Python Basics

Corey Fuller

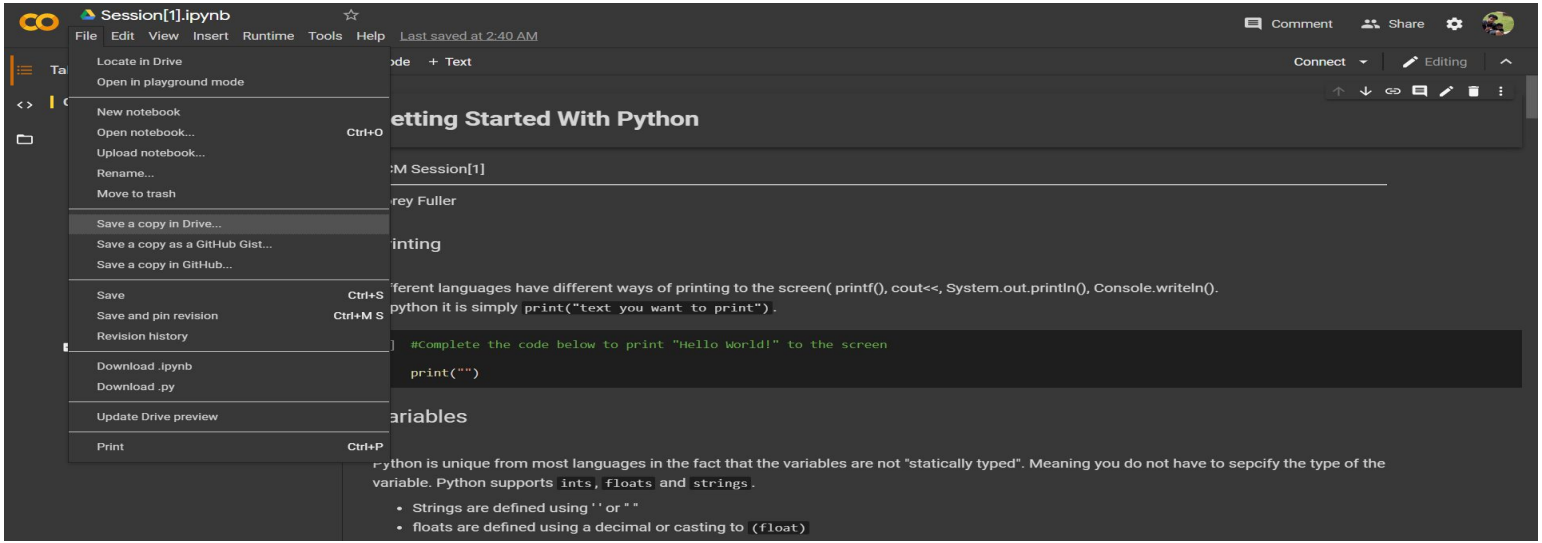
To Make Things Simpler...

- Go this link
  - Click the Session1 Folder
    - Click `SESSION1.md`
      - Click “Click here to access Session[1]'s interactive lesson file.”
- All future slides will be posted on GitHub, uploaded to the discord, and sent via email.

https://github.com/coreyFuller/ACM-Python.git

Working with Google Colaboratory

- Google Colab allows us to make these sessions more interactive
- Once the notebook is open (you've clicked the link), save a copy of the notebook to your Drive
    - This will allow you to edit your own personal copy and follow along



Let’s Get Started!!!

Printing

- Different languages have different ways of printing to the screen
  - printf(), cout<<, System.out.println(), Console.writeln()
- In python it is simply **print**("text you want to print").
- Last session we printed “Hello World”
- Try filling in the code on the notebook

Variables

- Python is unique from most languages in the fact that the variables are not "statically typed"
- This means you do not have to specify the data type of the variable.
  - Python supports **ints**, **floats**, and **strings**
- Strings are defined using ' ' or " "
- floats are defined using a decimal or casting to float()
- Python is an object oriented language so every variable is an **object**
- Try out the practice in the notebook

Math Operations

- Python supports all of the basic math operations just like any other languages
  - +, -, *, /, %, **
- Exponents operations are a ** b (a raised to the b power)
- Try to fill in the code on the notebook

Fun with Strings

- Python supports adding strings together much like in other high level languages, using the "+" operator
- Strings can be cast to other data types
- Strings are just an array, or list, of characters on the ASCII table.
    - They can be indexed into and have a set length
    - We can use this to our advantage

Lists

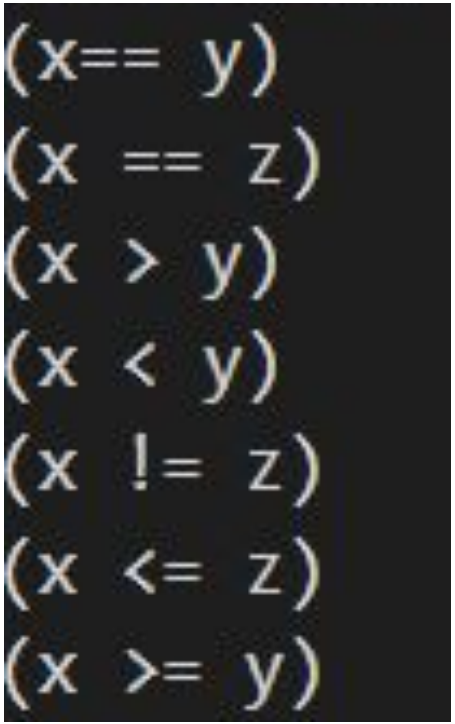
- Lists in python are just like arrays in other languages.
- Lists in python can have any data type in the same list and can be altered easily
- Lists are declared using []
- They can be defined "statically"
  - like mylist = [1,2,'Bob',4,6.9]
- or can be added/removed dynamically like
  - list = []
  - list.append('a')
  - list.insert(1,2)
  - list.remove('a')

Some Helpful List Functions

- .append(x) - adds x to the end of a list.
- .insert(i, x) - inserts x into position i in the list
- list[i] - tells you what's at position i in the list
- len(list) - tells you the length of the list
- list[a:b] - returns lists elements between a and b
  - can use just [a:] to return elements a and above in the list
  - can use just [:b] to return b and below in the list
- .remove(x) - removes x from the list if it exists
- .sort() - sorts list in ascending order
- Try out the notebook code

Booleans and Logic

- Booleans in python are either True or False
- Boolean Logic is also supported like other languages.
- Try out the notebook practice



if, elif, and else oh my! && “in”

- If statements take in boolean expressions
- If the condition inside of the if statement evaluates to True, the code underneath will run.
    - If not, elif (else if) take in an expressions and are checked next.
        - If that fails, else statements can dictate what happens next.
- In python, if/else statements use : instead of {}. The code inside the statement is indented underneath.
- Try out the practice
- The statement "in" can be used to determine if an element exists in a list or if a substring appears in a string.
- Try out the notebook practice

Loops

- Python has both for loops and while loops, just like many other languages.
- For loops
  - For loops in python are "ranged based" .
  - For example: “for i in range(0,5)” is equivalent to “for(int i = 0; i < 5; i++)” in Java, C, C++, C#, etc.
- While loops
  - While loops in python work just like other languages.
- Range based for loops
  - Ranged based for loops can be used to print elements of a list very easily
- Try out the notebook practice

Breaks and Continues

- Breaks can be used to immediately end loops.
- Continues can be used to skip the rest of the loop and immediately go back to the top of the loop.
- Run the script to see how it works
- Notice anything?

You’re all set to move on to the next section!

It will be on data structuring, objects, imports, and pip installs.