Table of Contents

*(**Note:** One Lesson in the Table of Contents above does not necessarily equate to one 1 hr 15 min 1-on-1 session. It will vary based on the pacing that the student can keep up with.)*

*(**Note:** You can click on the "Show document outline" button ( ) on the left to navigate this document in addition to the Table of Contents above.)*

Introduction

Why Coding?
Learning to code will force you to think like a computer. You are forced to think critically about creating a flow of logical instructions to get a desired result. If you leave anything ambiguous or unclear, your code won't work.

Also, coding skills will always be a plus to have in any career path, and it will usually be a requirement. For example, if you would like to become a doctor, you will be doing a lot of research assistant jobs for research labs. In these positions, you will have to learn Python to analyze DNA sequences and molecular bio-data. If you would like to be a hardware-focused engineer who likes to build robots, you will still need to program instructions and logic for your hardware creations.

Why Python?
Python is by far the easiest and most intuitive programming language to learn in my opinion. Python takes care of a lot of annoying memory/storage-related problems automatically in the background. The syntax for Python is also much closer to normal English, and any syntax-related problems are easier to understand and fix. We can spend less time on the syntax details, and more time on core coding concepts that you can use in any other programming language.

Why Data Analytics?
Data analytics projects are self-contained. You can do very cool projects by just downloading an Excel/csv file data set from the internet and conducting some data analysis with Python. In regards to careers, data analysis skills are also popular and valuable, and companies of all sizes are looking for people with these skills.

For web development projects, you would have to learn at least 3 languages (HTML, CSS, & JavaScript), and many, many tools on top of that. You would also have to get an understanding of how the internet and webpages work behind-the-scenes. Web development projects are great to pursue, but I don't think it is the best type of project space for someone new to coding. You can do a lot more with a lot less knowledge with data analysis.

Python has very powerful and popular tools for analyzing data. Therefore, data analytics projects are a natural next step after we understand the basics of coding.

Python Fundamentals

Lesson 1

(See lesson1.py)

Topics:

- Data Types
- Variables
- Operators
- Functions
- if-else statements
- For Loops

Data Types:

- int
- float
- string
- boolean

Variables:

- Dynamically-typed vs statically-typed

Operators:

- All are binary operators in the form **"x _ y"**
- + (Addition)
- - (Subtraction)
- * (Multiplication)
- Self Discover
  - // (Integer Divide)
  - / (Float Divide)
  - ** (Exponent)
  - % (Modulo)

Functions:

- Give input, get some output.
- Can use functions repeatedly. Don't have to re-write code each time.

if, else-if, else Statements:

- Helps control the flow of logic in our code.
- Comparators, logical operators, parentheses help define conditionals

For-Loops:

- Makes the computer do something repeatedly for some defined number of iterations.

Other: print(), type(), indenting, casting, parentheses, range()

Problem Set 1

(see problem_set1.py)
(see problem_set1_answers.py)

Outline:

- 10 Normal Problems
- 2 Challenge Problems

Problem Set 1 tests the topics from Lesson 1:

- Operators (Problems 1-4)
- Functions (Problems 5-8, 10 and onwards)
- if-else statements (Problems 7, 8, 10 and onwards)
- For-Loops (Problem 9 and onwards)

Lesson 2
(See lesson2.py)

Topics:

- Lists

Lists:

- List indexing
- List splicing
- Append vs concatenate
- For-loops + lists
    - Using range()
    - Using for-each
    - Using enumerate()
- Lists are heterogenous.
- List comprehensions

Other: in-place operations, scope, tuples, list(), len(), str()

Problem Set 2

(see problem_set2.py)
(see problem_set2_answers.py)

Outline:

- 10 Normal Problems
- 0 Challenge Problems

Problem Set 2 tests the topics from Lesson 2:

- List Indexing + Splicing (Problems 1-4)
- Iteration through a List (Problems 5-9)
- List Comprehensions (Problem 10)

<u>Lesson 3</u>
(See lesson3.py)

Topics:
- 2-D Lists
- User Input
- While Loops

<u>2-D Lists:</u>
- Initializing
- Indexing
- Splicing
- Appending + Removing
- Iteration w/ For-Loops

<u>User Input:</u>
- input()
- Cast user input (user input is always a str)

<u>While Loops:</u>
- Iterating a List
- User Input + While Loops Combo

Problem Set 3

(see problem_set3.py)
(see problem_set3_answers.py)

Outline:

- 5 Normal Problems
- 1 Challenge Problem

Problem Set 3 tests the topics from Lesson 3:

- 2-D Lists (Problems 1-4, Challenge Problem 1)
- User Input + While Loops (Problem 5)

Project 1 - Chess

(See project1_chess.py)

Description:
In this project, students will figure out how to model a chess board and chess piece movements using Python. This will require using all of the concepts learned in Lessons 1-3. Students will be recommended to use a 2-D list to model the chess board and functions to organize the logic for the movement patterns of each type of chess piece (Queen, King, Rook, etc.). However, the implementation is left up to the student.

After modeling the chess board and chess pieces, the student's code will take simple user input that will attempt to place chess pieces on the chess board. The student's code should remember which pieces have been placed so far and figure out whether the user's current piece can be placed at the indicated square. If the indicated square is empty, the student's code will place the new piece on that square and remember that the square is occupied for future user input. At this point, the student's code will also calculate and print how many squares the new piece can move based on its location, its piece type, and the vacancy of other squares on the chess board.

Main Concepts Used:
- if-else statements
- For Loops
- While Loops
- Functions
- 2-D Lists
- User Input/Output

<u>Lesson 4</u>
(See lesson4.py)

Topics:
- Sets
- Dictionaries
- Classes

<u>Sets:</u>
- O(1) add(), remove()
- O(1) contains/search using `in`
- O(1) search in sets versus O(n) search in lists
- Sets can't have any duplicates
- Sets have no implicit order
- Set Operations (union, intersection, difference, etc)

<u>Dictionaries:</u>
- Key must be unique, value does not need to be unique
- Key must be "hashable" and "immutable"
- Collection of all Keys is basically a set (since all keys must be unique)
- Iteration w/ For-Each Loop using key-set

<u>Classes:</u>
- __init__() & Global Fields (State)
- Methods (Behavior)
- Instantiating and using an object of a custom Class
- Scoping (global versus local)
- Mutable versus Immutable

Problem Set 4

(see problem_set4.py)
(see problem_set4_answers.py)

Outline:

- 3 Normal Problems
- 1 Challenge Problem

Problem Set 4 tests the topics from Lesson 4:

- Sets & Dictionaries (Problems 1 & 2, Challenge Problem 1)
- Classes (Problem 3)

<u>Project 2 - Bank</u>
(See project2_bank.py)

Description:
In this project, students will build the functionality of a commercial Bank. To simplify things, we assume all customers only have one type of account (not both a checking and savings account like most commercial banks).

Students will have to implement a Bank class and an Account class. The details of these classes are below. Much more details are provided in *project2_bank.py* for the student. The implementation/design decisions are largely left up to the student.

The functionality of the Bank will be assessed with a suite of private test cases. Some public test cases will be made available to the student to help test and debug their code. Students will largely have to rely on their own custom test cases to prepare their code to pass all of the private test cases.

Main Concepts Used:
- Classes
- Dictionaries
- Sets

Account class:
- __init__()
  - Unique ID of 9 integer digits
  - Name
  - Password
  - Account Balance
- get_password()
- get_name()
- change_password(new_pwd)
- deposit(amount)
- withdraw(amount)
- get_ID()

Project 2 - Bank (contd.)
(See project2_bank.py)

Bank class:

- __init__()
    - Use some data structure to store all Account objects
- deposit(ID, password, amount)
    - User of the Bank class must provide the right ID-password combination.
    - If the ID-password combo is valid, deposit the amount indicated into the account.
- withdraw(ID, password, amount)
    - Same as deposit(), just withdraw the amount from the account instead.
- change_pwd(ID, password, new_password)
- new_account(name, password, balance)
    - Returns an auto-generated, unique ID to the user for future calls to the Bank's functions.
- close_account(ID, password)

Data Science Lifecycle

Lesson 5

(See lesson5.py)

Topics:

- NumPy
    - ndarray common operations
        - Indexing
        - Splicing
        - Generating Data
    - ndarray vs Lists
        - Immutable vs Mutable
        - O(n) vs Amortized O(1) for add/remove
    - dtypes
- Pandas
    - Import from a csv
    - DataFrame common/useful operations
        - Drop columns
        - Filtering with masks
        - SQL Joins with merge()
    - DataFrame immutability

Problem Set 5

(see problem_set5.py)
(see problem_set5_answers.py)

Outline:

- 7 Normal Problems
- 0 Challenge Problems

Lesson 6

(See lesson6.py)

Topics:

- Linear Regression
- Logistic Regression

Linear Regression:

- Predictors and Response variables
- Simple Linear Regression with just 1 predictor
- "Line of Best Fit" (Minimizing Mean Squared Error, which is the Loss Function)
- Multiple Linear Regression
- Preparing data and running scikit-learn & statsmodels.api Linear Regression
- Hypothesis Testing for Significant Predictors (t-test, confidence intervals, and p-value)
- Interpreting Significant Predictors

Logistic Regression:

- Categorical versus Continuous Variables
- Logistic Regression versus Linear Regression
- Preparing data and running scikit-learn & statsmodels.api Logistic Regression
- Interpreting Categorical Significant Predictors
- Interpreting Continuous Significant Predictors

Project 3 - Kaggle

(See project3_kaggle.py)

Description:

In this project, students will choose one of the datasets below and conduct Linear or Logistic Regression to help answer some question they form about relationships between predictor and response variables in the dataset. Here are the datasets to choose from:

- Credit Card Customers: https://www.kaggle.com/sakshigoyal7/credit-card-customers
- Student Performance on Exams: https://www.kaggle.com/spscientist/students-performance-in-exams
- Real Estate Prices: https://www.kaggle.com/arslanali4343/real-estate-dataset

Main Concepts Used:

- NumPy
- Pandas
- Linear Regression
- Logistic Regression
- Hypothesis Testing
- Interpreting Predictors

<u>Lesson 7</u>
(See lesson7.py)

Topics:

- Scraping Data
- Data Visualization

<u>Scraping Data:</u>

- HTML Basics (Structure of HTML documents & common tags for data scraping)
- Scraping HTML page with BeautifulSoup
- JSON & REST API Basics
- GET Requests and scraping data from JSON output with BeautifulSoup

<u>Data Visualization:</u>

- Simple Scatter Plot with matplotlib
- Multiple Line Plot with Pivot Table DataFrame and matplotlib
- Extra matplotlib features with Axes & Legends
- Violin Plots with seaborn

<u>Project 4 - NBA</u>

(See project4_nba.py)

Description:

In this project, students will scrape NBA player season average stats from 2000-2019 from this website: https://www.basketball-reference.com/leagues/NBA_2001_per_game.html

Then, students will make 3 data visualizations of their own design on the compiled 20 years of NBA data. Students must explain what question they hoped to investigate with their visualization, and what they learned from creating and analyzing the visualization.

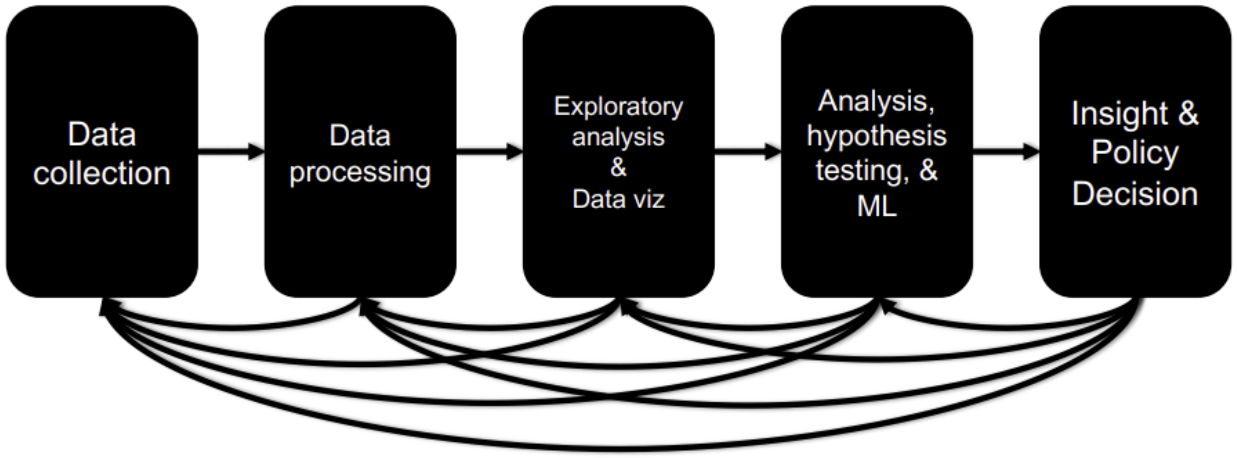
Students are recommended to use variables like Year or Age as the x-axis in visualizations, but the final choice is ultimately up to the student.

The hardest part of this project will be scraping and compiling the 20 years of data into one DataFrame, and then preparing that DataFrame for subsequent data visualizations.

Main Concepts Used:

- Scraping Data
- Data Visualization
- NumPy
- Pandas

Final Project


Data collection
Data processing
Exploratory analysis & Data viz
Analysis, hypothesis testing, & ML
Insight & Policy Decision

Description:
The Final Project is a chance for students to use all of the elements in the Data Science workflow we have learned in one project:

- Data Scraping/Collection w/ BeautifulSoup
- Data Transformation/Manipulation w/ Pandas & NumPy
- Data Visualization w/ matplotlib & seaborn
- Machine Learning & Hypothesis Testing w/ scikit-learn

The Final Project is very open-ended, and only has the following instructions and requirements:

- Find any dataset on the internet that interests you.
  - You can scrape data or simply download a .csv file. It doesn't matter.
- Ask 2-3 questions about the dataset. These are the hypotheses that you will test later.
  - Ex: "How does variable Y change as variable X changes?"
  - Ex: "I predict that variable Y will go up as variable X goes up because…"
- Create data visualizations that help give some insight in answering your questions.
- Run a Linear or Logistic Regression (or any other Machine Learning method you learn on your own) to conduct some hypothesis testing on your initial questions.
- Write a conclusion about what the results of your hypothesis testing indicate.
  - Were you right or wrong about your initial hypotheses/questions?
  - Were you able to find statistically-significant evidence to support your hypothesis?
  - What further steps can be taken in future studies to answer your questions?