Introduction:

Why Coding?

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

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 the DNA 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.

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()