Intro to Programming in Python

> "Programming languages allow us to formalize instructions and express logic, business rules, mathematics, processes, and automation instructions in one single language in a way where computers follows those instructions to produce utility for people." - Ryan Orsinger

- Python is the leading language in Data Science/Machine Learning
- One of the most beginner-friendly languages because it is close to natural language
- Industrial strength language for everything from heavy industry to PhD research

Setup your Python learning environment

1. Create an account on `https://kaggle.com`
2. Go to `https://www.kaggle.com/ryanorsinger/intro-to-python`
3. Click "Copy and Edit" to make your own copy

Using Jupyter Notebooks (called Kernels on Kaggle)

- Notebooks are made of cells. Each cell contains either code or text
- To run a cell, click on it and hit the Play button or "shift + Enter" on your keyboard
- Click into a cell to enter edit mode. You can enter or edit text/code

Data types and values

- Booleans (True or False values) denoted by `True` and `False`
- Numbers (integers and floats) `23`, `-5`, `3.141`, `0`
- Strings (text contained inside of 'single' or "double" quotation marks) `"hello"`
- `None` (signifies the absence of value)
- Lists contain values separated by a comma. `[1, 2, 3]` or `["Peter", "Paul", "Mary"]`
- Dictionaries are a type of labeled list. `{"make": "Toyota", model": "Supra"}`
- We often combine data types to have nested data structures. For example, data that you can visualize on a spreadsheet could be a list of lists or a list of dictionaries.
- `[[1, 2, 3], [4, 5, 6]]` is a list containing two lists.
- Functions are a named sequence of instructions that operate on inputs to produce outputs

Assigning and reassigning variables to point to values

`x = 2` The single equals symbol is called the `assignment operator`. Assignment means to point a variable name to a specific piece of data.

If `x` already points to a value, then `x = 5` reassigns `x` to be `5`.

Python evaluates the expression on the right and assigns that value to the variable on the left.

Operators and the results of operations

Operators in programming languages are like math operators like `+`, `-`, `*`, `/`, etc...

In programming, operators return a value with a data type. All values have a data type.

Boolean operators are represented by `and`, `or`, `not`, and the `!` operator.

Comparison operators compare two values and return a `True` or `False`

`==` is the equality comparison operator. `3 != 2` the "not equal to" operator.

Examples: `3 > 2` returns `True`, `2 == "2"` is `False`, `3 != "banana"` is `True`

Some Built-In Functions in Python

- `print()` prints whatever values we put into the parentheses.
- `type()` returns the name of the data-type of the values put into the parentheses.
- `len(["John", "Paul", "George", "Ringo"])` tells us the number of items on the list.
- https://docs.python.org/3/library/functions.html is a list of all built-in functions

Defining and Running Your Own Functions

- Functions take in inputs, perform a process or processes, and return the output.
- Think of functions like mathematical operators or commands.
- Functions **do** things or **compute** values.

```
# This block of code is how we would define a function to square a number
def square(input):
    return input * input
print(square(2)) # prints 4 when we "call" the square function with 5
print(square(square(3))) # prints 27 because we work from the inside out.
```

Python Documentation and Reference

Python Tutorial - https://docs.python.org/3/tutorial/index.html

Python Standard Library - https://docs.python.org/3/library/index.html

Python Language Reference - https://docs.python.org/3/reference/index.html#reference-index

Homework - This is also a part of Codeup's pre-work!

- Keep learning this material for two *huge* reasons:
  1. The ability to program, to read/write code, sets you apart in the marketplace.
  2. Data literacy is a highly marketable skill because *data is the new oil*.
- Goto `https://www.kaggle.com/ryanorsinger/101-exercises` for deep practice.