engineers live and breathe by a commercial tool called SolidWorks[11]. It is extremely expensive, but when you're making a $30M part, it's worth it. This class of tool includes sophisticated physics and material modeling so that as you adjust the shape and composition of your object you can also subject it to various conditions (like wind, temperature gradients, earthquakes) and make sure it survives or otherwise performs as required. You can also make sure the part is fabricable with the process you plan to build it with. Welcome to mechanical and civil/structural engineering.

For around the home, studio, and small shop, you can get started learning concepts of CAD and designing real things with an open-source parametric CAD system called FreeCAD[12]. It's not as sophisticated as the commercial tools, but it can still do an incredible amount. Many CAD experts have given it mediocre-to-poor reviews over the years but a growing excitement surrounds its recent releases and potential. Try it out right now via your package manager:

Table5: Installing FreeCAD

OS	Installation method
Windows	`choco install freecad`
Linux	`apt install freecad`
macOS	`brew install --cask freecad`

As with most valuable and powerful skills, CAD is non-trivial to pick up, requiring dedication to achieve expertise.

Notably, a Building Information Modeling (BIM) Workbench plug-in[13] to

[11] https://www.solidworks.com/
[12] https://www.freecadweb.org/
[13] https://github.com/yorikvanhavre/BIM_Workbench

FreeCAD has been under very active development and is something to watch for designing buildings more seriously. It could come in handy for that addition to your home or that new shed you've been thinking about.

The wide availability of FreeCAD combined with its scripting and parametric capabilities (meaning the when you adjust the height of a part, all interrelated parts will update automatically to fit) make it an intriguing candidate for distributed open-source hardware and structural projects, should such a thing ever be conceived. I envision a glorious future full of this kind of collaboration, such as in open-source international power plant design.

OpenSCAD is another excellent and available CAD tool that could be interesting to compare and contrast with FreeCAD as you're starting out. Yet another is SALOME[14], which comes out of the French nuclear industry, where it's used to make CAD models as well as to interface between many different numerical physics solvers.

Autodesk's commercial Fusion 360 has a free license for hobbyists and other non-professionals. This offers a pathway to learn a powerful commercial tool at low initial cost. Philosophically, the availability of this likely and unfortunately pulled some focus away from the development of the open-source tools.

Note: Speaking of science and engineering, OpenFOAM[15] is another stunning multi-platform open-source system made to perform computational fluid dynamics (CFD) calculations. If you've ever seen those animations showing high-color super-detailed airflow over an airplane wing or car, that's what CFD does. We use CFD all the time in the nuclear industry.

---

[14] http://www.salome-platform.org/
[15] https://openfoam.org/

The digital darkroom

In the olden days, photography ("*writing with light*") had two artistic phases: going out into the field and making beautiful compositions on film and then deciding how to expose and crop them to a print in the darkroom. Digital photography has been revolutionary, largely eliminating the darkroom step for most of us.

It's worth noting, however, that we do have digital darkrooms now in each of our computers, and the opportunities for artistic expression within are staggering. While anyone can take beautiful snapshots with their phone, those who are interested can *go deeper* and unlock the really fun and rewarding world of digital darkrooming.

The RAW advantage

When you take a digital picture, light comes through a lens and excites a bunch of little electric light buckets in a grid called a digital sensor. Each light bucket records how bright either the reds, greens, or blues are in each point on the grid (different colored filters are overlayed on the grid in a pattern, often as a *Bayer filter*). The size of the grid determines the resolution of the image you get, and that is often proportional to sharpness. Thus, more is considered better. You've heard these measured in megapixels ("millions of picture elements"). A 25-megapixel image may record one 14-bit number per pixel, so the total size of the file would be:

$$25e6 \times 14 \text{ bits} = 43.75 \text{ megabytes}$$

That's a lot of megabytes (MB). But if you look at the size of a picture from your phone, it's closer to 10 MB. BUT HOW? The answer is that the bits from the sensor get sent to a microprocessor in the camera that *compresses* them mathematically to smaller sizes using the JPEG algorithm. Conceptually, if an image has 500 zeros in a row, the RAW file has 500 repeated numbers whereas a compressed file just has a few bits that say: "500 zeros in a row". The compression used is *lossy*, meaning information is lost when you compress a RAW image to a JPG. Worse still, JPEG images only have 8-bits of information per pixel instead of

the full 12- or 14-bits your camera collected. More bits means more possible brightness values for each individual pixel.

To make a long story short, professional photographers shoot in RAW mode to skip this compression because they will process the image in their computer's digital darkroom. When you work in RAW you can do incredible things, like pull details out of underexposed shadows, change the white balance, reconstruct overexposed highlights, denoise, and much more. If you didn't have the exposure settings quite right on an incredible shot, RAW is very forgiving and will help you get the print just right.

If you want to try out the digital darkroom, set your camera to save files in RAW format (note that you won't be able to store as many on one memory card, but I'm not even worried because you can still fit a lot of RAW files on a 128 GB card).

Here is an example. First, an image as taken right out of the camera:

And the same photo after a bit of time in the digital darkroom:

Look at all that detail that was effectively black in the first one that was brought out!

How to get started in the digital darkroom

Most professional photographers use the proprietary and expensive programs from Adobe called Lightroom and Photoshop as their digital darkroom. These are considered the best and do everything you'll need. Open-source alternatives can get you 98% of the way there, and you can start with them right now. In particular, darktable[16] is an absolutely phenomenal RAW digital darkroom program that was until recently only available in Linux. Here's how to get it:

[16] https://www.darktable.org/

Table6: Installing darktable

OS	Installation method
Windows	`choco install darktable`
Linux	`apt install darktable`
macOS	`brew install --cask darktable`

Darktable is a powerful program that takes some orientation to get started with. Fortunately, well-produced tutorial videos on YouTube will walk you through various development workflows step by step. Harry Durgin is my favorite. He talks you through exactly what he's doing and why, while crickets chip in the background at his Hawai'i-based studio. His darktable edit playlist[17] at press time has 42 wonderful videos. This is the best way to see what the various tools are capable of doing.

The basic workflow is roughly as follows:

- Import photos into the *light table* view and rank them 1-5 by pressing the 1-5 buttons on your keyboard.
- Choose your favorite and double-click it to enter the *dark room* view where you'll edit a single photo.
- **Try out a few basic plug-ins to get started, including:**
  - Denoise (profiled) — This uses data generated by other users who have your same camera model to remove digital noise, smoothing our your picture, especially in low-light areas.
  - Lens correction — If your lens is in the database, this applies corrections to geometric distortions, chromatic aberrations, and vignetting.
  - Crop and rotate — Adjusts the crop of the photo to best tell the desired story. Remember the rule of thirds and other composition guidelines. Note that you can right-click/drag in this view and trace along a line that you intend to be

[17] https://www.youtube.com/playlist?list=PLsks-zRRM1ZVN_g7P6ZAsYVqTltmXyBjl

either vertical or horizontal and it will rotate the photo to make it so.

  - Tone curve — Adjusts the exposure to get the lighting just right. You can make the curve steeper to get more contrast.
  - Shadows and highlights — Brings the brights down and the shadows up, evening out the exposure a bit.
  - Color correction — Changes the overall colors in the image.
  - Levels — Brings the black point up so the blacks are really black.

- Press the `L` key to go back to the *light table*. Now click `export selected` near the bottom right to write out the final image as a developed Instagram-ready JPG.
- If desired, open the image in GIMP for final modifications and touch-ups.
- Repeat for however many photos you like.

If you have a few plug-ins you always activate, you can have them automatically turn on.

Darktable comes with some extremely powerful features. One impressive one is *parametric masks*, which allow you to apply various operations only to certain sections of the photo as defined by ranges of brightness or color. You can combine these with drawn masks to fine-tune the look you want. So if you want to brighten or adjust the reds, but not the greens, you have full elaborate control of doing so. The artistic flexibility will give you many new ways to express yourself. You really have to watch some of Harry's tutorial videos to see this in action.

Making podcasts, music, and sound effects

There's this great track on Daft Punk's *Random Access Memories* where Italian electronic dance music pioneer, Giorgio, explains how he got started making music with synthesizers. At the end of his monologue there's a perfect pause and then these epic synth arpeggios hit and it's just amazing. In Giorgio's pioneering days, the synthesizers were all

analog circuits, but way back in the 1980s, digital synthesizers became widespread. Today your own personal computer can churn out synth beats and sounds like you wouldn't believe.

The excitement of sound

In about 1992 *Sound Cards* were an exciting accessory you could add to home computers. I vividly remember our family getting a Sound Blaster Pro card and installing it in our DOS-powered CompuAdd computer. At the time it seemed that the possibilities were endless. This hardware to synthesize sound came with demo programs including[18] :

- A talking parrot that would repeat anything you said in the microphone in a high pitch.
- A talking psychiatrist program (*Dr. SBAITSO*) who would psychoanalyze you in a robot voice.
- An organ you could play that would auto-accompany you as you pressed buttons.
- A sound recording program where you could pitch-shift, reverse, and otherwise edit things you said into the microphone.

This was really exciting, fun, and inspiring. Even though computers making sound seems a bit mundane at first glance, it still can be awesome. Tools and processes have just gotten better with time. Most people don't look at their laptop and consider the possibilities as a sound studio. Maybe you will from now on.

Recording and processing audio samples

Recording and digitizing real sounds through a microphone is fundamental to audio on computers. So let's try it out. Audacity[19] is a popular and widely-available audio program that can do this well. Here's how to get it:

[18] There's a great video demo of how exciting the SBPro card was and all the little included goodies at https://www.youtube.com/watch?v=g15J44xB2zU.

[19] https://www.audacityteam.org/

Table7: Installing audacity audio software

OS	Installation method
Windows	`choco install audacity`
Linux	`apt install audacity`
macOS	`brew install --cask audacity`

Fire it up and press the big red circle *Record* button and just speak your name. You should see some audio waveforms showing up. Then press stop. (If you just flatline then you may have to hunt for your *Sound Settings* and select your microphone from the input section; just use a web search if you have trouble). You will see something similar to this:

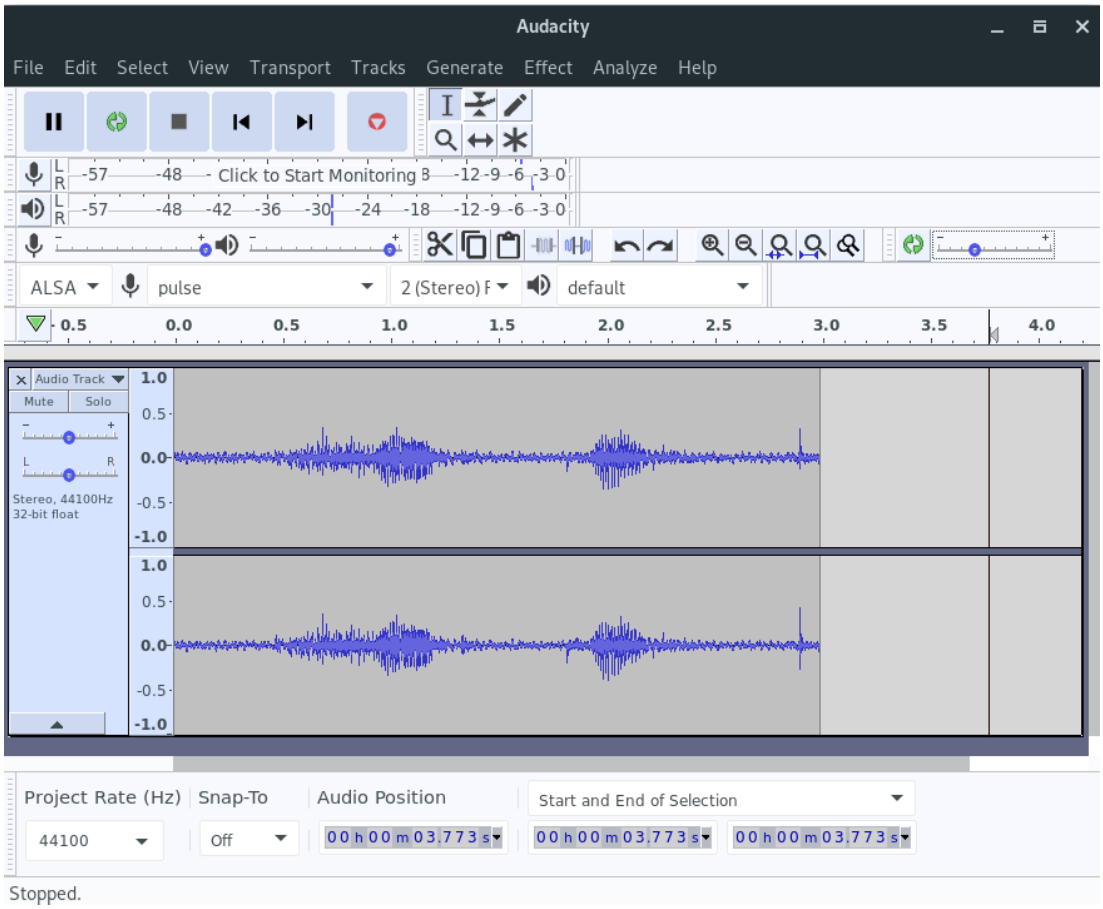


Figure9: A screenshot of Audacity with waveform after recording a sample.

Try playing it back (by pressing the green *Play* button). Now for the fun part. Select the whole waveform by clicking/dragging over it. In the menu, select `Effect → Reverse`. Play it back again and you'll hear your name reversed. Reverse it again to get back to normal.

Exercise

Now that you have heard your name spoken backwards, try recording yourself mimicking how it sounds backwards. Reverse it with the computer to see how you did. This is a decidedly fun party trick.

It's fun to try other effects as well, especially the `Change Pitch` one, creating laughs for days. As you dig into computer audio you'll also find ways to do live processing of what you say in the mic or play with an instrument. Beyond the hilarity of live pitch-shifting, your computer can be an entire post-recording or live effects system.

Recording for a podcast or radio show

If you want to make a podcast or radio show, Audacity is enough to get you started. You can chop up the recording, crop out parts you don't want, and add in a soundtrack. You can even remove background noise, which has an excellent effect on the listening quality. To do so you first have to profile the *room tone* by taking a recording of just it. Try this:

- Start a new recording in Audacity.
- For the first 10 seconds just sit there in silence, collecting ambient noise.
- Now say something useful (i.e. for your podcast) for at least another 10 seconds.
- In the waveform editor, select the portion that's just noise.
- Go to `Effect → Noise Reduction` and click `Get Noise Profile`.
- Now select the whole recording and go to `Effect → Noise Reduction` again.
- Click `Preview` to see how it will do and adjust the sliders if it doesn't yet meet your fancy.
- Click OK and behold your noise-free high-quality audio recording. You'll note that the waveform image shows much less noise in the gaps between your words.

The `Compressor` effect is also useful along these lines. It makes the volume more consistent so your listeners don't have to constantly futz with the volume level to hear you.

When you have the recording just right, choose `File → Export` and save the final result.

For complex projects with many tracks, a Digital Audio Workstation (DAW) such as **Ardour** becomes useful.

Synthesizers and music

There are lots of pieces involved in producing music on computers, so let's go through a few:

Table8: Components of a computerized music studio

Component	Purpose
Samples	Bring real-world sound into the computer by recording voices, acoustic instruments, sound effects, etc.
Synthesizer	Generates sound waves digitally based on keyboard (as in piano keyboard) input or programming
Drum machine	Special synthesizer that makes drum beats in loops
Digital Audio Workstation (DAW)	Records multiple tracks from synthesizers and/or microphones and allows you to move them around and modify them with various plug-ins to get them sounding just right. Often fully integrated with syths, drums, and plugins
VST Plug-ins	Add-ons that you can hook to your DAW to modify the sound of an instrument
VSTi Plug-ins	Synthesizer instruments that come packaged in the VST standard (making them easy to plug into any DAW that supports the VST standard)

Professional musicians invariably use professional audio software, and for good reason; it's quite polished. Ableton is only a few hundred dollars to get started with and, most musicians will tell you that it's hard to compete with.

That said, there's no harm in trying out the open-source offerings. Starting with drums, Hydrogen is multi-platform enough that we can all try it

out. It's not in the Windows or macOS package managers, so download it from here[20]. You can just start clicking where you want the various instruments to hit. Try matching my beat here (I changed the RES from 8 to 16 so I could put in 16th notes):

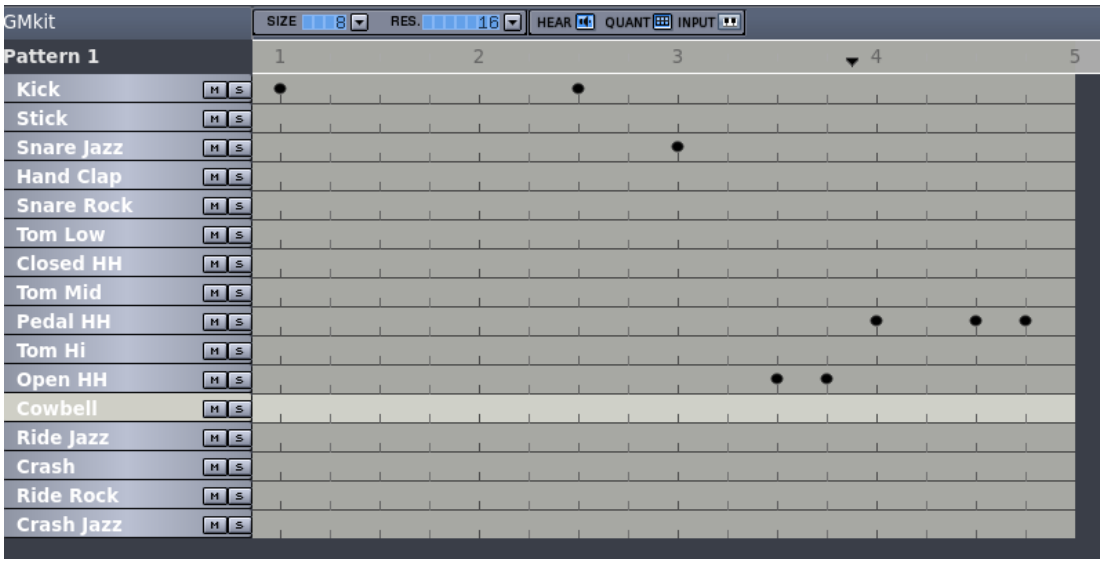


Figure10: Screenshot of a cool drum beat I whipped up in Hydrogen. Try it to hear it.

Press the little play/pause button to hear it. You can build dozens or hundreds of measures like this and use the grid above to schedule how they evolve with time. For instance, you can do 4 repeats of an introductory rhythm and then start activating more layers that build upon it. Fire up Audacity to record and plug in a microphone so you can make your first rap.

Exercise

Record your first rap using Hydrogen and Audacity.

We'll need to use a synth to put down a melody and bass line. Software synthesizers can be explored with a computer keyboard or even a mouse, but musicians will find this inadequate and want an older

[20] http://hydrogen-music.org/downloads/

kind of keyboard: that of a piano. MIDI keyboards are perfect for this. They have a piano keyboard but don't make any sound on their own; they're just a musical input device for your computer. They often include pressure-sensitive pads you can tap on for drums or other controls, a bunch of sliders, and dial controls. These can all be tied to whatever you want in the software synth. For instance, you can hook one of the sliders to something that changes some quality of the waveform and then control it during recording or a live show. There are specific MIDI ports on some sound cards, but many modern MIDI keyboards just use USB.

There are not many multi-platform open-source synthesizer systems/DAWs, but LMMS[21] has emerged as a high-quality, easy-to-get-started, and powerful system.

Table9: Installing LMMS

OS	Installation method
Windows	`choco install lmms`
Linux	`apt install lmms`
macOS	`brew install --cask lmms`

It's the kind of program that will have you playing the *Mission Impossible* theme in exciting glory within a half hour (it's in 5/4 time, FYI). The documentation is good, and this tutorial[22] will have you rocking out. Try it out, it will absolutely bring a smile to your face and get your creative, expressive wheels turning into overdrive.

[21] https://lmms.io

[22] https://docs.lmms.io/user-manual/4-production/4.6-your-first-song-with-lmms

Figure11: A screenshot from making the Mission Impossible theme in LMMS

DJing a party or show

DJing a good party or show requires a lot of work to set the mood and shift it through emotional highs and lows all while keeping a coherent, continuous story or pace going. To produce such a show requires either two turntables that you swap between, or a laptop. If you go the laptop route, you will find some excellent software tools that can get you up and MCing in no time.

We'll highlight the cross-platform open source Mixxx[23] program here. It's available in your package manager and is designed to help you get started even if you've never DJed before.

[23] https://www.mixxx.org

Table10: Installing Mixxx

OS	Installation method
Windows	`choco install mixxx`
Linux	`apt install mixxx`
macOS	`brew install --cask mixxx`

This program will scan your music library and present you with the list of songs. You can drag one to the player on the left and another to the player on the right. There's a left/right cross-fader control in the middle that will let you switch volume from one side to the other gradually. Here's the idea:

- Load a song in the left player and drag the cross-fader all the way to the left. Press the play button under the `CUE` icon. You should hear it start playing, and the waveform on top will start moving.
- Try clicking/dragging the moving waveform back and forth jerkily a little (trust me).
- Load up the next song in the right player and drag it to where you want it to start from. If you want, you can rig this up with dual soundcards so you can preview what you'll play next in your headphones while still playing over the main room system (this is why DJs always are holding up headphones to their ears and bobbing their heads).
- Adjust the tempo if needed, or press `SYNC` to match the same tempo as the currently playing song.
- Press the play button on the right to get it rolling (it's still silent due to the cross-fader).
- When the time is right, drag the cross-fader right until the song on the right is the only one heard.
- Repeat this process, but this time going from right to left.
- Do not stop until either the world ends or the sun comes up.

You can do this live or use something like Audacity to record your show in advance if you're putting out a mash-up album or whatever.

It looks like this:

Figure12: A screenshot of Mixxx playing a fairly eclectic show

Movies

The movie industry is large and lucrative. Software is absolutely key as the green screen has taken over. As such, professional software tools in all elements of movie production are in fierce competition. All the best tools are commercial, so if you want to go into professional movie production, you will likely need to learn tools like Adobe Premiere, DaVinci Resolve, Lightworks, Autodesk Maya, Flowbox, ZBrush, etc.

That said, a few accessible superpower tools do exist for people outside the movie production industry.

Format conversion, trimming, and time-lapses

The most universal command-line video tool is called ffmpeg[24]. It's useful for recording, converting, and streaming video (and audio).

[24] https://ffmpeg.org/

Table11: Installing ffmpeg video utility

OS	Installation method
Windows	`choco install ffmpeg`
Linux	`apt install ffmpeg`
macOS	`brew install ffmpeg`

`ffmpeg` can slice and dice videos. For example, if you have a long video (`input.mp4`) that you only want a two-minute clip from (between `12:21` and `14:21`, for example), you can use a command along the lines of:

```
ffmpeg -ss 00:12:21 -t 00:14:21 -i input.mp4 -acodec copy \
    -vcodec copy -async 1 section.mp4
```

This will create a new video called `section.mp4`.

Note: If the video doesn't end up cutting right, then there probably is no *keyframe* at the time you cut it. To get around this you have to re-encode the video by omitting the `acodec` and `vcodec` option, which will take some processing time.

To convert from one video format to another, you specify the video codec (`-c:v`) and/or audio codec (`-c:a`). The commands are like:

```
ffmpeg -i movie.avi -c:v libx264 -preset ultrafast movie.mp4
```

`ffmpeg` can make time-lapse videos from image sequences. This is useful in research environments (e.g. stitching together stills from a simulation) or in hobby or art spaces just the same (e.g. combining a year's worth of security camera stills into a year-in-review project). The input options to `ffmpeg` are very versatile, but also complex and frequently changing. Thus, it's best to use web searches to find the best command for what you want to do. For instance, if you search something like `ffmpeg stitch photos into time-lapse` you may find something like:

```
ffmpeg -framerate 9 -pattern_type glob -i 'ice-move/shed-*.jpg' \
       -vcodec libx264 -crf 15 -filter "minterpolate='fps=30':
```

(continues on next page)

(continued from previous page)

```
↪'mi_mode=blend'"\
      -pix_fmt yuv420p ice-move.mp4
```

This will take all files called `shed-[anything].jpg` in the `ice-move` folder and combine them into a new time-lapse movie called `ice-move.mp4`. This actually does interpolation between frames, giving a really smooth-looking time-lapse.

You can even make high-quality animated GIFs for posting on the internet. With `ffmpeg`, web searches and StackOverflow are your friends. All the options are described in elaborate detail at https://ffmpeg.org/ffmpeg.html.

Video editing

When you're going to produce a movie from video you shot around town for whatever reason, you need graphical video editing software to arrange the clips how you want them, to add smooth transitions (fade, blur, etc.), to add rolling credits at the end, and to insert a soundtrack.

There really is no universal standard multi-platform video editing software. Each OS has its favorites. OpenShot[25] is one open source option available on all platforms that is well-worth trying out. It's user-friendly, and has most of the features you'll want for video editing. It's a bit slow, and I got frustrated with how frequently it crashed a few years back, but they've been working hard on stability since then. There's absolutely a strong potential.

Table12: Installing OpenShot video editor

OS	Installation method
Windows	`choco install openshot`
Linux	`apt install openshot`
macOS	Go to https://www.openshot.org/download/ and run the installer

[25] https://www.openshot.org/

If you're on Linux, KdenLive[26] is also worth a look. It seems significantly faster. Blender actually has full-featured video editing capabilities as well, and is quite stable. The user interface may take extra work and training to get used to, but it may actually be the most stable multi-platform experience. Tutorial playlists like Mikeycal Meyers'[27] can get you operational fairly quickly.

Other than that, Windows Movie Maker on Windows and iMovie on Mac are very polished and usable. The real point is to get that phone camera out and try to make little movie productions. As much as we have video cameras in our pockets these days, it's a bit uncommon to see someone put a film together. You have all you need at your fingertips already, so making a movie could be a fun activity for some rainy day. At least personally, I really cherish the old home videos I have from my childhood a lot more than the photos.

Making games

In the 1990s I was hoping to learn how to make video games. I got a book about doing it in the `C` programming language and it started with putting the graphics card into VGA mode `13h` and writing pixels to the double-buffer one at a time. In other words, it was a bit inaccessible for me at the time, and I never got very far.

With the advent of widely available *game engines*, things have changed dramatically. These are software systems that make games. They come with so many features that you can make simple games without writing any code whatsoever. Of course, for advanced games and logic, code will eventually be needed.

Unity3D[28] is such a game engine that makes games for phones, computers, and game consoles. A free beginner version lets you try it out (though you do have to pay once your games start making over $100k on app stores). The free version is more than sufficient to start going

[26] https://kdenlive.org/
[27] https://www.youtube.com/playlist?list=PLjyuVPBuorqIhlqZtoIvnAVQ3x18sNev4
[28] https://unity3d.com

through some of the training material. The tutorials[29] will walk you through the basics and whet your appetite so you can see if you want to keep going with it or not.

This high-level game programming is a fun way to start. That said, you'll still gain much in going under the hood to learn what's really going on. The world will always need low-level developers to keep the lights on for the rest of us. For that, you'll need the chapter on *Programming*.

From cropping photos to DJing parties, I hope you've enjoyed this taste of what kind of creative things you can do on your computer. We'll continue along these lines in the next chapter as we learn about getting the written word out there.

[29] https://unity3d.com/learn/tutorials

6

Publishing

I once had the pleasure of visiting the Plantin-Moretus Museum in Antwerp, Belgium. It's a UNESCO World Heritage site that used to be a 16th-century printing facility. Before visiting, I didn't realize how difficult it was to publish the written word and sheet music. A punchcutter started by carving each letter on the end of a hard steel punch, often after softening it in a flame. These punches were then pounded into softer metal like copper to create an imprint. The imprint was then placed in a hand mould, where still softer metal was poured in, cooled, and then taken out. This seemingly round-about process was designed to mass-produce many exact copies of what the punchcutter had laboriously carved.

These pieces of type were organized in cases, with the case of larger letters placed on the compositor's upper rack and the case of smaller letters placed on the lower rack (hence "upper case" and "lower case" letters). Once numerous copies of each letter in different sizes and variants were cast, they'd be painstakingly arranged by compositors and placed into a string-bound type galley, proof-read, placed in the press, inked, and pressed onto many copies of paper. Then there was the binding and transportation. Woo-eee!

Advances in printing did occur between Gutenberg and the digital age, but the advent of computers and the internet was by far the most revolutionary change the industry has experienced.

Publishing online

The internet is an amazing, almost magical publishing venue. Anyone with a phone can type something in and it will be available instantly to people all around the world. The vastness introduces dual problems. Authors need to ensure people around the world want to and can find their material. Meanwhile, readers need to filter through an increasing amount of noise to find what they are interested in.

In the following sections, we will explore a few options for publishing online.

Sign up for a web publishing service

Aside from posting on Facebook, the easiest and fastest way to get your words online is to sign up with a service provider who offers publishing. Wordpress[1] is a famous example of such a service that is used to make some large fraction of the world's web pages. Medium[2] is another. Wix is another. These are polished services that will guide you through everything you need to get your words out there. You will be able to type your content in a nice web-based form that has all the formatting options you're used to from a word processor. If you get a free account, they may show advertisements to people who visit your page.

Getting your own web address

If you want the `www.yourname.com` domain name, you have to pay someone a modest fee (typically around $10/year) to register the name. The Domain Name System (*DNS*) behind this works much like a phonebook for the internet. I have used Namecheap[3] to buy my domain names for decades and have never had a complaint. Countless others exist as well. When you go to these services (called *domain registrars*), they

[1] https://wordpress.com
[2] https://medium.com/
[3] https://www.namecheap.com/

will present you with a search form. You type in the name you are seeking and it will tell you whether it's available, and how much it will cost. Many names are taken, but if you do find one that's really catchy, they may charge you a premium. If `yourname.com` is not available, you may be able to get `yourname.io` or even `yourname.horse`, among many other *top-level domains*, as these suffixes are called.

Once you find a name and buy it, you configure it by pointing the name to your server, or by telling anyone who goes to your name to forward the lookup request to some other name server. You can generally point your own domain name to any of the polished web services mentioned above. On Wordpress, for example, you will point your domain's lookup servers to `ns1.wordpress.com` and Wordpress will make sure to forward people on to your page. Behind the scenes, something like this will happen:

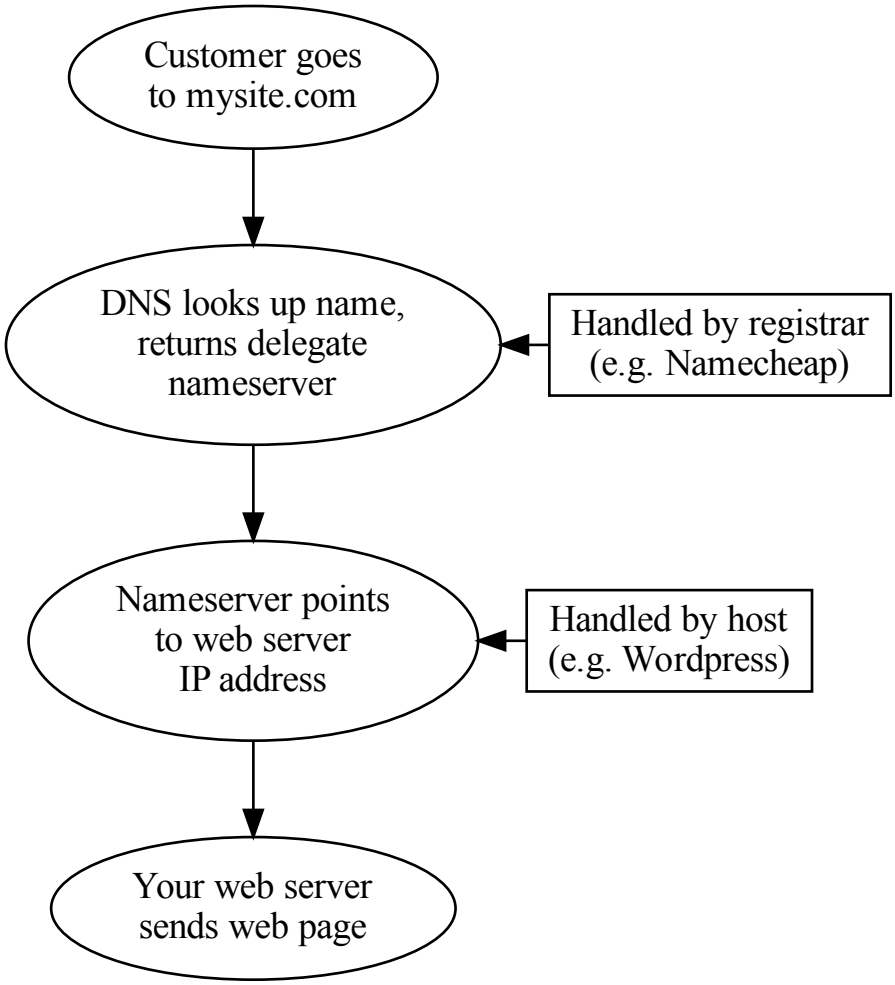


Figure1: Behind the scenes with domain names

Make your own web page

If you want more control and flexibility, and are willing to learn more to get it, you can always make your own web page. Do this if you either want to make a hobby of web development, or you are interested in going professional someday in tech. You need two things:

- An internet-accessible place to put your content (a web server)
- The ability to make web-browsable content that displays in a web browser

Web hosting companies will allow you to put files and content management systems on their servers for a modest fee. PC Magazine ranked HostGator, Dreamhost, and Hostwinds highest in 2019. Once you have your server space, you point your domain name to it. Some people even use their home computers as servers and leave them on all the time for this, but many internet service providers frown upon this.

Now you have to put content up. You may create the content on your local computer and then upload it to the web server using a secure file transfer program such as SSH or WinSCP.

Writing a web page from scratch

Web browsers read a markup language called HyperText Markup Language (HTML). The most rudimentary way to make a web page is to write HTML by hand in a text editor. It looks like this:

```
    <html>
    <head>
        <title>My neat web page</title>
    </head>
    <body bgcolor="green">
        <h1>Welcome to my web page</h1>
        <hr />
        <p>I am a business consultant and you can hire me.␣
↪Please contact me at <a
        href="mailto:myemail@myserver.tld">my email address</a>.
    </body>
    </html>
```

You can save this to a file called `bad_webpage.html` and load it in your browser (by double-clicking the file, usually). It looks like this:

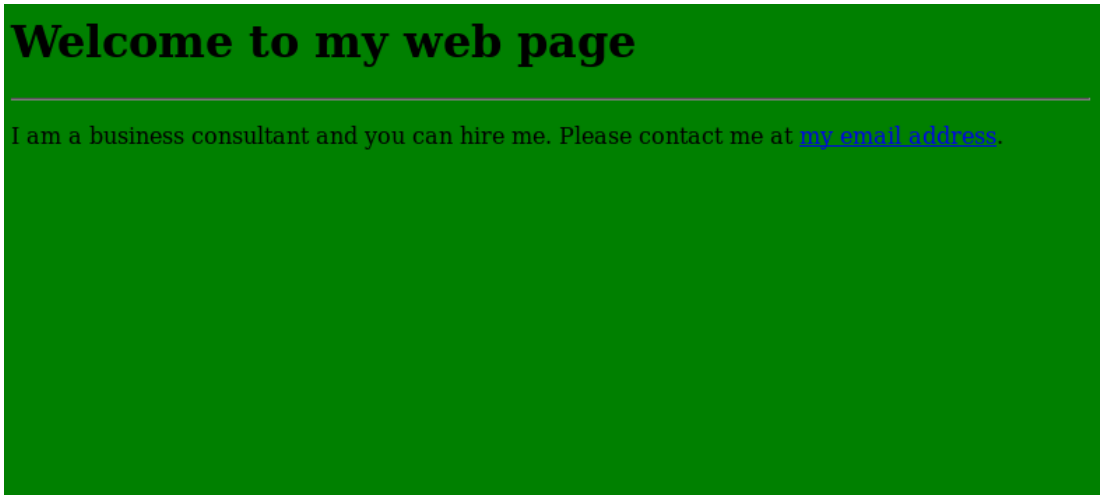


Figure2: A terrible yet nostalgic web page

Exercise

Write a similar web page in a text editor and open it up in your web browser.

If you upload this to your web server, you will have your own web page, written from scratch by you.

In reality, very few people do it this way anymore. It's just too hard to update hundreds of pages that may get broken links. Or if you want to change the look and feel of the site, you can't be expected to change each page individually. There is, however, value in knowing the basics of HTML because the syntax and features are often used in other tools.

Using a Content Management System

Wordpress is an example of a *content management system* (*CMS*) that accommodates the management of web pages. Besides signing up with Wordpress.com as discussed above, you can also install the open-source Wordpress application on your own server and publish through it. Literally hundreds of other self-hosted CMSs like Joomla offer a balance between control and ease. Many web hosts provide single-click installations of a variety of CMSs so you can try a few out as a learning process, and then choose one to deploy for your next business, fan-club, or project.

Static site generators

As CMSs got more complicated and heavier-weight, another approach emerged: the path of the static site generator. Where Wordpress requires a resource-intensive and sometimes-slow database system, static site generators target the convenience and power of a CMS without needing a central database.

These make even more sense today due to the prevalence of cloud computing platforms. If your website gets featured on a popular news aggregator, the traffic spike can bring your little server to a halt. With static sites, you can just ramp up and down how many duplicate servers you're using willy-nilly and toss in a load-balancer that distributes readers to each server equally, and you'll handle the traffic just fine. One excellent example of a static site generator is called Jekyll[4].

These systems are non-trivial to set up and get going with, but they are quite nice. We're using Jekyll for https://whatisnuclear.com, for example.

[4] https://jekyllrb.com/

Other Topics

This just scratches the surface of web publishing. There's a whole world of front-end design and development covering things like JavaScript and Cascading Style Sheets (CSS) that allow you to tweak and polish every aspect of what the web user sees. Entire back end systems make web pages interactive. You'll read more about that in *Programming*.

Publishing PDFs

Web pages aren't for every kind of document. Reports, articles, and books are often better suited for documents that come across the internet as PDFs and look great when printed. Here we'll learn how to efficiently build beautiful PDF documents.

This is particularly interesting to all you scientists out there. Here we'll learn about some systems that make it easy to get journal manuscripts into the right format, with all the cross-references just right and all the citations in the right format. It's not your job to get a bibliography formatted correctly; that's the computer's job.

The systems we'll learn about also excel at generating stunning, elegant, and unique resumes.

Introduction to LaTeX

In this section, we'll learn about LaTeX, a publishing system infamous for having ardent followers while being considered too annoying to deal with by most people. We'll show what it can do, explain why some people love it, show you some recent advances in potential workflows, and give you enough information to make informed decisions about using it or not.

Note: The TeX in LaTeX is related to a Greek root that means both art and technology. It's pronounced roughly like "teck". The La part is

pronounced “lah” or “lay”, you get to choose which one you prefer.

LaTeX and related technologies essentially *build* documents from a descriptive pure-text language, similar in concept to the raw HTML file seen in *Writing a web page from scratch*. This enables some efficiencies, including:

- Table, Figure, Reference, and Section numbering is totally automatic. Cross-references are updated for you, so you never have to re-number anything if you add a new table.
- The format of the bibliography is auto-generated. Various publishers give out bibliography style definition files so you can seamlessly switch between them.
- Text-based source can be version-controlled in detail for the ability to rewind in time and collaborate with others (see *Using git*).
- Equations are typeset beautifully. LaTeX is particularly popular in math journals.
- The layout of the document and letter spacing is handled for you.
- Documents can be assembled from multiple places, so the cover page manager can define the cover page and it will automatically be used in any documents that pull it in.
- Documents or reports can be automatically generated or updated by a program (more useful for updating engineering reports based on simulations than for traditional publishing).

LaTeX has notable downsides:

- The syntax of LaTeX source has a learning curve that most people are not willing to climb given how easy well-known alternatives are (though recent advances in lightweight markup languages and document conversion with `pandoc` are alleviating this, as you’ll soon see).
- It can be difficult to get a document to render exactly to your liking. Inserting images can be a pain compared to point-and-click solutions (fortunately, point-and-click LaTeX editors exist these days).
- Most people don’t know how to install LaTeX or run it (we will solve this problem momentarily).

How to run LaTeX

Let's go through a basic LaTeX workflow.

Table1: Installing LaTeX (warning: it's huge and this may take a while)

OS	Installation method
Windows	`choco install miktex`
Linux	`apt install texlive-latex-extra`
macOS	`brew install --cask mactex`

Make a text file called `mypub.tex` and put something along these lines in it:

```
\documentclass{article}

\title{On the writing of \LaTeX{} and building of a PDF}
\author{Your name here}
\date{July 2035}

\begin{document}

\maketitle
\section{Writing the source file}
\label{sec:writing}
Writing the source may look odd at first, but once you start, it
 ↪'s not so
bad. Also, there are WYSIWYG\footnote{What you see is what you␣
 ↪get} editors that make
it easier.

One thing that's really nice about \LaTeX{} is math. Here is an␣
 ↪equation:

\begin{displaymath}
    R = \sum_{g=0}^{G-1} N_i \phi_g \sigma_g
\end{displaymath}
```

(continues on next page)

(continued from previous page)

```
\section{Another section}
As you saw in Section~\ref{sec:writing}, you can make equations\
↪cite{brown2018endf}.

\bibliography{myrefs}
\bibliographystyle{ieeetr}

\end{document}
```

Also, make a separate text file for the references called `myrefs.bib`. This file can be built with a reference manager (like Zotero[5] with the Better BibTeX plugin[6] or JabRef[7]) but for starters let's fill it manually. You can find citations in the proper format through Google Scholar[8]. Find the reference you want, click the little " icon, and then click the `BibTeX` link at the bottom. It will present you with text that you can copy and paste directly into `myrefs.bib`. Choose your own journal citation to use, or if you want to type one in yourself, it looks like this:

```
@article{brown2018endf,
  title={ENDF/B-VIII. 0: The 8 th Major Release of the Nuclear␣
↪Reaction Data Library with CIELO-project Cross Sections, New␣
↪Standards and Thermal Scattering Data},
  author={Brown, David A and Chadwick, MB and Capote, R and␣
↪Kahler, AC and Trkov, A and Herman, MW and Sonzogni, AA and␣
↪Danon, Y and Carlson, AD and Dunn, M and others},
  journal={Nuclear Data Sheets},
  volume={148},
  pages={1--142},
  year={2018},
  publisher={Elsevier}
}
```

Note how that article name (`brown2018endf`) is the thing we referred to in the citation in section 2 of our source file.

Let's build the document. From the command line in the folder where

[5] https://www.zotero.org/
[6] https://github.com/retorquere/zotero-better-bibtex
[7] https://www.jabref.org/
[8] https://scholar.google.com

we made our two files, run:

```
latexmk -pdf mypub
```

Now you should see `mypub.pdf` in that directory, which looks like this:

On the writing of LaTeX and building of a PDF

Your name here

July 2035

1 Writing the source file

Writing the source may look odd at first, but once you start, it's not so bad. Also, there are WYSIWYG[1] editors that make it easier.

One thing that's really nice about LaTeX is math. Here is an equation:

$$R = \sum_{g=0}^{G-1} N_i \phi_g \sigma_g$$

2 Another section

As you saw in Section 1, you can make equations[1].

References

[1] D. A. Brown, M. Chadwick, R. Capote, A. Kahler, A. Trkov, M. Herman, A. Sonzogni, Y. Danon, A. Carlson, M. Dunn, *et al.*, "Endf/b-viii. 0: The 8 th major release of the nuclear reaction data library with cielo-project cross sections, new standards and thermal scattering data," *Nuclear Data Sheets*, vol. 148, pp. 1–142, 2018.

[1] What you see is what you get

Figure 3: A beautifully-rendered document with automatic references and citations.

Official LaTeX templates

Most major journal publishers give out LaTeX style files to authors. These put the formatting completely in their control; you just write your content. For example, Science Magazine offers their template here[9]. Universities also typically have these for dissertations.

Using LaTeX from a GUI

Modern GUIs for editing LaTeX files are quite good. They let you leverage most of the ease of a word processor with the added power of LaTeX. They'll automatically insert section labels and offer dialogs to help you build tables, insert figures, provide spell-checking, give integrated PDF previews, and so on. TeXstudio[10] is the most popular GUI TeX editor. It is available in your package manager if you'd like to try it out.

Table2: Trying out TeXstudio

OS	Installation method
Windows	`choco install texstudio`
Linux	`apt install texstudio`
macOS	`brew install --cask texstudio`

Another option is to use LyX[11], which actually keeps the TeX code behind the scenes for the most part. It is the closest thing to a word processor that uses LaTeX. I wrote my dissertation with LyX.

For web-based collaboration, ShareLaTeX[12] is the cutting edge.

See also: Wikipedia's comparison of TeX editors[13].

[9] http://www.sciencemag.org/authors/preparing-manuscripts-using-latex
[10] https://www.texstudio.org/
[11] https://www.lyx.org/
[12] https://www.sharelatex.com/
[13] https://en.wikipedia.org/wiki/Comparison_of_TeX_editors

Easier workflows with lightweight markup languages

Even though it's very powerful, I cannot deny that writing in pure LaTeX can be tedious. Nor is it overly versatile; it directly produces PDF documents but not web pages or other commonly-desired formats from a single source. The concept of *lightweight markup languages* (LMLs), also known as *humane* markup languages, can help get the powers of LaTeX without as much pain.

Two popular LMLs are Markdown and reStructuredText (RST). Markdown is particularly simple, while RST is slightly more complicated (but also significantly more featureful). It's becoming more common for people to write technical documentation, reports, web pages (often for static site generators), and even books in LMLs.

RST is usually associated with a tool called Sphinx[14] which was originally created to help write documentation for the Python programming language. It includes facilities for cross-references, indices, code snippets, glossaries, and citations, plus it can output to HTML, LaTeX, ePub, and more. As it turns out, this system is useful for more than just technical documentation and finds wide usage in other things. Indeed, this book itself was written in RST and processed through Sphinx to ePub, PDF, and HTML formats!

Pandoc[15] is an outstanding tool that simply converts various document formats between one and the other. It's a good way to go from an LML to HTML for the web, LaTeX for print or PDFs, ePub for e-books, or even MS Word (and vice versa). You can try making a diary in Markdown and render it to a beautiful PDF with pandoc on demand. It does an incredible job and continues to just surprise me by working so well. Let's convert that LaTeX sample from above to a Word docx file.

[14] http://www.sphinx-doc.org/en/master/index.html
[15] https://pandoc.org/

Table3: Installing pandoc document converter

OS	Installation method
Windows	`choco install pandoc`
Linux	`apt install pandoc`
macOS	`brew install pandoc`

Then run the following in the command line (in the same folder as `mypub.tex` from above):

```
pandoc -s mypub.tex -t docx -o mypub.docx
```

The options are `-s` for source, `-t` for `to`, and `-o` for output file name. Open this up in Word or LibreOffice and voila! Baffling, isn't it?

Best of all, these kinds of workflows can be revision-controlled and collaborated upon using `git`, which you'll read about soon.

It must be stated that Microsoft Word has improved mightily through the years in terms of section numbering, cross-referencing, and citations. It integrates with many powerful reference management tools like EndNote and really isn't all that bad. For the data-centric world of the near future, where information presentation must come from diverse sources and come together elegantly and collaboratively, the tools we've discussed here unlock many superpowers.

Publishing eBooks (and books)

The advances in publishing since the days of punchcutting have brought us to a situation where it's quite straightforward to publish an entire book digitally and in paperback. Even a few years ago, it could require significant investments to publish a book, whereas now digital delivery costs nearly nothing and print-on-demand services will print only as many books as are ordered. You still have to invest in writing, obviously, and possibly cover design, advertising, editing, and so on.

Self-publishing a book rather than getting a publishing company to do it is an exciting venture. You get to maintain full creative and financial

control, but you're on the hook for making sure everything is good, and for advertising/promoting the book. If nothing else, it's fairly low-risk if you don't invest *too* much on artists, editors, advertisements, or your book tour. The exciting point is that, with a computer in hand, you have everything you need to self-publish your first book.

E-readers use a variety of formats, many of which are locked down with special encryption called Digital Rights Management (DRM) that prevents people from copying books left and right. Before getting converted to a locked-down file, manuscripts often pass through the `ePub` format, which is similar to HTML and has the benefit over the PDF format of being *reflowable* (so it works with different font and screen sizes).

Many authors write manuscripts for eBooks in a word processor and use services to convert them into the `ePub` and/or the DRM-friendly formats. For example, Kindle Direct Publishing[16] offers tools that can convert `.docx` files from MS Word into the format necessary for publishing Kindle eBooks and printed-on-demand paperbacks. They've automated much of the work, and provide great tutorials and videos to walk people through all steps. They even have a Cover Creator tool, but I bet you can do a great job on your own with Inkscape for your first book. If you plan to sell a lot of copies, engaging a professional cover artist will run between $300-$1200. Besides KDP, Lulu.com[17] offers similar services in case you'd like to shop around.

Barring those services, the `LaTeX` system discussed previously is perfectly capable of creating ready-for-print physical books, but it won't be very useful for getting something onto an e-reader (PDFs generally look bad on e-readers specifically because they are not reflowable). Perhaps surprisingly, the Sphinx system and the ReStructuredText LML are a fairly powerful combination for self-publishing. It's advanced enough to let you add metadata that will generate the entire book including an index, a table of contents, cross-references, footnotes, images, hyperlinks, and a glossary in `ePub` (for e-readers), `PDF` (for print), and `HTML` (for the web). Once you have an `ePub` file, Amazon provides something called `KindleGen` on Windows, macOS, and Linux that will convert it to a `mobi` format appropriate for uploading to Amazon/KDP.

For example, to have an index point to the proper page numbers for

[16] https://kdp.amazon.com/en_US
[17] http://www.lulu.com

some keywords, you simply list them before the paragraph that they appear in, like this:

```
.. index::
    single: Horses

The joy of horses
-----------------
Horses are fine and majestic animals.
They have four legs and love oats.
```

Very rich indices can be generated by populating a manuscript with this kind of metadata.

Getting the formatting exactly right is non-trivial. Neither is grammar checking (though check out `LanguageTool`). Thus, I do not recommend this approach to anyone who is not keenly interested in "going it alone" and having full control with an open-source stack. It's certainly possible.

This concludes the chapter on publishing. You've learned many new and interesting ways to get the written word out to the public. I hope the opportunity arises for you to exercise these tools.

7 Programming

The *real* power of computers, of course, comes from programming them. Knowing how to program even a little bit unlocks untold opportunities to make the computer work better for you in the most personal of ways. Besides, programming skills also unlock very well-paying jobs. The advent of Machine Learning-assisted programming is clearly going to make an impact on the art of programming in general, but in order to be ML/AI-assisted, you'll want to have a solid foundation in the basics regardless. In this chapter, you'll try a version control system called `git`, write a few simple Python programs, find resources on machine learning, and be introduced to a web app framework called Django.

Tracking changes of anything

Even if you never program, you will benefit from knowing about version control systems (VCS). A VCS allows you to create a rewindable timeline of anything you're working on. Think of it like the "Track Changes" feature of a word processor, but more powerful and for more than just documents.

VCSs are generally associated with programmers because they *really* benefit from having a rewindable timeline of their software. This comes in handy often, such as when a program that used to work no longer works. Programmers look at the timeline of the source code changes,

rewind it, and play it back, all the while running their program to find out where the bug was introduced.

Some people keep their résumé in a VCS. They want to see it progress, and also "branch" it to make slightly different versions for various job applications.

A VCS is also wonderful for collaboration. You know how people e-mail around versions of documents like `Proposal_FINAL_final_nt_jrc2_draft.docx`? A VCS can make that kind of operation much better. Everyone can work on their own copy of a document and then these can be merged together. One hiccup with this approach is that VCS works best with pure text files so if you want to go this route, consider a lightweight markup language like ReST or Markdown (see *Publishing*).

Some notable VCSs include git[1], mercurial[2], subversion[3], and Perforce[4]. Git is a fairly universal favorite. It's the eponym of the famous GitHub, which Microsoft acquired in 2018 for several billion dollars.

Using git

Git has a slight reputation of being hard to learn. Fear not, it's not that bad. Git is made for the command line, but can be used graphically if you prefer. For the most common uses, you only need to know a few commands. Here they are:

[1] https://git-scm.com/
[2] https://www.mercurial-scm.org/
[3] https://subversion.apache.org/
[4] https://www.perforce.com/

Table1: Some key git commands

Command	Purpose
`git init`	Initialize a new `git` repository in the current folder
`git clone`	Make a local copy of a remote repository (e.g. from GitHub or your team)
`git add`	Mark one or more files to be part of a new commit that is in process
`git commit`	Take all marked changes and package them as a discrete “change” to be remembered forever.
`git pull`	Update the local repository with any changes someone else made on the remote server.
`git push`	Send all our local commits to the remote server (for collaboration)
`git log`	View all changes you or anyone else has made
`git checkout`	Check out a previous commit or separate branch (for going back in time or trying alternate paths)

The best way to learn is to try it out and experiment.

Table2: Installing git

OS	Command
Windows	`choco install git`
Linux	`apt install git`
macOS	`brew install git`

In a new empty folder, run `git init` to start a new repository. Create a new text file called `file1.txt` in the folder using your text editor and write a few lines in of your choosing. For example:

```
Uncle Dave: Married to Rose; likes putt-putt
Aunt Malia: Went to University of Michigan
Cousin Jingjing: From Hawai'i
```

(continues on next page)

(continued from previous page)

```
Third-cousin Pete: Enjoys cooking
```

Stage it with:

```
git add file1.txt
```

Commit it with:

```
git commit -m "Initial commit of relatives"
```

Make some changes to the file. Delete a line or add a new one. View the differences from the last commit by running:

```
git diff
```

Stage and commit the new changes (by repeating the steps above). Now let's say you want to go back to the initial version of the file by rewinding it to the first commit. First, you need to find out what the commit's "address" is, which is a long string of letters and numbers. Run:

```
git log
```

to get a list of commits. Copy the first 8 or so letters/numbers from the first commit's address and fill them in for your version of the following command (your address will be different):

```
git checkout a3e2b54
```

Now look at the file and you'll see that it's at its original commit. Go back to the latest commit with:

```
git checkout master
```

There, you've used most of the primary `git` operations now. Undoubtedly some of these commands will feel a bit mysterious still. There's a whole free book on it on the official git web page called Pro Git[5] and countless tutorials on the internet. We'll do a similar example but on a collaborative repository in the next section.

[5] https://git-scm.com/book/en/v2

Besides working on text files, there is also an extension called Git Large File Storage[6] (LFS) that can be used to store big binary assets efficiently in a git repository (graphics design files, science files, etc.). Git-annex[7], as alluded to in *Backups*, can track information about large sets of files without storing the files themselves in the git repository. This was originally built for managing metadata and synchronization of photo and music libraries. It's finding uses in advanced areas like big-data research as well.

Fixing something in a GitHub repository

GitHub[8] is a website (now owned by Microsoft) that stores thousands of projects' git repositories. You can clone, pull from, and push to it to collaborate on building software. As an exercise let's see if we can contribute to an open-source project.

First, you'll need a GitHub account (use your password manager to make your password!). Next, find a repository to send a pull request too. You can either make your own repository and use it, or actually make a contribution to another team's. One really easy one to try would be to search GitHub for common misspellings of a word and then just make a change to correct it. It's best to find misspellings in comments so you don't risk messing up the code. Even better, add some clarification to the documentation of a project you've been using a little.

Note: Make sure the repository has recent activity or else the maintainer may not ever respond to your pull request.

---

[6] https://git-lfs.github.com/
[7] https://git-annex.branchable.com/
[8] https://github.com/

Table3: Workflow for collaborating with people on GitHub

Operation	Description
Fork the target repository	Choose a repository to update and navigate to it. Click the "Fork" button in the GitHub web page. This will make a copy of the repository on your own account.
Clone your copy of the repository to your machine	From the command line, run `git clone https://github.com/user/repo.git` filling in your username and repo.git according to your fork (you can get this address easily by clicking the green "clone" button in GitHub)
Make the modifications to the source code on your computer	Navigate to the file(s) you want to change, change it, and save it.
Stage and commit your changes to your local repository	Run `git add -u` to *stage* all changed files and then run `git commit -m "Updated spelling"` (use an appropriately descriptive commit message).
Push the changes up to your personal fork of the repository on GitHub	Run `git push origin`. Now your change is live on GitHub but only on your fork of the original project. You may have to type your GitHub password at this stage[10].
Create a pull request to communicate with the original team	There should be a button on your GitHub now that says "Pull Request". Click it and fill in the form with details of what you changed and why. This will now enter into a review/approval process with the original team. If they accept it, your change will go live.

[10] You can alternatively set up SSH keys to avoid typing in passwords all the time. There are many tutorials on this available.

That's it. As usual, this is just scratching the surface, but these are the fundamentals. If you can do this, you are basically ready to participate in collaborative development or publishing using git. And with that, let's get to the actual programming.

Why program?

Programming is the Holy Grail of digital superpowers. Things like Machine Learning, gaming, business process, social networking, and so on are rooted in programming. Of course, it is an entire discipline that can take years or even decades to *truly* master, but you can do meaningful things very readily, and go from there as suits your situation. If you want, you can make an entire well-paid career out of it. Or you can just dink around.

I'm fairly convinced that programming should be a fundamental topic in middle schools all the way up, possibly even displacing some required math curriculum that very few people end up using. I say this as a Ph.D. hard-science engineer: I have never ever had to do long division of polynomials and I never will. But boy have I seen people in all walks of life struggle with things where a bit of programming exposure would have helped.

The single best way to learn how to program is to recognize that you have a problem you want to be solved, and that a program could help you solve it. For example, a friend of mine who is now an applied mathematics professor at Harvard gave me his favorite math riddle once:

> Given two 8's and two 3's, combine them with the four basic math operations (+, −, ÷, ×) to get 24. You have to use each. For example, 8+8+3+3 is a valid guess, but it's wrong because that equals 22, not 24. Similarly, 8×3=24 is invalid because you only used one 8 and one 3.

I struggled for a while and decided that a program guessing all the potential options would be useful. In the end, it worked!

More practically, a friend was doing some medical research using a large dataset of pharmacy refill data and was going through this

painstaking and error-prone process in a spreadsheet to quantify potential medication gaps. She explained her process in detail and then I helped her whip up a little program that did what she was doing. Thus, she was liberated to tweak and debug her *process*. She ran the program dozens of times, analyzing the algorithm she was using and coming up with interesting conclusions.

Personally, I've spent many years writing programs that help simulate nuclear reactors. Once we have a virtual nuclear reactor we can adjust its design and perform "experiments" that would cost billions of dollars to run in the real world until we think our design is optimal. We have to put a lot of effort into making sure the models match reality (e.g. by simulating things that really operated and comparing results to measurements). The synergy between physical models, numerical simulations, and experiments is a real work-horse in science and engineering across the board.

Then, of course, professional programmers build and maintain all the massive software systems behind a large fraction of our everyday experiences.

Programming languages

Hundreds of *programming languages* exist. Each of them has the goal of translating (or *compiling*) human input (*source code*) into electrical operations the computer can perform. Their differences arise from different trade-offs their creators were aiming for:

- Is it better to have a really fast-running program that can crash without saying why, or one that runs slowly but tells you in detail where it was when something went wrong to help with debugging?
- Should the same source code run on many types of computers, or should the programmer have to maintain different versions for different machines?
- Should the programmer have to wait 10 minutes between changing the source code and running the program so the compiler can optimize aggressively, or should it be instantaneous?

Very low-level programming languages like *assembly* run extremely quickly, but require customizations for every model of CPU and are challenging to write. Conversely, high-level languages like *JavaScript* are pleasant to read but generally slow. One of the more recent and very popular languages, *Rust*, attempts to let programmers write very "safe" code (if it compiles it will have fewer bugs than usual) while still running extremely quickly, and in parallel.

Among the data science, automation, web, and engineering industries (and many others!), the *Python* language has been doing very well in the past decade or two. It's a high-level interpreted language that runs slowly on its own. But it's so popular that people have interfaced it nicely with some lower-level screaming-fast math, graphics, and machine learning libraries, rendering it a pretty good workhorse. There's also a vibrant collection of other modules people have made that do all sorts of things like statistical analysis, graph creation, and web apps. Here, we're going to try out Python. Depending on your end goal, another language may be better.

Writing Python programs

Before writing a Python program, we need to download the Python interpreter. It's what translates Python code into actual operations your computer can perform.

Note: You can skip this step if you already have Python installed. Find out by running `python -V` in your command line and seeing if it throws an error.

Table4: Installing Python

OS	Command
Windows	`choco install python`.
Linux	It's almost definitely already installed
macOS	It's almost definitely already installed

Run `python -V` to make sure it works. If you just execute `python` at the command line, you'll be presented with a Python *prompt* that looks like `>>>`. Typing `exit()` will exit out when you're done. You can type *Python commands* here, and they will execute one after the other. For example, we can use it as a calculator. First just with numbers:

```
>>> 2+2
4
```

All programming languages have *variables* where you can store the result of some operation in a name like `x` or `number_of_customers`:

```
>>> x=2
>>> y=5
>>> x+y
7
```

You can also define *sequences* and then repeat an operation on all elements in the sequence in a loop:

```
>>> names = ["Ford", "Honda", "Jeep", "Chevy"]
>>> for name in names:
        print(name.upper())
FORD
HONDA
JEEP
CHEVY
```

Built-in libraries provide the ability to do useful things like sample random numbers. When you run these, you will see different numbers.

```
>>> import random
>>> random.random()
0.8254737295983463
>>> random.random()
0.2630839054490208
>>> random.randint(1,100)
88
>>> random.choice(['Pizza','Ramen','Eggs','Pho'])
'Eggs'
```

Random numbers are fundamental to things like games and cryptogra-

phy. That last line is basically a universal 8-ball, which is useful if you need help deciding what's for dinner.

Note: At this point, you're well poised to run through the official Python tutorial[11]. It mostly targets people who have programmed a little before, but even if you haven't, it's easy to follow along and get some exposure to the syntax and possibilities.

Running commands line-by-line in the Python prompt is useful for exploring and simple tasks, but real programs are written as sets of commands in text files. See *Text editors and extensions* to make sure you have an appropriate program for creating source code.

A program to approximate π

Let's make a real program that does something neat: approximates π with the Monte Carlo method. This leverages the fact that computers cannot get bored to approximate a fundamental constant of nature basically by throwing darts.

Imagine you are blindfolded and throwing darts at a perfectly square board with a side length of 1 m and with a circle drawn on it that just touches the edges. If you throw randomly, the number of darts that land inside the circle will be proportional to the area of the circle, while the number of darts that land anywhere on the board will be proportional to the area of the square. If you throw millions upon millions of darts, these approximations will become pretty good, and your counts will tell you what π is.

[11] https://docs.python.org/3/tutorial/introduction.html

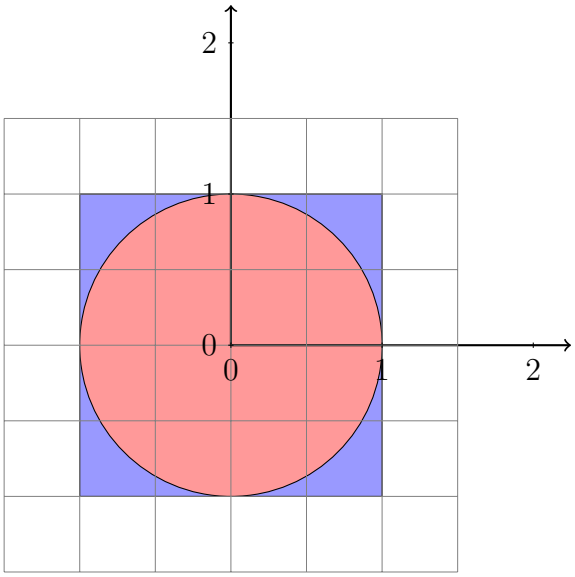


Figure1: The dartboard

$$\text{Square Area} = s^2$$

$$\text{Circle Area} = \pi \left(\frac{s}{2}\right)^2$$

Dividing those equations gives:

$$\frac{\text{Square Area}}{\text{Circle Area}} = \frac{4}{\pi}$$

or

$$\pi = 4 \times \frac{\text{Circle Area}}{\text{Square Area}}$$

Instead of throwing the darts ourselves, we can write a program to simulate the dart tosses and approximate the area ratio. To do so, we put the following in a new text file called `darts.py`. Note that all lines beginning with # are just comments and are not executed (so you don't have to type them).

```
import random

RADIUS = 1.0   # define a constant

num_in_circle = 0 # initialize our count

# ask user how many darts to toss
num_darts = int(input('Number of darts: '))

for _individual_dart in range(num_darts):
    # for each dart toss, grab two random
    # numbers between 0 and 1.
    x = random.random()
    y = random.random()

    # Anyone remember the distance formula?
    # We can skip sqrt here because sqrt(1) = 1
    d = x**2 + y**2
    if d < RADIUS:
        # increment count in circle
        num_in_circle = num_in_circle + 1

pi = 4*num_in_circle/float(num_darts)
print('Pi is approximately: {}'.format(pi))
```

Run the program with `$ python darts.py` at the normal command prompt, not the Python one. If you still see `>>>` at your command line, run `exit()`. It should give:

```
$ python darts.py
Enter the number of darts: 100000
Pi is approximately: 3.14504
```

Pretty close! The correct answer is `3.1415926...`. If you only run a few darts, the answer will be much more wrong (try it!). Ah, the central limit theorem in action!

My computer only takes a second to do 1 million dart tosses, but don't do too many, it may take too long! Remember: `Ctrl-C` will abort a running program if you get stuck.

The program only samples numbers between 0 and 1, so we're really

only operating in the first quadrant of the dartboard (top left). This is OK because the dartboard is symmetrical.

What if we want to plot our results? We'll need some third-party libraries to do that (things that are part of the Python community but not included in Python itself). To get them, use the *Python package manager*, which is called `pip`. It's very similar to the system package managers we've been using, but it just pulls in Python modules. We need two things to have plotting:

```
pip install --user numpy
pip install --user matplotlib
```

We have to modify the program to keep track of the `(x, y)` coordinates and then plot them in the end. So we'll make *lists* to store each coordinate. Here goes (this time as `darts_plot.py`):

```
import random

import matplotlib.pyplot as plt

RADIUS = 1.0

coords_inside = []
coords_outside = []

num_darts = int(input('Number of darts: '))

if num_darts > 1000000:
    # error checking
    raise RuntimeError('Too many particles.')

for _individual_dart in range(num_darts):
    x = random.random()
    y = random.random()

    d = x**2 + y**2
    if d < RADIUS:
        # add to the list of coords inside circle
        coords_inside.append((x,y))
    else:
```

(continues on next page)

(continued from previous page)

```
        # also track outsiders for plotting
        coords_outside.append((x,y))
pi = 4*len(coords_inside)/float(num_darts)
print('Pi is approximately: {}'.format(pi))

# Make plot
# unpack x's and y's into their own lists
xinside, yinside = zip(*coords_inside)
plt.plot(xinside, yinside, 'b.',label='Inside')

xoutside, youtside = zip(*coords_outside)
plt.plot(xoutside, youtside, 'gx',label='Outside')
plt.grid()
plt.legend()
plt.title('Approximation of $\pi$')
plt.xlabel('x (m)')
plt.xlabel('y (m)')
plt.show()
# uncomment to save as file
#plt.savefig('darts.png', dpi=200)
```

Warning: For this version, you better not try running too many darts or else your machine really might crash.

And here is the result:

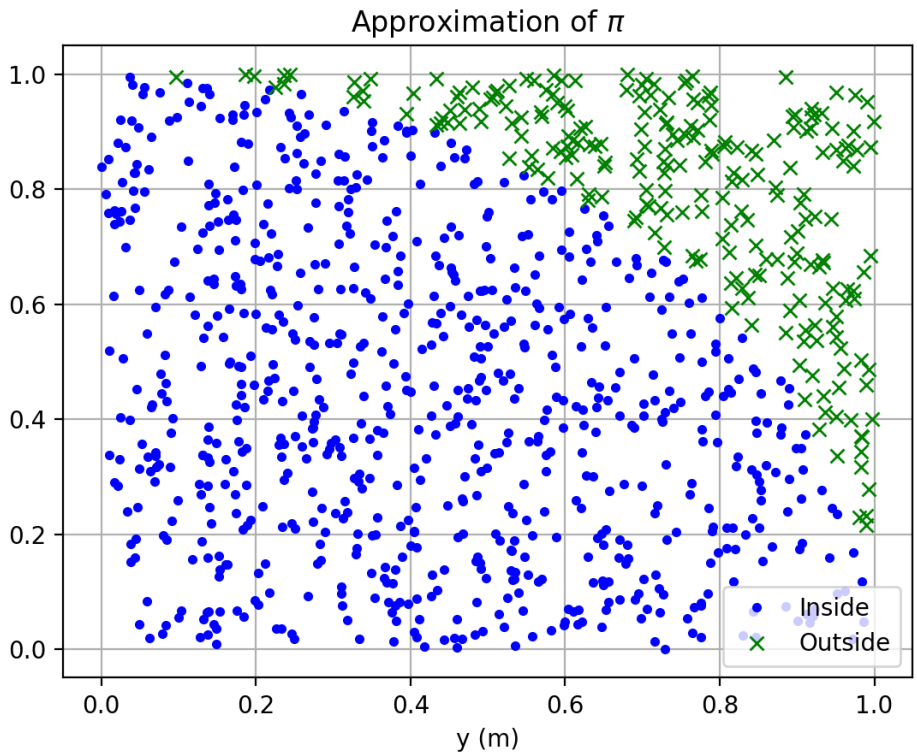


Figure2: Plot of 10,000 darts hitting or missing.

Pretty fancy! This is admittedly a very inefficient method to approximate π. The fundamental method of generating random numbers and running computerized trials, however, is extremely powerful and valuable. First used to model thermonuclear weapons on the original mainframes in the 1950s, these methods are used all over science, engineering, risk management, insurance, and finance. Now you're well on your way to becoming a high-powered quant mega-millionaire on Wall Street.

For more inspiration about what kind of plots you can make with Python (as well as example code to learn how to do it), check out the matplotlib gallery[12] and plotly[13].

[12] https://matplotlib.org/stable/gallery/

[13] https://plotly.com/python/

A linear regression

Python is great for stats. This is often useful for your research teams and around the house (I once used regressions to estimate what time the furnace had to turn itself on to get up to the desired temperature given the outdoor temperature each morning during the Michigan winter). Let's do a simple example of reading data in directly from Excel. The pandas[14] library provides high-level data analysis capabilities, but we're just going to use it for reading from Excel:

```
pip install --user pandas
pip install --user xlrd
```

We need some data to process. Open up your favorite spreadsheet program (Excel or Open Office Calc will work fine, for instance) and let's generate some noisy data. Let's use a line from this formula:

$$y = mx + b + \epsilon$$

where:

$$\begin{aligned} m &= 3.5 \\ b &= 2.3 \\ \epsilon &= \text{Uniform noise from -4 to 4} \end{aligned}$$

Note: If you have your own data you want to do a regression on, just load that up instead!

In a new spreadsheet, label the first two columns `X` and `Y`. In the first column, enter the formula: `=RAND()*5` to get random x-values between 0 and 5. Click the little black box in the bottom right corner of the cell and drag it down 30 or more cells to get lots of random X values. Then in the `Y` column enter our line as `=3.5*A1 + 2.3 + (RAND()*8-4)`[15]. Double-click its little black box to fill down. Save the file as `data.xlsx`. It will look like this:

[14] https://pandas.pydata.org/

[15] To get our noise from -4 to 4 we do a random number between 0 and 8 and subtract 4 from it.

	A	B
1	X	Y
2	2.72337737	9.96801996
3	4.48134377	16.3558232
4	1.20537228	3.55701239
5	4.80170672	15.5727303
6	1.74394316	5.27918594
7	3.79467892	17.9391624
8	0.09536572	0.50969187
9	3.73022051	14.4704398
10	0.63912977	2.12314499
11	1.46911346	7.2990603
12	0.77471679	3.95056734
13	0.54363541	7.48054793
14	1.00705999	2.30254887
15	1.67254084	4.65955513
16	4.35097146	15.6558039
17	0.40116527	6.60823643
18	4.45521565	21.4728294
19	2.83531263	13.4286586
20	1.42569761	8.60714881
21	4.86064028	19.1467933
22	4.05399503	19.7532507
23	0.60356886	4.77548493
24	4.73506626	15.8056171
25	0.68888264	2.9597832
26	0.83118009	7.91973463
27	3.92834423	14.1075571
28	3.76204074	15.6992754
29	3.45298583	13.437501

Figure3: Example noisy data simulating our line. Yes we could use Python to make these data but many of us receive data in spreadsheets.

To do the regression:

```
from scipy import stats
import pandas

# Read data
data = pandas.read_excel('data.xlsx')

# Grab the columns we want
x = data['X'].values
y = data['Y'].values

# Compute the regression
```

(continues on next page)

(continued from previous page)

```
results = stats.linregress(x,y)

# unpack results into useful names
(slope, intercept, r_val, p_val, std_err) = results

# Display results
print('Slope:     {}'.format(slope))
print('Intercept: {}'.format(intercept))
print('R-squared: {}'.format(r_val**2))
print('P-val:     {}'.format(p_val))
```

Here's the result with my data:

```
Slope:     3.473511906229649
Intercept: 1.8192562724560553
R-squared: 0.861400909580644
P-val:     1.1563714298035524e-12
```

That's all it takes to do a regression! We can plot it too, by adding this code:

```
# Make a plot while we're at it.
import matplotlib.pyplot as plt
import numpy
modelX = numpy.linspace(0,5,10)
modelY = modelX * slope + intercept
fig = plt.figure(dpi=300)
plt.plot(x,y,'o', label='Data')
plt.plot(modelX, modelY, '--',label='Model')

# Add labels & formatting fluff
plt.title('Beautiful regression')
plt.xlabel('X')
plt.ylabel('Y')
plt.grid(linestyle='--')
plt.legend()
plt.text(3.1,3.0,'m={:.3f}'.format(slope))
plt.text(3.1,2.0,'b={:.3f}'.format(intercept))

plt.savefig('regression.png')
```

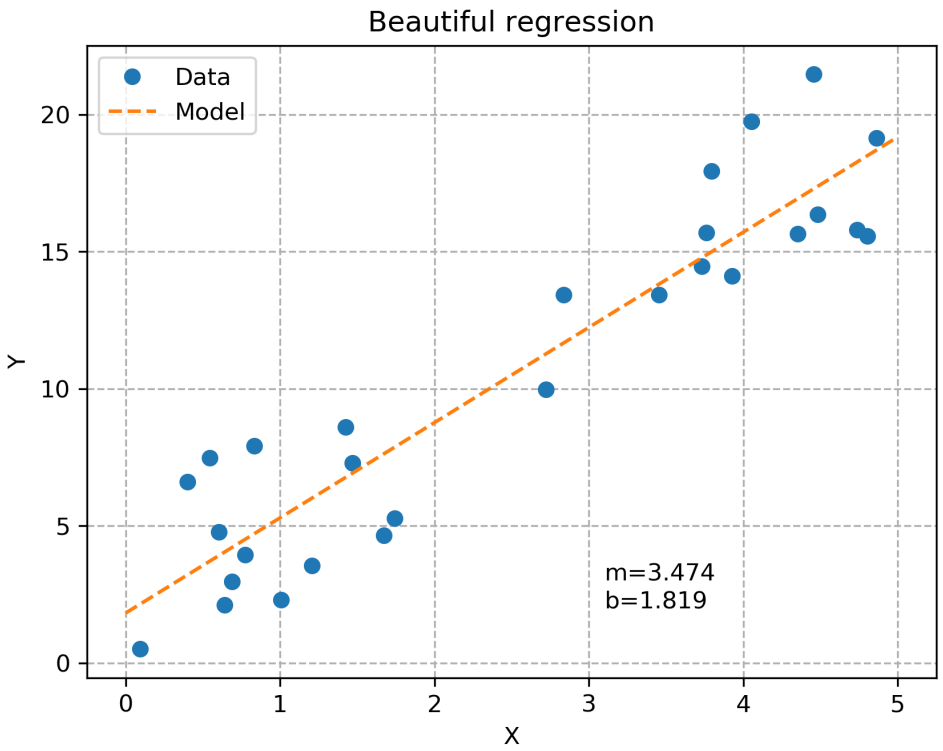


Figure4: The random data plotted with the best linear fit.

Hey, not bad! Note that by defining the plot in code, we can afford to spend time polishing its look to be very professional because we only have to do that part once. We can then throw a billion different data sets at this program and not have to spend a second plotting the results; we already did it once, the computer's got it from here.

Exercise

Make the noise go from -1 to 1 and see if your r^2 value gets closer to 1.0!

Note: Another great tool for data processing is called R. There's plenty of debate about whether Python's better than it. It's fair to say that

Python is sufficiently good at stats for production research and has the advantage of being very general-purpose, so if you learn it, you get stats plus anything else it can do, whereas R is more specialized for stats only.

The bridge to machine learning

From self-driving cars to face recognition to the Terminator, machine learning (ML) is quite a hot topic. The reason it's gotten so good is that computer graphics cards have gotten extremely fast at doing matrix-vector multiplication, largely driven by the gaming industry (where this math is required for shading as players move around). Combine this with the universal function (capable of mapping any arbitrarily-complex input to an arbitrarily-complex output) that is the Convolutional Neural Network and massive datasets to train them on (e.g. vast quantities of selfies) and you've got yourself a machine learning revolution.

One good way to get your hands dirty in ML is to dedicate yourself to some fast.ai courses[16]. These free online courses are cutting-edge, and require roughly 8-hours per week for two months or so. Not everyone's computer is going to be good at training on large datasets (unless you're a gamer), but you can try it out (I got my computer working on them). Better yet, they set up a cloud environment at Amazon Web Services where you can just log into computers "in the cloud" that are more than capable of doing the exercises in the class.

Note: Much ML work is done through Python, which interacts with heavier-weight libraries under the hood. That's another good reason to choose Python as a first programming language.

A simpler introduction to computer vision (sort of related to ML) is to play around with the OpenCV[17] library. It has good Python bindings and you can even get it going on a $35 Raspberry Pi mini-computer. I've seen people make license plate readers, recognize family members'

[16] https://www.fast.ai/
[17] https://opencv.org/

faces, analyze video of experiments to collect data, and lots of other useful things.

A trivial (non-ML) example usage of it is to get some data out of a movie file. I had a high-speed video of a light mill and wanted to see how fast it was spinning by analyzing the video[18]. OpenCV is perfect for this. I figured out which pixel I wanted to watch and then extracted intensity vs. time with a simple program:

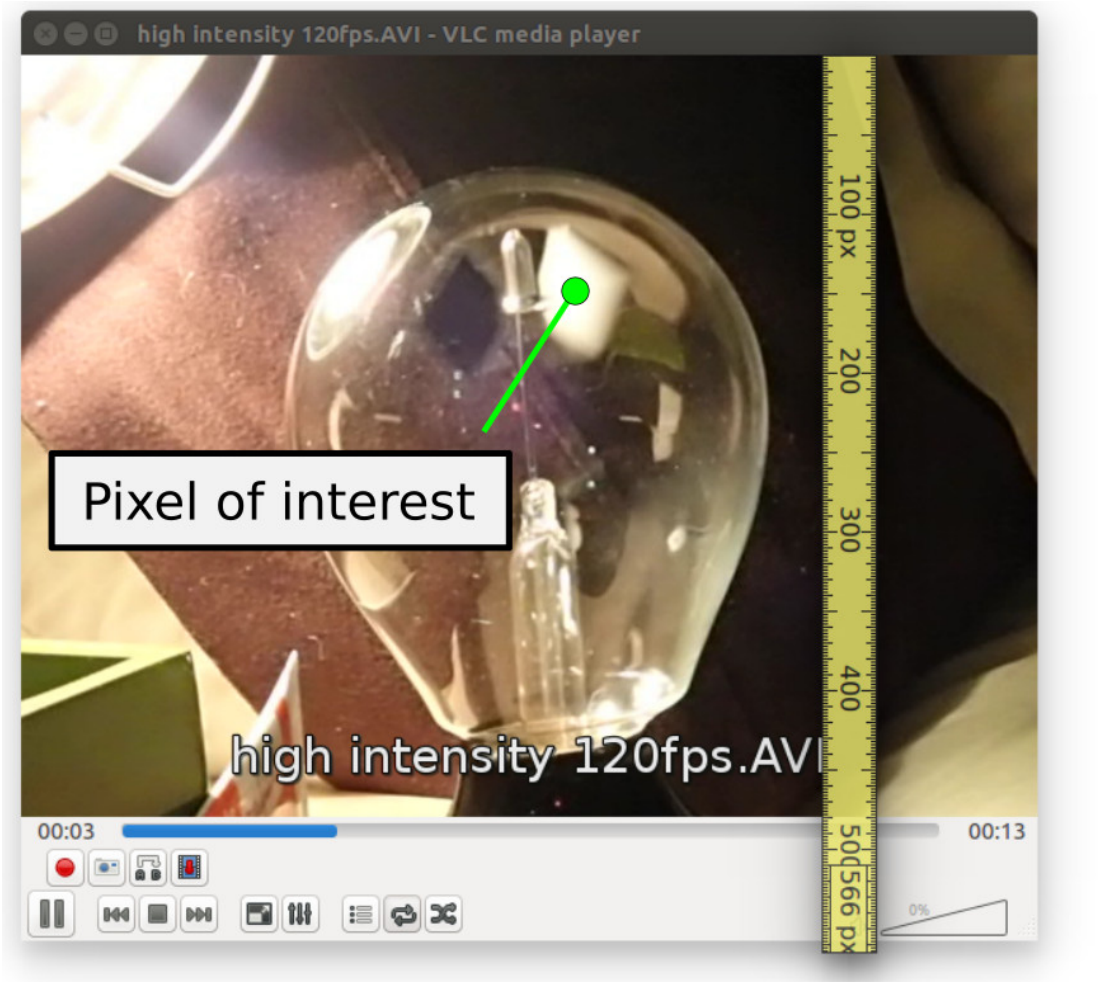


Figure5: Figuring out which pixel to read.

[18] There's a full blog post about this at https://partofthething.com/thoughts/reading-out-a-crookes-radiometer-light-mill-with-python-and-opencv/

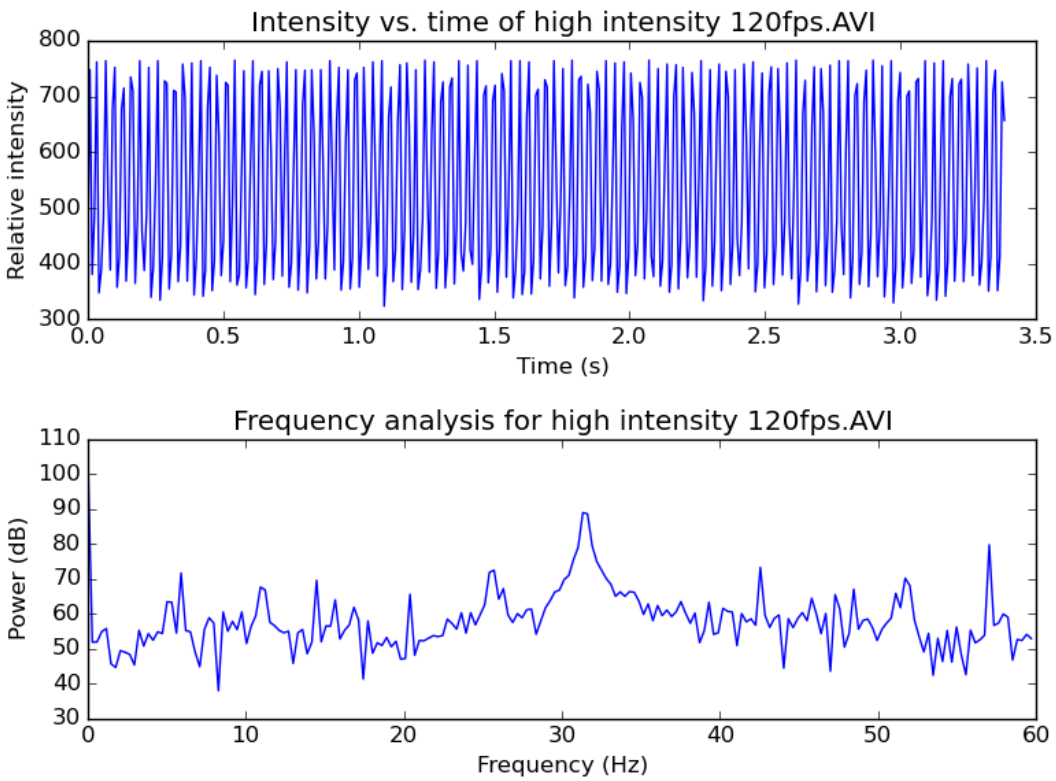


Figure6: Plotting and analyzing the intensity data. Looks like it's spinning at about 32 vanes per second. These devices, actually called Crookes radiometers, were originally scientific equipment that could be combined with a stroboscope to measure the intensity of light.

The (partial) code for reading this looks like this:

```
import cv2  # bring in the OpenCV library

def measureIntensities(videoFileName, pointOfInterest):
    """Find the intensity of a point of interest in each frame"""
    video = cv2.VideoCapture(videoFileName)
    intensities = []
    while video.isOpened():
        _returnCode, frame = video.read()
        if frame is None:
            break
```

(continues on next page)

(continued from previous page)

```
        pixelValues = frame[pointOfInterest]
        intensities.append(pixelValues.sum())
    video.release()
return intensities
```

If you have video of something and want to quantify some element of it with a computer, OpenCV is your ticket. Once you are comfortable with simple things like this, you can try out facial recognition and license-plate reading with it as well.

Graphical User Interfaces

Making desktop applications GUI with dialogs, buttons, menus, and forms can make your programs more accessible and user-friendly. Most programming languages enjoy GUI frameworks that do most of the hard work of making such interfaces. In Python, we have Kivy, wxPython, PyQt, Tkinter, and dozens of others. I picked up wxPython first because of its excellent demo app that lets you try out all its hundreds of widgets interactively with sample code. Kivy is nice in that it can even run on cell phones. PyQt is especially polished, though it requires a commercial license if used in commercial software (not that there's anything wrong with that).

Perhaps the most universal and modern GUI system is a web browser, which brings us to the next topic.

Web applications

Almost every web page you go to these days is actually a web application, meaning it's running some program behind the scenes and presenting the results to you through a web browser like Firefox or Chrome. Almost always, there's a database sitting on the back end containing the information. For instance, Facebook is conceptually a web-based view of user data stored in a database and governed by rules and process.

The possibilities with web apps are nearly endless. Unfathomable yet-to-be-created businesses and services will be web apps. A home inventory could be a web app. Twitter is a web app. Reddit is a web app. The programs driving web apps are written in many varieties of programming languages, just like offline applications. Through the years, though, various frameworks have evolved that contain commonly-needed pieces of web applications, like user authentication, secure form submission, and secure database access. Starting with a web framework is a great way to become productive at making safe and performant web apps rapidly. The hundreds of web frameworks out there all have distinct pros and cons. We will highlight only one of them here.

Instagram and the Washington Post are among the thousands of web apps built upon a web framework called Django[19] ("The D is silent"). Django is surprisingly nice to use. Having come from a Python background, the fact that it uses Python made it extra easy for me to pick up. I went through their Writing your first Django app tutorial[20] and was off and running. This was really an exciting moment for me because I could just feel my mind expanding... "So many new possibilities. I can make any web app out there with this thing!" It's the closest thing to leveling-up I think I'll ever experience.

Django is extremely well documented and has a fun, engaged community. For example, Django Girls[21] is a non-profit centered on teaching Django to girls around the world while building interest in programming.

Note: One mental model that's often used in these kinds of applications is the "Model-Viewer-Controller" (MVC) pattern. The model is the database organized somehow, the viewer is the web page rendered with the data in it, and the controller is the code running to figure out what to show when, and what you are and are not allowed to do. By separating these units out, the code remains easy to maintain. Django is designed around MVC.

The sheer magnitude of capabilities Django and its add-ons bring is im-

[19] https://www.djangoproject.com/

[20] https://docs.djangoproject.com/en/2.1/intro/

[21] https://djangogirls.org/

pressive. Once you get to the point where you're defining models, a lot of impressive systems activate automatically. For example, let's say you were making a discussion forum where you had made some models that look like this:

```
class Post(models.Model):
    title = models.CharField(max_length=300)
    poster = models.ForeignKey(User)
    posted = models.DateTimeField()
    votes = models.IntField(default=0)

class Comment(models.Model):
    title = models.CharField(max_length=300)
    poster = models.ForeignKey(User)
    post = models.ForeignKey(Post)
    posted = models.DateTimeField()
    votes = models.IntField(default=0)
```

The automatic administration panel would allow you to populate the database with information. It presents you with web-based widgets that are appropriate for each data type. The `DateTimeField` has a calendar pop-up picker and the `ForeignKeys` have drop-downs with links to the other instances. You can build custom forms for your users that automatically get these widgets as well. You can also use the Django REST framework (an add-on) which will set up an entire system to interact with this app from other places (other web apps, desktop programs, phones, etc.). In other words, all the things you typically need to be done with a web app have built-in or readily available solutions.

I simply cannot emphasize how empowering this is. If you learn it, you'll start wanting to automate all sorts of inefficiencies around your office, research team, warehouse, or wherever you find them. The official documentation cannot be improved upon and so to get started, simply head on over to it and begin. Having Python installed from the previous section is all you need to get going. I look forward to hearing about the wonderful things you do with these systems.

Note: Sysadmin skills in running a web server are useful for setting up production-caliber Django apps, though you can start building apps right away using the built-in test server that comes with Django.

And there you have it, the ultimate computer superpower. The skills you've started here can change the world, or simply make your life a little better. I hope you'll find some exciting, interesting, and helpful uses of these technologies.

8

Robotics and Hardware

So far we've dealt exclusively in software. Whole new ballparks of fascinating capabilities can be reached by looking to hardware as well. Hardware can interact *physically* with us and our environment. And what a time to be alive in the hardware realm! Of course, to begin interacting with the physical world, it's best to start with some user-friendly hardware. The exploration of hardware in this chapter builds on the skills introduced in *Programming*, since some coding is required. Fortunately, completely novice programming experience is more than enough to begin the journey. In fact, hardware projects often provide a thrilling motivation for expanding one's programming skills.

Smartphones have an astounding number of hardware sensors and input devices on them, but unfortunately there's no system that I'm aware of that provides entry-level access to the data, other than the Unity3D system discussed in *Making games*. Thus, in order to try out some hardware-related superpowers, it may be necessary to buy some (relatively inexpensive) hardware systems. This chapter is therefore simply explanatory.

Warning: Electricity is dangerous and can kill you and/or damage property. Taking an electronics course is a good way to get started with the kinds of projects discussed in this chapter if you are inspired to do so.

The era of cheap, user-friendly microcontrollers

Since the early 2000s, the market has been flooded with circuit boards containing a low-powered CPU, some memory, digital and analog input/output ports (for interacting with the external world), and network capabilities. These are basically mini-computers that you can program with your normal computer and then disconnect to put in a robot, your garden, an art installation, and so on.

One such product is the Arduino[1], an open-source microcontroller (the hardware design is open as well as the software) that really started the cheap-microcontroller revolution. To use, you attach various sensors, buttons, lights, or other circuits to it and then write a small program (called a *sketch*) on the computer. You transfer the program via the USB port to the Arduino, where it gets stored on the board and begins executing. These sketches are generally written in the `C` programming language, which is lower-level than Python (`C` is more powerful and faster but harder to read and write). Getting started, many users keep it attached to the computer's USB port as a power source and/or to have it interact with the computer while executing. For robots and other things away from the computer, a battery can be used for power while the embedded program runs.

[1] https://www.arduino.cc/

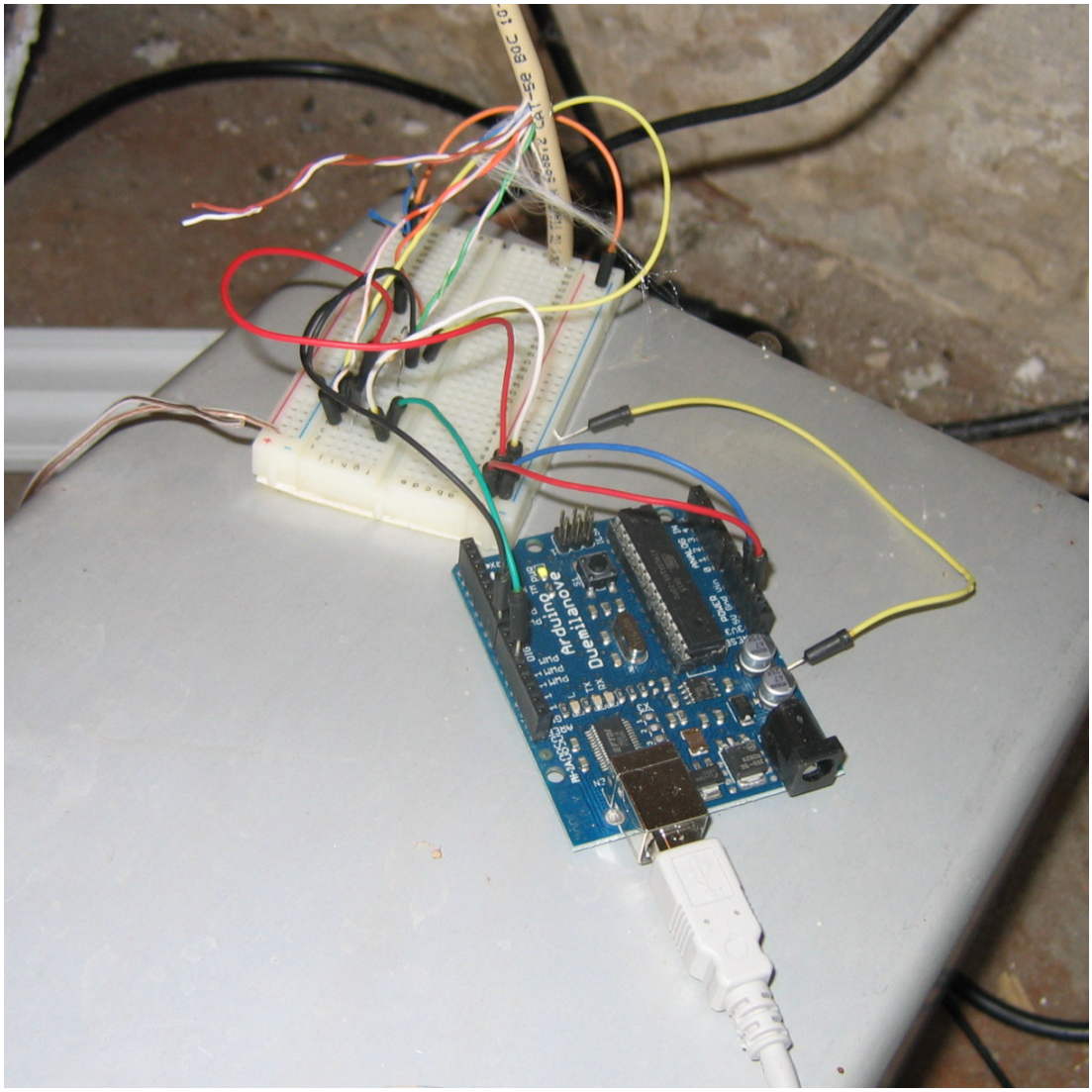


Figure1: An old Arduino 2009 connected to temperature sensors and relay controllers that fully automate the temperature of an ancient hot tub, allowing soakers to send text messages to adjust the temperature while they're on their way home. The computer reads the e-mails in a Python program, reads the current hot tub temperature in the tub, and turns the heater relay on or off to reach the target temperature.

Large collections of add-on boards (called *shields*) have been built and are on the market that add additional features to Arduinos.

The Raspberry Pi[2] is another extremely popular single-board computer. It has a much more powerful CPU than most Arduinos, an Ethernet port, a Wi-Fi card, an HDMI video port, an audio output port, and a port for an optional camera. It can run many of the same programs you use on your daily PC, like web browsers and word processors. It's used for computer science education and has an absolutely huge hobbyist following. A Pi can even be set up to play collections of old video games (SNES anyone?). People run their entire home automation systems and self-hosted cloud drives with them (more on that later). Raspberry Pis are certainly more complex systems than Arduinos, but the extra power makes them practical for running higher-level programming languages (like Python), and the built-in peripherals make it more flexible.

Some basic peripherals

You'll need some hardware components to fully embrace the hardware superpowers. Here's a list of some useful peripherals.

- **LEDs** — Little lights that you can turn on and off as indicators in a game, on a control panel, or for thousands of other uses.
- **Resistors** — fundamental electronic components that resist the flow of electricity. Often needed to keep too much electricity from flowing (e.g. often used in series with LEDs). Get a variety pack.
- **Prototype board** — a board that you can plug circuit components into to make connections, but that's easy to tear down as well. Used for experimenting and learning. Often when a design is finished, it's transferred from a proto-board to a custom circuit board with solder.
- **Relay board** — A series of electromagnetic switches that can be flipped on or off by small voltages (i.e. from a microcontroller). With relays, you can use a microcontroller to switch on and off something large like a heater or a large light. As always, electricity is dangerous and can kill you. Do not hook a relay up to some appliance without appropriate training and safety precautions. You

[2] https://www.raspberrypi.org/

can often hear relays clicking on and off, and you may recognize the sound from your turn signal relays in your car clicking on and off.

- **PIR motion sensor** — Passive Infrared motion sensors are those white plastic motions sensors you see all over. They are cheap and sensitive. You can build things that respond to motion with them, like a security system or a Halloween product that yells when people walk by.
- **Sonar** — Sonar uses sound reflections to measure distances. You can mount sonar peripherals on your projects to see how far away they are from a wall, or to increase audio pitch as something gets closer, or whatever.
- **Photoresistor** — Responds to light. Can be used to build a flashlight-tag game, respond to darkness, or turn off a light when the sun comes up.
- **Transistors** — Fundamental components similar to relays but with no moving parts and a continuum of responses, rather than just on and off. These are the basis of digital technology behind all of computer technology, but can also be useful for sending more current than a microcontroller can safely provide to a bright LED or a series of LEDs, among nearly infinite other things. You will need some for various examples during your training.
- **Patch cables** — You'll want a bunch of short multi-colored cables to connect your hardware pieces together. These are available with double-male, double-female, and mixed end connectors, and all three styles come in handy.
- **Power supplies** — While a lot of fun, small stuff can run on USB power, you'll likely want more power supplies for certain projects. For example, the LED light strips generally require 12V power (not the 5V that comes out of a USB port) and a lot of amps. If you want to control LED light strips you'll need a power supply and a relay board. These power supplies are exactly what you plug your laptop and phone in with, but they have different voltages and amp ratings which can be chosen based on your needs.

An interactive art installation

I'd like to share an example of what kinds of things become possible with the superpowers we are discussing. A Sweden-based artist named Mary Coble happened upon a project I had done using a Raspberry Pi to blink a laser in Morse code to communicate with light. She had used Morse code in previous projects and wanted to set a new one up. Patrons would type protest chants into a form on their phones, and the installation would beam them out in Morse code across the town. I agreed to help out, and so began a rewarding and exciting foray into the world of the art technician.

The project wasn't simple. Mary had to do all the artistic design and physical implementation. I provided a shopping list (including a Raspberry Pi, LED lights, and relays), instructions on how to set up the Pi and wire the lights, and remote software support. Once Mary had the Pi on her network, I was able to remotely connect and add the software I had written to receive messages from a web page and switch the relays on and off accordingly.

In the end, the system had the following schematic:

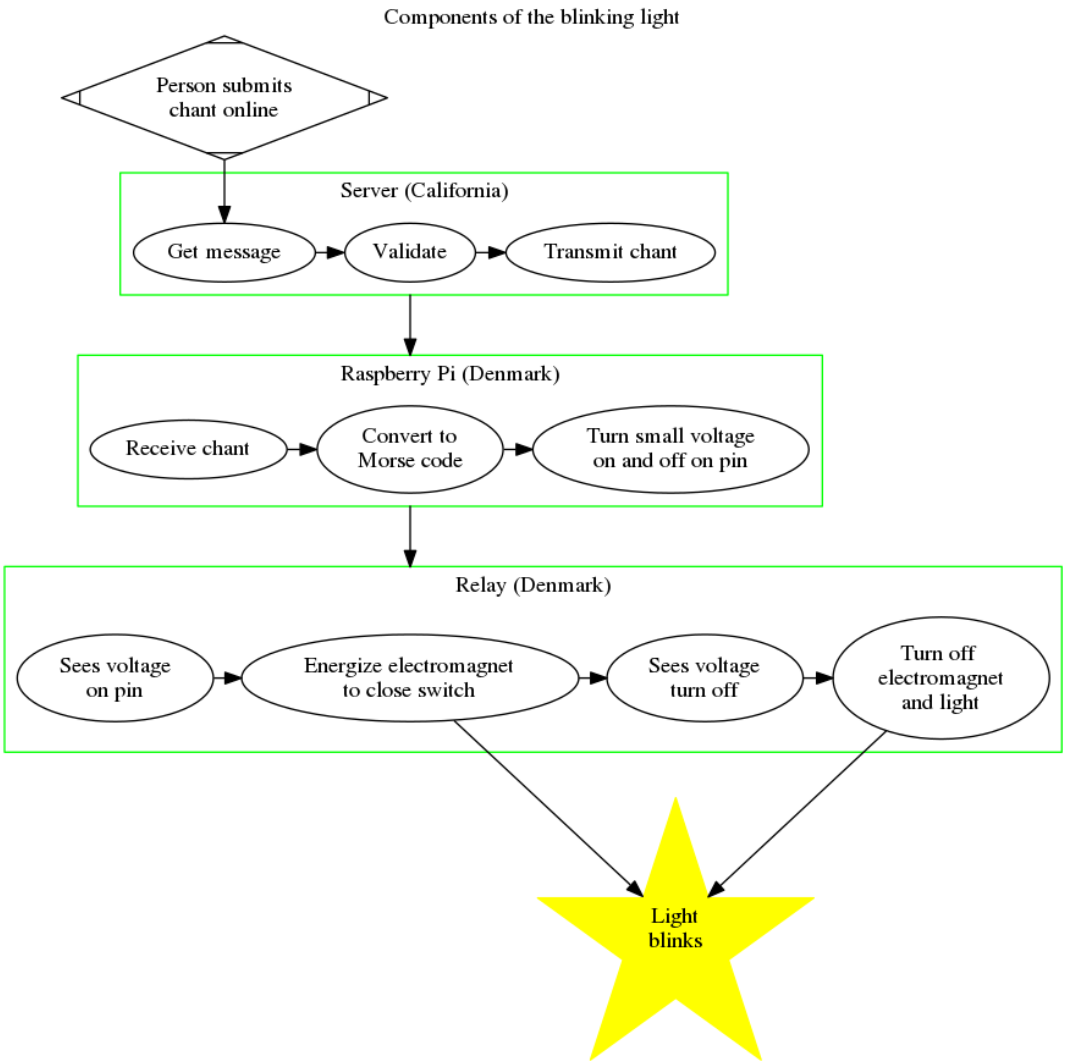


Figure2: The various interconnects in the PULSE system backend

It went well, and Mary did an even larger version of the system in her *SHOUT FIRE* exhibit at the Röda Sten Konsthall, in Göteborg, Sweden.

Note: For remote installations of any kind, a *reverse SSH tunnel* can be useful. Wherever your controller is plugged into a network, it will seek out and connect to your server. Once it does, you can connect to the remote device through the tunnel. If you want to change the behavior of the art installation once it's installed, this can be a good option for

doing so.

Working with this artist from across the Atlantic, whom I had never met in person, in such an engaging and cool project was really just wonderful. It felt like the internet really was bridging gaps in the way people had dreamed of in the early days. What an experience![3]

Even cheaper microcontrollers

One of the more astounding pieces of hardware is the Espressif 8266, or ESP8266. This is a very tiny board with a processor and onboard Wi-Fi capabilities. In bulk, it only costs $4! It's now been succeeded by the ESP32 which has Bluetooth as well. You can find a single unit of either in a nice package online for about $10. These are super lower power and can be put in about anything you can imagine. These little guys can be the whole brains and communication system behind whatever Wi-Fi capable little gadget you can dream up if you were interested in going into the business of building and selling products. Since they're so cheap, it's no problem to dedicate them to seemingly trivial applications. For example, you can use them:

- To add an internet-controlled thermostat to an old boiler/furnace. It's basically a $4 Nest system
- To add a wireless opening sensor to a cabinet (which could be used to set off an alarm or otherwise keep a child or friend away from something)
- As a doorbell extender that senses when the doorbell goes off and plays a recording of the doorbell on the stereo upstairs
- As the brains behind a star-tracking camera mount to keep the camera pointed at the same star for many minutes while the world turns

That last example deserves more explanation…

[3] The longer version of this story is available at https://partofthething.com/thoughts/helping-an-artist-with-a-morse-code-protest-chant-installation-in-denmark/

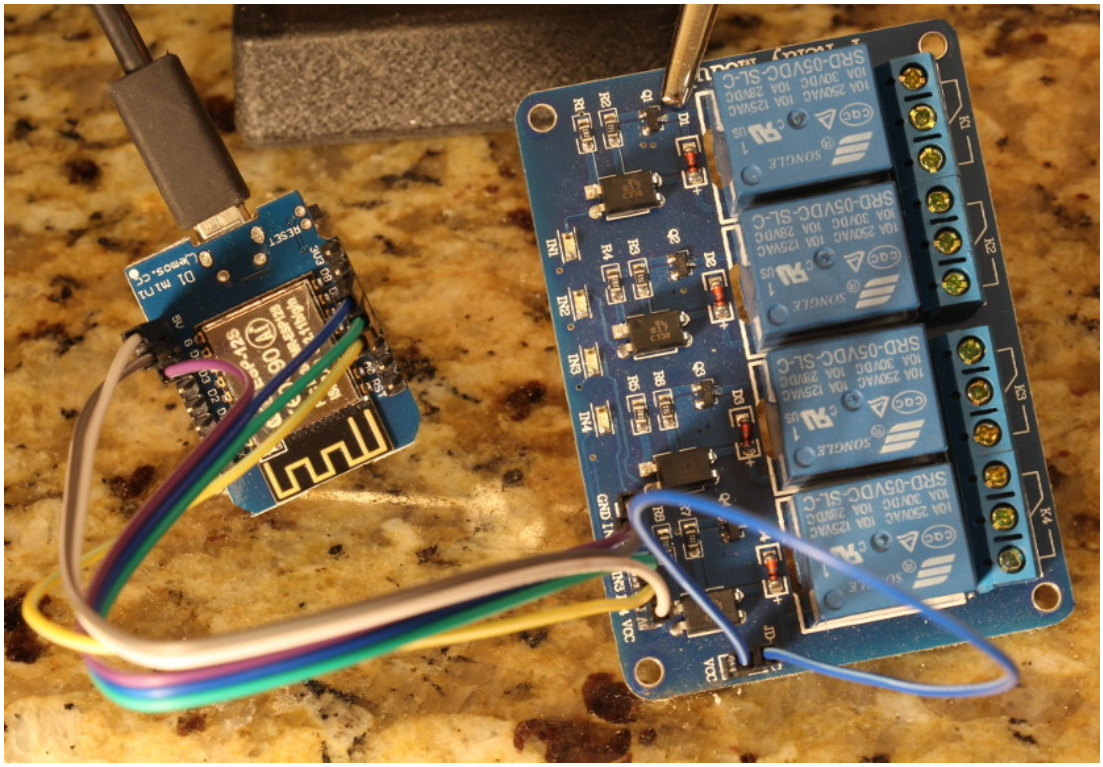


Figure3: A ESP8266 microcontroller hooked to a set of relays. This combo allows you to make a Wi-Fi controlled smart switch on a very tight budget, all while learning a ton and paving the road for even more creative ideas down the road.

A star-tracking camera mount for astrophotography

The Earth rotates 360° every 24 hours, making it difficult to take a long-exposure photograph of stars with the shutter open more than a few seconds without getting streaks. A simple solution is to put the camera on a mount that rotates the same exact speed but in the opposite direction, canceling out the rotation and keeping the stars deadlocked with the lens. You can buy a camera mount that does this, or you could make one yourself using your digital superpowers. One simple mechanism is to put two boards together with a hinge and have a motor slowly turn a

screw to push the boards apart. This is called a barn-door mechanism, and looks schematically like this:

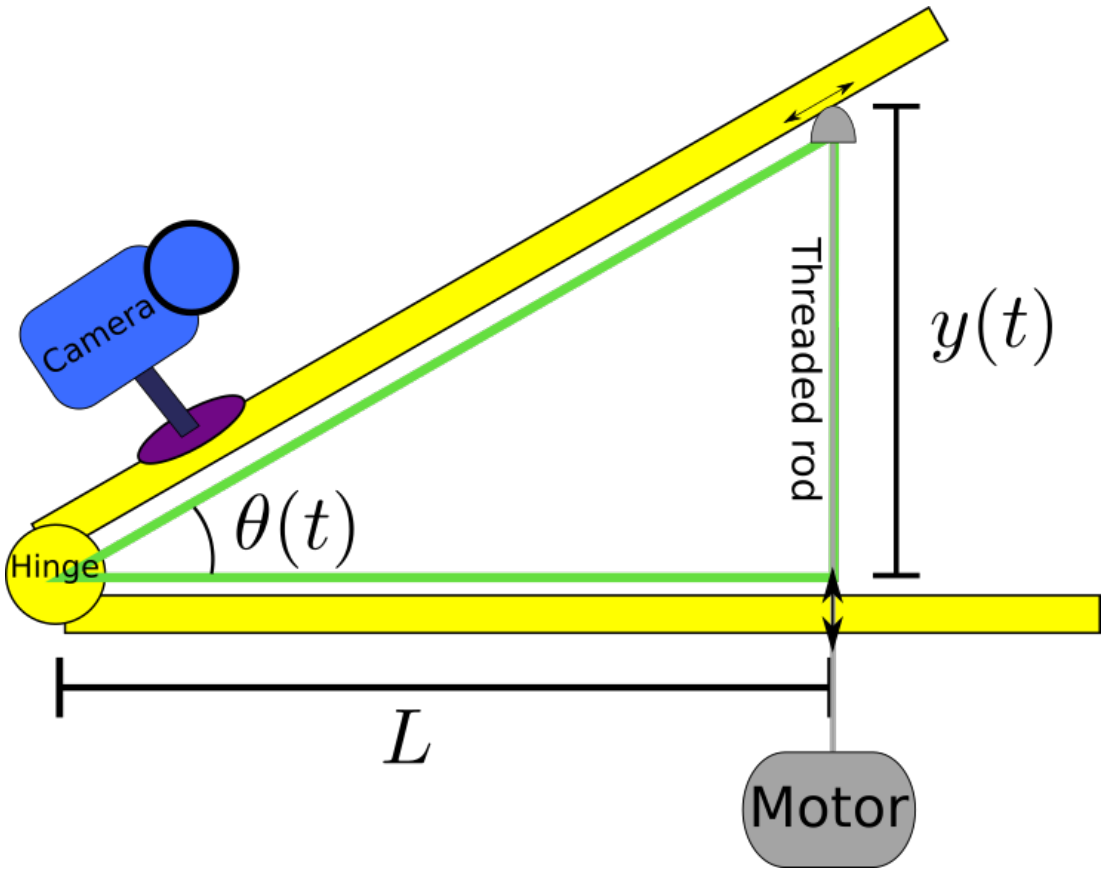


Figure4: A schematic of a barn-door star-tracking camera mount.

If you make the distance L equal to 29 cm, a motor that turns at 1 RPM will give you very close to the right speed. In that case, this project is fairly simple and no fancy software is needed. But that rate will only work for short exposures. If you want very long exposures, the motor has to speed up as the screw is inserted more. Mathematically, just note that the long arm of the triangle in the figure gets longer as the screw slides along the bottom of the upper board to see why.

If you use a stepper motor controlled by a microcontroller (like the ESP8266), you can precisely control the speed with which the motor turns, and you can even speed it up as time goes on to do the tangent

correction (as its called) in software!! This is extremely cool and works very well.

Note: By the way, the Sun is a star like any other, and this project can be adapted with larger motors to track it with a solar-panel, increasing the efficiency of your solar installation. The software is the same!

Detailed pictures, source code, and construction steps for doing this exact project are available under an open-source license at https://github.com/partofthething/startracker

Figure5: A photo of a working star-tracking camera mount.

Controlling hardware directly from your laptop

If you'd like to interface with some hardware but you don't want or need to bother with microcontrollers, you can just use your laptop directly for many of these activities. You do need a bridge to the hardware world.

When I first read about Bunny Hwang's "heirloom laptop" that he added General Purpose Input/Output (GPIO) ports to, I was really jealous and thought that was a great idea. That would allow you to control relays, read signals, blink lights, read data from various sensors, and all sorts of things people do with microcontrollers right on a laptop. It turns out that you can get a USB device that adds GPIO ports to any computer quite easily.

The Adafruit FT232H breakout[4] is one such device. Most of the demos you may find online about doing such things with an Arduino or Raspberry Pi can also probably be done with a board like this and software on your computer.

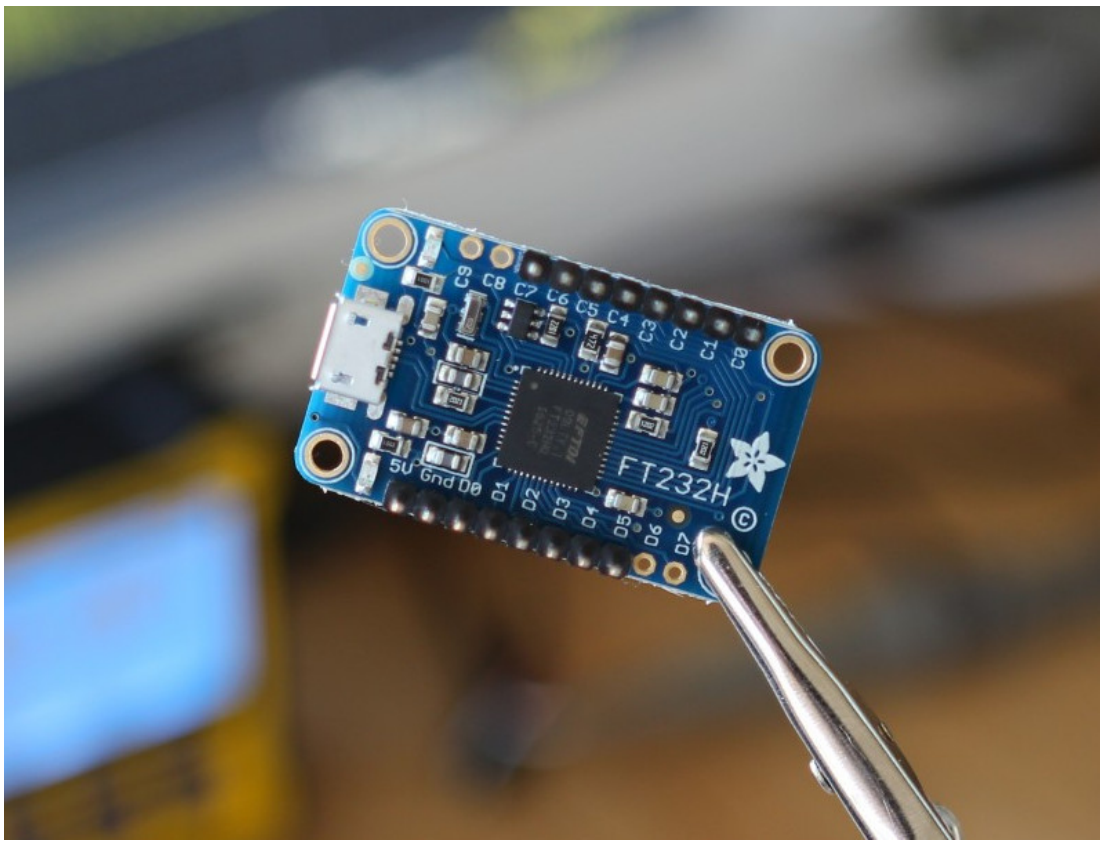


Figure6: The FT232H breakout board, a general-purpose GPIO system for your USB port.

[4] https://learn.adafruit.com/adafruit-ft232h-breakout/overview

For example, if you have a simple digital level and you'd like to read the data in live to support some experiment or other project, this breakout board is perfect. An example usage would be to precisely measure and calibrate the rate of change of the angle in a star-tracker like the one just discussed.

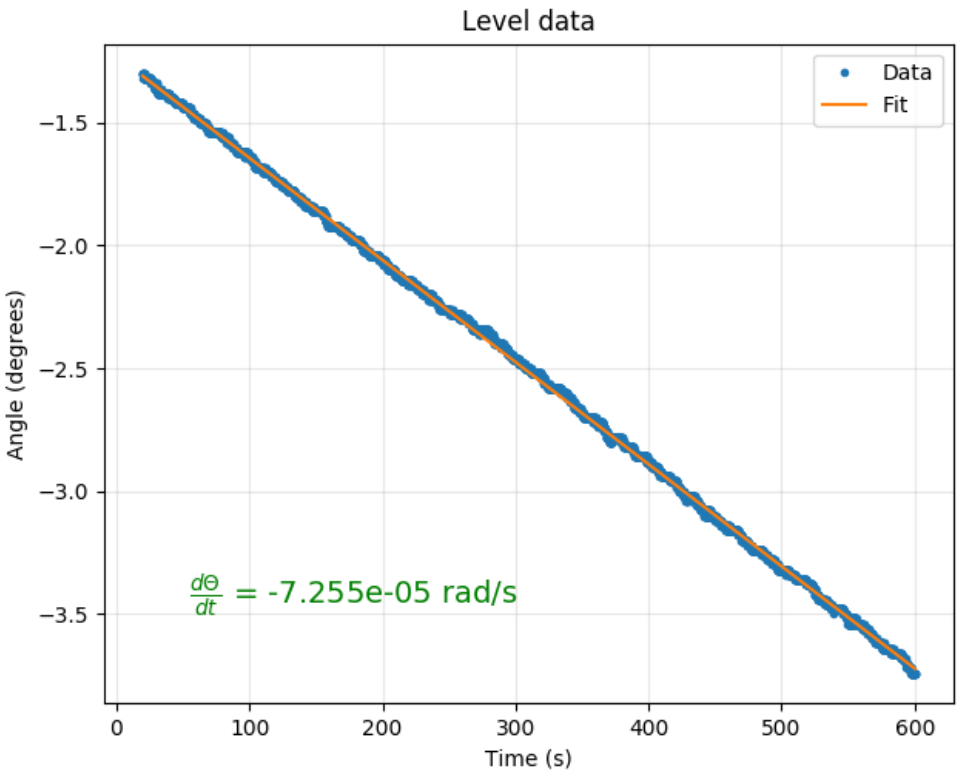


Figure7: Level data with best fit line for calibrating a star-tracker. The Python code that read these data and made this plot are included in the star-tracking code referenced in the previous section, in the `calibration` folder.

This chapter briefly introduced you to a variety of ways to bridge the gap between computers and the physical world. I hope these concepts help bring you an "Ah-ha!" moment regarding something you've been dreaming of starting. Many devices on the shelves of stores can be built

using combinations of the components you read about here. What new product can you dream up?

9 Self-Hosting

With all the news about this company breaching trust and that company getting itself breached by hackers, some of us may feel the urge to take back some control of our data by keeping it on machines we operate and/or own. This practice is called *self-hosting*. Fortunately, those who want to take a deep dive will find plenty of options. This chapter will explore but a few.

This is a bonus chapter intended for advanced readers who are either very interested in the hobby elements of computing or who are thinking of becoming IT professionals and want to get their hands dirty.

Warning: By self-hosting your data you may avoid the prying eyes of companies out there, but you take on the full responsibility of protecting what you host. It's a constantly changing world out there. New threats come up fairly regularly. It's often safest to leave hosting important data or processes to professionals. But that's also a lot less fun and educational.

Getting your own server

The first step in self-hosting is to get or choose your server (or servers).

Perhaps the easiest way to do this is to pay a Virtual Private Server (VPS) company money. They'll spin up a fresh (virtual) server in one of their data centers and give you login credentials. You can log in from your terminal and begin immediately. Companies like DigitalOcean[1] and Linode[2] will get you up and running in mere minutes starting around $5/month. The big companies (Amazon, Google, Microsoft) offer similar services. In this option, you're still trusting the VPS host with your data but you're much more unique, and there's at least a little security in obscurity. This is a very good option for trying things out in any case.

You could get a server going in your own home. This would involve either configuring an old PC or laptop you have sitting around as a server, buying something like a Dell PowerEdge entry-level server, or cobbling one of your own together by choosing components and installing them in a case. This option is not ideal for hosting things like web pages and e-mail (your home IP address changes frequently, and many ISPs don't allow hosting), but it can be really nice for personal things like file storage, home automation, ad-blocking, or a VPN.

[1] https://www.digitalocean.com/
[2] https://www.linode.com/

Figure1: This is what it looks like to buy components of a computer (minus hard drives) before assembling it. These things effectively snap together but you should enlist the help of a friend who has done something similar before for your first time.

You could also buy a rack-mount server and pay a data center to let you put it in there. That will give you reliable power, cooling, and networking but will be pretty expensive.

Most hard-core of all is to build your own rack and get a business internet connection from your ISP that does explicitly allow hosting. Do this if you want to start your own data center or something.

Once you have a server up and running, you have to choose which services and programs to run on it and then install/configure them.

Well-polished self-hosted catch-all

Before mentioning the low-level self-hosting options below you should know that Nextcloud[3] is a very good option to get started in self-hosting. It can run on your server and will provide personal cloud-based file synchronization (like Dropbox) as well as calendar and contacts. The company behind it offers Nextcloud support to businesses who want to run it on their own servers, but anyone can run a copy of it themselves on their own system, including a Raspberry Pi. Nextcloud was forked from their now-competitor, OwnCloud[4], which offers similar features. If you go this route, you'll find the client programs in your package manager and can easily install them to start syncing.

Self-hosted home automation and security

Computers can do really wonderful things around the home when hooked to various sensors and switches. Home automation is a major industry with major players. I think it's fair to be skeptical of putting control and surveillance of our own home in the hands of other people. Additionally, if some server in Texas goes down for whatever reason, you don't want your house to stop being smart only because it's totally dependent on the cloud! Fortunately, the world of self-hosting for home automation is extremely advanced, active, and welcoming.

Home Assistant[5] is an incredible open-source home automation driver system that was at one point among the top active repositories on GitHub. You can install it on a computer in your home (even on a cheap little Raspberry Pi) and have it do all kinds of things. All data stays right at home. Here are just a few examples of what you can configure it to do:

- Turn your lights on at a certain time and play nice music to help you wake up
- Turn lights on in the living room when you walk in

[3] https://nextcloud.com/
[4] http://owncloud.com/
[5] https://www.home-assistant.io/

- Alert you if it's going to be extra rainy today
- Advise you about your commute options
- Play a Common Loon call[6] the second the Sun sets every day
- Turn your dumb furnace into a smart furnace, heating up every morning before you wake
- Arm a security system when all family members' cell phones have disconnected from the Wi-Fi network (and also at night when everyone goes to bed)
- Disarm a security system when any family member reconnects to the Wi-Fi (so you never have to type a code or anything)
- Trigger an alarm that turns lights on, blares a siren, and e-mails pictures to you from cameras when a door is opened or motion is sensed while armed (after a brief warning period)
- Control your TV, stereo, and air conditioner as you desire (e.g. cool it down when you start your commute on hot days)
- Remind you on certain days of events on your calendar, or when it's garbage day
- Provide gentle beep noises whenever any door is opened or closed when you're home
- Rig up a motion-activated jump-scare on a TV for a Halloween party
- Dim the lights at night, and play some cricket noises for ambiance
- Remind you randomly to do pushups
- Monitor your household energy usage
- Turn a closet light off 20 minutes after it's been turned on
- Turn lights on and off randomly when you're away to confuse would-be burglars

[6] https://www.allaboutbirds.org/guide/Common_Loon/sounds

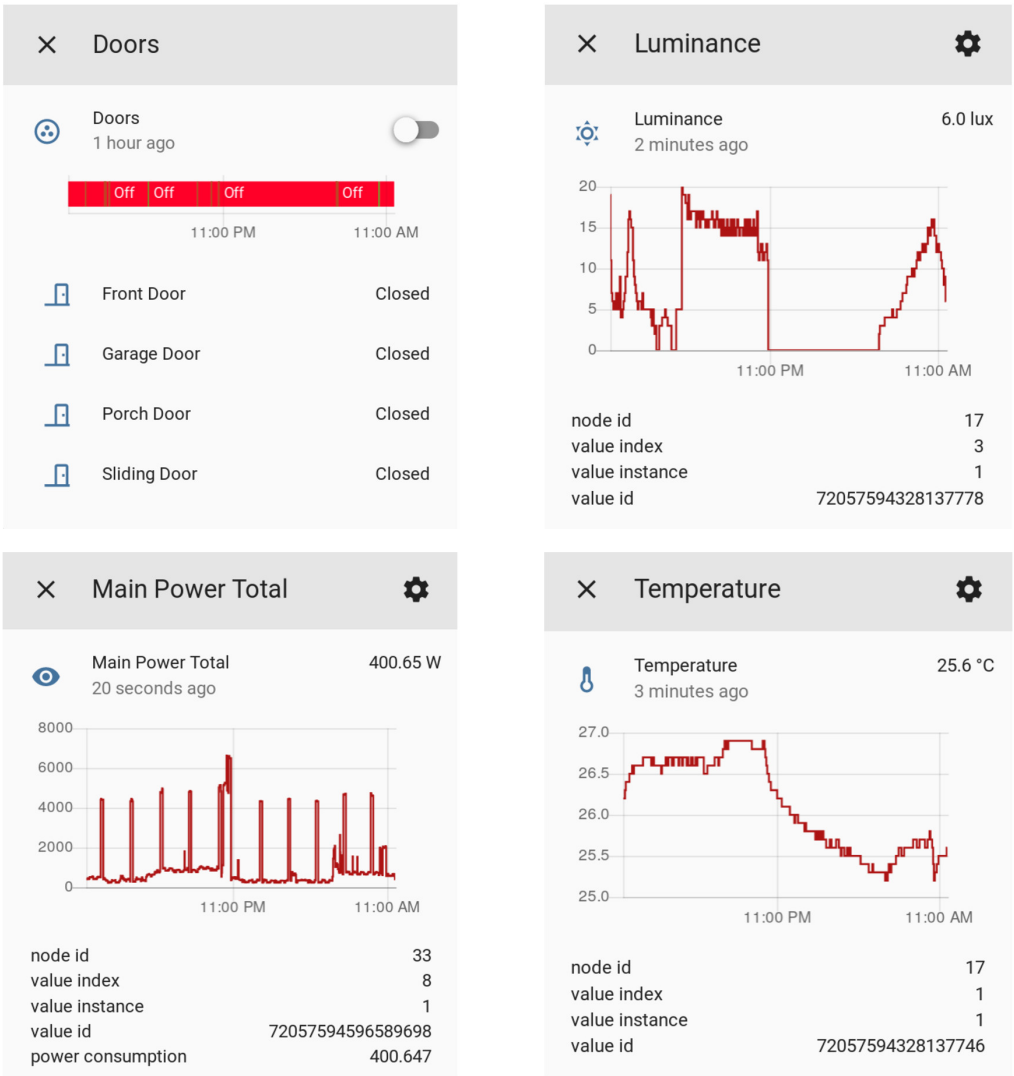


Figure2: A few sensor readings from Home Assistant.

Here are some steps to try out Home Assistant:

- Get a Raspberry Pi kit including a memory card, power supply, and case. It should run you about $50. This is a full-on Linux computer, but tiny and cheap. Once you set it up, install it near a stereo and connect the audio output of the Pi to an audio input of your stereo. This will allow it to alert you of things audibly.
- Get a USB Z-Wave dongle like the **Aeotec Z-Stick Gen5, Z-Wave**

Plus USB. Z-Wave is a wireless technology not unlike Wi-Fi but on a different radio frequency. It's designed for low-bandwidth mesh-networks, meaning a signal can reach the main access point by relaying from device to device if the access point isn't in direct range.

- Get a few Z-Wave door sensors, switches, and motion sensors (as budget allows). The easiest switches to install are basically things you plug into the wall and then plug your lamp (or whatever) into a socket it provides.
- Install Home Assistant. I prefer the "Virtual Environment" installation method even though the "docker" option may be a little simpler. Set up the configuration for Z-Wave.

Figure3: A few good Z-Wave sensors and switches. The plug-in ones are very convenient for adding smart-capabilities to normal things (like Christmas lights), and the battery-powered door and motion/temperature/humidity sensors last far longer than one might expect.

Home Assistant uses text files in the `YAML` format for configuration. The best way to get the hang of it is to look through a lot of the example configurations people have posted online. Then, try to make your own automation. The best part about the community is that you can ask, and

they'll really help guide you through. There's even a chat room for help. Here's one that plays a little beep every time someone opens a door:

```
shell_command:
  door_chime: play -v 0.6 /pool/Systems/AptPi/Videos/beep.mp3

automation:
    - alias: Doorbell
      trigger:
        - platform: state
          entity_id: binary_sensor.sliding_door
          from: 'off'
          to: 'on'
        - platform: state
          entity_id: binary_sensor.front_door
          from: 'off'
          to: 'on'
        - platform: state
          entity_id: binary_sensor.garage_door
          from: 'off'
          to: 'on'
        - platform: state
          entity_id: binary_sensor.porch_door
          from: 'off'
          to: 'on'
      action:
        service: shell_command.door_chime
```

These structured-text input files are yet another example of the utility of pure text files.

Set up your own VPN Server

You may have a workplace that lets you connect to their VPN to get onto the corporate network and access your files. If you're feeling up to it, you can actually set up a personal VPN server for yourself in your home using an open source VPN system called OpenVPN[7]. You can get all the benefits from *Using a VPN Service* for free while also accessing

[7] https://openvpn.net/

your home network from wherever you are, and quite securely! Who can you trust to not snoop on you better than yourself[8]? People who are into security cameras, home automation, and other advanced home networking would like this "road warrior" option. You can even install a VPN server on some of the more powerful home Wi-Fi routers themselves (some even come with a VPN server capability out of the box!).

Setting up a VPN server may be cheapest if you install an after-market *firmware* system on your home router. This isn't particularly easy and can damage your router if you do it wrong, so I can't recommend it unless you have a backup to fiddle with, and are willing to struggle a little while reading forums on advanced troubleshooting. The standout aftermarket router firmware is called OpenWrt[9]. Once you install it on a supported router, setting up the VPN server functionality requires a few steps:

- Generating cryptographic keys for the server and all the clients you wish to connect with (all laptops, phones, etc. you have, and maybe your friends' too if you want to charge them).
- Entering the server keys and settings into the router's configuration (through the router web interface)
- Adjusting the firewall on the router appropriately
- Downloading and installing OpenVPN clients on all your laptops and phones (use the app called *OpenVPN Connect*)
- Copying the cryptographic client keys to each device and importing them into the client
- Keeping track of your home's public IP address, typically by setting up something called *Dynamic DNS*

The tutorials about doing this are very long[10], commiserate with how complex these systems are. Although that particular one is a bit longer than it needs to be (no use of the LuCI user-interface!?). It looks like the internet as a whole is short of a great OpenVPN server tutorial at the moment. Of course, you can also install OpenVPN on any other server,

[8] Of course your ISP can always snoop on your metadata. Unless you use the ultimate in network security: The Onion Router.

[9] https://openwrt.org/docs/guide-quick-start/begin_here

[10] https://oldwiki.archive.openwrt.org/doc/howto/openvpn-streamlined-server-setup

including a VPS. That may honestly be the best bet for privacy, but you'll miss out on accessing your home devices.

Your own webserver

As discussed in *Publishing*, putting web pages online requires server space. Beyond paying someone to configure and run the servers for you, you can also configure and run your own. The `apache` and `nginx` systems are among the most popular for web servers. If you do end up going deep with self-hosting, it's very useful to have one of these systems operating, even if you don't have a public web page to host. Since these systems are the first line of defense for the majority of the internet against attackers, they are made with security in mind. You can hide many of the self-hosted applications discussed here behind these efforts. Something called a *reverse proxy* comes in handy for this, where you run your self-hosted application on your server, but only expose it to the public internet through a specially-guarded pathway going through `apache`.

While many self-hosted applications do come with their own web-server system, it's a simple fact that no matter which one it is, it's not as widespread as `apache`. Thus, `apache` has had more eyes on it, so when vulnerabilities are found, it gets patched very rapidly.

Hosting your own contacts and calendar

One of the major societal advances evident when one loses a phone is that when you get a new one and sign in, your contacts and calendar often are still there. This is because they are "in the cloud" or as we learned earlier "on someone's server somewhere". If you have your own server, you can get the same capability without giving anyone else your info.

The internet standards governing the exchange of contact and calendar information are called CardDAV and CalDAV. Most calendar and contact apps on phones or computers will be able to interact with any server

that speaks these languages. One great open-source CalDAV/CardDAV server is called Radicale[11]. It's written in Python and can run on any kind of machine.

It's nearly trivial to get it up and running at home in a way that will only be accessible from your home network. However, it's even better to have it running on the public internet so that you can update your calendar while you're away from home. To do that requires you to configure a web server and use a reverse proxy as mentioned above. The details are out of the scope of this book, but I encourage you to look into it if it's something you think you're interested in. Personally, I love that kind of thing.

Personal cloud for documents and photos

As mentioned above, Nextcloud is the easiest way to get started with your own cloud file synchronization system.

For all you advanced web server operators out there, Seafile[12] is another excellent option. It has phone and desktop clients for all major systems, and can auto-upload photos taken on your phone.

Self-hosting your own e-mail

No one recommends self-hosting your own e-mail anymore. In fact, most corporate IT departments don't even want to host e-mail; many of them are migrating to cloud-based services maintained by a few major companies. However, for those of us who really want to achieve the ultimate badge of honor in self-hosting (and the associated inability to get your e-mails through to a bunch of people), just know that it is certainly possible. `Dovecot` and `postfix` are the server components, and `spamassassin` with Bayesian filtering is surprisingly effective against spam. As you configure your server, send a test e-mail to services such as https://mail-tester.com, which will analyse it and produce a report

[11] https://radicale.org/

[12] https://www.seafile.com

telling you of any server configuration issues you still have to work out. Once you get a 10/10 score, you're good to go.

That wraps up this bonus chapter. I hope this brief tour will inspire the brave and curious among you to investigate the fascinating world of self-hosting.

10

Conclusion

From slinging PDFs to DJing a dance to making the next great web app, the tools and practices you've now experienced are true digital superpowers. You have completed the first step to becoming a digital superhero (or mastermind, or wizard, or whatever form of expert you aspire to become). While you now know some of these tools as well as you ever will need, mastery in all of them is readily attainable through additional practice and effort. You will find that mastery will come naturally as you reach for these tools again and again. You'll start out using one little feature (perhaps the one you learned in this book), and you'll wonder if there's a tweak that gives you a slightly different behavior. You'll build skill upon skill this way, personally crafting your superpowers to meet your specific needs, interests, and dreams.

Go forth and fulfill those dreams! Good luck.

11

Glossary

2-factor authentication
: 2FA is a means for a computer user to identify themself and authenticate using both something they know (usually a password) with something they have (often a phone or keychain with changing digits on it).

Bit
: A single one or zero; the fundamental digit in the binary number system.

Byte
: Eight bits together, representing a number between 0 and $2^8 - 1$ or 255

Bitmap
: Computer graphics represented as a connected grid of pixels of different colors that make up an image. Used often to represent photographs on computers.

Cascading Style Sheets
: A system of text files that define the look and feel of a web page separately from the files and/or data that make up the content of the page.

CMS
: A content management system is Web-based software that assists in building and maintaining websites in user-friendly ways.

Cloud
: Always-available computer servers running on the internet. Often used as an alternate place to store files or e-mails rather than on a local hard disk drive.

Command line
: The interface on many computer systems that allows the user to enter text commands to be processed. Infamous for looking hacky but actually being extremely expressive and useful.

Central Processing Unit
: The CPU is the "brain" of a computer, it's a microprocessor that crunches all the numbers. Made of billions and billions of little transistors etched onto a silicon wafer.

Dark web
: Any part of the internet that isn't readily available for browsing by the public. Technically includes your friends' e-mails since not everyone is authorized to see them. Typically used in practice to refer to the black market networks running alongside more legitimate services in The Onion Router.

Database
: A computer system designed to receive, store, and retrieve large amounts of information from multiple concurrent other computer systems.

DNS
: The Domain Name System is a "phonebook" for computer networks that maps readable names like `www.webpage.com` to computer-usable IP addresses like `255.255.255.20`.

EFF
: Electronic Frontier Foundation[1] is a nonprofit defending digital privacy, free speech, and innovation.

Encryption
: The process of hiding information from prying eyes on network or elsewhere using math. Can also validate that information was sent unchanged from a particular individual.

GNU

[1] https://www.eff.org/

Gnu's Not Unix (GNU) (recursive acronym) is a free-software collaboration project that started in 1983 and continues to this day. Many free, open-source tools are developed under the GNU umbrella.

GUI
A Graphical User Interface uses a monitor and a mouse or touchscreen to provide a visual-based experience in using a computer program rather than a purely text-driven environment. Windows and macOS are GUI OSs, while DOS was not (if you recall DOS).

Hard drive
The device in a computer that stores information like documents, photos, programs, and so on for long periods of time. Contrast with RAM (see Memory).

IP address
An Internet Protocol address defines where a computer is on a computer network, similar to a street address in real life. It is numerical, either four numbers between 0 and 255 separated by periods (IPv4) or eight groups of four hexadecimal digits separated by colons (IPv6). All domain names and other human-readable network locations are mapped to IP addresses via DNS.

ISP
An Internet Service Provider is a company you pay to deliver a connection to the public internet. Examples include Comcast, CenturyLink, Charter, etc.

LaTeX
A document typesetting and publishing system that separates content from presentation.

Lightweight markup language
A markup language with an intentionally-simple syntax, often to accommodate rapid creation by a human writer.

Linear regression
The process of finding the equation of a line that best fits a given set of data points.

Linux
An open-source operating system that powers many servers that make up the internet, all Android smart-phones, all of the top 500

largest supercomputers on Earth, and a small but growing number of PCs.

macOS
: A proprietary operating system made by Apple and used on Apple computers and iPhones.

Markup language
: A way to express something to a computer using a text file with special text-based annotations. For instance, while italics cannot be represented directly in a pure text file, a word could be indicated as italicized by surrounding it in in `<i>` tags. Markup languages are used often for rich data exchange (e.g. `HTML` and `XML`).

Memory
: Hardware that can store large chunks of information on a computer for some period of time. Memory is typically used for Random Access Memory (RAM), which is a short-lived fast-acting memory. There's also hard-drive memory which is slower, larger, and cheaper.

Network Attached Storage
: A NAS is a bank of large hard drives that are made available on a network, often used for backups of photos and videos.

Noise
: Random information present on any analog signal such as the hiss on the radio or the snow on old TVs

Onion router
: A web-browsing technology that offers increased anonymity and security over normal browsing. It hides the web pages you're browsing from the infrastructure and hides your place of origin from the web pages you browse.

Operating system
: An OS is a set of software that runs a computer. It interacts with all hardware and provides the environment in which other programs run. Examples are Windows, macOS, and Linux.

Package manager
: A program that can go out on the internet and find a different program, download it, install it, configure it, and (later, if desired)

uninstall it. These pre-date *app stores* but are very similar in concept. These allow for easy program management, often though at least nominally more secure channels than random downloads. Each OS has its own package managers.

Password manager
: A program or service that stores various passwords for various services in one place. It also has facilities for generating random passwords to avoid the problems of password-reuse.

Programming
: Writing a sequence of operations for a computer to perform. The process of creating software.

Regular expressions
: A general pattern representing some combination of letters, numbers, punctuation, and other characters that is used for advanced searching and replacing in text and data processing.

Router
: A network component that receives packets and forwards them to a client or other network that it knows about but perhaps the sender does not (and vice versa).

Secure Shell (SSH)
: A means for accessing the command line of a computer that's connected to you by a network that maintains strong security along the network as commands and responses are passed back and forth.

Static site generator
: A program or system that converts lightweight markdown language code and images into a full HTML website. It's effectively a lightweight CMS.

Text editor
: A program that views and modifies text files.

Text file
: A universal file that only contains sequences of characters without any embedded information about font, style, etc. These are very useful and fundamental as an interface file between various steps of data processing, publishing, programming, etc.

Top-level domain

A TLD is the suffix of a domain name like `.com` or `.edu` or even `.horse`.

Vector graphics
: Computer graphics defined by mathematical equations rather than grids of pixels. For certain types of images (e.g. icons, logos) they are nice because they use very few bits but can be zoomed in on infinitely. Contrast with Bitmaps.

Virtual machine
: A computer program that emulates the hardware of a different computer system.

VPN
: A Virtual Private Network is a technology that securely connects through the public internet to servers you are specially authorized on. Used frequently for businesses, allowing employees to connect to corporate intranet resources like shared drives while away on travel. Also offered as a commercial service to help add network security and privacy for consumers.

VST
: Virtual Studio Technology[2] is a specification that defines how a large set of software *plugins* work together while composing music on a computerized studio.

Windows
: A proprietary operating system made by Microsoft that runs on most consumer and office PCs.

[2] https://en.wikipedia.org/wiki/Virtual_Studio_Technology

Index

Non-alphabetical

A

B

C

D

E

O

P

Q

R

S

T

U

V

W