

Welcome to Python

No Sneks Allowed

Richard Morrill

Fordham University CS Society

Thursday, November 21st 2019



Hello, Python

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



CS SOCIETY
FORDHAM UNIVERSITY

Why Python?

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

¹Well, of a very specific type. . .

Why Python?

- 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

¹Well, of a very specific type. . .

Why Python?

- 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

¹Well, of a very specific type. . .

Why Python?

- 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. . .

Why Python?

- 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. . .

Why Python?

- 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?



CS SOCIETY
FORDHAM UNIVERSITY™

Why not Python?

It's SLOW AF



CS SOCIETY
FORDHAM UNIVERSITY™

Why not Python?

It's SLOW AF

It hides a lot of the “fun” from you



CS SOCIETY
FORDHAM UNIVERSITY

What You Need

- A Text Editor
- A Python Interpreter



CS SOCIETY
FORDHAM UNIVERSITY™

What You Need

- A Text Editor
- A Python Interpreter

Why were your frigging instructions so damn complicated then?

- ²Anybody want an event on Docker?

- Builds Character
- Docker is Great ²

²Anybody want an event on Docker?

Why my frigging instructions were so damn complicated

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

²Anybody want an event on Docker?

Why my frigging instructions were so damn complicated

- 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 Docker?

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 need



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

⁴Optional

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 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:

- Open a terminal (VsCode menu bar → Terminal → New Terminal)
- Type `python --version`
- You should see “Python 3.7.5”⁸

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

Python Has all the Basics

- It's got functions

⁹Not covered today

Python Has all the Basics

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

⁹Not covered today

Python Has all the Basics

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

⁹Not covered today

Python Has all the Basics

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

⁹Not covered today

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

```
1 # This file demonstrates variables in Python
2 # You don't need to use a 'var' keyword
3 # or declare a type
4 a = 1
5 b = "A string"
6 c = {
7     "This is": "A dictionary",
8     "It's pretty much": "the same thing as a JS object"
9 }
10 # This is a list
11 d = [a, b, c]
12 # This is fine, you can change type on the fly
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?

```
1 print("A string")
2 print(1)
3 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: boolean (True / False)
- You can define them literally:

```
1 condition = True
2 if condition:
3     print("If statements are pretty easy to deal with")
4 else:
5     print("Just don't forget the colons and indent")
```

¹¹I'll go over all the available comparisons later

Conditionals

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

```
1 condition = True
2 if condition:
3     print("If statements are pretty easy to deal with")
4 else:
5     print("Just don't forget the colons and indent")
```

- Or use the result of a comparison¹¹

```
1 if "something" == "not something":
2     print("This will never execute")
3 else:
4     print("This will always execute")
```

¹¹I'll go over all the available comparisons later



CS SOCIETY
FORDHAM UNIVERSITY

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

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
```



CS SOCIETY
FORDHAM UNIVERSITY

Functions

```
1 def no_arguments():
2     # Do stuff...
3     return 5
4
5 def no_return(var_1, var_2):
6     # Do stuff
7     var_1 += 3
8     print(var_2 + var_1)
9     # Notice there's no return statement
10
11 # If you want, you can still return early from a function
12 def print_conditional(var_1):
13     if (var_1 == 5):
14         return
15     print(var_1)
```


Loops

- Do the same thing over and over

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

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
```



CS SOCIETY
FORDHAM UNIVERSITY

While Loop Example

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

- Anybody see the potential issue here?



CS SOCIETY
FORDHAM UNIVERSITY

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



CS SOCIETY
FORDHAM UNIVERSITY

For Loop Example

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



CS SOCIETY
FORDHAM UNIVERSITY

For Loop Example

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

If you want to exit early

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



CS SOCIETY
FORDHAM UNIVERSITY

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)
```



CS SOCIETY
FORDHAM UNIVERSITY™

Numbers

- Most languages have a bunch of number types



CS SOCIETY
FORDHAM UNIVERSITY™

Numbers

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

Numbers

```
1 # An integer
2 a = 4
3 print(a)
4 # A floating point number
5 b = 4.3
6 print(b)
7 # Round down a float
8 print(int(4.3))
9 # A string
10 c = "1000"
11 print(c * 5)
```



CS SOCIETY
FORDHAM UNIVERSITY™

Numbers

```
1 # An integer
2 a = 4
3 print(a)
4 # A floating point number
5 b = 4.3
6 print(b)
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



CS SOCIETY
FORDHAM UNIVERSITY™

Numbers

```
1 # An integer
2 a = 4
3 print(a)
4 # A floating point number
5 b = 4.3
6 print(b)
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

```
1 # So, convert a string to a number
2 print(int(c) * 5)
```

Arithmetic Operators

```
1 # All of these work exactly like you'd expect
2 print(2 + 3)
3 print(10 / 2)
4 print(3 * 7)
5 print(3 - 19)
6 # If you're used to C++, this might work differently
7 # than you'd expect
8 print(11 / 2)
9 # If you want integer division, use //
10 print(11 // 2)
11 # Python also has a cool exponentiation operator
12 print("3^3 = {}".format(3**3))
```



CS SOCIETY
FORDHAM UNIVERSITY

Logical Operators

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



CS SOCIETY
FORDHAM UNIVERSITY

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`



CS SOCIETY
FORDHAM UNIVERSITY

What is a string?

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



CS SOCIETY
FORDHAM UNIVERSITY™

What is a string?

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



CS SOCIETY
FORDHAM UNIVERSITY™

String Operations

```
1 a = "This is a string"
2 b = "this is another string"
3 # We can concatenate them together
4 c = a + ", and " + b
5 # We can split them apart
6 for w in a.split():
7     print(w)
8 # We can change case
9 print(a.upper())
10 print(a.lower())
11 # we can test if a string contains another
12 if "is" in a:
13     print("Yay!")
14 else:
15     print("Boo!")
```



CS SOCIETY
FORDHAM UNIVERSITY

What is a list?

- An ordered collection of values
- Holds any type



CS SOCIETY
FORDHAM UNIVERSITY

List Operations

```
1 a = [66.25, 333, 333, 1, 1234.5]
2 print a.count(333), a.count(66.25), a.count('x')
3 # 1st Argument index of element before which to insert
4 a.insert(2, -1)
5 a.append(333)
6 # Find index of first item with that value
7 a.index(333)
8 # Remove item with that value
9 a.remove(333)
10 a.reverse()
11 a.sort()
12 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)