Beginning Python

WELCOME!

TODAY'S CLASS OBJECTIVES

Today

- **Course Overview** - Tips and goals
- **Intro to Terminal** - basics of working with the terminal on Mac or Windows machine
- **Git and Github** - Version control of our code for collaborating and done with Terminal. Useful for your online resume
- **'Hello World' -** writing our first program in Jupyter/Anaconda, committing it to your personal repository

0

Course Overview

12 weeks

YOU SHOULD HAVE INSTALLED

Install These in the Background

- **Anaconda** - for Jupyter notebooks
- **Git** - Windows or Mac - git-scm.com
- **GitHub Account** - Setup an online account

And be able to get to Terminal...

- **Terminal -** Windows or Mac

COURSE GOALS

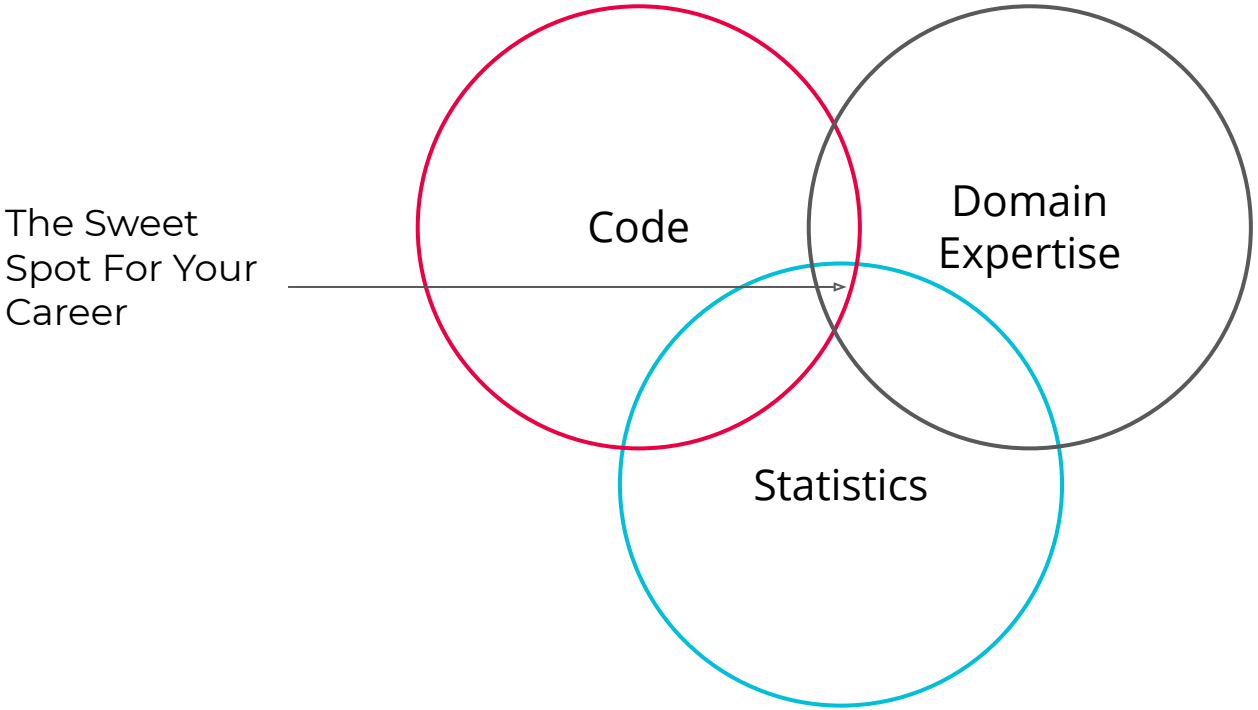
The 80/20 Python Summer

- **Know Your Tools** - Basics: Terminal, Git, GitHub, and Anaconda. Can setup a repository, pull/push code, work with notebooks

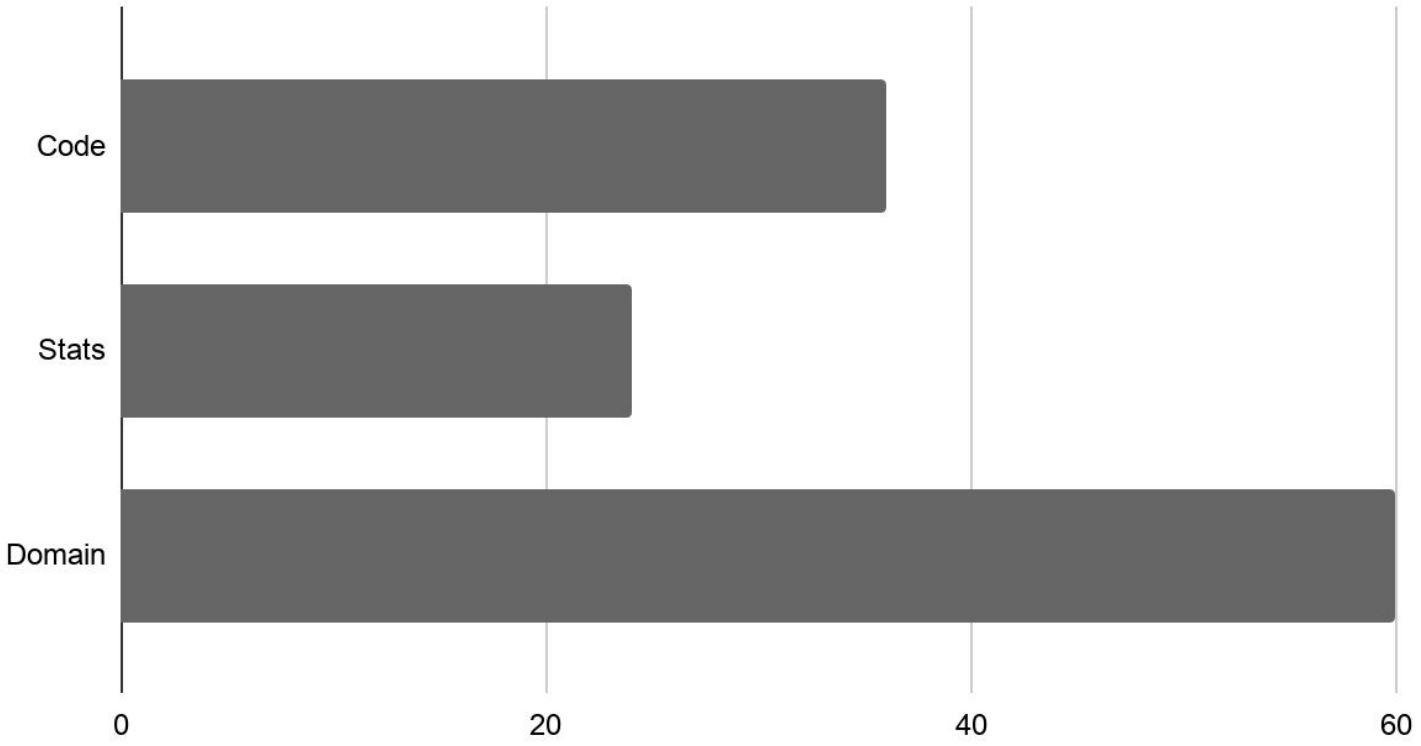
- **Python** - Familiar with how Python works (logic and syntax) to use 20% for 80% results. Can look at a chunk of sample code or write your own to answer a question of data

- **Data & Visualization** - Can wrangle data, clean it, and visualize it

THE BIG PICTURE

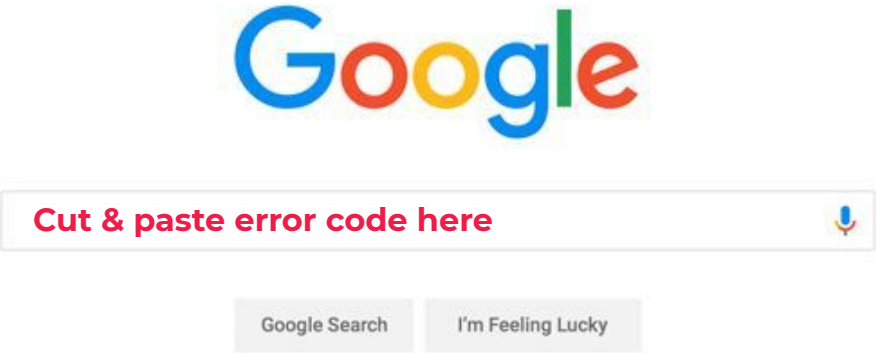


(Rough) Timelines for Proficiency



OUR GOAL

Getting really good at Googling (seriously)



HOW TO LEARN

Pick a Project

1. **Start Brainstorming** - What might be an interesting question to answer of some data that matters to *you*?

2. **Write It Out** - Pen and paper is more powerful than the keyboard as it forces you to think through each step

3. **Observe & Assemble** - Identify what concepts are tough and then assemble your own resources via Google Doc (YouTube, Kaggle, etc)

4. **Plan for Forgetting** - You'll forget a bunch. Build periodic review time into your schedule

SPECIFICALLY

Tips That Help

1. **Pomodoro Technique** - Study for 25 min and take 5 min break. Repeat for up to 2 hours to stay focused.

2. **Code + Review Everyday** - bite-size is easier than all you can eat

3. **Be Patient** - Lot's of ways to solve problems and when Googling you'll think everyone is a genius when you see elegant code

4. **Design Your Learning Program** - Per #2 - Notes? Flashcards? Custom cheatsheets? Modify as needed for how you learn best

NITTY GRITTY

Phase I Foundations - Weeks 1-6

1. **Workflows** - Learning Your Tools
2. **Python Basics** - Variables, data types, functions, data structures

Phase II Data - Weeks 7-10

3. **Manipulation** - Munging and wrangling data
4. **Dataframes and Viz** - Pandas and visualization

Phase III Final Project - Weeks 11-12

5. **Inspiration to Insight** - Full Jupyter Notebook working end to end

1

Terminal

Controlling without a mouse

Terminal

Why This Matters

Opens up a ton of functionality for Python libraries, version control (Git), and working with data/files super efficiently.
(And all the cool tech kids do it this way)

1. No different than mouse
2. Based off Unix (before we had GUI's)
3. A few key commands gets you 80% of the way there
4. Makes you much more efficient with GitHub, scripting, etc.
5. $ means that is the command prompt line

TERMINAL KEY COMMANDS

Getting Around Your Computer

- **$ pwd -** The "Where the #&@% am I?" command. Stands for Print Working Directory. A directory is just like a folder on your computer

- **$ ls** - lists all the files in that directory (folder). **$ls -a** lists all the hidden files in a directory. Useful for any random files not organized

- **$ cd -** "change directory" - let's you navigate to another folder (just like double clicking does). Ex: **$ cd Your_Favorite_Folder** puts you in that folder on your machine.

- **$ cd /** is the "Toto take me home" command and take you to root

TERMINAL KEY COMMANDS - Your Turn

1. Open up your Terminal (Windows and Mac differ)
2. Figure out where you are (type '**pwd**') - do you recognize any files?
3. List all the files in your current directory including hidden ones (type '**ls**').
4. Bonus Points: Change to a different directory (hint: type "**cd folder/sub-folder**'). Then type "cd .." for the Toto take me home to the parents

Cont'd - TERMINAL KEY COMMANDS

Moving/Copying Stuff

- **$ mv -** "move" - **$ mv <file_name> <new_location>**

- **$ cp -** "copy" - copy a file or directory. Specify file name and where you want it copied to. Ex: **$ cp file.py DataScience/Terminal**

- **$ man -** "manual" - this is the owners manual, so if you can remember what a command does, simply go **$ man cp** - and it will spit back all the info

Cont'd - TERMINAL KEY COMMANDS

Making/Destroying Stuff On Your Computer

- **$ mkdir -** "make directory" - creates a folder for you in whatever directory you are in (which is why 'pwd' is important!)

- **$ touch -** 'touch by you, the creator' which creates life for a new file. Ex: $ touch ilovePython.py - creates a new file. Can double check by typing '$ ls' afterwords

- **$ rm -** "remove" - let's you remove files or directory (folder). Ex: **$ rm i_love_python.py**. **Use VERY CAREFULLY!** Note: Can remove a whole directory: **$ rm -r ThisDirectoryIsDead**

TERMINAL KEY COMMANDS - Your Turn

1. Identify where you are (**$ pwd).**
2. Navigate to an area you'd like to store your class files on (this can change later) **$ cd DIRECTORY NAME**
3. Create a new directory on your computer called "MSBA_Class1" **$ mkdir MSBA_Class1** this can live wherever you like on your machine
4. Verify this directory was created using **$ ls**
5. Create a new file using **$ touch sacrificial_file.py**
6. Verify this was created using **$ ls**
7. Remove this file using **$ rm Sacrifical_File.py**
8. Verify using **$ ls -a**

2

Git & GitHub

A Way to Save Your Code & Collaborate

Git & GitHub 101

WHAT THEY ARE

1. **Git -** Git is a free and open source distributed version control system designed to handle collaboration on code repositories. (Have to install on your machine).

2. **GitHub** - An online repository where you can store your code, collaborate, track changes, and save versions. (Have to sign up for a free account)

3 Parts to a Git Project

WHAT THEY ARE

1. **A Working Directory:** where files are created, edited, deleted, and organized (Hint: $ cd MSBA_Class1)
2. **A Staging Area:** where changes that are made to the working directory are listed
3. **A Repository:** where Git permanently stores changes as different versions of the project

The Git workflow consists of editing files in the working directory, adding files to the staging area, and saving changes to a Git repository.

Your Turn

LET'S DIVE IN

1. **Install Git** - On your computer (if you haven't already).

2. **GitHub** - An online repository where you can store your code, collaborate, track changes, and save versions. (Have to sign up for a free account)

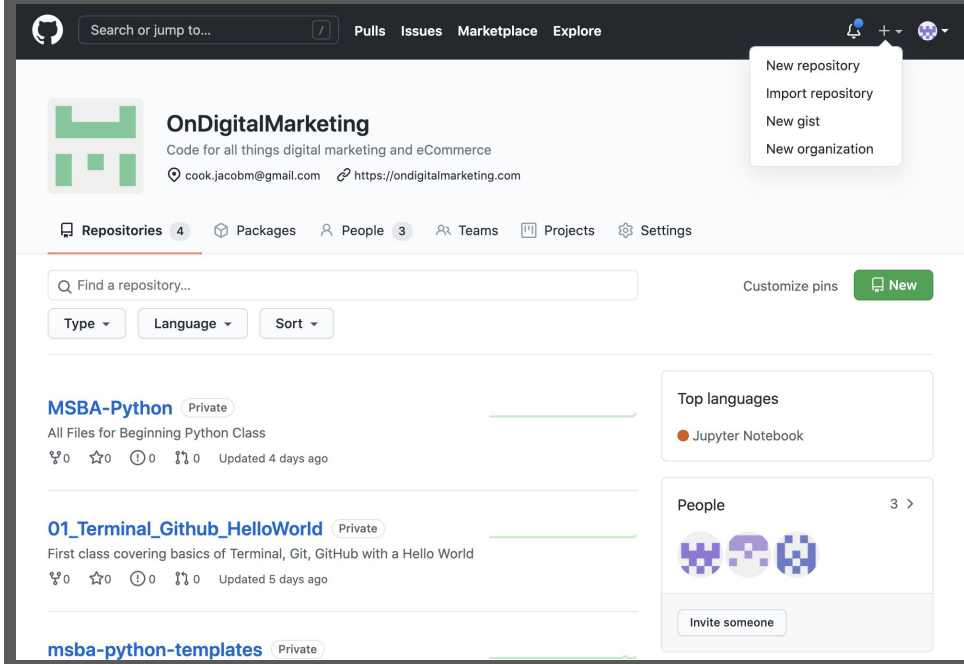
3. **Our First Repository**

1. Create a Repository on GitHub

Why This Matters

Let's us store all our code online

Step by Step:
https://docs.github.com/en/github/creating-cloning-and-archiving-repositories/creating-a-new-repository



1b. Or Locally

Why This Matters

The git init command creates or initializes a new Git project, or repository. It creates a .git folder with all the tools and data necessary to maintain versions. This command only needs to be used once per project to complete the initial setup.

$ git add . -adds everything to our repo

Source: Github

```
$ cd /MSBA_Class1

$ git init

$ git add .
```

2. Clone Your Repo on GitHub

Why This Matters

We have a way to match GitHub to our local machine and all the Git goodness for version control

Step by Step:
https://docs.github.com/en/github/creating-cloning-and-archiving-repositories/cloning-a-repository

```
$ cd /MSBA_Class1

$ git clone
https://github.com/YOUR
-USERNAME/YOUR-REPOSITO
RY
```

3. Fire Up Anaconda

Why This Matters

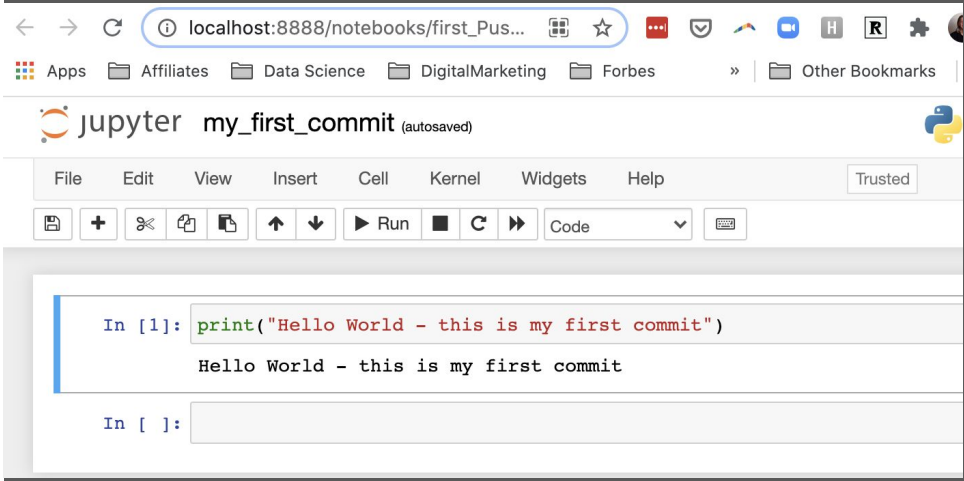
Here's where we'll code our first Python lines

```
$ jupyter notebook
```

4. Hello World

Why This Matters

A basic file we'll save to our repo




```
print("Hello World - this is my first commit")
```

5. Push to GitHub

Why This Matters

We stage and then commit our work back up to GitHub to keep our versions under control

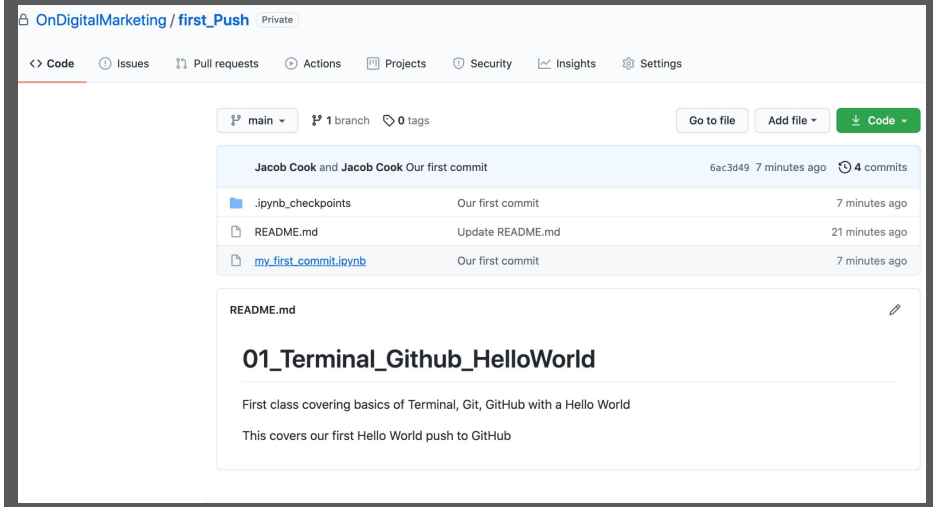
Step by Step:
https://docs.github.com/en/github/getting-started-with-github/pushing-commits-to-a-remote-repository

```
$ ls -la

$ git add -A

$ git commit -m “Our first commit!”

$ git push origin
```

6. Verify on GitHub

Why This Matters

Check to see our code made it up online like we expected




```
$ git status
```

KEY TAKEAWAYS

1. Develop Your Learning Plan & Rhythm
2. Terminal is a super power to master
3. Git + GitHub is foundational for MSBA and coding in general

For Next Week...

WHERE WE'RE GOING

1. **Tools -** Week 1 assignment will be very basic and the easiest of the course. We'll simply be building repos on GitHub, pulling/pushing, and moving around in Terminal.

2. **Coding** - We'll start coding and working with the basics

Required Prep for Week 2

- Read Ch 1: Python Basics on Automate the Boring Stuff (ignore references to Mu as we'll use Jupyter notebooks via Anaconda).
- Watch accompanying Video 2 on YouTube
- Optional: Byte of Python for brief overviews