Introduction to Programming in Python Part 1a

By William Chan

Introduction: William Chan

- Developer with 5 years in a variety of programming languages such as Python, PHP, JavaScript, Ruby, and more recently Go
- Currently, a Software Engineer for the Platform team at ShopKeep POS and a freelancer
- Instructor of classes in front end development, HTML & CSS, JavaScript & AJAX, and PHP & MySQL at Baruch College and General Assemb.ly
- Interests in software design, best practices, development, and programming languages

Objectives of the Workshop

- Becoming a “practical” programmer by understanding the environments, tools, and frameworks in which developers work with
- Learn the software development life cycle
- Learn the general constructs shared by most programming languages
- Learn the procedural and object oriented paradigms to programming

Common Programming Paradigms

1. Procedural Programming
    - Code is executed in a linear fashion top down
2. Functional Programming
    - Functions can be composed from other functions to allow for code reuse
3. Object Oriented Programming
    - Forming objects to compose modular components to allow for code reuse

Common Constructs of Programming

1. Variables and data types
2. Conditionals
3. Loops and flow control
4. Functions
5. Classes and object oriented programming

Introduction to Python

- Python can be written procedurally
- Python can be written in an object oriented fashion
- Python is also dynamically typed

What does it mean to write code “procedurally”?

- Based on code written in a sequential fashion with functions calling other functions
- A simple example for adding 5 to a given number:

```
def add5(num):
    return num + 5


if __name__ == '__main__':
    fifteen = add5(10)
    print(fifteen)
```

What does it mean to write code in a “object oriented” way?

- Declaring “classes” of objects and use the class to create an object of that class
- Object oriented programming promotes the use of objects for modularizing programming
- A simple example of a vehicle object that has no properties and does nothing:

```
class Vehicle(object):
    pass
my_car = Vehicle()
```

What does it mean to be dynamically typed?

- Errors are typically caught at run time instead of at compile time. This also makes it an “interpreted” language and not a “compiled” language.
- An example run time error:

```
def append_world(some_string):
    return some_string.upper() + ' World'


if __name__ == '__main__':
    print(append_world('hello'))  # no run time error
    print(append_world(10))       # run time error
```

In summary, Python...

- can be written procedurally by just calling standalone functions
- can define new functions given another function
- can define classes to group functionality and data in objects
- catches errors at run time due to its dynamic typing nature