Welcome to Python No Sneks Allowed

Richard Morrill

Fordham University CS Society

Thursday, Novemeber 21st 2019



Hello, Python

```
1 def main():
2    print("Hello World!")
3
4 if __name__ == "__main__":
5    main()
```



Python is an interpreted programming language / scripting language (i.e. an interpreter runs your code)

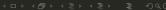


¹Well, of a very specific type...

- Python is an interpreted programming language / scripting language (i.e. an interpreter runs your code)
 - No need to write "boilerplate" like in C++
 - Much harder to crash your computer
 - Much harder to leave critical security vulnerabilities¹



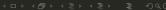




- Python is an interpreted programming language / scripting language (i.e. an interpreter runs your code)
 - No need to write "boilerplate" like in C++
 - Much harder to crash your computer
 - Much harder to leave critical security vulnerabilities¹
- Python is EVERYWHERE

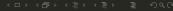






- Python is an interpreted programming language / scripting language (i.e. an interpreter runs your code)
 - No need to write "boilerplate" like in C++
 - Much harder to crash your computer
 - Much harder to leave critical security vulnerabilities¹
- Python is EVERYWHERE
 - Machine Learning
 - Data Science
 - Some random utility some guy wrote for one very specific problem

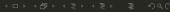




¹Well, of a very specific type...

- Python is an interpreted programming language / scripting language (i.e. an interpreter runs your code)
 - No need to write "boilerplate" like in C++
 - Much harder to crash your computer
 - Much harder to leave critical security vulnerabilities¹
- Python is EVERYWHERE
 - Machine Learning
 - Data Science
 - Some random utility some guy wrote for one very specific problem
- Python is Great for Personal Projects





¹Well, of a very specific type...

- Python is an interpreted programming language / scripting language (i.e. an interpreter runs your code)
 - No need to write "boilerplate" like in C++
 - Much harder to crash your computer
 - Much harder to leave critical security vulnerabilities¹
- Python is EVERYWHERE
 - Machine Learning
 - Data Science
 - Some random utility some guy wrote for one very specific problem
- Python is Great for Personal Projects
 - Easy to start fast
 - No need to worry about advanced CS stuff



¹Well, of a very specific type...

Why not Python?



Why not Python?

It's SLOW AF



Why not Python?

It's SLOW AF
It hides a lot of the "fun" from you



What You Need

- A Text Editor
- A Python Interpreter





What You Need

- A Text Editor
- A Python Interpreter

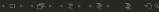
Why were your frigging instructions so damn complicated then?





Builds Character

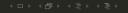




²Anybody want an event on Docker?

- Builds Character
- Docker is Great ²

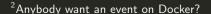


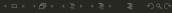


²Anybody want an event on <u>Docker?</u>

- Builds Character
- Docker is Great ²
- I wanted to remove ALL variables from the process for beginners







- Builds Character
- Docker is Great ²
- I wanted to remove ALL variables from the process for beginners
- You could've skipped them anyways





²Anybody want an event on <u>Docker?</u>

Setup Steps: Get Resources

- All resources at: https://github.com/fordham-css/Python_Workshop
- Either use git clone or download it as a zip file
- Hint: You can download things from terminal on most *NIX systems and in Powershell with \$ wget URL
- Navigate to the resources directory
- You'll be opening VsCode (After Python / Docker is set up) in this directory, by
 navigating here in Terminal and typing \$ code . or through the GUI on Windows (Ask
 me)³

³If you open VsCode in the main folder of the repo it will attempt to install a ton of stuff you don't **CS SOCIETY**

Setup Steps: Python

- Install Text Editor (VsCode Preferred): https://code.visualstudio.com/download
- Install Docker⁴, find correct link in setup Doc





⁴Optional

⁵That's "resources"

Setup Steps: Python

- Install Text Editor (VsCode Preferred): https://code.visualstudio.com/download
- Install Docker⁴, find correct link in setup Doc
- If using Docker:
 - VsCode Required
 - Make sure you open it in the right directory⁵
 - Install Extension "Remote Containers"
 - Click Orangey Thingy at Bottom Left
 - Select "Reopen in Container"
 - Wait...
 - You're ready



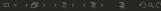
⁵That's "resources"



Setup Steps: Python

- If not using Docker:
 - Install Miniconda: https://docs.conda.io/en/latest/miniconda.html
 - Navigate to "resources" folder inside wherever you downloaded the repo
 - Open your text editor and wait





Setup Steps: VsCode

- If you're using Docker this will already be configured for you
- F1
- > Python: Select Interpreter
- Click the one that that has conda/miniconda somewhere⁶
- F1
- > Python: Select Linter
- Click "pylint"
- It will prompt you to install PyLint, let it⁷



⁶Ask for help if you don't see it

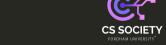
⁷If it asks whether you want to use conda or pip say conda

You did it!

You now should all have a working Python setup. We'll continue once everybody has gotten it working.

To ensure it actually is working:

- $\bullet \ \, \mathsf{Open} \,\, \mathsf{a} \,\, \mathsf{terminal} \,\, (\mathsf{VsCode} \,\, \mathsf{menu} \,\, \mathsf{bar} \,\, \to \, \mathsf{Terminal} \,\, \to \, \mathsf{New} \,\, \mathsf{Terminal})$
- Type python --version
- You should see "Python 3.7.5" 8



⁸Don't sweat it if your number is slightly lower

It's got functions







- It's got functions
- It's got variables

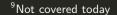






- It's got functions
- It's got variables
- It's got operators







- It's got functions
- It's got variables
- It's got operators
- It's got classes and OOP⁹



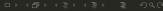




Indented Syntax

- If you're used to C++ this will trip you up
- Python relies on indent to tell which code is in which block.
- You may use either tabs or spaces, but you must be consistent
- Keep top-level code right against the left margin
- This will all make more sense once you see examples





Variables

- For those that haven't programmed before:
 - Not "unknowns" like in math
 - A place to store pieces of information
- Open basics/variables.py

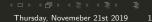




Variables

- For those that haven't programmed before:
 - Not "unknowns" like in math
 - A place to store pieces of information
- Open basics/variables.py





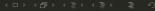
Variables

```
4 a = 1
5 b = "A string"
6 C = {
  "This is": "A dictionary".
   "It's pretty much": "the same thing as a JS object"
9 }
o # This is a list
11 d = [a, b, c]
13 a = b
```

Printing

- It's nice to see the results of our program.
- We do this (in basic programs) by printing to the terminal.
- Anybody know why it's called that?





Printing

- It's nice to see the results of our program.
- We do this (in basic programs) by printing to the terminal.
- Anybody know why it's called that?

```
print("A string")
print(1)
print([1, 4, 5, 6])
```

- You can print nearly anything, with varying degrees of success
- Try printing some things, see what works well and what doesn't



Control Flow

• In its simplest form, a Python script is just a list of instructions the interpreter does sequentially.



¹⁰Who wants to tell me what each of these is?

Control Flow

- In its simplest form, a Python script is just a list of instructions the interpreter does sequentially.
- You control the order in which it executes statements using: 10
 - If Statements
 - Loops
 - Functions





¹⁰Who wants to tell me what each of these is?

Conditionals

- There's another variable type: <u>boolean</u> (True / False)
- You can define them literally:

```
1 condition = True
```

```
2 if condition:
```

- $_{
 m 3}$ print("If statements are pretty easy to deal with")
- 4 else:
- print("Just don't forget the colons and indent")



¹¹I'll go over all the available comparisons later

Conditionals

- There's another variable type: <u>boolean</u> (True / False)
- You can define them literally:

```
1 condition = True
```

- 2 if condition:
- print("If statements are pretty easy to deal with")
- 4 else:
 - print("Just don't forget the colons and indent")
- Or use the result of a comparison¹¹

 1 if "something" == "not something":
 - print("This will never execute")
- 3 else:
- 4 <u>print("This will always</u> execute")

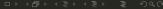
 11 I'll go over all the available comparisons later



Functions

- For those that haven't programmed before:
 - A set of statements that can be run over and over again
 - Called by name
 - Optionally takes input and returns output





Thursday, Novemeber 21st 2019

Functions

- For those that haven't programmed before:
 - A set of statements that can be run over and over again
 - Called by name
 - Optionally takes input and returns output

```
1 def example_function(var_a, var_b):
2 '''This function takes 2 arguments'''
3
4  # It also returns 2 values
5  return var_b, var_a
```





Functions

1 def no_arguments():

```
return 5
5 def no_return(var_1, var_2):
7 var_1 += 3
8 print(var_2 + var_1)
12 def print_conditional(var_1):
if (var_1 == 5):
     return
print(var_1)
```

Thursday, Novemeber 21st 2019

Loops

Do the same thing over and over



21 / 33

¹²Although that number can be modified while they run, and they can exit early

Loops

- Do the same thing over and over
- While Loops:
 - Run until condition is no longer true
 - Won't even run once if condition isn't true



¹²Although that number can be modified while they run, and they can exit early

Loops

- Do the same thing over and over
- While Loops:
 - Run until condition is no longer true
 - Won't even run once if condition isn't true
- For Loops:
 - Run a fixed number of times¹²
 - In Python almost always over a list of something



21 / 33

¹²Although that number can be modified while they run, and they can exit early

While Loop Example

```
1 b = 5
2 while(b > 0):
3     b = b - 1
```



While Loop Example

```
1 b = 5
2 while(b > 0):
3 b = b - 1
```

Anybody see the potential issue here?



While Loop Example

```
1 b = 5
2 while(b > 0):
3 b = b - 1
```

- Anybody see the potential issue here?
- While loops always need a good exit condition



For Loop Example

```
1 a = [5, 7, 8]
2 for num in a:
3 print(num)
```



For Loop Example

```
_{1} a = [5, 7, 8]
2 for num in a:
   print(num)
 If you want to exit early
_{1} a = [5, 7, 8]
2 for num in a:
    if num == 7:
      break
   print(num)
```



For Loop Example

If you want to run a fixed number of times without a list:

- 1 for i in range(9):
- 2 print(i)



Most languages have a bunch of number types

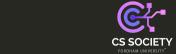


- Most languages have a bunch of number types
- Python only has 3 (that you can see):
 - int
 - float
 - complex





```
2 a = 4
3 print(a)
_{5} b = 4.3
6 print(b)
8 print(int(4.3))
\frac{10 \text{ c}}{\text{c}} = "1000"
n print(c * 5)
```



 $_{2} a = 4$ 3 print(a)

5 b = 4.36 print(b)

```
8 print(int(4.3))
\frac{10 \text{ c}}{\text{c}} = "1000"
n print(c * 5)
  →Somebody run that last line
```

Thursday, Novemeber 21st 2019

2 a = 4 3 print(a)

5 b = 4.3
6 print(b)

2 print(int(c) * 5)

```
7 # Round down a float
8 print(int(4.3))
9 # A string
10 c = "1000"
11 print(c * 5)

→Somebody run that last line
```



Arithmatic Operators

```
2 print(2 + 3)
3 print(10 / 2)
4 print(3 * 7)
5 print(3 - 19)
8 print(11 / 2)
10 print(11 // 2)
_{12} print("3^3 = {}".format(3**3))
```

Thursday, Novemeber 21st 2019

Logical Operators

- Used to compare values:
 - Equality: 3 == 5
 - Greater / Less Than: 2 > 4
 - Greater Than or Equal: 5 >= 8





Logical Operators

- Used to compare values:
 - Equality: 3 == 5
 - Greater / Less Than: 2 > 4
 - Greater Than or Equal: 5 >= 8
- Combining logical statements
 - And: 2 < 8 and 4 == 4
 - Or: "a" == "b" or 5 == 5

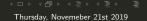




What is a string?

- If you haven't programmed before:
 - A sequence of <u>characters</u>
 - Used to store text





What is a string?

- If you haven't programmed before:
 - A sequence of <u>characters</u>
 - Used to store text
- Closely related to lists





String Operations

```
1 a = "This is a string"
2 b = "this is another string"
_{4} c = a + ", and " + b
6 for w in a.split():
     print(w)
9 print(a.upper())
print(a.lower())
12 if "is" in a:
print("Yay!")
14 else:
print("Boo!")
```

Thursday, Novemeber 21st 2019

What is a list?

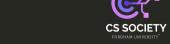
- An ordered collection of values
- Holds any type





List Operations

```
_{1} a = [66.25, 333, 333, 1, 1234.5]
2 print a.count(333), a.count(66.25), a.count('x')
_4 a.insert(2. -1)
5 a.append(333)
6 # Find index of first item with that value
7 a.index(333)
* # Remove item with that value
9 a.remove(333)
a.reverse()
n a.sort()
<sub>12</sub> a.pop()
```



You Have the Tools

• All I've given you today is the tools, it's up to you to figure out how to actually use them





You Have the Tools

- All I've given you today is the tools, it's up to you to figure out how to actually use them
- Project Ideas:
 - Simple Calculator
 - Guessing Game





You Have the Tools

- All I've given you today is the tools, it's up to you to figure out how to actually use them
- Project Ideas:
 - Simple Calculator
 - Guessing Game
- Possible Future Events
 - Scikit (Data Science)
 - Game Making
 - Tensorflow (Neural Networks)



