GENERAL ASSEMBLY

Intro to Python

Housekeeping items:

Internet connection

Wifi: **GA-Guest**

Password: **yellowpencil**

Please download

Class Resources: **https://bit.ly/2OVEGtC**

Anaconda: **https://www.anaconda.com/download**

Agenda:

1. Introductions
2. Getting started
3. Why Python?
4. Python basics
5. Using 3rd party modules
6. . . .
7. Recap
8. Questions?

Introductions:

I am: *Mike Sanders*

- Software Developer at Pivotal Labs
- Started teaching at GA in 2017
- Full stack application developer since 2006
- Contact me: sanders.michael.j@gmail.com

Introductions:

You are: . . .

- What do you do today?
- What do you hope to learn from this class?
- Where do you want this to take you?

Getting Started

Class Resources: **https://bit.ly/2OVEGtC**

Contents:

1. Class slides: **Intro_to_Python.pdf**
2. Example scripts: **examples directory**

Let’s install Python!

Getting Started

Next, you will need:

1. An editor: **https://www.anaconda.com/download**
2. Other editors:

Sublime Text	https://www.sublimetext.com/3
Atom	https://atom.io
PyCharm	https://www.jetbrains.com/pycharm/download
Visual Studio Code	https://code.visualstudio.com/Download

Getting Started

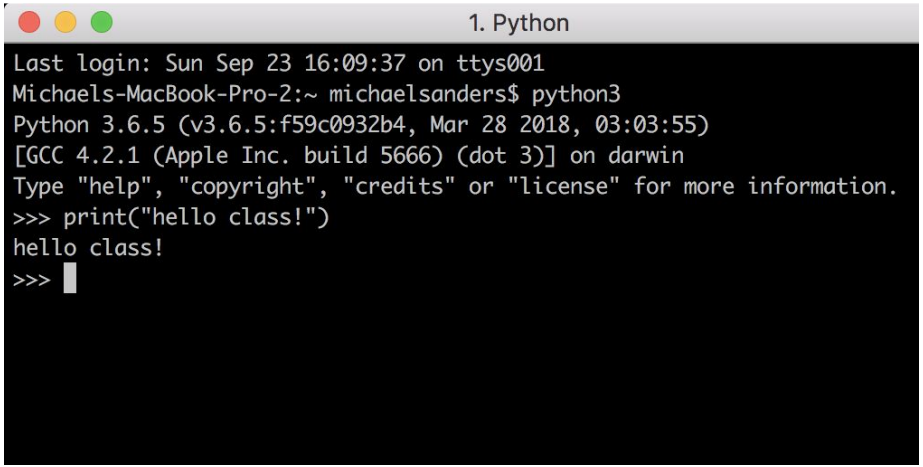
Python interpreter

1. Open a command shell and type ‘python3’

2. This opens an interactive Python session that allows you to execute code

3. Any code that you write will disappear once the session ends

Example:



Getting Started

Python scripts

1. Save your source code in a file with the ***.py*** extension

2. Navigate your terminal session to the folder where your script was saved

3. Execute using:
   python3 <script_name>

Example:

hello_world.py

```
print("Hello world!")
```

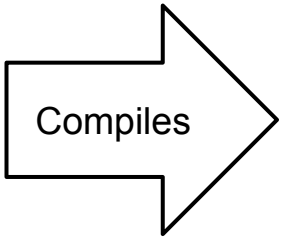
1. bash

```
Michaels-MacBook-Pro-2:Desktop michaelsanders$ pwd
/Users/michaelsanders/Desktop
Michaels-MacBook-Pro-2:Desktop michaelsanders$ ls
Intro_to_Python hello_world.py
Michaels-MacBook-Pro-2:Desktop michaelsanders$ python3 hello_world.py
Hello world!
Michaels-MacBook-Pro-2:Desktop michaelsanders$
```

Why Python?

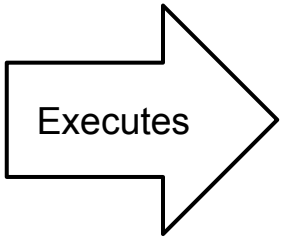
Compiled languages:

C++, Java, etc...



gcc, javac, etc...

.exe, .jar, etc...

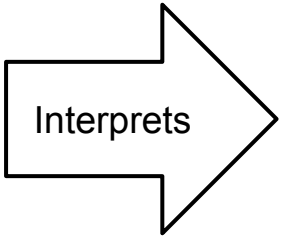


User's machine

Why Python?

Interpreted languages:

Javascript, Python, etc...



Javascript engine, Python interpreter, etc...

User's machine

Why Python?

Compiled vs Interpreted:

<u>Compiled software:</u>

1. Only needs to be compiled once
2. Typically executes faster
3. Must be redistributed whenever an update is released

<u>Interpreted software:</u>

1. Is much more flexible
2. Is inherently slower
3. Must be interpreted every time it runs

Both have their place! Which one you use depends on the problem you are solving for!

Why Python?

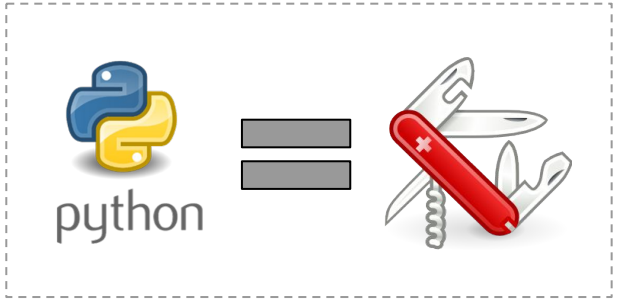
Widely applicable

Development Community

Best of Both Worlds

Scripted + Object Oriented

Why Python?



INDUSTRIAL
LIGHT & MAGIC

Python basics

Comments:

1. Start with a # symbol
2. Prevent code from executing
3. Are used to provide contextual information

Example:

comments.py

```
# This is a comment
print('This is a line of code')

# print('This will not execute')
```

Python basics

Variables:

1. Used to store a value
2. Names can start with letters or underscores and can contain numbers
3. Should not be Python keywords
4. Basic types are numbers, strings, and booleans

Example:

variables.py

```python

name = "Mike Sanders"
age = 42
print("1st pass:", name, age)

name = 42
age = "Mike Sanders"
print("2nd pass:", name, age)

bool_true = True
bool_false = False
print(bool_true, "and", bool_false)
```

Python basics

Variables:

1. Lists = [ ]
2. Tuples = ( )
3. Dictionaries = { }
4. Hold multiple values
5. Are very similar in use

Example:

other_variables.py

```python

a_list = ['1', '2', '3']
print(a_list[0])

a_tuple = (1, 2, 3)
print(a_tuple[1])

a_dict = {
    'one': '1',
    'two': '2',
    'three': '3'
}
print(a_dict['three'])

```

Python basics

Loops:

1. Used to perform an operation on each member of a collection of data
2. Two main types:
   a. ***while***
   b. for . . . in

Example:

while_loop.py

```python
numbers = [1, 2, 3, 4, 5]
sum = 0
i = 0

while i < len(numbers):
    sum += numbers[i]
    i += 1

print(sum)
```

Python basics

Loops:

1. Used to perform an operation on each member of a collection of data
2. Two main types:
   a. while
   b. for . . . in

Example:

for_in_loop.py

```python
numbers = [1, 2, 3, 4, 5]
sum = 0

for number in numbers:
    sum += number

print(sum)
```

Python basics

If statements:

1. Allow us to make decisions based on a certain criteria
2. ***Use conditional tests to determine whether or not to execute a block of code***
3. Only one code block gets executed
4. Can have a default block if none evaluate to true

Example:

if_statement.py

```
value = 10

if value < 11:
    print(str(value), 'is less than 11')

if not (value > 11):
    print(str(value), 'is not greater than 11')
```

Python basics

If statements:

1. Allow us to make decisions based on a certain criteria
2. Use conditional tests to determine whether or not to execute a block of code
3. ***Only one code block gets executed***
4. Can have a default block if none evaluate to true

Example:

if_else.py

```
value = 50

if value > 100:
    print('That\'s a big value!')
elif value > 10:
    print('That\'s a medium value!')
```

Python basics

If statements:

1. Allow us to make decisions based on a certain criteria
2. Use conditional tests to determine whether or not to execute a block of code
3. Only one code block gets executed
4. ***Can have a default block if none evaluate to true***

Example:

if_else.py

```
value = 3

if value > 100:
    print('Large')
elif value > 50:
    print('Medium')
else:
    print('Small')
```

Python basics

Functions:

1. Named blocks of code that perform a specific job
2. Called using the name of the function
3. Alleviates the need for writing the same code multiple times
4. May return a value

Example:

function.py

```
1
2  def print_my_age(age):
3      print('I am: ', str(age))
4
5  print_my_age(15)
6  print_my_age(42)
7
```

Python basics

Functions:

1. Named blocks of code that perform a specific job
2. Called using the name of the function
3. Alleviates the need for writing the same code multiple times
4. ***May return a value***

Example:

function.py

```
1
2  def get_age_text(age):
3      return 'I am: ' + str(age)
4
5  age_text = get_age_text(21)
6  print(age_text)
7
```

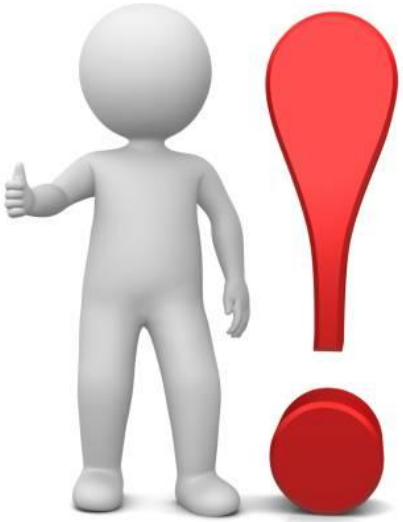
Python basics

So, how do I take advantage of the Python development community?

Using 3rd party modules

Let's use Python's package manager!

1. In a terminal shell:
    a. *pip3 install <package_name>*
2. Anaconda Navigator
    a. *The Environments tab*
3. import <package> [as <alias>]



Let’s graph some data using Python!

Recap:

1. How to install and verify Python
2. What tools can be used for coding in Python
3. How to execute Python code: Interpreter vs Scripts
4. Different applications of Python
5. Basics of Python
6. How to use external modules in our Python programs
7. How to work with data in a realistic scenario

Questions?