The Python Book

Python for Information Systems Majors

James Davis, PhD

Contents

Preface

Admittedly, this book represents my opinionated technology stack for developing applications. - Dr. Davis

This book is intended to give Information Systems students a foundational understanding of programming and application development. To do so, we will need to use a programming language to better illustrate logic, data structures, and other important concepts. While there are multiple options, to me, we need a language that is robust, versatile, popular, and free. There are other languages, such as Ruby, that meet these requirements but the rising popularity of Python plus hundreds of readily available packages gives Python an edge above the competition.

A simple search of ‘python popularity’ will return pages of articles, blogs, and information about The Incredible Growth of Python.

As a comment on the versatility of Python, it comes preloaded on many systems from the Raspberry Pi to the MacBook Pro. It is popular within the scientific community, tech startups and the top software companies including:

- Google
- Instagram
- Spotify
- Quora
- Netflix

- Dropbox
- Reddit

Chapter 1

Technology Stack

A technology stack is the combination of all the products and programming languages used to create an application. For many applications, the 'stack' used in its development might include:

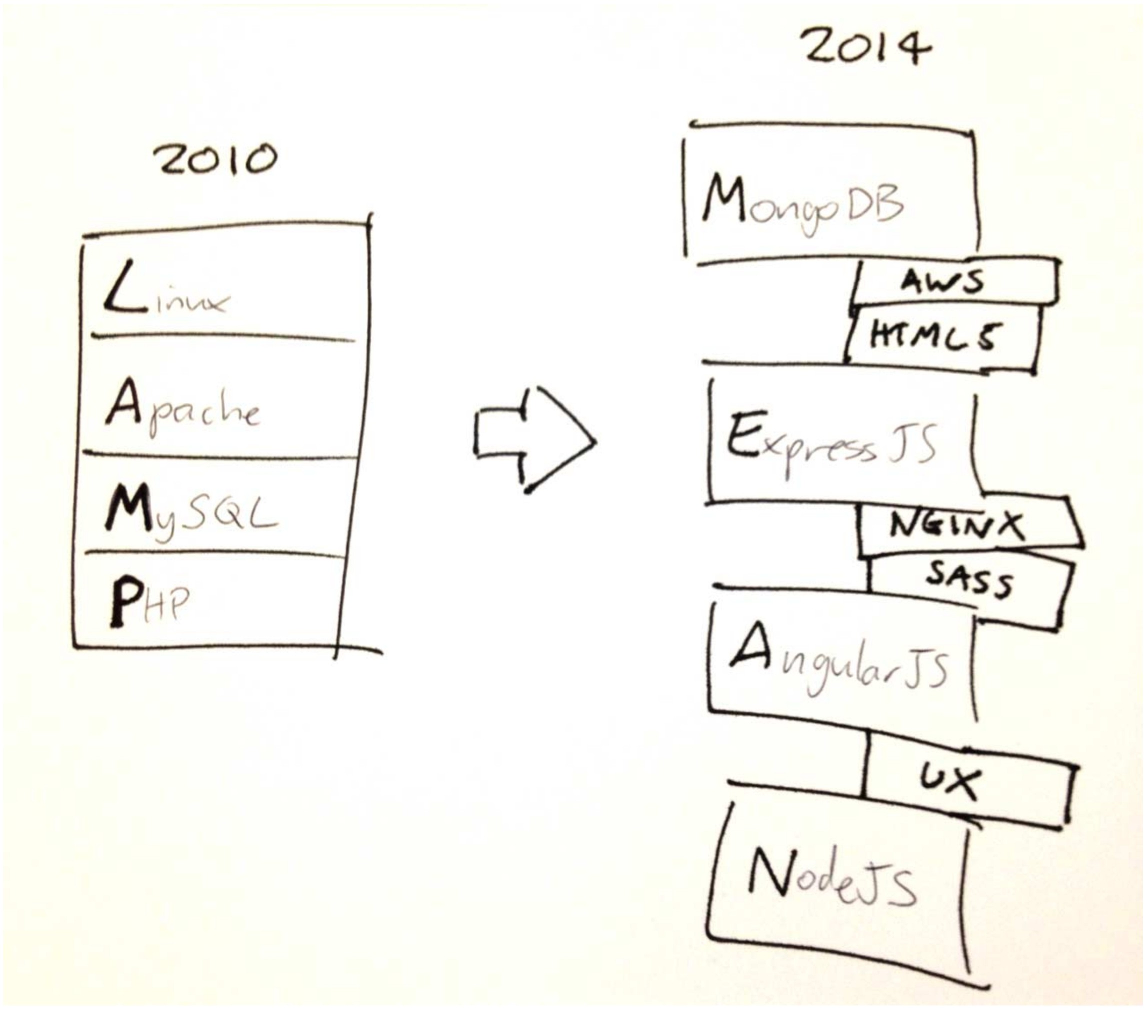
- Database management system (DBMS)
- Language(s) for backend logic
- Language(s) for frontend logic or presentation
- Web server
- Operating system

1.1 Full Stack Development

You may have also heard the term 'full-stack developer'. A full-stack developer is knowledgable in all aspects of software development from concept to a finished product. The key here is 'knowledgable'. It is acceptable that you might not know the answer to all technical questions but you should be able to find answers in a timely manner.

The opposite of a full-stack developer would be someone with a very specialized and limited skillset. A database administrator (DBA) will have a very deep understanding of his/her particular database management system (DBMS) along with the concept of table spaces, triggers, SQL, etc. but probably has a minimal understanding of UI/UX, responsive frontends, or javascript.

Over the years, the technology stack a developer was required to know has gotten more complex.



Other important skills for a full-stack developer to know:

- Server, network, and hosting environment
- Relational and non-relational databases

- How to interact with APIs and the external world
- User interface and user experience
- Quality assurance
- Security concerns throughout the program
- Understanding customer and business needs
- Version control

1.2 Demystifying Stacks

Geek speak, or technical jargon, is the terms, phrases, and expressions that the members of the technology community use in their, or if you identify as a techie, ‘our’, communication.

To the outsider or someone new to community, this jargon can be confusing, intimidating, and possibly a barrier. I’ve found comparing technology stacks to a bento box helps to remove some of the mystery of our field.

Box 1.1. Bento

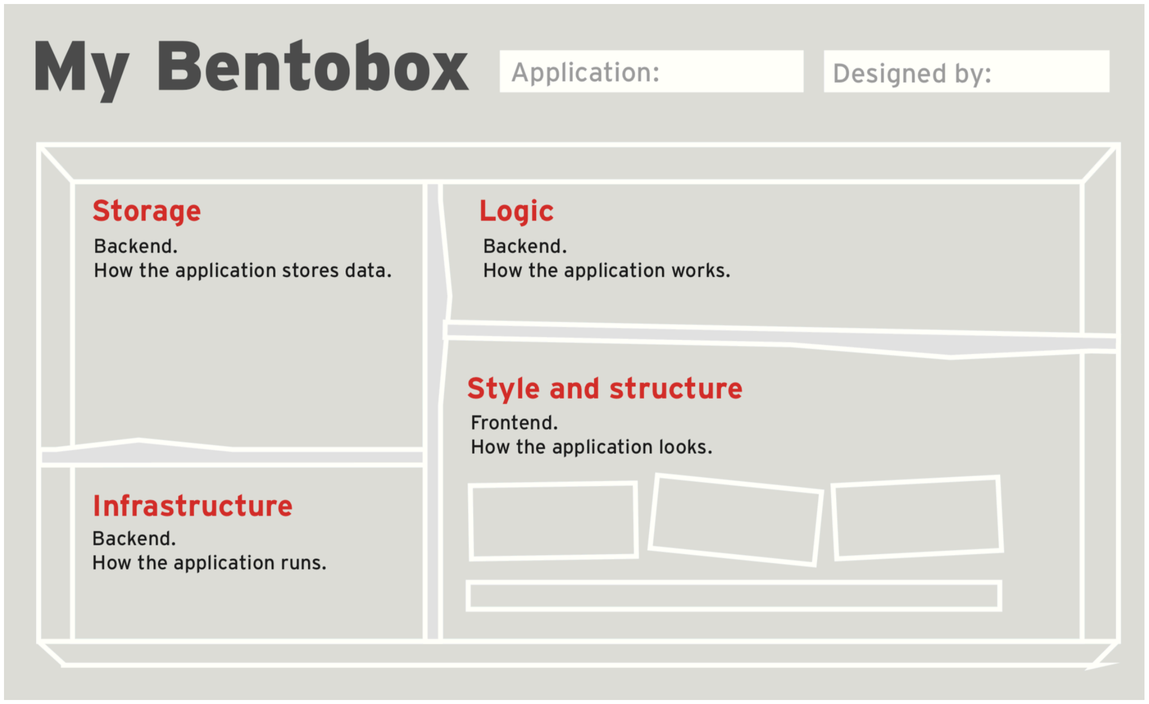
Bento is a popular box meal common in Japanese cuisine. It brings together rice or noodles, fish or meat, with pickled and cooked vegetables, in a yummy box.

You need a balanced mix of things. Its a puzzle - putting everything together in the box. Ekiben - content which is arranged in the most efficient, graceful manner. The bento is presented in a simple, beautiful, balanced way. Nothing lacking. Nothing superfluous. Not decorated, but wonderfully designed.

Likewise, we can think of the technology stack as a bento consisting of our protein, starch, veggies, and lagniappe[1]. Our technology bento will be used to

[1]The word comes from the Louisiana French referring to the extra items.

classify the technology as storage, infrastructure, logic, and style/structure.



1.3 What I use for development

The Golf Channel has a segment called "What's in the bag?" where they analyze the clubs used by a particular professional golfer. During the course of this book, I'm going to give you my "What's in the bag?" for developing a software product. Like pro golfers, my selections change as the technology changes. Here is what I'm currently using for development...to include writing this book!

1.3.1 My Machine

Most developers I know working at startups or active in the open source community use Mac computers [2]. Those not using a Mac tend to select some flavor

[2]This is completely my observation with no quantitative support.

of Linux (with Ubuntu being a personal favorite). I was a long term ThinkPad owner (dual booting between Windows & Ubuntu) but made the switch to a MacBook in 2013 and haven't looked back. When working in software development, you will eventually want to create a mobile app. The only way to create and deploy an app for the iOS is by using a Mac. With my MacBook Pro, I can do web development, android development, and ios development.

1.3.2 Text Editor

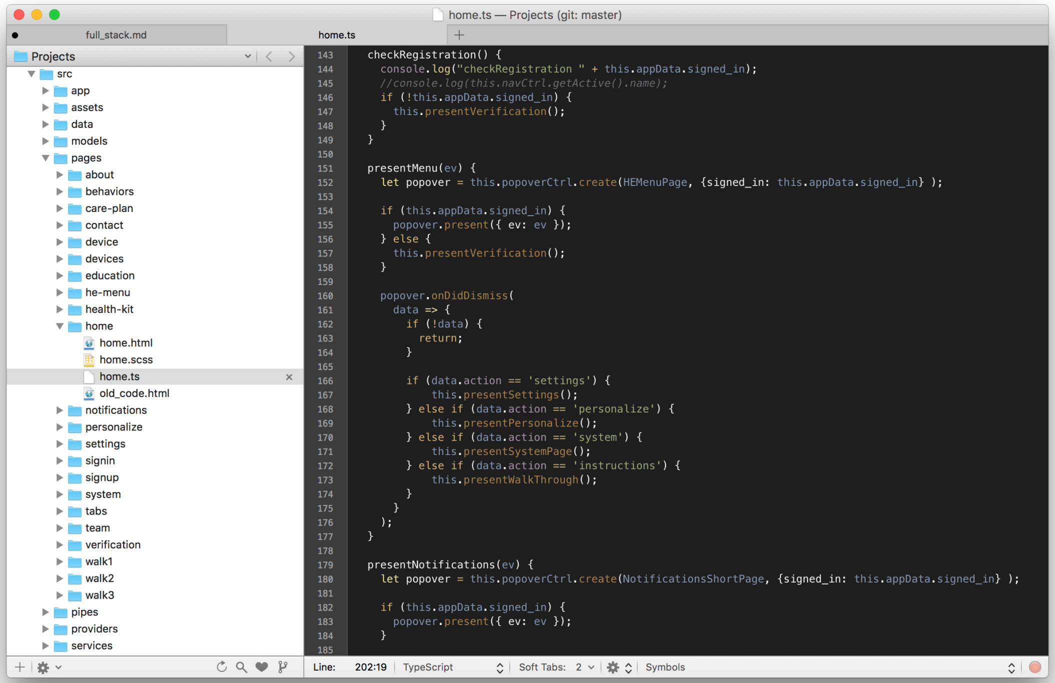
Yes, a text editor is exactly what it sounds like. It edits text. In coding, the term 'editor' has a slightly different meaning. Used in software development, it is an application used for editing your code files. Most web development languages are interpreted, not compiled, so there isn't a need for a robust integrated development environment (IDE). For the past 5 years, I have been using TextMate as my coding editor.

You might be wondering why would you need a special code editor, rather than using something like Word or Notepad.

The main reason is that code needs to be plain text, and the problem with programs like Word is that they don't actually produce plain text, they produce rich text (with fonts and formatting) along with other meta data created as part of the file.

Another reason is that code editors are specialized for editing code, so they can provide helpful features like highlighting code with color according to its meaning, or automatically closing tags for you.

I've found most developers become very passionate about their particular editor. Soon, you'll come to think of your trusty code editor as one of your favorite tools.



```
  checkRegistration() {
    console.log("checkRegistration " + this.appData.signed_in);
    //console.log(this.navCtrl.getActive().name);
    if (!this.appData.signed_in) {
      this.presentVerification();
    }
  }

  presentMenu(ev) {
    let popover = this.popoverCtrl.create(HEMenuPage, {signed_in: this.appData.signed_in} );

    if (this.appData.signed_in) {
      popover.present({ ev: ev });
    } else {
      this.presentVerification();
    }

    popover.onDidDismiss(
      data => {
        if (!data) {
          return;
        }

        if (data.action == 'settings') {
            this.presentSettings();
        } else if (data.action == 'personalize') {
            this.presentPersonalize();
        } else if (data.action == 'system') {
            this.presentSystemPage();
        } else if (data.action == 'instructions') {
            this.presentWalkThrough();
        }
      }
    );
  }

  presentNotifications(ev) {
    let popover = this.popoverCtrl.create(NotificationsShortPage, {signed_in: this.appData.signed_in} );

    if (this.appData.signed_in) {
      popover.present({ ev: ev });
    }

```

VS Code (recommended)

Visual Studio Code is a free editor from Microsoft. It is popular in the .NET community but a little too heavy for my needs. If plan on using any of the Visual Studio products or .NET in the future, I would recommend VS Code.

Atom (recommended)

Atom is a code editor created by GitHub. It's free, open-source and available for Windows, OS X and Linux. If you are not already attached to an editor, I would recommend Atom.

Eclipse IDE

Eclipse is an integrated development environment (IDE) which is much more than a code editor. IDEs are typically associated with compiled languages (such as Java) but can still be used with other languages. I have used Eclipse in the past but it is very heavy in terms of application size.

Box 1.2. Who cares what I think?

Who is this guy and why should you care what I think about development? If you are one of my students, you are a captive audience and your grade is an incentive to care. Otherwise, here is some more information about me. I haven't always been an academic and these are the cliff notes. I've been writing code since the mid-1980s. As a kid, my parents bought me a Commodore Vic-20 and I would write games for me and my friends. In college, I majored in math while taking programming classes in Fortran, Pascal, COBOL, and C. There was a 4ish year hiatus from coding because I used the army to help pay for college (whole different story for a completely different book) but quickly returned to software development after my time in the military. While working on an MS in Computer Science, I paid the bills by developing software for the healthcare industry. Not knowing when to quit, I completed the MS and picked up a PhD program in Information Systems.

Somewhere in graduate school things got a bit more interesting. I started working with some fellow researchers on social networking algorithms and knowledge networks. Long story made short, we found success in a startup venture developing an application to track the knowledge networks of organizations. We had a good exit, I did some more army stuff, worked as an enterprise architect for a bit, and then returned to academics while still partaking in the occasional side venture.

LinkedIn

Chapter 2

Version Control, Git and GitHub

Version control is a system that records changes to a file or group of files. It allows developers to maintain a log of code changes with minimal effort. More importantly, it allows multiple developers to work on the same project at the same time and provides a robust method for sharing & merging work.

2.1 Not having version control

How many versions of code can exist? Even on a small project there could be dozens of versions. Imagine being in an IT Department with a desktop computer at a "work" location and another at home. Most of us want continue working in the evenings so you put the code for a project on a USB drive and bring it home. At home, you put the updated code on another computer and do some additional development. At this point, there are two versions of the code at three locations (work computer, thumb drive, & home computer). As this process continues and you rotate through multiple USB drives, your code can get disjointed and you will find yourself spending time & effort researching file modification dates to realign code. This complexity would increase exponentially when working with multiple developers.

As projects scale, your team will likely require testing, staging, and production environments each built from a similar code base at different time periods. Maintaining and cascading changes would be problematic even in the smallest of development teams.

2.2 Advantages of version control

A robust version control system, such as git, supports multiple development versions (one for each developer) along with as many staging, testing, and production environments that are desired. It also provides the foundation for other agile software development practices such as test driven development (TDD), continuous integration, and continuous deployment.

2.3 Git

Git is the most commonly used version control platform. Its amazingly fast, its very efficient with large projects, and it has an incredible branching system for non-linear development.

2.3.1 Installing Git

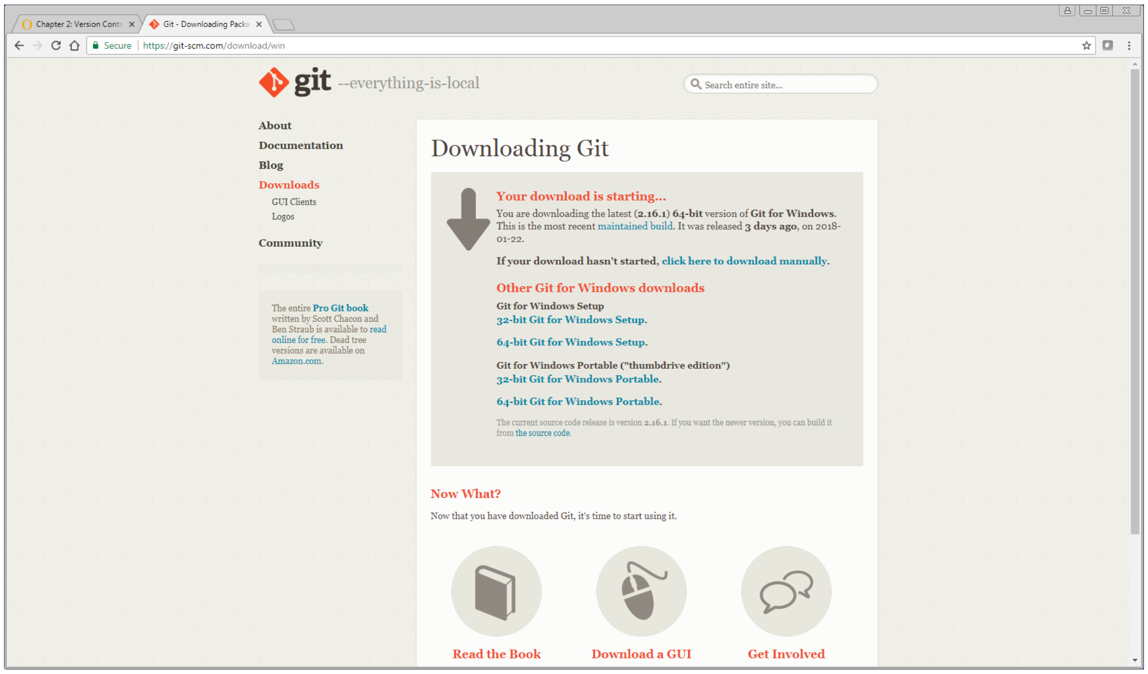
If you have done some coding in the past, you may already have git on your system. Git is already packaged with Xcode and some other development environments.

```
$ git --version
```

If you don't have a version, git is relatively easy to install.

Install on Windows

The recommended method for utilizing git on Windows is through an installer available at the Git website, git-scm.com.



The entire **Pro Git book** written by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on Amazon.com.

Downloading Git

Your download is starting...

You are downloading the latest (**2.16.1**) **64-bit** version of **Git for Windows**. This is the most recent maintained build. It was released **3 days ago**, on 2018-01-22.

If your download hasn't started, click here to download manually.

Other Git for Windows downloads

Git for Windows Setup
32-bit Git for Windows Setup.

64-bit Git for Windows Setup.

Git for Windows Portable ("thumbdrive edition")
32-bit Git for Windows Portable.

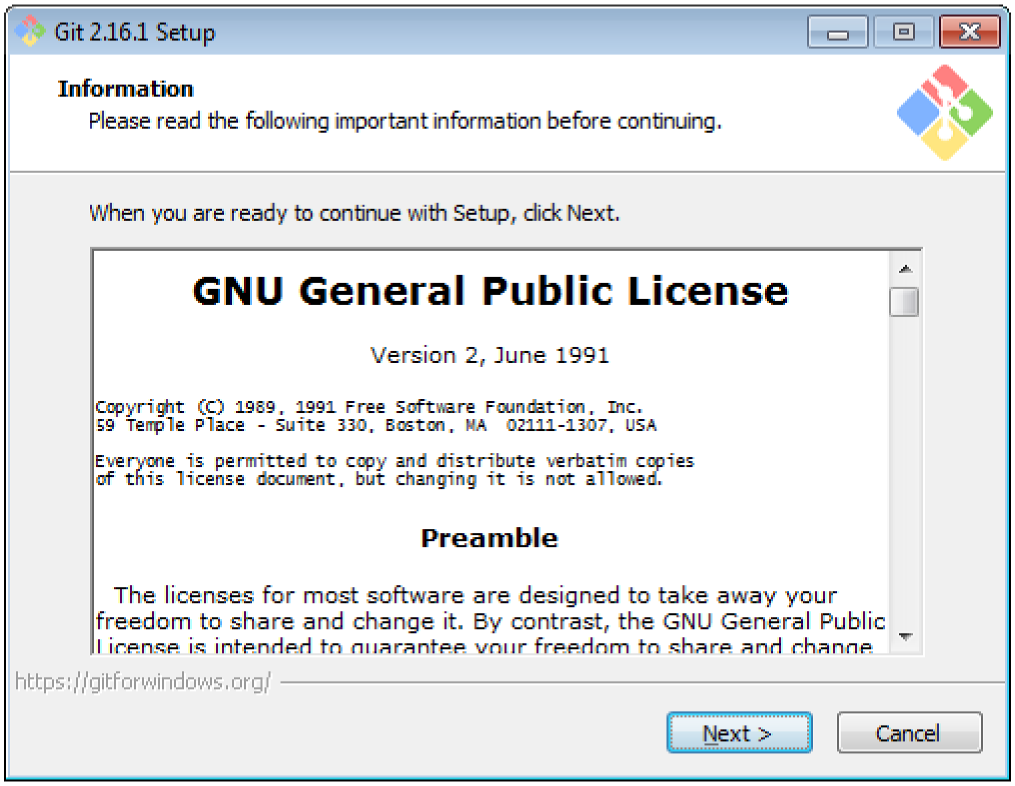
64-bit Git for Windows Portable.

The current source code release is version **2.16.1**. If you want the newer version, you can build it from the source code.

Now What?

Now that you have downloaded Git, it's time to start using it.

Start the setup installer. You should accept the default settings during the installation with a few exceptions noted below.



Information

Please read the following important information before continuing.

When you are ready to continue with Setup, click Next.

GNU General Public License

Version 2, June 1991

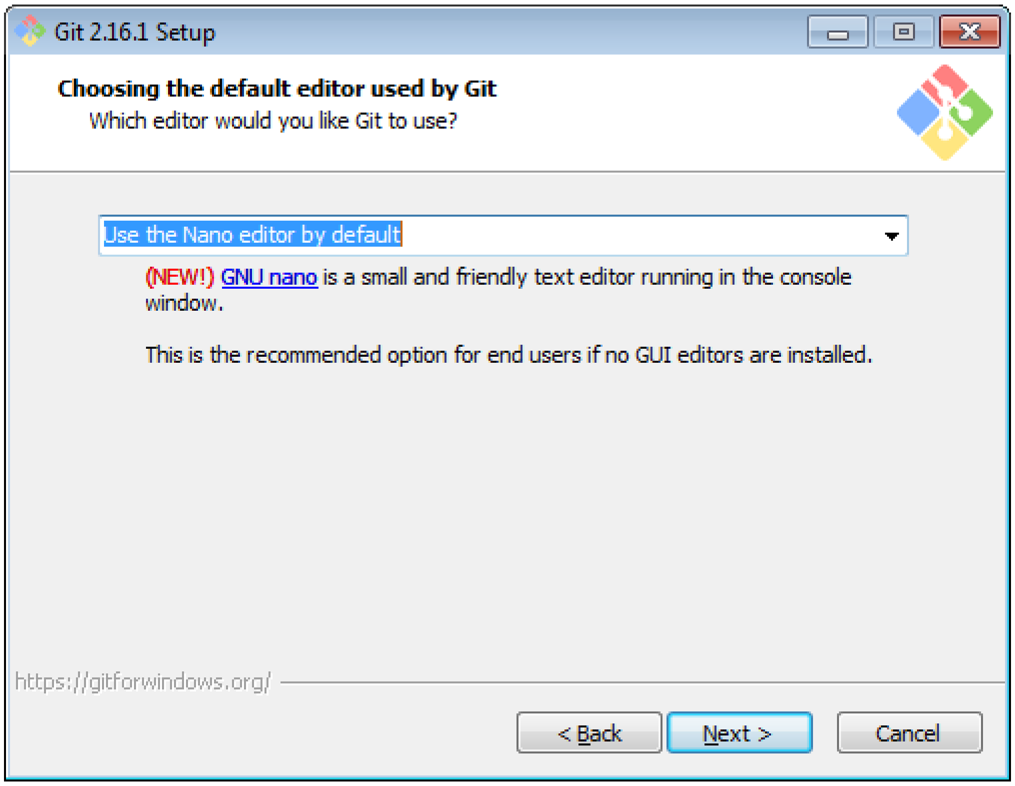
```
Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place - Suite 330, Boston, MA  02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.
```

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change

https://gitforwindows.org/

I would suggest changing the default editor to something other than VIM.


Git 2.16.1 Setup

Choosing the default editor used by Git
Which editor would you like Git to use?

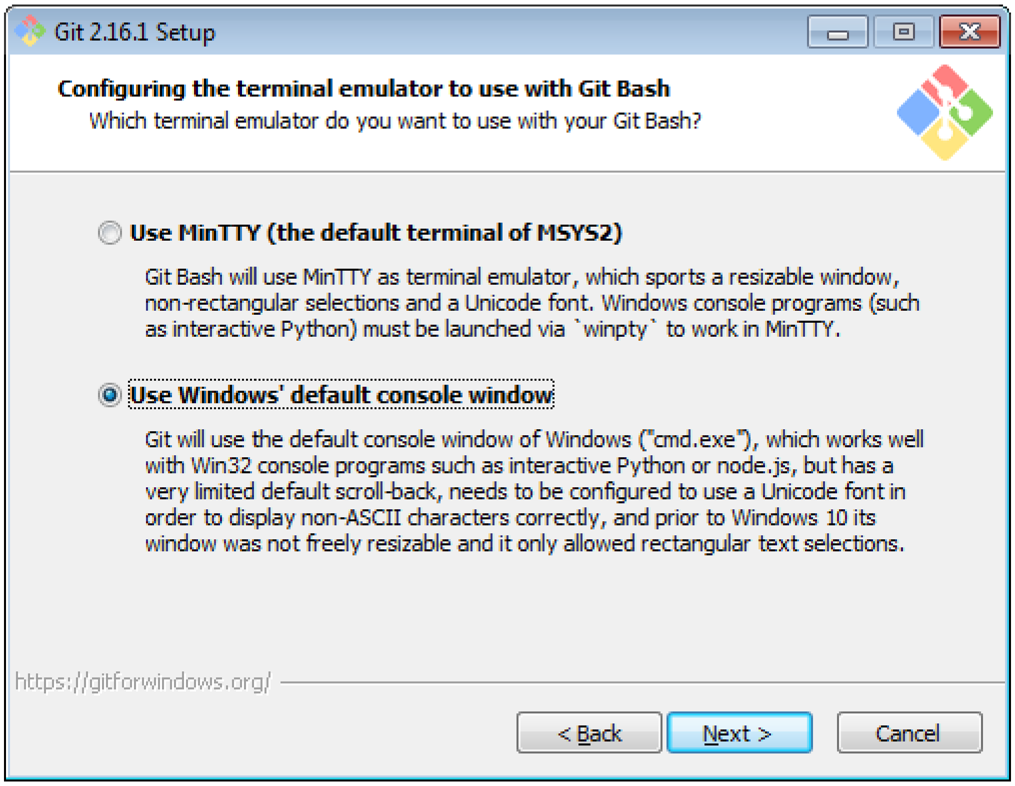
Use the Nano editor by default

(NEW!) GNU nano is a small and friendly text editor running in the console window.

This is the recommended option for end users if no GUI editors are installed.

https://gitforwindows.org/

Select 'Use Windows default console window'. This is important to properly launch interactive Python in later chapters of the book.



Configuring the terminal emulator to use with Git Bash
Which terminal emulator do you want to use with your Git Bash?

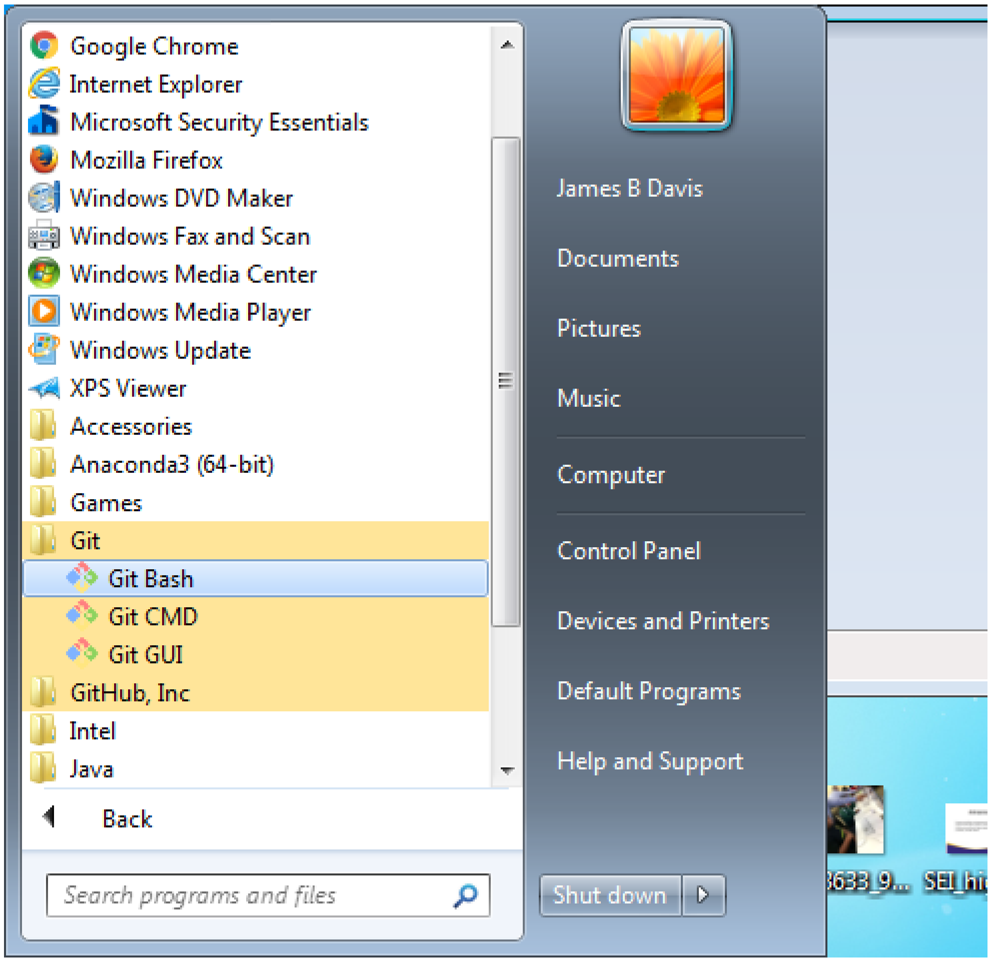
( ) **Use MinTTY (the default terminal of MSYS2)**

Git Bash will use MinTTY as terminal emulator, which sports a resizable window, non-rectangular selections and a Unicode font. Windows console programs (such as interactive Python) must be launched via `winpty` to work in MinTTY.

[✓ **Use Windows' default console window**]

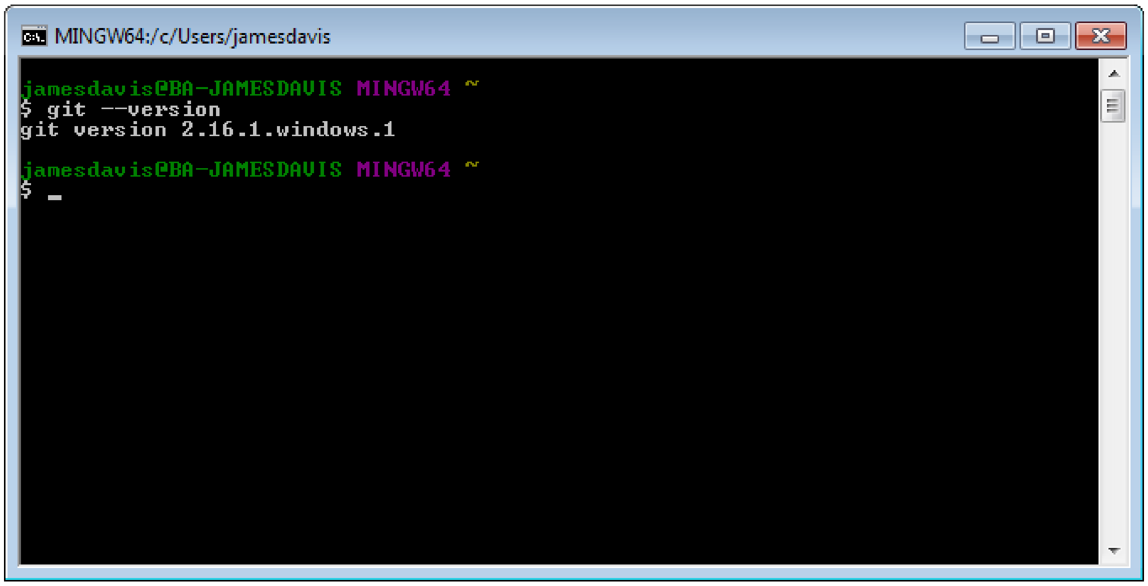
Git will use the default console window of Windows ("cmd.exe"), which works well with Win32 console programs such as interactive Python or node.js, but has a very limited default scroll-back, needs to be configured to use a Unicode font in order to display non-ASCII characters correctly, and prior to Windows 10 its window was not freely resizable and it only allowed rectangular text selections.

After installation is complete, you can interact with git by selecting ‘Git Bash’ in your windows menu.



This bash prompt will provide the command line interface for git commands. Type ‘git –version’ to confirm git is intalled.

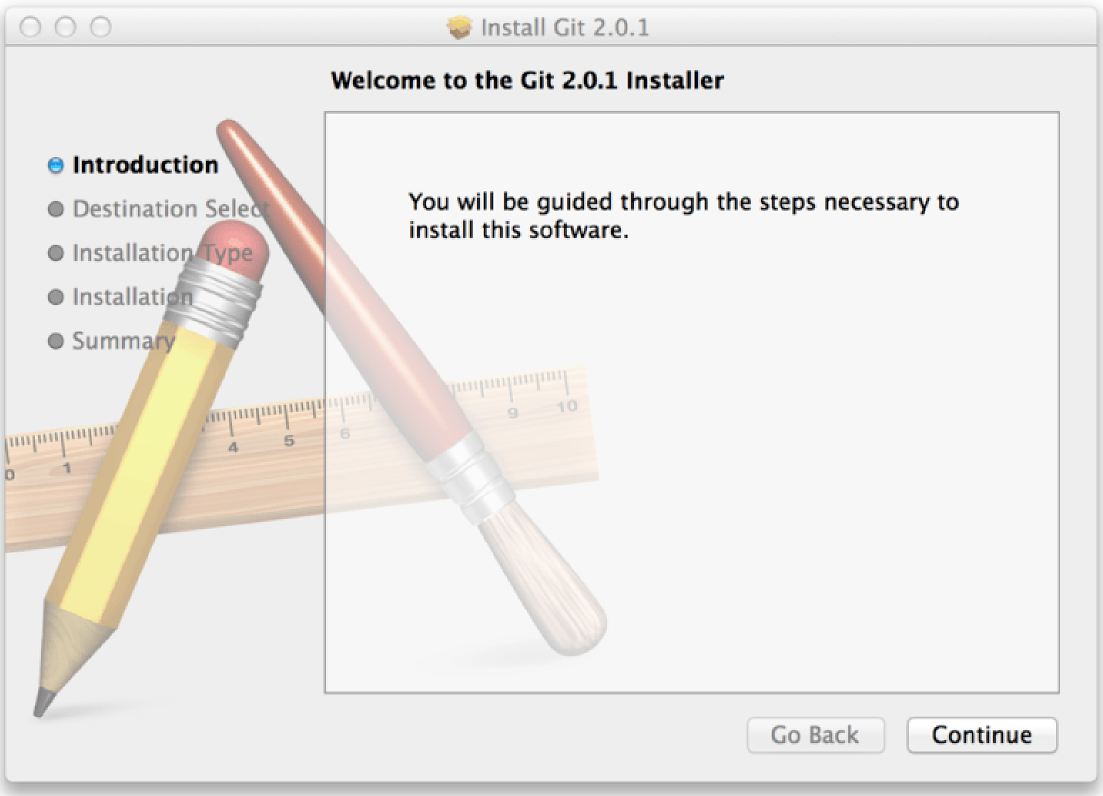
```
$ git --version
git version 2.16.1.windows.1
```



```
jamesdavis@BA-JAMESDAVIS MINGW64 ~
$ git --version
git version 2.16.1.windows.1

jamesdavis@BA-JAMESDAVIS MINGW64 ~
$ _
```

Install on Mac

Typing the ‘git –version’ command above should prompt you to install it. If not, a macOS Git installer is maintained and available for download at the Git website, at git-scm.com.



You will be guided through the steps necessary to install this software.

Install on Ubuntu

```
$ sudo apt-get install git-all
```

For other flavors of Linux, checkout this website for more information git-scm.com.

2.3.2 Git basics

After you install Git, set your name and password. Git uses this information to track changes and it is immutably baked into your committed code.

```
$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com
```

Note: Use your name and email address for the above and not "John Doe".

You can confirm your Git configuration with the list tag.

```
$ git config --list

credential.helper=osxkeychain
user.name=John Doe
user.email=johndoe@example.com
core.editor=mate -w
core.repositoryformatversion=0
core.filemode=true
core.bare=false
core.logallrefupdates=true
core.ignorecase=true
core.precomposeunicode=true
remote.origin.url=git@github.com:johndoe/test.git
remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*
branch.master.remote=origin
branch.master.merge=refs/heads/master
```

Now that you have Git properly setup, change to a directory where you want to utilize version control. For me, I my coding projects in a 'Projects' folder.

```
$ cd Projects/mysite
```

This particular folder contains a basic Django project but you can use any project you may want to add version control.

```
$ ls
blog                db.sqlite3        manage.py        mysite                notes.txt
```

To setup this project with Git, I run the init command.

```
$ git init
```

Which returns something like this.

```
Initialized empty Git repository in /Users/james/Projects/mysite/.git/
```

The ‘git init’ command created a new subdirectory named .git that contains the skeleton files fora Git repository. At this point, nothing in your project is tracked, yet. . . .

To add files to the Git repository, you can type ‘git add .’ which will add all the files or you can add the files by name.

```
$ git add .
```

Files have been added to Git but not commited. To finish the commit process, add the commit command.

```
$ git commit -m "initial commit for the mysite project"
```

If you are ever unsure about the status of your Git repository, you can run the status command.

```
$ git status
On branch master
nothing to commit, working tree clean
```

Excellent! You can now create Git repositories and commit code. Next we will learn how to integrate with a remote repository.

2.4 GitHub

GitHub is a cloud hosting service for git repositories. It is mostly used for computer code although it can be used for other types of version control. **Note: this book is maintained using Git & GitHub as well as the code for the hosting platform, Softcover.** GitHub offers all of the distributed version control and source code management functionality of Git as well as adding its own features.

Signing up for a GitHub account is straight forward, easy, and free. After you get an account I would envite you to participate in one of their tutorials such as GitHub Hello World. The Hello World tutorial covers some basics about how to create a repository and branching. We will cover more about cloning, branching, pull requests, merging, and other Git commands later in this book.

2.4.1 Authenticating with GitHub

When you connect to a GitHub repository from Git, you'll need to authenticate with GitHub using either HTTPS or SSH...***with HTTPS being the recommended method***.

Connecting over HTTPS (recommended)

If you clone with HTTPS, you can cache your GitHub password in Git using a credential helper.

Connecting over SSH

If you clone with SSH, you must generate SSH keys on each computer you use to push or pull from GitHub.

2.4.2 Celebrate

Congratulations, you now have Git and GitHub all set up! You can also checkout the Git Cheatsheet and GitHub Flow documents. Don't worry about mastering the Git commands and flow of versioning control. It will start meaning more to you later.

Chapter 3

Build and Host a Static Website

Everybody has to start somewhere. . . so let's start with a static website!

3.1 Static vs Dynamic Websites

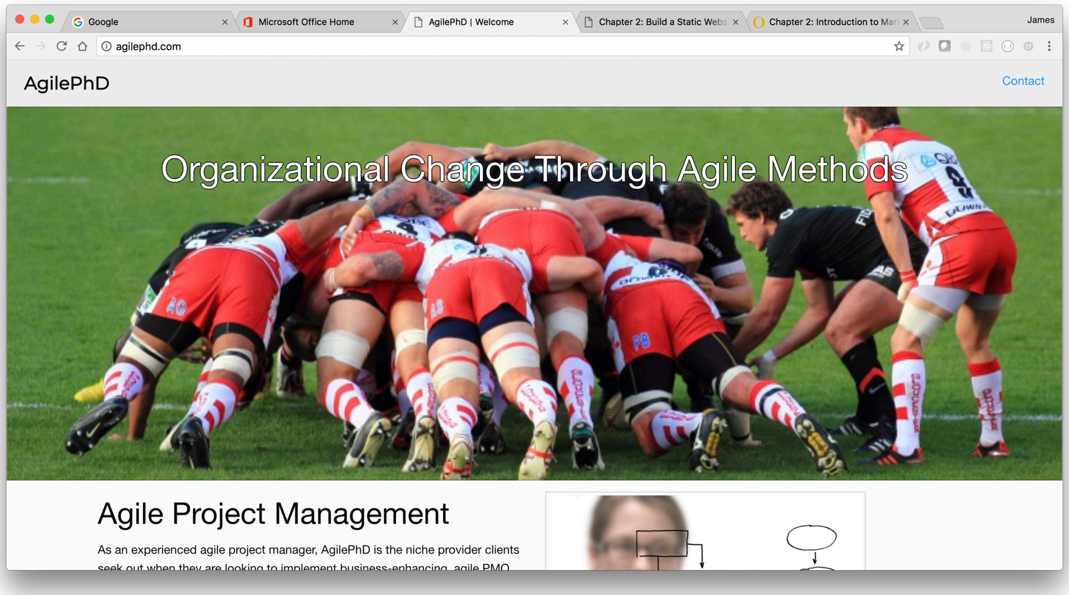
Static means the pages delivered to the client are the same for everyone. Dynamic is the opposite and each client may receive different content. In general, web applications are dynamic because users are authenticated against a database and receive content based on their account, privileges, and/or roles. A website advertising the hours for a local hardware store could be static in that every one visiting that website sees the same information. Static web servers give everyone the same html file. Dynamic servers involve logic, database connections, and provide changing content by building the html file 'on the fly'. . . aka dynamically.

3.2 Our First Gig

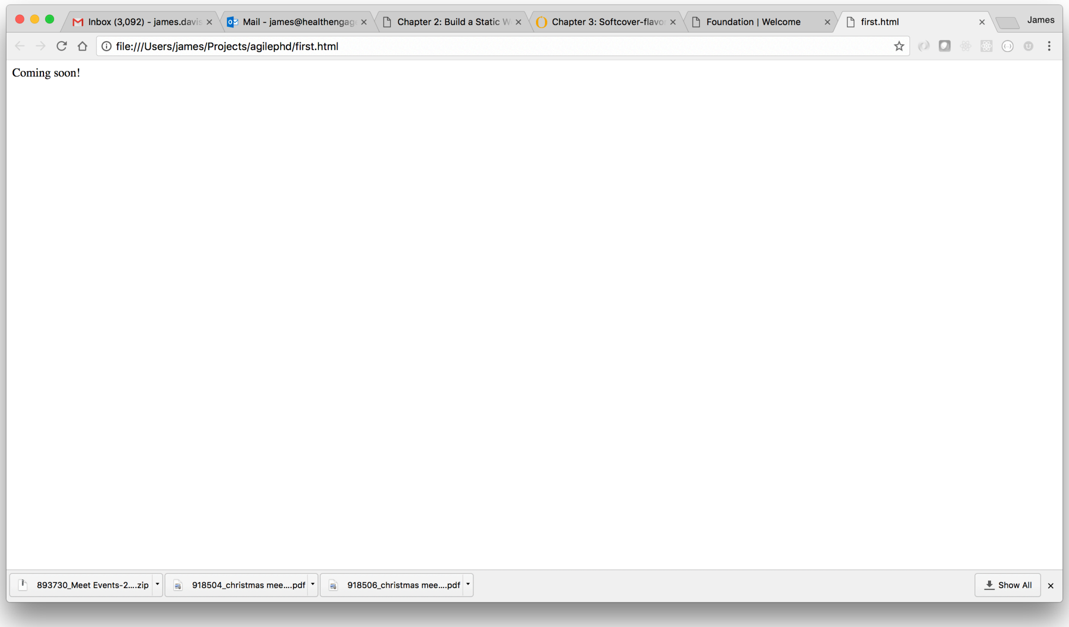
As an example, this chapter will focus on building a website for a technology consulting service, AgilePhd. **Note: This exercise assumes that you are fa-**

miliar with creating GitHub repositories. If not, try working through the GitHub Hello World tutorial.

The finished product will look something like this:


Google
Microsoft Office Home
AgilePhD | Welcome
Chapter 2: Build a Static Webs
Chapter 2: Introduction to Mar
James
agilephd.com
AgilePhD
Contact
Organizational Change Through Agile Methods
Agile Project Management
As an experienced agile project manager, AgilePhD is the niche provider clients

…but you usually start with something like this:


Inbox (3,092) - james.davis
Mail - james@healthengag
Chapter 2: Build a Static W
Chapter 3: Softcover-flavo
Foundation | Welcome
first.html
James
file:///Users/james/Projects/agilephd/first.html
893730_Meet Events-2....zip
918504_christmas mee....pdf
918506_christmas mee....pdf
Show All

Coming soon!

Generated with this minimal code:

Listing 3.1: index.html

```
<html>
<body>
  Coming soon!
</body>
</html>
```

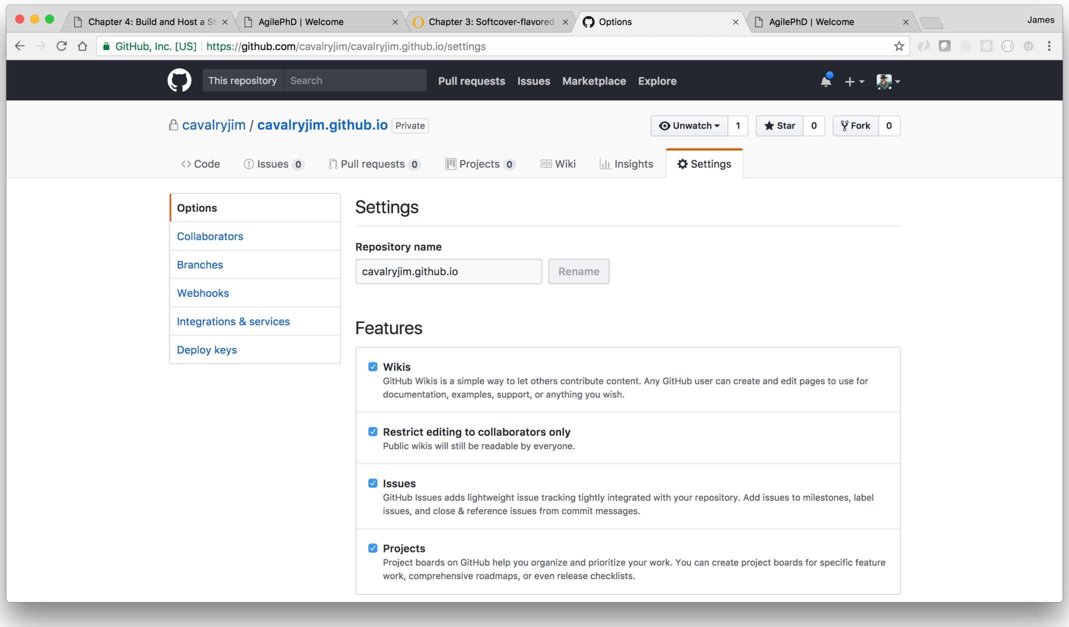
Inside your Projects folder, create a subfolder called 'agilephd' and copy the above code into a file called index.html.

We are going to use a service called GitHub Pages so let's create a git repository for this code. You get one freely hosted website per GitHub account. Head over to GitHub and create a new repository named username.github.io, where username is your username on GitHub.

```
$ git init
$ git add .
$ git remote add origin git@github.com:username/username.github.io.git
```

```
    (where username is your username)
$ git push -u origin master
```

Goto your new GitHub repository and visit the settings page. Scroll down to the ‘GitHub Pages’ section and publish the page.



https://github.com/cavalryjim/cavalryjim.github.io/settings

cavalryjim / cavalryjim.github.io Private

Settings

Repository name

cavalryjim.github.io

Features

✓ Wikis
GitHub Wikis is a simple way to let others contribute content. Any GitHub user can create and edit pages to use for documentation, examples, support, or anything you wish.

✓ Restrict editing to collaborators only
Public wikis will still be readable by everyone.

✓ Issues
GitHub Issues adds lightweight issue tracking tightly integrated with your repository. Add issues to milestones, label issues, and close & reference issues from commit messages.

✓ Projects
Project boards on GitHub help you organize and prioritize your work. You can create project boards for specific feature work, comprehensive roadmaps, or even release checklists.

Now you should be able to view your ‘coming soon!’ page at username.github.io

3.3 Plagiarize

In technology, plagiarism is not only tolerated but it is encouraged and celebrated. Open source communities endorse this practice by publicly posting code. All that is asked in return is that you share your enhancements with the community creating a win-win atmosphere.

Don’t spend time reinventing the wheel. Find a solution that works and borrow heavily. That is exactly where we are going to start with our static website.

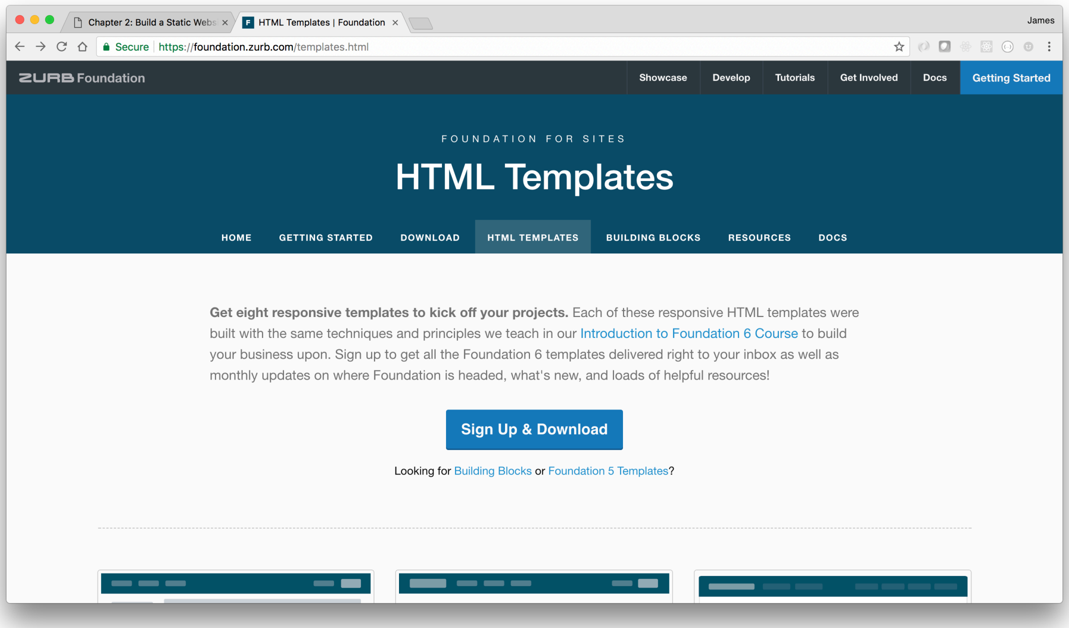
3.4 Zurb Foundation

HTML is the language of the web. Web-servers know how to send it and browsers know how to interpret it. HTML is great. HTML is awesome. . . but writing it can be a bit mundane and slow. Did I mention you should borrow from others? No, it is not stealing and it is not wrong. We all do it and as a community, we all 'borrow' from each other. Let's start by borrowing from a company that has made it easy to find their code.

Zurb is a design firm that creates awesome websites for clients. They are also the creators of Foundation, a responsive front-end framework. Zurb's Foundation or Zurb Foundation or just Foundation is a javascript library that helps your web pages look better. Foundation, and other responsive frameworks such as Bootstrap, make it easy to design beautiful responsive websites, apps and emails that look amazing on any browser or any device.

Check out Zurb to learn more about the framework (but don't spend too much time as it is not important for this course).

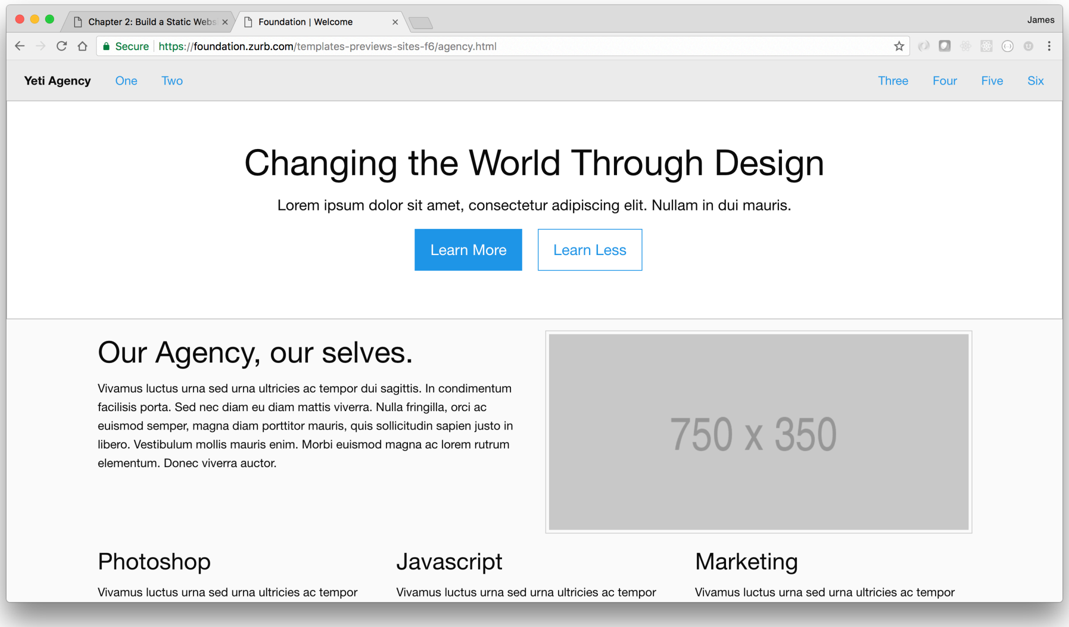
Zurb provides several templates of their framework. Sign up to download all the templates or you can scroll down to view demos.



Get eight responsive templates to kick off your projects. Each of these responsive HTML templates were built with the same techniques and principles we teach in our Introduction to Foundation 6 Course to build your business upon. Sign up to get all the Foundation 6 templates delivered right to your inbox as well as monthly updates on where Foundation is headed, what's new, and loads of helpful resources!

Looking for Building Blocks or Foundation 5 Templates?

For the consulting firm gig, let's start with the 'agency' template.


Chapter 2: Build a Static Web
Foundation | Welcome
James
Secure https://foundation.zurb.com/templates-previews-sites-f6/agency.html
Yeti Agency
One
Two
Three
Four
Five
Six
Learn More
Learn Less
750 x 350

Changing the World Through Design

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam in dui mauris.

Our Agency, our selves.

Vivamus luctus urna sed urna ultricies ac tempor dui sagittis. In condimentum facilisis porta. Sed nec diam eu diam mattis viverra. Nulla fringilla, orci ac euismod semper, magna diam porttitor mauris, quis sollicitudin sapien justo in libero. Vestibulum mollis mauris enim. Morbi euismod magna ac lorem rutrum elementum. Donec viverra auctor.

Photoshop

Vivamus luctus urna sed urna ultricies ac tempor

Javascript

Vivamus luctus urna sed urna ultricies ac tempor

Marketing

Vivamus luctus urna sed urna ultricies ac tempor

For the cost of signing up with Zurb (which was free), we now have some robust HTML code to build upon.

Listing 3.2: index.html

```
<!doctype html>
<html class="no-js" lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Foundation | Welcome</title>
    <link rel="stylesheet"
      href="https://dhbhdrzi4tiry.cloudfront.net/cdn/sites/foundation.min.css">
  </head>
  <body>

    <!-- Start Top Bar -->
    <div class="top-bar">
      <div class="top-bar-left">
        <ul class="menu">
          <li class="menu-text">Yeti Agency</li>
          <li><a href="#">One</a></li>
          <li><a href="#">Two</a></li>
        </ul>
      </div>
      <div class="top-bar-right">
        <ul class="menu">
          <li><a href="#">Three</a></li>
          <li><a href="#">Four</a></li>
          <li><a href="#">Five</a></li>
          <li><a href="#">Six</a></li>
        </ul>
      </div>
    </div>
    <!-- End Top Bar -->

    <div class="callout large">
      <div class="row column text-center">
        <h1>Changing the World Through Design</h1>
        <p class="lead">
          Lorem ipsum dolor sit amet, consectetur adipiscing elit.
          Nullam in dui mauris.
        </p>
        <a href="#" class="button large">Learn More</a>
        <a href="#" class="button large hollow">Learn Less</a>
      </div>
    </div>

    <div class="row">
      <div class="medium-6 columns medium-push-6">
        <img class="thumbnail" src="https://placehold.it/750x350">
      </div>
      <div class="medium-6 columns medium-pull-6">
        <h2>Our Agency, our selves.</h2>
        <p>
          Vivamus luctus urna sed urna ultricies ac tempor dui sagittis.
          In condimentum facilisis porta. Sed nec diam eu diam mattis viverra.
          Nulla fringilla, orci ac euismod semper, magna diam porttitor mauris,
          quis sollicitudin sapien justo in libero. Vestibulum mollis mauris enim.
          Morbi euismod magna ac lorem rutrum elementum. Donec viverra auctor.
        </p>
      </div>
    </div>

    <div class="row">
      <div class="medium-4 columns">
        <h3>Photoshop</h3>
        <p>Vivamus luctus urna sed urna ultricies ac tempor dui sagittis.
        In condimentum facilisis porta. Sed nec diam eu diam mattis viverra.
        Nulla fringilla, orci ac euismod semper, magna.</p>
      </div>
      <div class="medium-4 columns">
        <h3>Javascript</h3>
```

```
        <p>Vivamus luctus urna sed urna ultricies ac tempor dui sagittis.
        In condimentum facilisis porta. Sed nec diam eu diam mattis viverra.
        Nulla fringilla, orci ac euismod semper, magna.</p>
      </div>
      <div class="medium-4 columns">
        <h3>Marketing</h3>
        <p>Vivamus luctus urna sed urna ultricies ac tempor dui sagittis.
        In condimentum facilisis porta. Sed nec diam eu diam mattis viverra.
        Nulla fringilla, orci ac euismod semper, magna.</p>
      </div>
    </div>

    <hr>

    <div class="row column">
      <ul class="vertical medium-horizontal menu expanded text-center">
        <li><a href="#"><div class="stat">28</div><span>Websites</span></a></li>
        <li><a href="#"><div class="stat">43</div><span>Apps</span></a></li>
        <li><a href="#"><div class="stat">95</div><span>Ads</span></a></li>
        <li><a href="#"><div class="stat">59</div><span>Cakes</span></a></li>
        <li><a href="#"><div class="stat">18</div><span>Logos</span></a></li>
      </ul>
    </div>

    <hr>

    <div class="row column">
      <h3>Our Recent Work</h3>
    </div>

    <div class="row medium-up-3 large-up-4">
      <div class="column">
        <img class="thumbnail" src="https://placehold.it/550x550">
      </div>
      <div class="column">
        <img class="thumbnail" src="https://placehold.it/550x550">
      </div>
      <div class="column">
        <img class="thumbnail" src="https://placehold.it/550x550">
      </div>
      <div class="column">
        <img class="thumbnail" src="https://placehold.it/550x550">
      </div>
      <div class="column">
        <img class="thumbnail" src="https://placehold.it/550x550">
      </div>
      <div class="column">
        <img class="thumbnail" src="https://placehold.it/550x550">
      </div>
      <div class="column">
        <img class="thumbnail" src="https://placehold.it/550x550">
      </div>
      <div class="column">
        <img class="thumbnail" src="https://placehold.it/550x550">
      </div>
      <div class="column">
        <img class="thumbnail" src="https://placehold.it/550x550">
      </div>
      <div class="column">
        <img class="thumbnail" src="https://placehold.it/550x550">
      </div>
      <div class="column">
        <img class="thumbnail" src="https://placehold.it/550x550">
      </div>
      <div class="column">
        <img class="thumbnail" src="https://placehold.it/550x550">
      </div>
    </div>

    <hr>

    <div class="row column">
      <ul class="menu">
        <li><a href="#">One</a></li>
        <li><a href="#">Two</a></li>
        <li><a href="#">Three</a></li>
        <li><a href="#">Four</a></li>
      </ul>
    </div>

    <script src="https://code.jquery.com/jquery-2.1.4.min.js"></script>
    <script src="https://dhbhdrzi4tiry.cloudfront.net/cdn/sites/foundation.js">
    </script>
    <script>
      $(document).foundation();
    </script>
  </body>
</html>
```

Copy / paste the above code into your index.html file and push to github.

```
$ git add .
$ git commit -m "updating page with a template from Zurb Foundation"
$ git push
```

Now you should be able to view the Zurb Foundation template page at username.github.io

Time to modify the template for the AgilePhD consulting firm.

Listing 3.3: index.html

```html
<!doctype html>
<html class="no-js" lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>AgilePhD | Welcome</title>
    <link rel="stylesheet"
      href="https://dhbhdrzi4tiry.cloudfront.net/cdn/sites/foundation.min.css">
    <link rel="stylesheet" href="https://s3.amazonaws.com/agilephd/styles/main.css">
    <link href="https://fonts.googleapis.com/css?family=Montserrat" rel="stylesheet">
  </head>
  <body>

    <!-- Start Top Bar -->
    <div class="top-bar">
      <div class="top-bar-left">
        <ul class="menu">
          <li class="menu-text agilephd">AgilePhD</li>
        </ul>
      </div>
      <div class="top-bar-right">
        <ul class="menu">
          <li><a data-open="contactModal">Contact</a></li>
        </ul>
      </div>
    </div>
    <!-- End Top Bar -->

    <div class="callout large scrum">
      <div class="row column text-center bottom">
        <h1 class="scrum_h1 bottom">Organizational Change Through Agile Methods</h1>
      </div>
    </div>

    <div class="row">
      <div class="medium-6 columns medium-push-6">
        <img class="thumbnail pm_image"
          src="https://s3.amazonaws.com/agilephd/project_management.jpg">
      </div>
      <div class="medium-6 columns medium-pull-6">
        <h2>Agile Project Management</h2>
        <p>
          As an experienced agile project manager, AgilePhD is the niche provider
          clients seek out when they are looking to implement business-enhancing,
          agile PMO services that improve project and portfolio performance.
        </p>

      </div>
    </div>
    <div class="row">
      <div class="medium-4 columns">
        <h3>Project Management</h3>
        <p>
          Speed is what companies want as they adopt Agile. Traditional
          plan-driven methodologies like Waterfall make a lot of up front
          assumptions and leave no flexibility to adapt to the changing
          needs of the customer.
        </p>
      </div>
      <div class="medium-4 columns">
        <h3>Software Development</h3>
        <p>
          It's all about that MVP (minimally viable product) and getting
          it in front of your customers.  I love building technology products.
        </p>
      </div>
      <div class="medium-4 columns">
        <h3>Cloud Services</h3>
        <p>
          As an early adopter several cloud technologies, AgilePhD can
          help you develop a scalable solutions allowing you to focus
          on your core business.
        </p>
      </div>
    </div>

    <hr>

    <div class="row column">
      <ul class="vertical medium-horizontal menu expanded text-center">
        <li>
          <a href="#"><div class="stat">600+</div><span>Students Trained</span></a>
        </li>
        <li>
          <a href="#"><div class="stat">20+</div><span>Web App Projects</span></a>
        </li>
        <li>
          <a href="#"><div class="stat">12+</div><span>Mobile App Projects</span></a>
        </li>
      </ul>
    </div>

    <hr>

    <div class="row column">
      <h3>Recent Web Apps</h3>
    </div>

    <div class="row medium-up-3 ">
      <div class="column">
```

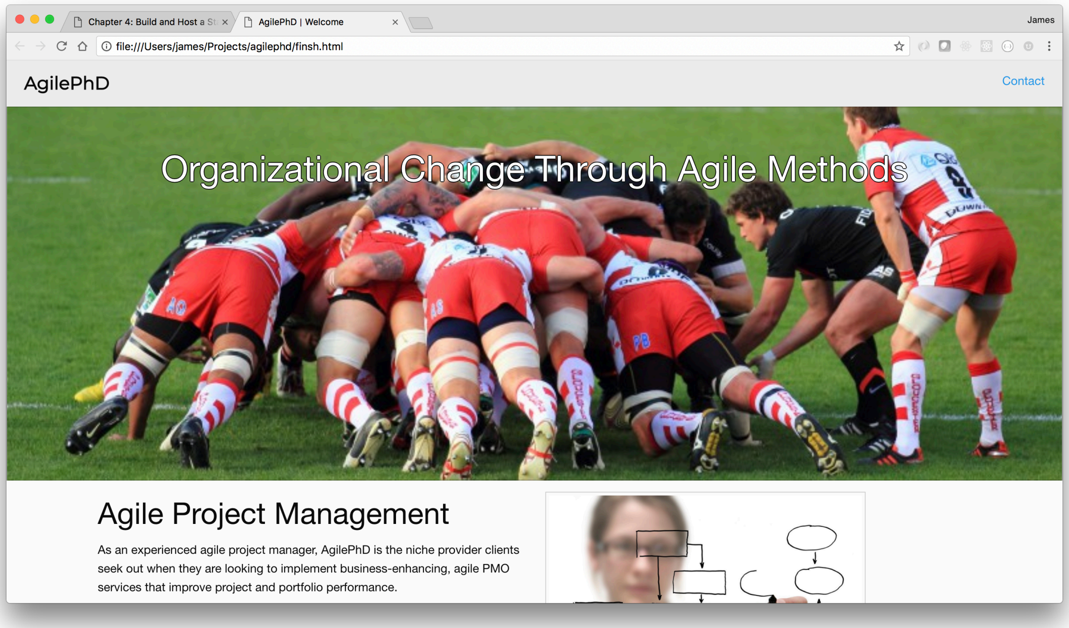
```
        <img class="thumbnail"
          src="https://s3.amazonaws.com/agilephd/damagelist.png">
      </div>
      <div class="column">
        <img class="thumbnail"
          src="https://s3.amazonaws.com/agilephd/health_engagements.png">
      </div>
      <div class="column">
        <img class="thumbnail"
          src="https://s3.amazonaws.com/agilephd/animalminder_2.jpg">
      </div>
      <div class="column">
        <img class="thumbnail"
          src="https://s3.amazonaws.com/agilephd/sabot_solutions.png">
      </div>
      <div class="column">
        <img class="thumbnail"
          src="https://s3.amazonaws.com/agilephd/sicklewell.png">
      </div>
      <div class="column">
        <img class="thumbnail"
          src="https://s3.amazonaws.com/agilephd/rails_girls.png">
      </div>

    </div>

    <hr>

    <div class="row column">
      <h3>Recent Mobile Apps</h3>
    </div>

    <div class="row small-up-2 medium-up-5 ">
      <div class="column">
        <img class="thumbnail"
          src="https://s3.amazonaws.com/agilephd/damagelist_ios_3.jpg">
      </div>
      <div class="column">
        <img class="thumbnail"
          src="https://s3.amazonaws.com/agilephd/btr_crime.jpeg">
      </div>
      <div class="column">
        <img class="thumbnail"
          src="https://s3.amazonaws.com/agilephd/seattle_crime.jpeg">
      </div>
      <div class="column">
        <img class="thumbnail"
          src="https://s3.amazonaws.com/agilephd/sfo_crime.jpeg">
      </div>
      <div class="column">
        <img class="thumbnail"
```

```
          src="https://s3.amazonaws.com/agilephd/sicklewell_ios.png">
      </div>

    </div>

    <hr>

    <div class="row column">
      <ul class="menu">

        <li>
          <a href="https://www.linkedin.com/in/jdavisphd/" class="" target="_blank">
            <img src="https://s3.amazonaws.com/agilephd/webicon-linkedin-m.png">
          </a>
        </li>
        <li>
          <a href="https://github.com/cavalryjim" class="" target="_blank">
            <img src="https://s3.amazonaws.com/agilephd/webicon-github-m.png">
          </a>
        </li>
      </ul>
    </div>

    <div class="reveal" id="contactModal" data-reveal>
      <h4>Contact information</h4>
      <ul class="no-bullet">
        <li>Dr. James Davis, PhD, PMP, CSP, CSM</li>
        <li>
          <a href="mailto:james@agilephd.com?Subject=AgilePhD" target="_top">
            james@agilephd.com
          </a>
        </li>
        <li><a href="tel:+12258920230">+1.225.892.0230</a></li>
      </ul>
      <button class="close-button" data-close aria-label="Close modal" type="button">
        <span aria-hidden="true">&times;</span>
      </button>
    </div>

    <script src="https://code.jquery.com/jquery-2.1.4.min.js"></script>
    <script src="https://dhbhdrzi4tiry.cloudfront.net/cdn/sites/foundation.js"></script>
    <script>
      $(document).foundation();
    </script>
  </body>
</html>
```

Update the index.html file with the new code (from above) and push to github.

```
$ git add .
$ git commit -m "customizing index.html for AgilePhd"
$ git push
```

When viewed in a browser, username.github.io should look something like this:


Chapter 4: Build and Host a St
AgilePhD | Welcome
James
file:///Users/james/Projects/agilephd/finsh.html
AgilePhD
Contact
Organizational Change Through Agile Methods

Agile Project Management

As an experienced agile project manager, AgilePhD is the niche provider clients seek out when they are looking to implement business-enhancing, agile PMO services that improve project and portfolio performance.

3.5 Hosting Webpages

Now that we have a decent static webpage, it is time to host it.

We are living in the glory days of technology. While there are unlimited options for hosting websites, we are going to focus on options that are 1) free and 2) worthy your time & effort.[1]

[1]If you are in a startup organization, titles really don't matter and nothing should be considered 'beneath' you. You and your co-founders should be willing to do everything and anything to see the venture move forward.

Box 3.1. Hosting Services

Word Press, Wix, Squarespace, along with most Domain Registration Services, have wizard-style website creation abilities. While this may work for some situations, I've found these 'wizard' solutions to be very heavy and overly constraining. As the technical lead for your venture, developing your own webpages gives you flexibility with hosting options and features.

While laws vary from state to state, content, such as a website, is owned by the person or entity that created it. This is more reason to for you to develop your own websites and/or applications. If you do happen to utilize outside services, read the contract and consult an attorney as needed.

If you didn't already think of yourself as a web developer, now you can. You have created an HTML page and published it at username.github.io (where 'username' is your github username). While this may be sufficient for some projects or personal pages, you usually want your page accessible through a custom URL. For this gig, I want to use 'agilephd.com'.

There are dozens, if not hundreds, of domain registration firms. I've used several and at the moment, GoDaddy is my favorite. If you don't already have one, create an account with GoDaddy.

GoDaddy allows users to search for domain names. If your desired domain name isn't available, GoDaddy offers suggestions. I was lucky and registered 'agilephd.com' years ago.

Box 3.2. Domain Names

What's in a domain name? Names can be meaningful or meaningless. IBM.com and LSU.edu are very meaningful names. I imagine the discussion for lsu.edu started with "We are Louisiana State University and must have lsu.edu".

To me, other domain name, such as heroku.com, are a less meaningful in that the founders likely started with the question “What domain names are available? Heroku.com is available and sounds cool... let’s get it.”

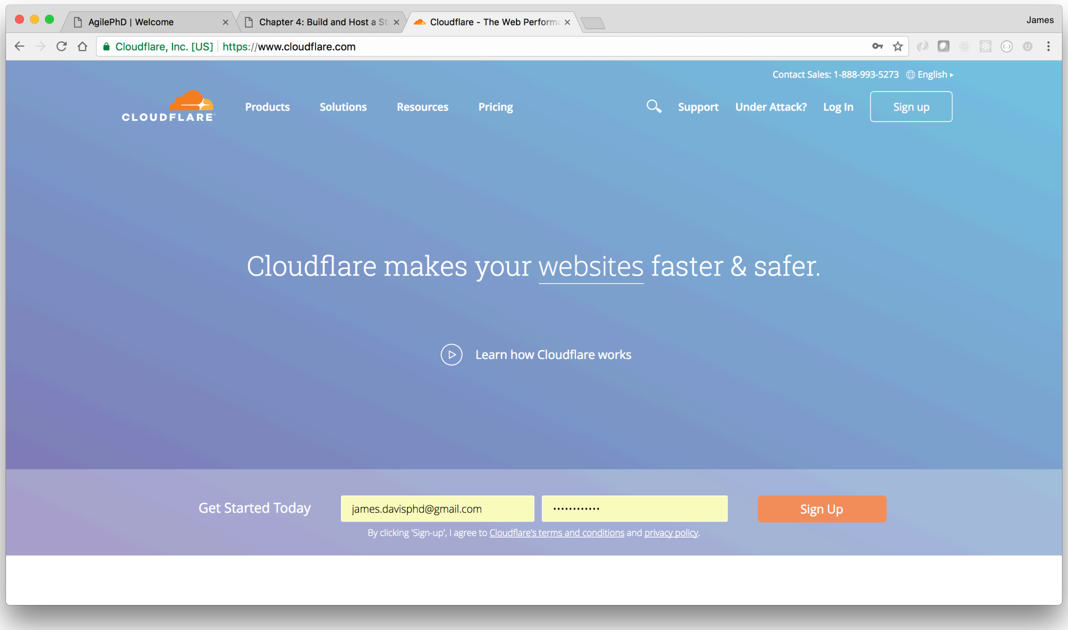
After finding the appropriate domain name, you register it with GoDaddy. During the registration process, domain registration firms are usually very persistent about up-selling additional services to you. These services may include hiding your registration name, SSL, coming soon pages, templates, and web development services. You don’t need any of those and can opt out.

Once you have purchased a domain name, you must keep it registered with a registration service to maintain ownership.

3.6 The Connection

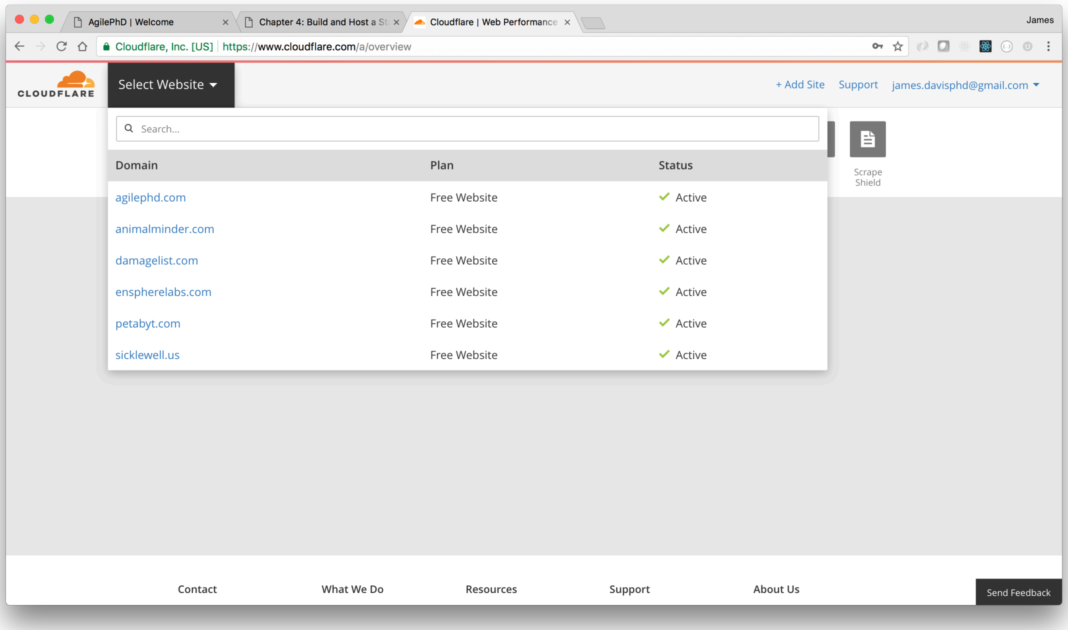
The last step is to select the DNS servers for your newly registered domain such that it ‘points’ to the location of your content. You could configure GoDaddy to use GitHub’s DNS servers but I use a service called CloudFlare.

3.6.1 Configure CloudFlare


AgilePhD | Welcome
Chapter 4: Build and Host a St
Cloudflare - The Web Perform
James
Cloudflare, Inc. [US] https://www.cloudflare.com
Contact Sales: 1-888-993-5273 English
CLOUDFLARE
Products
Solutions
Resources
Pricing
Support
Under Attack?
Log In
Sign up
Cloudflare makes your websites faster & safer.
Learn how Cloudflare works
Get Started Today
Sign Up
By clicking 'Sign-up', I agree to Cloudflare's terms and conditions and privacy policy.

james.davisphd@gmail.com

Create an account with CloudFlare and add your website to your account.



Domain	Plan	Status
agilephd.com	Free Website	✓ Active
animalminder.com	Free Website	✓ Active
damagelist.com	Free Website	✓ Active
enspherelabs.com	Free Website	✓ Active
petabyt.com	Free Website	✓ Active
sicklewell.us	Free Website	✓ Active

Like many other SaaS solutions, CloudFlare utilizes the freemium model. At the free level, you get basic routing and CloudFlare's robust protection service. In the past, we had to buy expensive hardware solutions from companies like Cisco and configure complicated firewall servers. CloudFlare keeps your application safer (there is no such thing as completely safe) from threats such as DDOS attacks and other evilness in cyber world.

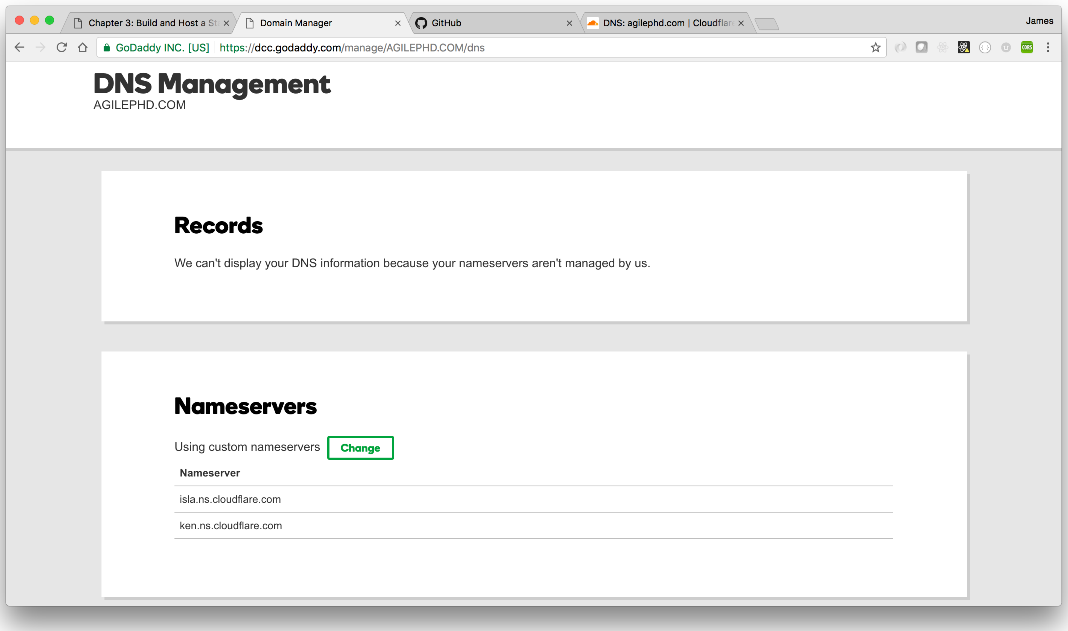
Box 3.3. Nameservers

To use Cloudflare, you need to change your domain's authoritative DNS servers, which are also referred to as just 'nameservers'. This change would be made in your hosting service account (GoDaddy).

As a reference, these are an example of Cloudflare nameservers you may be assigned.

```
isla.ns.cloudflare.com
ken.ns.cloudflare.com
```

In my GoDaddy account, I change the DNS settings to the nameservers provided by CloudFlare.



DNS Management

AGILEPHD.COM

Records

We can't display your DNS information because your nameservers aren't managed by us.

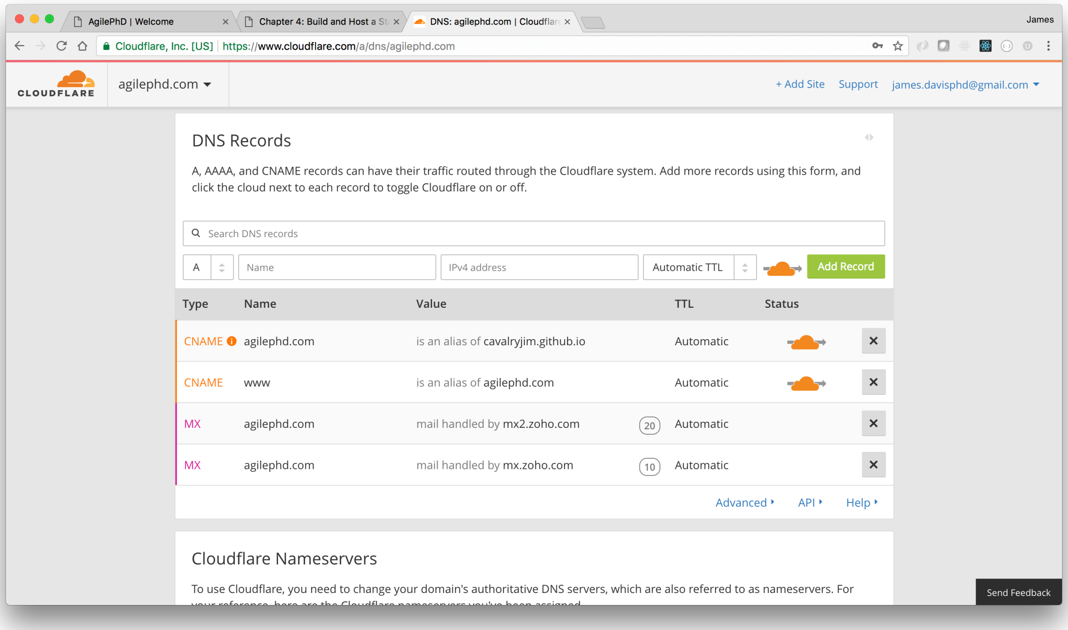
Nameservers

Using custom nameservers

Nameserver
isla.ns.cloudflare.com
ken.ns.cloudflare.com

Back at CloudFlare page for your website, select the DNS page. Similar to the image below, I will add CNAME and MX records routing traffic through the Cloudflare system.

- CNAME agilephd.com is an alias of cavalryjim.github.io
- CNAME www is an alias of agilephd.com
- MX agilephd.com mail handled by mx2.zoho.com
- MX agilephd.com mail handled by mx.zoho.com



DNS Records

A, AAAA, and CNAME records can have their traffic routed through the Cloudflare system. Add more records using this form, and click the cloud next to each record to toggle Cloudflare on or off.

Type	Name	Value		TTL	Status
CNAME	agilephd.com	is an alias of cavalryjim.github.io		Automatic	
CNAME	www	is an alias of agilephd.com		Automatic	
MX	agilephd.com	mail handled by mx2.zoho.com	20	Automatic	
MX	agilephd.com	mail handled by mx.zoho.com	10	Automatic	

Cloudflare Nameservers

To use Cloudflare, you need to change your domain's authoritative DNS servers, which are also referred to as nameservers. For

Box 3.4. DNS Record Types

A Canonical Name or ‘CNAME’ record is a type of DNS record that maps an alias name to a true or canonical domain name. An ‘A’ record returns a 32-bit IPv4 address such as ‘192.0.2.23’. An ‘MX’ record is a mail exchange record that is used for mapping email traffic.

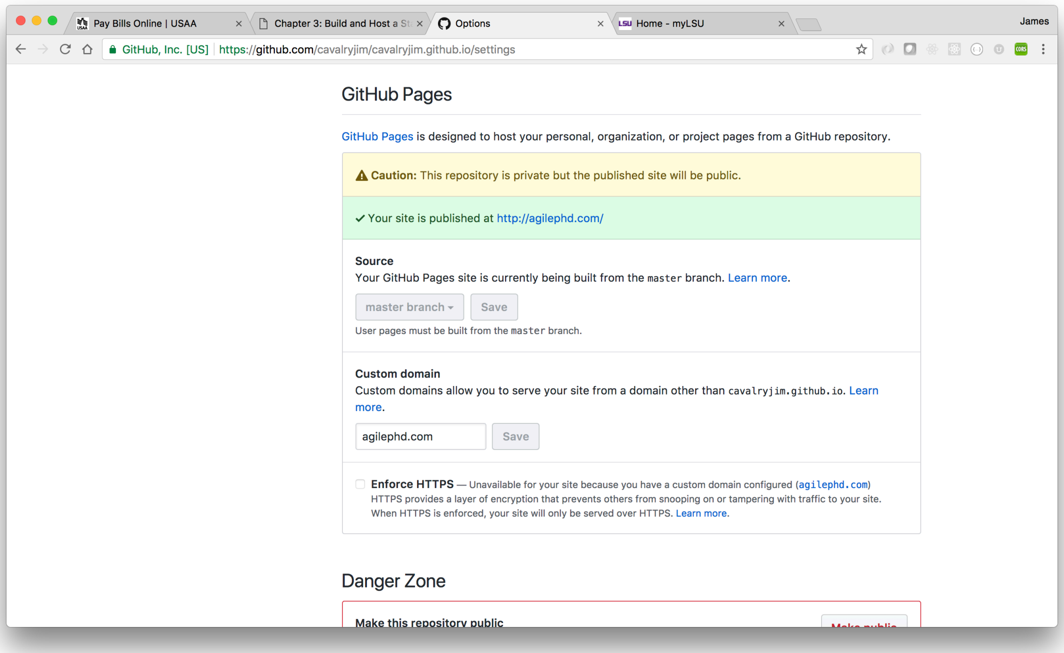
```
NAME                    TYPE   VALUE
--------------------------------------------------
bar.example.com         CNAME  foo.example.com
foo.example.com         A      192.0.2.23
example.com             MX     mx.mail.com
```

DNS changes can take a few minutes to a few hours to propagate through the ‘Internet’. This delay is caused by Internet Service Providers and other entities caching DNS settings. **Mental note: Don’t make DNS changes before**

an important demo of your product.

3.6.2 Update GitHub

Lastly, you need to let GitHub know that you are using a custom domain name.


Pay Bills Online | USAA
Chapter 3: Build and Host a St
Options
Home - myLSU
James
GitHub, Inc. [US] https://github.com/cavalryjim/cavalryjim.github.io/settings

GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

Caution: This repository is private but the published site will be public.

✓ Your site is published at http://agilephd.com/

Source
Your GitHub Pages site is currently being built from the `master` branch. Learn more.

master branch ▾ Save

User pages must be built from the `master` branch.

Custom domain
Custom domains allow you to serve your site from a domain other than `cavalryjim.github.io`. Learn more.

agilephd.com Save

Enforce HTTPS — Unavailable for your site because you have a custom domain configured (`agilephd.com`)
HTTPS provides a layer of encryption that prevents others from snooping on or tampering with traffic to your site.
When HTTPS is enforced, your site will only be served over HTTPS. Learn more.

Danger Zone

Make this repository public

At this point, you are ready to call your client (or co-workers) and tell them the new website is done and has been deployed. Good job!

Chapter 4

Sprint Zero

In agile software development frameworks, such as Scrum, development is conducted in time-boxed work iterations called Sprints. Sprint Zero is all the work that needs to happen before you can start actually development[1].

4.1 Python versions

Technology is constantly changing. That the great thing about our industry and the frustrating thing about our industry. In Python, there was a bit of a riff when the language went from version 2.x to 3.x. The debate is mostly over but know that some popular packages were slow to embrace Python 3 and developers were forced to choose between versions.

[1]Some agile practitioners argue there is no such thing as Sprint Zero. My response to them would be - this is my book and I'm going to use the term as I see fit.

Despite a few staunch holdouts, Python 3 is the present and future of the language. You can read more about the version debate at wiki.python.org.

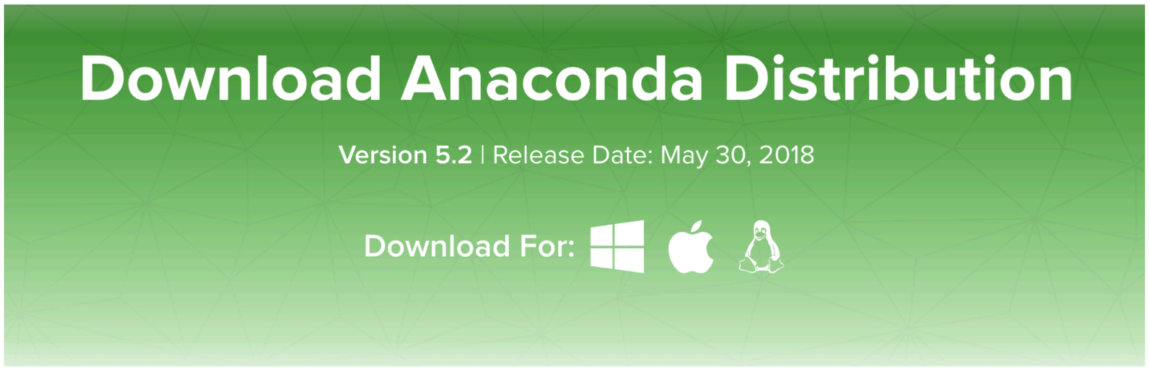
For this book, we're going to use Python 3. If you're using MacOS or Linux, you probably have a version of Python already installed. Verify by typing 'python –version' in the terminal.

```
$ python --version
Python 3.6.3
```

If you have Python version 3 or later, you are good and can skip the section on installing Python.

4.1.1 Anaconda Distribution

If you don't have Python version 3.6 or later on your computer, the Anaconda Distribution is the recommended installation. Anaconda is an open source distribution of Python that comes with many of the popular data science packages.


Download Anaconda Distribution
Version 5.2 | Release Date: May 30, 2018
Download For:

The Anaconda Distribution is available at https://www.anaconda.com/.

Once installed, you should be able to verify the Python version in the terminal by using `python -version` or `python3 -version`.

```
$ python3 --version
Python 3.6.3
```

Anaconda comes with all of the Python packages we will be using in this course. You can view a list of the packages at https://docs.anaconda.com/. We will be using:

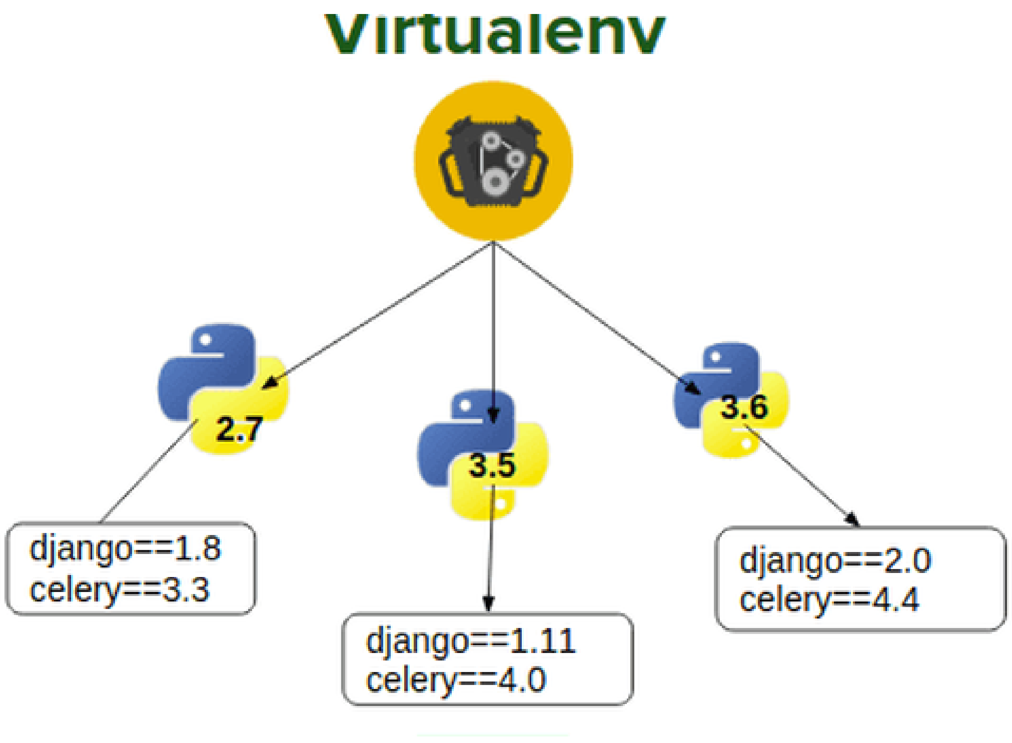
- sqlalchemy
- pandas
- scikit-learn
- beautifulsoup4
- flask
- django
- jupyter
- . . . and many others

4.1.2 Python PIP

PIP is the recommended tool for installing Python packages. It comes with the Anaconda Distribution. Verify that pip is installed with `pip -version` or `pip3 -version`.

```
$ pip --version
pip 9.0.1

$ pip3 --version
pip 9.0.1
```

4.1.3 Pyenv, Pipenv & Virtualenv (optional)

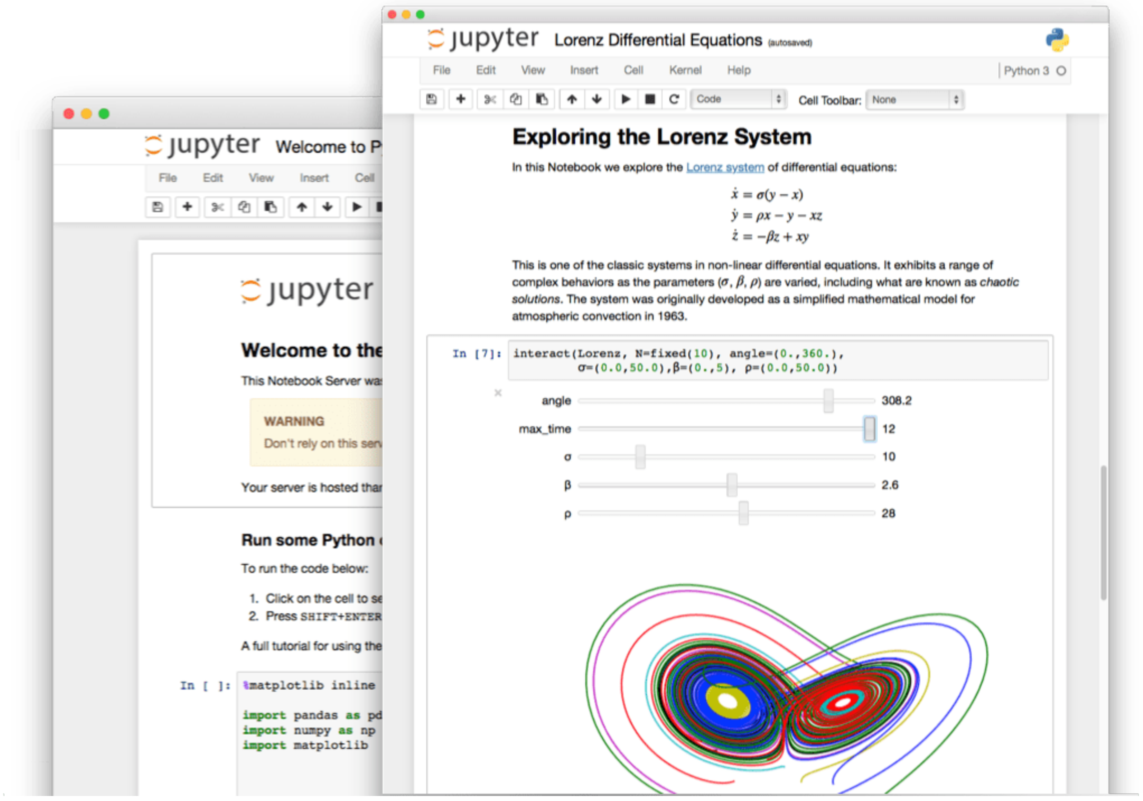


If you expect to use Python outside of this course, you are encouraged to isolate your Python environments. Isolating your environments allows you to work with different versions of packages for different projects. It also allows you to share the exact package versions being used on a project with other collaborators.

Using isolated environments is optional and you can easily transition when the need arises. You can find more information about at the following websites:

- https://github.com/pyenv/pyenv
- https://pipenv.readthedocs.io/
- https://virtualenv.pypa.io/

4.2 Jupyter Notebook



jupyter

Welcome to the

This Notebook Server wa

WARNING
Don't rely on this serv

Your server is hosted thar

Run some Python

To run the code below:

1. Click on the cell to se
2. Press SHIFT+ENTER

A full tutorial for using the

```
In [ ]: %matplotlib inline

        import pandas as pd
        import numpy as np
        import matplotlib
```

Exploring the Lorenz System

In this Notebook we explore the Lorenz system of differential equations:

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = -\beta z + xy$$

This is one of the classic systems in non-linear differential equations. It exhibits a range of complex behaviors as the parameters (σ, β, ρ) are varied, including what are known as *chaotic solutions*. The system was originally developed as a simplified mathematical model for atmospheric convection in 1963.

```
In [7]: interact(Lorenz, N=fixed(10), angle=(0.,360.),
                 σ=(0.0,50.0),β=(0.,5), ρ=(0.0,50.0))
```

```
angle     308.2
max_time  12
σ         10
β         2.6
ρ         28
```

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. It is popular among data scientists and comes as part of the Anaconda Distribution. We will be using Jupyter later in this book. Confirm Jupyter's installation with `jupyter -version`

```
$ jupyter --version
4.4.0
```

4.3 SQLite

SQLite is a compact, cross platform, self-contained relational database management system that is available in the public domain. SQLite is included with Mac OS X by default. It is located in the /usr/bin directory and called sqlite3.

```
$ sqlite3 --version
3.16.2 2017-01-06 16:32:41 a65a62893ca8319e89e48b8a38cf8a59c69a8209
```

Using SQLite, users can create file-based databases that can be transported across machines, platforms, etc. The only thing needed to then view or edit these databases is the SQLite command line program, a GUI tool capable of communicating with SQLite, or python packages (such as sqlalchemy) that were created for database interactions.

4.3.1 SQLite on Windows

Instructions for installing SQLite on Windows will be provided if needed. SQLAlchemy may provide the required functionality needed to interaction with a SQLite file.

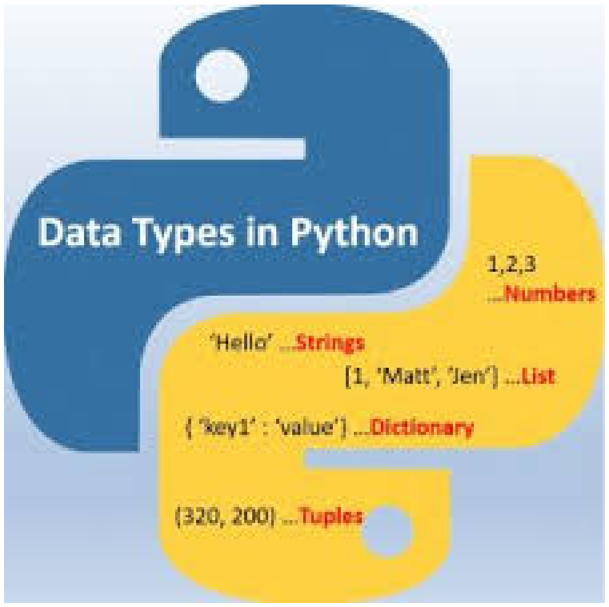
4.4 Celebrate

You are complete with Sprint Zero and ready to move forward!

Chapter 5

Data Types

This section will introduce the common data types used in Python. Additional information about Python data types can be found at docs.python.org.



The data types to be discussed in this chapter include:

- boolean
- numeric

- string
- datetime

Box 5.1. Interacting with Python (REPL)

At this point, you should have a working Python 3 interpreter at hand. If you need help getting Python set up correctly, please refer to Chapter 4 to get started.

The most straightforward way to start talking to Python is in an interactive Read-Eval-Print Loop (REPL) environment. That simply means starting up an interactive shell and typing commands to it directly.

From the terminal, you can start an interactive shell by typing the command **python** or **python3**.

```
$ python
Python 3.6.3 (default, Nov  2 2017, 10:31:58)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.42.1)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

If you are not seeing the >>> prompt, then you are not talking to the Python interpreter. If you are seeing the prompt, youre up and running! Try running some simple python commands.

```
>>> print("Hello, World.")
Hello, World.
>>> x = 2
>>> y = 6
>>> y + x
8
>>> exit()
```

As a note, use the **exit()** command to get out of the Python shell environment.

5.1 Boolean

Boolean values are the two constant objects `True` and `False`.

They are used to represent truth values (other values can also be considered true or false).

In numeric contexts (for example, when used as the argument to an arithmetic operator), they behave like the integers 1 and 0, respectively.

The built-in function `bool()` can be used to cast any value to a Boolean, if the value can be interpreted as a truth value.

They are written as `True` and `False`, respectively.

Open an interactive Python environment using `python` or `python3` inside your terminal.

```
>>> t = True
>>> t + t
2
>>> f = False
>>> f
False
>>> f + f
0
>>> type(t)
<class 'bool'>
>>> type(f)
<class 'bool'>
>>> bool(1)
True
```

Boolean values respond to logical operators `and` / `or`

```
>>> True and False
False
>>> True and True
True
>>> False or True
True
>>> False or False
False
```

5.2 Numeric

Number data types store numeric values. They are **immutable** data types, means that changing the value of a number data type results in a newly allocated object.

Box 5.2. Python 2 vs Python 3

As an example of the differences between Python versions, Python 2 had two integer types `int` and `long`. These have been unified in Python 3, so there is now only one type, `int`.

5.2.1 Integer

Integers, or ints, are positive or negative whole numbers with no decimal point. Use `int()` to cast other data types as an integer, if the value can be interpreted as an integer.

```
>>> x = 17
>>> y = 3
>>> x + y
20
>>> x - y
14
>>> x * y
51
>>> x / y
5.666666666666667
>>> x // y
5

""" You can also cast other data types as integers """
>>> z = "22"
>>> type(z)
<class 'str'>
>>> a = int(z)
>>> type(a)
<class 'int'>
```

5.2.2 Float

Floating point real values, also called floats, represent real numbers and are written with a decimal point dividing the integer and fractional parts. Floats may also be in scientific notation, with e indicating the power of 10 (2.5e2 = 2.5 x 102 = 250). Use **float()** to cast other data types as a float, if the value can be interpreted as an float.

```
>>> c = 8.0
>>> d = 20.2223
>>> pi = 3.14
>>> type(c)
<class 'float'>
>>> c * pi
25.12
>>> d / c
2.5277875
>>> d // pi
6.0
>>> d / pi
6.440222929936306
>>> i = int(c)
>>> i
8
```

5.3 String

String literals, or just strings, in python are surrounded by either single quotation marks, or double quotation marks.

'hello' is the same as **"hello"**.

Strings can be output to screen using the print function. For example: **print("hello")**.

Like many other popular programming languages, strings in Python are arrays of bytes representing unicode characters. However, Python does not have a character data type, a single character is simply a string with a length of 1. Square brackets can be used to access elements of the string.

```
>>> h = "hello world!"
>>> type(h)
<class 'str'>
>>> h[0]
'h'
>>> h[-1]
'!'
>>> len(h)
12
>>> h.upper()
'HELLO WORLD!'
>>> h.lower()
'hello world!'
```

5.4 DateTime

All of the above data types come "built-in" with Python. While this makes Python a very effective language, you will likely require some additional data types not included with Python, such as dates and times.

The **datetime** module supplies classes for manipulating dates and times in both simple and complex ways.

```
>>> import datetime
>>> d = datetime.datetime.now()
>>> type(d)
<class 'datetime.datetime'>
>>> print(d)
2018-09-06 09:51:01.246077
>>> d.year
2018
>>> print(d.strftime("%m/%d/%Y"))
09/06/2018
```

5.5 Conclusion

Using Python, you will have many data types available to use. Some of the data types come "built-in" to the Python language while others may require importing an additional package.

Box 5.3. Interacting with Python (Command Line)

Entering commands to the Python interpreter interactively is great for quick testing and exploring features or functionality.

Eventually though, as you create more complex applications, you will develop longer bodies of code that you will want to edit and run repeatedly. You clearly dont want to re-type the code into the interpreter every time! This is where you will want to create a reusable script file.

Using whatever code editor youve chosen, create a script file called **hello.py** containing the following:

Listing 5.1: hello.py

```
print('Enter your name:')
x = input()
print('Hello, ' + x)
```

Now save the file, keeping track of the directory or folder you chose to save into.

Start a command prompt or terminal window. If the current working directory is the same as the location in which you saved the file, you can simply specify the filename as a command-line argument to the Python interpreter: **python hello.py**

```
$ python hello.py
Enter your name:
James
Hello, James
```

Chapter 6

Python Containers

Containers are built-in Python data structures that allow other data types to be organized, assessed, and manipulated. In this chapter, we explore the most commonly used data structures: the **list** and the **dictionary**.

Lists and dictionaries provide powerful ways to organize data in useful and interesting applications. In addition to exploring the use of lists and dictionaries, this chapter also introduces some simple control statements.

6.1 Lists

A **list** is a sequence of data values called items or elements. An item can be of any type.

A Python list is similar to a list you would make in the real-world:

- shopping list
- to-do list
- roster for a team
- guest list for a party

6.1.1 List structure

The logical structure of a list resembles the structure of a string. Items in a list are ordered by position. Each list item has a unique index specifying its position. Like many programming languages, Python is 0-index based, meaning list indexes start at 0, not 1. The index for a list starts with 0 and counts upward.

A list is written as a bracketed sequence of data separated by commas. Here are some examples:

[1971, 1989, 1994] # A list of integers
[‘bananas’, ‘apples’, ‘oranges’] # A list of strings
[[4, 5], [200, 343]] # A list containing two other lists

When using variables, a list is defined by either using the bracket notation **[]** or by casting any iterable sequence with the **list()** function.

Start an interactive Python session by using the `python` command in a terminal.

```
>>> a = [1, 2, 3, 4]
>>> a
[1, 2, 3, 4]
>>> type(a)
<class 'list'>
>>> b = list("Hello world!")
>>> b
['H', 'e', 'l', 'l', 'o', ' ', 'w', 'o', 'r', 'l', 'd', '!']
>>> type(b)
<class 'list'>
```

For convenience, we will be using the **range()** function to assist with list creation. The range function takes one to three arguments.

- list(range(4)) # create a list up to, but not including, the integer 4.
- list(range(2, 6)) # create a list of integers starting at 2 and going to, but not including, 6.
- list(range(3, 18, 3)) # create a list starting at 3, going to 18, and using steps of 3.

```
>>> r = list(range(3, 18, 3))
>>> r
[3, 6, 9, 12, 15]
>>> type(r)
<class 'list'>
>>> len(r)
5
>>> 12 in r
True
>>> 13 in r
False
```

6.1.2 Modifying lists

At any point in a list's existence, elements can be inserted, removed, or changed. The list will maintain its identity but the contents can change.

```
>>> t = [34, 12, 24, 77, 234, 65]
>>> type(t)
<class 'list'>
>>> t[1]
12
>>> t[1] = 9
>>> t
[34, 9, 24, 77, 234, 65]
```

Note that the subscript operation `t[1]` refers to the element's position and the target of the assignment.

Use the **split** function to extract a list of words from a sentence.

```
>>> s = "This is a sentence with seven words."
>>> words = s.split()
>>> words
['This', 'is', 'a', 'sentence', 'with', 'seven', 'words.']
```

The **list** object includes several methods for inserting and removing elements.

List Method	Results

list.append(element)	Adds element to the end of the list
list.extend(a_list)	Adds another list to the list
list.insert(index, element)	Inserts element at index
list.pop()	Removes and returns the element at the end of the list
list.pop(index)	Removes and returns the element at index

The **append** method takes the new element as an argument and adds it the the end of the list. The method **insert** does something similar but it takes the index and new element as arguments and adds the element to the given position by shifting the remaining elements to the right. The **extend** method takes another list and adds those elements to the end of the list.

```
>>> e = [1, 2, 3, 4, 5, 6]
>>> e.append(7)
>>> e
[1, 2, 3, 4, 5, 6, 7]
>>> e.extend([8, 9, 10])
>>> e
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> e.insert(5, 5.5)
>>> e
[1, 2, 3, 4, 5, 5.5, 6, 7, 8, 9, 10]
```

The **pop** method is used to remove an element from a list. If the index position is not specified, **pop** removes and returns the last element of the list. If a position is specified, **pop** removes the element at that location and returns it. Remaining elements would then be shifted one position to the left.

```
>>> e
[1, 2, 3, 4, 5, 5.5, 6, 7, 8, 9, 10]
>>> e
[1, 2, 3, 4, 5, 5.5, 6, 7, 8, 9, 10]
>>> e.pop()
10
>>> e.pop(4)
5
>>> e
[1, 2, 3, 4, 5.5, 6, 7, 8, 9]
```

6.1.3 Sorting

The **list** object has a **sort** method that will arrange its elements in numeric or alphabetical order.

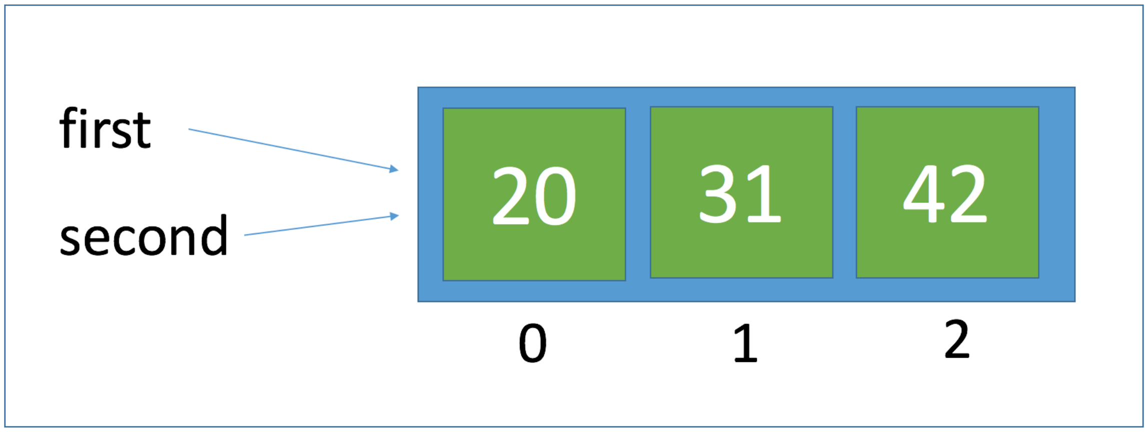
```
>>> f = [23, 12, 500, 3.3, 42, 92, 7]
>>> f
[23, 12, 500, 3.3, 42, 92, 7]
>>> f.sort()
>>> f
[3.3, 7, 12, 23, 42, 92, 500]
```

6.1.4 Aliasing

Not all variable names refer to different variables. When two identifiers refer to the same variable (and therefore value), this is known as an **alias**.

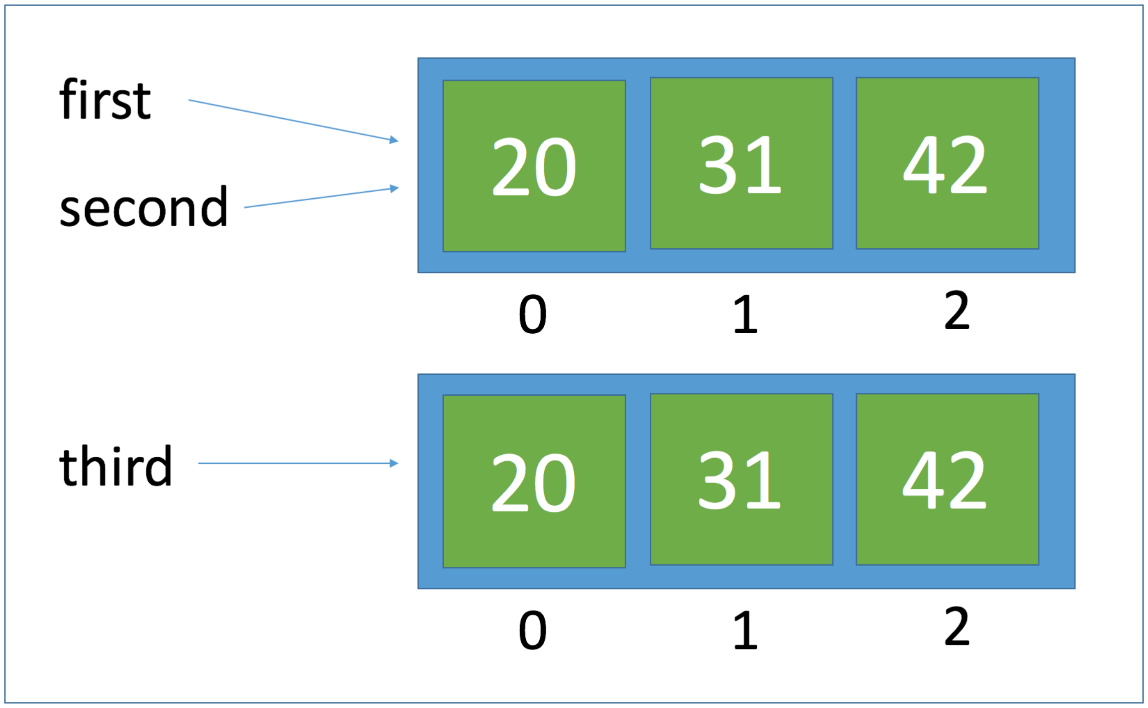
```
>>> first = [20, 31, 42]
>>> second = first
>>> first
[20, 31, 42]
>>> second
[20, 31, 42]
>>> second.append(53)
>>> second
[20, 31, 42, 53]
>>> first
[20, 31, 42, 53]
```

In the example above, a single list object is created with two names, or aliases. When an element is appended to the **second** list, the **first** list changed also. This happens because the variables **first** and **second** refer to the exact same list object.



If you do not want an alias, you can pass the source list to a call of the **list** function.

```
>>> third = list(first)
>>> third
[20, 31, 42, 53]
```



6.1.5 Equality

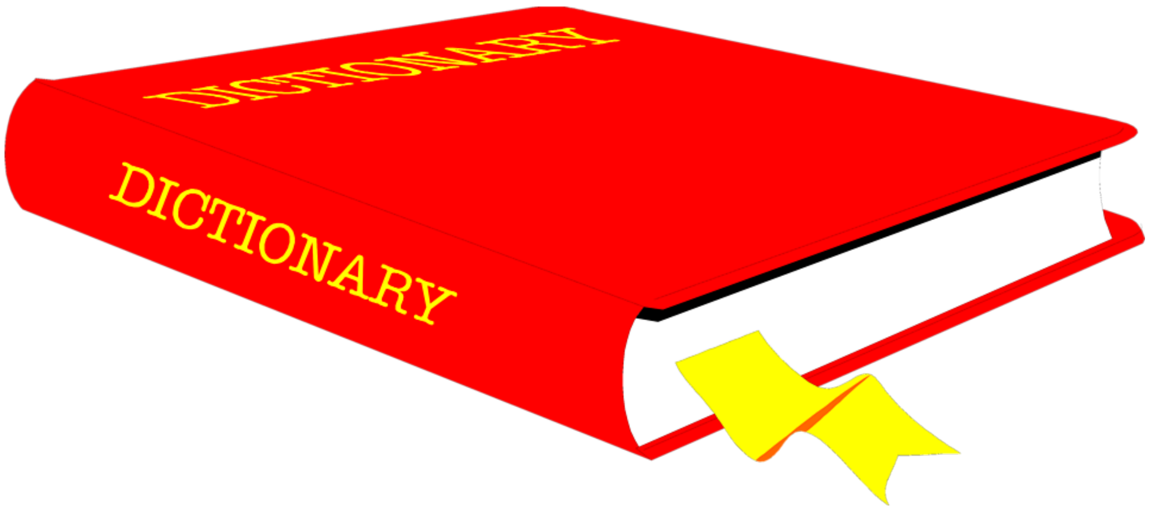
Frequently, programmers need to check for equality between variables (are the values equal). There could also be circumstances when you need to know not only if the values are equal but do the variables refer to the same object.

The **==** operator returns **True** if two values are equal, or the lists have the same structural equivalence. The Python **is** operator returns **True** if two variables refer to the same object.

```
>>> first == second
True
>>> first == third
True
>>> first is second
True
>>> first is third
False
```

6.2 Dictionaries

A **dictionary** is a collection which is unordered, changeable and indexed.



6.2.1 Dictionary structure

A dictionary, or dict, organizes data values by association with other data values rather than by sequential position. A dictionary is a collection which is unordered, changeable and indexed. In Python dictionaries are written with curly brackets, and they have keys and values. Dictionaries are initialized using curly braces **{ }**.

```
>>> capitals = {'United States': 'Washington, DC','France': 'Paris','Italy': 'Rome'}
>>> capitals
{'United States': 'Washington, DC', 'France': 'Paris', 'Italy': 'Rome'}
>>> type(capitals)
<class 'dict'>
>>> capitals['Italy']
'Rome'
>>> capitals['Spain'] = 'Madrid'
>>> capitals
{'United States': 'Washington, DC', 'France': 'Paris', 'Italy': 'Rome',
'Spain': 'Madrid'}
>>> 'Germany' in capitals
False
>>> 'Italy' in capitals
True
>>> morecapitals = {'Germany': 'Berlin','United Kingdom': 'London'}
>>> capitals.update(morecapitals)
>>> capitals
{'United States': 'Washington, DC', 'France': 'Paris', 'Italy': 'Rome',
'Spain': 'Madrid', 'Germany': 'Berlin', 'United Kingdom': 'London'}
>>> len(capitals)
6
```

6.2.2 Dictionary methods

Python has a set of built-in methods that you can use on dictionaries.

Dictionary Method	Results
dict.copy()	Returns a copy of the dictionary
dict.fromkeys()	Returns a dictionary with the specified keys and values
dict.get()	Returns the value of the specified key
dict.items()	Returns a tuple for each key value pair
dict.keys()	Returns the dictionary’s keys

dict.pop()	Removes the element with the specified key
dict.popitem()	Removes the last key-value pair
dict.setdefault()	Returns the value of the specified key. If the key does not exist: inser
dict.update()	Updates the dictionary with the specified key-value pairs
dict.values()	Returns a list of all the values in the dictionary

Let's try some of these functions.

```
>>> capitals.items()
dict_items([('United States', 'Washington, DC'), ('France', 'Paris'),
('Italy', 'Rome'), ('Spain', 'Madrid'), ('Germany', 'Berlin'),
('United Kingdom', 'London')])
>>> capitals.keys()
dict_keys(['United States', 'France', 'Italy', 'Spain', 'Germany',
'United Kingdom'])
>>> capitals.values()
dict_values(['Washington, DC', 'Paris', 'Rome', 'Madrid', 'Berlin', 'London'])
>>>
```

6.2.3 Loop through a Dictionary

As an iterable object, we can loop, or iterate, through a dictionary using a **for** loop. With the **for** loop we can execute a set of statements, once for each item in a list, tuple, set etc.

```
>>> for key in capitals.keys():
...     print(key)
...
United States
France
Italy
Spain
Germany
United Kingdom

>>> for value in capitals.values():
...     print(value)
...
Washington, DC
Paris
Rome
Madrid
```

```
Berlin
London

>>> for key, value in capitals.items():
...     print(key, value)
...
United States Washington, DC
France Paris
Italy Rome
Spain Madrid
Germany Berlin
United Kingdom London
```

Box 6.1. Reading from a file

In Python you need to give access to a file by opening it. You can do it by using the open() function. Open returns a file object, which has methods and attributes for getting information about and manipulating the opened file.

Using the **with** statement, you get better syntax and exceptions handling. In addition, it will automatically close the file. The with statement provides a way for ensuring that a clean-up is always used.

```
>>> file = open("welcome.txt")
>>> data = file.read()
>>> print(data)
Hi, this is a text file!

>>> file.close()  # It's important to close the file when you're done with it
```

Opening a file using with is as simple as: `with open(filename) as file`:

```
>>> with open("welcome.txt") as file: # Use file to refer to the file object
...     data = file.read()
...     print(data)
```

Chapter 7

Control Statements

I've got my own mind. Wanna make my own decisions. When it has to do with my life. I wanna be the one in **control**. —Janet Jackson, *Control*

7.1 Conditional Statements

It is very common for programs to execute statements based on some conditions. In this section we will learn about Python's **if**, **else**, and **elif** statements.

Box 7.1. Relational operators

Relational operators are used to compare values. It either returns **True** or **False** according to the condition.

Operator	Meaning
>	Greater than - True if left operand is greater than the right
<	Less than - True if left operand is less than the right
==	Equal to - True if both operands are equal
!=	Not equal to - True if operands are not equal
>=	Greater than or equal to - True if left operand is greater than or equal to the right
<=	Less than or equal to - True if left operand is less than or equal to the right

7.1.1 The *if* statement

The **if** statement evaluates a boolean condition. If the results of the evaluation is **True**, the code block following the **if** statement is executed. If the evaluation is **False**, the code block is not executed.

Listing 7.1: control.py

```python
today = "Tuesday"

if today == "Tuesday":
  print("We get to write code!")
```

Create a file using the code above and run the command `python control.py` in your terminal.

```
$ python control.py
We get to write code!
```

7.1.2 Adding *else*

The **else** statement only works when following an **if** statement and the block code following the **else** gets executed only when the **if** statement evaluates to **False**.

Listing 7.2: control.py (modified)

```
today = "Wednesday"

if today == "Tuesday":
  print("We get to write code!")
else:
  print("Time to learn Scrum!")
```

Running the modified file, should provide this output:

```
$ python control.py
Time to learn Scrum!
```

7.1.3 The *elif* statement

Similar to the **else** statement, an **elif** statement only works when following an **if** statement but the **elif** allows you to evaluate another boolean condition. If the **elif** statement evaluates to **True** the code block following will be executed.

Box 7.2. Membership operators

Keywords **in** and **not in** are membership operators in Python. They are used to test whether a value or variable is found in a container (string, list, or dictionary). In a dictionary we can only test for presence of key, not the value.

Operator	Meaning	Example
in	True if value/variable is found in the sequence	5 in x
not in	True if value/variable is not found in the sequence	5 not in x

Listing 7.3: control.py (modified)

```
today = "Saturday"
code_days = ["Tuesday", "Thursday"]
scrum_days = ["Monday", "Wednesday", "Friday"]

if today in code_days:
  print("We get to write code!")
elif today in scrum_days:
  print("Time to learn Scrum!")
else:
  print("Time to relax!")
```

Make the above changes to the control.py file and run it in the terminal.

```
$ python control.py
Time to relax!
```

Let's use the **if-elif-else** statement combination to help with assigning grades.

Listing 7.4: letter_grade.py (modified)

```
num = int(input("Enter your numeric grade: "))
if num >= 90:
  letter_grade = 'A'
elif num >= 80:
  letter_grade = 'B'
elif num >= 70:
  letter_grade = 'C'
elif num >= 60:
  letter_grade = 'D'
else:
  letter_grade = 'F'

print("Your letter grade is " + letter_grade )
```

7.2 For Loops

Thus far, we have limited ourselves to short sequences of instructions that are executed one after the other. Next we will introduce repetition statements which repeat an action. Each repetition of the action is known as a **pass** or an **iteration**.

The Python **for** loop is a control statement that supports a definite number of iterations.

Listing 7.5: for_loop.py

```python
weekdays = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday"]

for day in weekdays:
  print(day)
```

for loops are also great for iterating over numbers and making calculations.

Listing 7.6: avg_loop.py

```python
l = [5, 6, 10, 20]

sum = 0
for num in l:
    sum += num

avg = sum / len(l)
print(avg)
```

7.3 While Loops

The **for** loop executes for a definite number of iterations. In some situations, the number of iterations needed is not predictable. The logic needed requires a condition to be met before completing its work.

Listing 7.7: blackjack.py

```
import random

card_values = [2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 10, 10, 11]
sum = 0
turn = "hit"
while turn == "hit":
  sum += random.choice(card_values)
  print("You currently have: " + str(sum))
  turn = input("What do you want to do? ")

if sum >= 21:
  print("You went over 21!")
else:
  print("You stopped at: " + str(sum))
```

As a final note on **while** loops, the command **break** can also be used to exit a loop.

7.4 Nested Loops

Loops can be nested in that you can put any type of loop inside of any other type of loop. For example a for loop can be inside a while loop or vice versa. Let's expand the the simple Blackjack code such that it allows us to play multiple games.

Listing 7.8: blackjack.py (modified)

```
import random

card_values = [2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 10, 10, 11]
play_again = 'yes'
while play_again == 'yes':
  sum = 0
  turn = "hit"
  while turn == "hit":
    sum += random.choice(card_values)
    print("You currently have: " + str(sum))
    turn = input("What do you want to do? ")

  if sum >= 21:
    print("You went over 21!")
  else:
    print("You stopped at: " + str(sum))

  play_again = input("Want to play again? ")
```

Chapter 8

Files

Thus far, most of our input & output has been limited to the command line. In many situations, we will need to work with larger sets of data. Python easily allows for reading from and writing to files.

The **open()** method returns a file object, and is most commonly used with two arguments: **open(filename, mode)**.

The first argument is a string containing the filename. The second argument is another string containing a few characters describing the way in which the file will be used. *mode* can be `'r'` when the file will only be read, `'w'` for only writing, and `'a'` opens the file for appending; any data written to the file is automatically added to the end. `'r+'` opens the file for both reading and writing.

8.1 Reading from a file

As mentioned above to read a file, use the **open(filename, 'r')** method. For reading a file, the mode argument is optional; `'r'` will be assumed if its omitted.

Listing 8.1: read_file.py

```
file = open("text_file.txt")

for line in file:
    print(line)

file.close()
```

8.2 Writing to a file

When opening a file using **open(filename, 'w')**, the returned file object can be used to write information to the file but any existing files with the same name will be erased when this mode is activated.

Listing 8.2: write_file.py

```python
students = [ "Jennifer", "Craig", "Robert", "Boudreaux", "Thibodeaux"]
file = open("output_file.txt", "w")
file.write("Here is a list of names: \n")

for s in students:
    file.write(s + "\n")

file.close()
```

8.3 Appending a file

Similar to writing to a file, you can also append new text to the already existing file instead of overwriting an existing file.

Listing 8.3: append_file.py

```python
more_students = ["Sonja", "Melissa", "Karen"]
file = open("output_file.txt", "a")
file.write("Here are some more names:\n")

for s in more_students:
    file.write(s + "\n")

file.close()
```

8.4 The *with* statement

When finished working with a file, use the **close()** method to end the interaction with the file. This terminates the file object and makes the file available to other resources.

While it is a good programming habit to close files after use, we, as busy human beings, can sometimes forget to close a file after opening it.

Luckily, Python has a **with** keyword that helps us better manage file resources. It is good practice to use **with** when using file objects. The advantage is that the file is properly closed after its code block has completed.

Listing 8.4: with_open.py

```
with open("text_file.txt") as file:
    for line in file:
        print(line)
```

8.5 JSON

With files, the **read()** and **write()** methods only use strings. This can be problematic when trying to use files to store or share data that isn't easily converted to and from a string.

Rather than having users constantly writing and debugging code to save complicated data types to files, Python allows you to use the popular data interchange format called **JSON (JavaScript Object Notation)**. The standard module called `json` can take Python data hierarchies, and convert them to string representations; this process is called serializing. Reconstructing the data from the string representation is called deserializing.

Listing 8.5: create_json.py

```
import json

student_scores = {'name': 'John', 'scores': [90, 88, 92]}
with open("student_scores.json", "w") as file:
    json.dump(student_scores, file)
```

Listing 8.6: read_json.py

```
import json

with open("student_scores.json", "r") as file:
    student_scores = json.load(file)

print("The object type is: ", type(student_scores))
print(student_scores)
```

This simple serialization technique can handle lists and dictionaries, but serializing arbitrary class instances in JSON requires a bit of extra effort. The Python documents reference for the json module contains an explanation of this.

Box 8.1. Language of APIs

JSON is the language of devices communicating over the Internet. Roughly 80% of all existing APIs use JSON and it is very likely that over 95% of all new APIs are using JSON. Anyone looking for a career in technology or data should be familiar with the producing and consuming JSON.

Here is an example of a JSON feed from an OpenData portal: Baton Rouge Crime

8.6 Other file types

Data can be shared using multiple other file types, such as **.csv**, **.xls**, and **.xlsx** files. To properly read these specifically formatted files, Python usually requires additional packages that can easily be added with the **import** command.

Listing 8.7: mpg.py

```
import csv

with open('mpg.csv') as file:
    mpg_reader = csv.reader(file)
    for row in mpg_reader:
        print(row)
```

Chapter 9

Database Connections

Most modern applications contain some sort of database back-end. In general, users never directly interact with the database. Instead, the database is accessed indirectly through an external application that handles connections and logic.

9.1 Structured Query Language

Structured Query Language (SQL) is a data-specific language used in programming and designed for managing data held in a database management system (DBMS). As a programming language, SQL is very powerful when used for searching or manipulating data. While most DBMSs adhere to the ANSI SQL standards, there are varying dialects of SQL based on the organization creating the DBMS. Some DBMS platforms you may encounter:

- Oracle
- SQL Server

- MySQL
- PostgreSQL
- MongoDB
- SQLite

SQL statements can range from a very simple statement such as the example in code listing 9.1 to a more advanced example such as code listing 9.2

Listing 9.1: Basic SQL statement

```
SELECT * FROM employees;
```

Listing 9.2: Advanced SQL statement

```
SELECT
  e.employee_id AS "Employee #"
  , e.first_name || ' ' || e.last_name AS "Name"
  , e.email AS "Email"
  , e.phone_number AS "Phone"
  , TO_CHAR(e.hire_date, 'MM/DD/YYYY') AS "Hire Date"
  , TO_CHAR(e.salary, 'L99G999D99', 'NLS_NUMERIC_CHARACTERS = ''.,
    '' NLS_CURRENCY = ''$''') AS "Salary"
  , e.commission_pct AS "Comission %"
  , 'works as ' || j.job_title || ' in ' || d.department_name
    || ' department (manager: ' || dm.first_name || ' ' || dm.last_name
    || ') and immediate supervisor: ' || m.first_name || ' '
    || m.last_name AS "Current Job"
  , TO_CHAR(j.min_salary, 'L99G999D99', 'NLS_NUMERIC_CHARACTERS = ''.,
    '' NLS_CURRENCY = ''$''') || ' - ' ||
      TO_CHAR(j.max_salary, 'L99G999D99', 'NLS_NUMERIC_CHARACTERS = ''.,
      '' NLS_CURRENCY = ''$''') AS "Current Salary"
  , l.street_address || ', ' || l.postal_code || ', ' || l.city || ', '
    || l.state_province || ', '
    || c.country_name || ' (' || r.region_name || ')' AS "Location"
  , jh.job_id AS "History Job ID"
  , 'worked from ' || TO_CHAR(jh.start_date, 'MM/DD/YYYY') || ' to '
    || TO_CHAR(jh.end_date, 'MM/DD/YYYY') ||
    ' as ' || jj.job_title || ' in ' || dd.department_name
    || ' department' AS "History Job Title"
    FROM employees e
    -- to get title of current job_id
```

```
      JOIN jobs j
        ON e.job_id = j.job_id
    -- to get name of current manager_id
      LEFT JOIN employees m
        ON e.manager_id = m.employee_id
    -- to get name of current department_id
      LEFT JOIN departments d
        ON d.department_id = e.department_id
    -- to get name of manager of current department
    -- (not equal to current manager and can be equal to the employee itself)
      LEFT JOIN employees dm
        ON d.manager_id = dm.employee_id
    -- to get name of location
      LEFT JOIN locations l
        ON d.location_id = l.location_id
      LEFT JOIN countries c
        ON l.country_id = c.country_id
      LEFT JOIN regions r
        ON c.region_id = r.region_id
    -- to get job history of employee
      LEFT JOIN job_history jh
      ON e.employee_id = jh.employee_id
    -- to get title of job history job_id
    LEFT JOIN jobs jj
      ON jj.job_id = jh.job_id
    -- to get namee of department from job history
    LEFT JOIN departments dd
      ON dd.department_id = jh.department_id

ORDER BY e.employee_id;
```

Being familiar with multiple programming languages is a good thing and I would encourage interested programmers to learn as much SQL as they feel comfortable learning. Learning some SQL will make application developers better at thinking about data and interacting with a DBMS.

The downside is that maintaining a proficiency in multiple languages can be difficult…especially, in a fast changing sector!

Luckily, we are part of an open source community that seeks to improve our abilities.

9.2 SQLAlchemy

SQLAlchemy is a Python package that provides a nice Pythonic way of interacting with databases. So rather than dealing with the differences between specific dialects of traditional SQL, you can leverage the Pythonic framework of SQLAlchemy to streamline your workflow and more efficiently query your data.

The SQLAlchemy package may need to be installed before use. Using the Python package manager **pip**, type `pip install sqlalchemy` in your terminal.

```
$ pip install sqlalchemy
Collecting sqlalchemy
  Downloading https://files.pythonhosted.org/packages/25/c9/b0552098cee325425a61efd
  f380c51b5c721e459081c85bbb860f501c091/SQLAlchemy-1.2.12.tar.gz (5.6MB)
    100% || 5.6MB 226kB/s
Building wheels for collected packages: sqlalchemy
  Running setup.py bdist_wheel for sqlalchemy ... done
  Stored in directory: /Users/james/Library/Caches/pip/wheels/ed/bd/2e/d3874a6e97b8
  cc71e7e177c8d065ead30f67f380c4d9bbadaa
Successfully built sqlalchemy
Installing collected packages: sqlalchemy
Successfully installed sqlalchemy-1.2.12
```

SQLAlchemy includes the dialects for more popular DBMS platforms and has extensions for some of the more obscure databases.

9.3 Connecting to a database

To start interacting with the database we first we need to establish a connection.

Listing 9.3: Example Connection

```python
import sqlalchemy as db
engine = db.create_engine('dialect+driver://user:pass@host:port/db')
```

9.4 Viewing Table Details

SQLAlchemy can be used to automatically load tables from a database using something called reflection. Reflection is the process of reading the database and building the metadata based on that information.

Listing 9.4: census_metadata.py

```
import sqlalchemy as db

engine = db.create_engine('sqlite:///census.sqlite')
connection = engine.connect()
metadata = db.MetaData()
census = db.Table('census', metadata, autoload=True, autoload_with=engine)

print(census.columns.keys())

print(repr(metadata.tables['census']))
```

9.5 Querying

SLQAlchemy supports querying of data from a DBMS. If needed or desired, you can open a connection and use SQL for the query.

Listing 9.5: census_sql.py

```
import sqlalchemy as db

engine = db.create_engine('sqlite:///census.sqlite')
connection = engine.connect()
results = connection.execute("select * from census")
for row in results:
    print(row)

connection.close()
```

A better solution would be to allow SQLAlchemy to handle the SQL statement.

Listing 9.6: census_query.py

```
import sqlalchemy as db

engine = db.create_engine('sqlite:///census.sqlite')
connection = engine.connect()
metadata = db.MetaData()
census = db.Table('census', metadata, autoload=True, autoload_with=engine)

# Equivalent to 'SELECT * FROM census'
query = db.select([census])

proxy = connection.execute(query)
print(proxy)

result = proxy.fetchall()

print(result[:10])
```

proxy: The object returned by the .execute() method. It can be used in a variety of ways to get the data returned by the query.

results: The actual data asked for in the query when using a fetch method such as *.fetchall()* on a ResultProxy.

9.5.1 Dealing with Large ResultSet

On occasion, you may encounter very large tables. SQLAlchemy has *.fetchmany()* to load optimal no of rows and overcome memory issues in case of large datasets

9.6 Filter

Data can be filtered.

Listing 9.7: census_filter.py

```
import sqlalchemy as db

engine = db.create_engine('sqlite:///census.sqlite')
```

```
connection = engine.connect()
metadata = db.MetaData()
census = db.Table('census', metadata, autoload=True, autoload_with=engine)

# Equivalent to:
# SELECT * FROM census WHERE sex = F
#query = db.select([census]).where(census.columns.sex == 'F')

# Equivalent to:
# SELECT state, sex FROM census WHERE state IN (Texas, New York)
query = db.select([census.columns.state, census.columns.sex])
        .where(census.columns.state.in_(['Texas', 'New York']))

proxy = connection.execute(query)
print(proxy)

result = proxy.fetchall()

print(result[:10])
```

9.7 Join

If you have two tables that already have an established relationship, you can automatically use that relationship by just adding the columns we want from each table to the select statement.

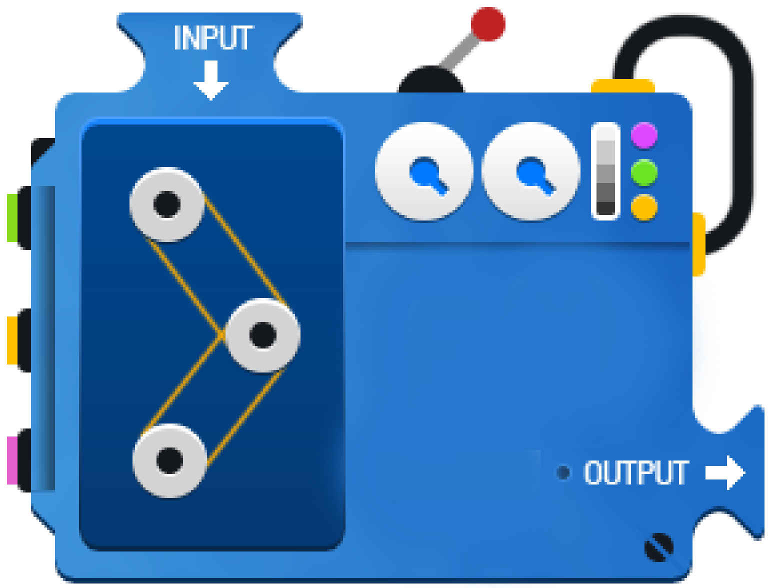
Listing 9.8: census_join.py

```
import sqlalchemy as db

engine = db.create_engine('sqlite:///census.sqlite')
connection = engine.connect()
metadata = db.MetaData()
census = db.Table('census', metadata, autoload=True, autoload_with=engine)
state_fact = db.Table('state_fact', metadata, autoload=True, autoload_with=engine)

query = db.select([census.columns.pop2008, state_fact.columns.abbreviation])
result = connection.execute(query).fetchall()

print(result[:10])
```

Chapter 10

Functions

Most programs perform tasks that can be broken down into several subtasks. For this reason, programmers usually break down programs into small manageable pieces known as functions. A *function* is a group of statements, known as a code block, that exist within a program for the purpose of performing a specific task. Instead of writing a large unstructured sequence of commands, it should be written as several smaller functions with each one performing a specific part of the overall task.



Also, functions are a key way to define interfaces so programmers can reuse or share their code.

10.1 Built-in Functions

Python has several built-in functions that you have already been using.

```
>>> len('Hello World!')
12
>>> type(2.0)
<class 'float'>
>>> round(3.14)
3
>>> round(3.14, 1)
3.1
```

10.2 Writing Functions in Python

As we have seen on previous tutorials, Python makes use of blocks.

A block is a area of code of written in the format of:

```
block_head():
    first line of code
    second line of code
    ...
```

Box 10.1. naming_conventions

Most programming languages have a naming convention that is used for creating the names of variables and functions. Two popular conventions are *snake case* and *camel case*. **Snake case**, also called *underscore case*, uses an underscore '_' when naming a function that contains multiple words. An example would be 'compute_pay()' or 'complete_order()'.

The **camel case** naming convention uses capitalization of the first letter of following words. Using the same examples from above would be ‘computePay()’ or ‘completeOrder()’.

The Python community predominately uses *snake_case*.

10.3 Void Functions

A *void function* executes a code block and then terminates. It does not return anything.

Listing 10.1: function_print.py

```python
# Define the function
def message():
  print('I like coding.')
  print('Python is my favorite language!')

# Call the function
message()
```

The program above only has one function, but it is common to define multiple functions in a program. There is usually a *main* function that is called when the program starts. The main fuction would then call other functions.

Listing 10.2: function_two.py

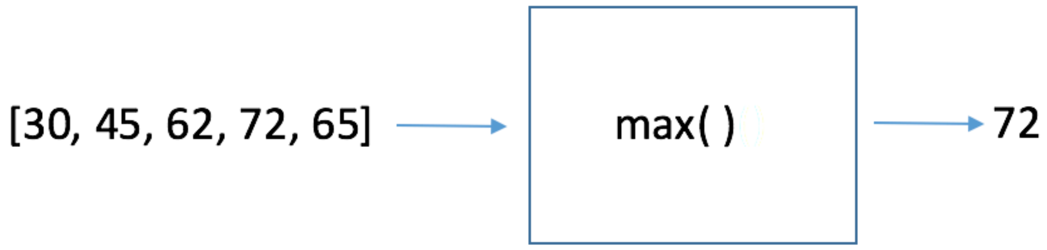
```python
# This program has two functions.
# First, define the main() function.
def main():
  print('I have a message for you.')
  message()
  print('See you later!')

# Define the function
def message():
  print('I like coding.')
  print('Python is my favorite language!')
```

```
# Call the function
main()
```

10.4 Value-returning Functions

As the name sounds, *value-returning functions* return a value when called. The returned value can be any of the variable types we've covered (int, str, float, dict, list) or an object.

Let's write an actual function for finding the maximum value from a list of numbers. For this example, we will use the height, in inches, for my family.



Listing 10.3: function_max.py

```
def main():
  fam = [40, 55, 62, 72, 65]
  tallest = max(fam)
  print("The tallest is ", tallest)

def max(l):
  m = l[0]
  for h in l:
    if h > m:
      m = h
  return m

main()
```

Functions can accept input, execute code, return values, or call other functions.

10.5 Passing Arguments to Functions

Sometimes it is useful not only to call a function, but also to send some data to the function. Additional information sent to the function is called *arguments*.

Listing 10.4: function_multi_arguments.py

```python
# This program demonstrates a function that accepts two arguments.
def main():
  print('The sum of 12 and 45 is')
  calculate_sum(12, 45)

# The calculate_sum function accepts two arguments.
def calculate_sum(num1, num2):
  result = num1 + num2
  print(result)

main()
```

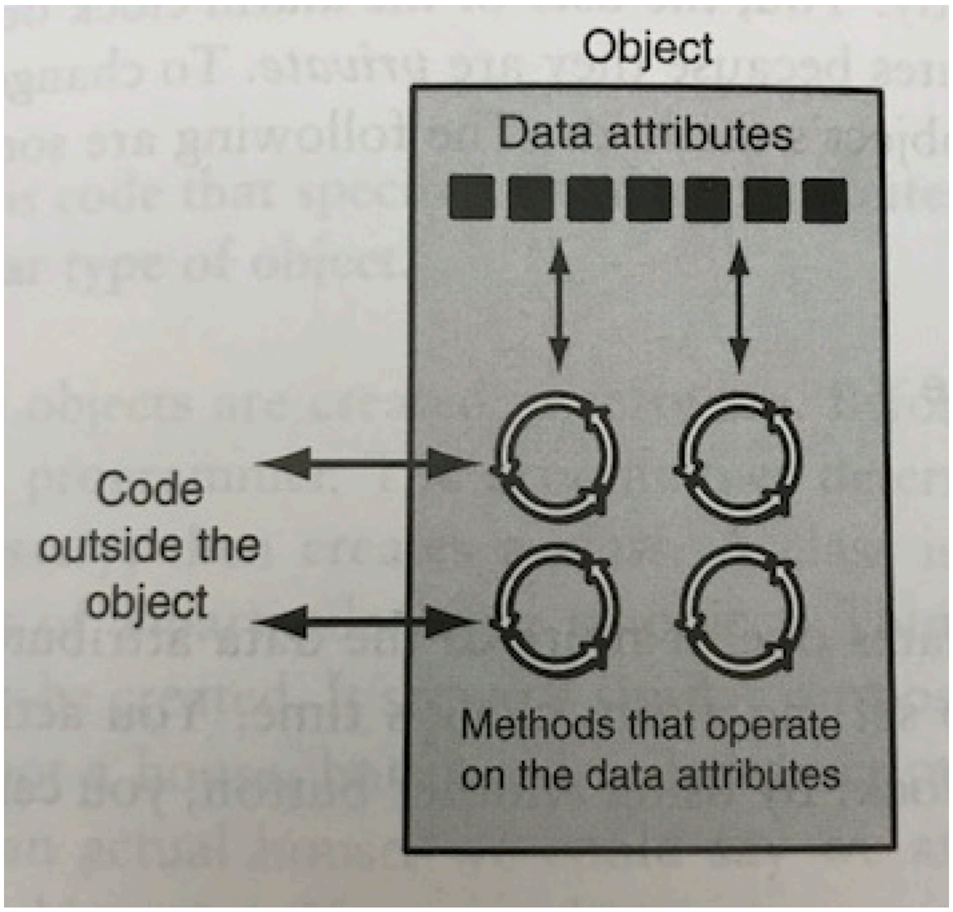
Chapter 11

Classes

In science, we use classifications to help describe the objects we are studying. Zoologist use terms such as mammals, birds, fish, reptiles, amphibians in the description of animals. This is the **class** of the animal. Knowing an animals class will provide a basic understanding of that species.

In *object oriented programming*, classes provide a means of bundling data and functionality together. An *object* is a software entity that contains data and the ability to execute functions. The data contained in an object is known as the object's *data attributes*. Those attributes are simply variables that reference data. The functions that an object can call are known as *methods*.

11.1 Data & code

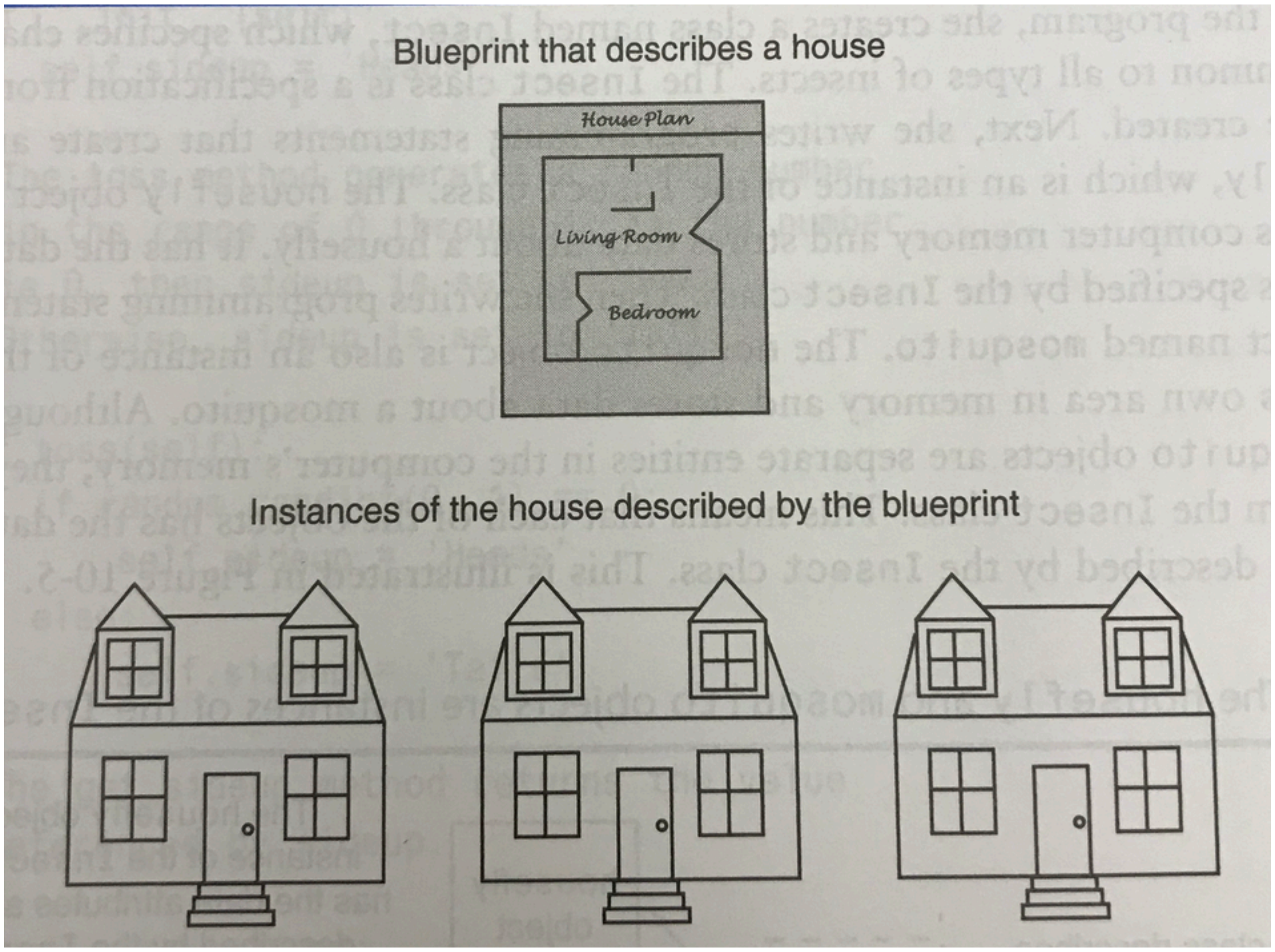


Utilizing objects helps to separate code from the data. Before object oriented programming, code was written in a more procedural format with variables and other data structures, such as lists or dictionaries. Changes at an application's data-layer could cause multiple errors depending on the application size. With objects, changes in the database are handled inside the object's class thus minimizing the effort required to accommodate changes.

11.2 Object reusability

In addition to solving problems of code and data separation, object oriented programming empowers developers to reuse objects. An object is not meant to be a stand-alone program but it can be reused in multiple other programs.

11.3 Defining the object


Blueprint that describes a house
House Plan
Living Room
Bedroom
Instances of the house described by the blueprint

Objects are defined a class. A class is code that specifies the data attributes and methods of a particular type of object. Think of a class as a "blueprint" from which an object is created. In construction, a blueprint is used to describe a structure, such as a house, that will be built. The blueprint itself is not a house object, but a description of a house. When the blueprint is used to build an actual house, it can be called an *instance* of the house described by the blueprint.

Perhaps an even simpler example would be a cookie cutter and a cookie. The cookie cutter would be the class that is used to create the cookies.

Listing 11.1: class_dog.py

```python
class Dog:

    # the __init__ method initializes the object.
    def __init__(self, name):
        self.name = name
        self.tricks = []    # creates a new empty list for each dog

    def add_trick(self, trick):
        self.tricks.append(trick)
```

To view the class, we will use the REPL / interactive Python. Inside a terminal, type 'python'.

```
>>> from class_dog import Dog
>>> d = Dog('Eli')
>>> type(d)
<class 'class_dog.Dog'>
>>> d.name
'Eli'
>>> d.tricks
[]
>>> d.add_trick("roll over")
>>> d.add_trick("fetch ball")
>>> d.tricks
['roll over', 'fetch ball']
```

So, a class is a description of an object's characteristics. When used in a program, a class can be used to create one or more objects. Each object created from a class is called an *instance* of the class.

11.4 Class definitions

To create a class, we write a *class definition*. A class definition is a set of statements that define a class's methods and data attributes.

Suppose you are writing an application to simulate a Magic 8 Ball. In the application, we need to repeatedly 'shake' the Magic 8 Ball and each time return an answer. Let's see what that class might look like.

Listing 11.2: class_magic8ball.py

```python
import random

class Magic8Ball:
    answers = ["It is certain.", "It is decidedly so.", "Without a doubt.",
               "Yes - definitely.", "You may rely on it.", "As I see it, yes.",
               "Most likely.", "Outlook good.", "Signs point to yes.",
               "Yes.", "Reply hazy, try again", "Ask again later.",
               "Better not tell you now.", "Cannot predict now.",
               "Concentrate and ask again.", "Cannot predict now.",
               "Concentrate and ask again.", "Don't count on it.",
               "My reply is no.", "My sources say no.",
               "Outlook not so good.", "Very doubtful."]

    # the __init__ method initializes the object.
    def __init__(self):
        self.answer = ""

    def shake(self):
        self.answer = random.choice(self.answers)

    def get_answer(self):
        return self.answer
```

Get ready for some Magic 8 Ball action and use the interactive Python in your terminal.

```
>>> from class_magic8ball import Magic8Ball
>>> ball = Magic8Ball()
>>> type(ball)
<class 'class_magic8ball.Magic8Ball'>
>>> ball.get_answer()
''
>>> ball.shake()
>>> ball.get_answer()
'Yes - definitely.'
```

11.5 Class inheritance

Classes can inherit from other classes. A class can inherit attributes and methods from another class, called the *superclass*. A class which inherits from a