# Introduction to Python

**Mr. Kasey P. Martin, MIS**

## About Me

- Master of Information Systems - University of the Philippines
- Bachelor of Science in Computer Science - University of the Philippines
- Python Experience (since 2013):
  - Geoprocessing
  - Web Scraping
  - Scientific Computing
  - App Development
  - Data Science

## Outline

- Python Background
- Environment Installation
- Jupyter Notebook Overview
- Python Script Overview
- Python Language Essentials

## Background

- Created by Guido van Rossum
- Open-source, general purpose, high-level
- Procedural, functional, object-oriented
- Interpreted language
- Two versions: 2.7 and 3.x (We will use 3.x)

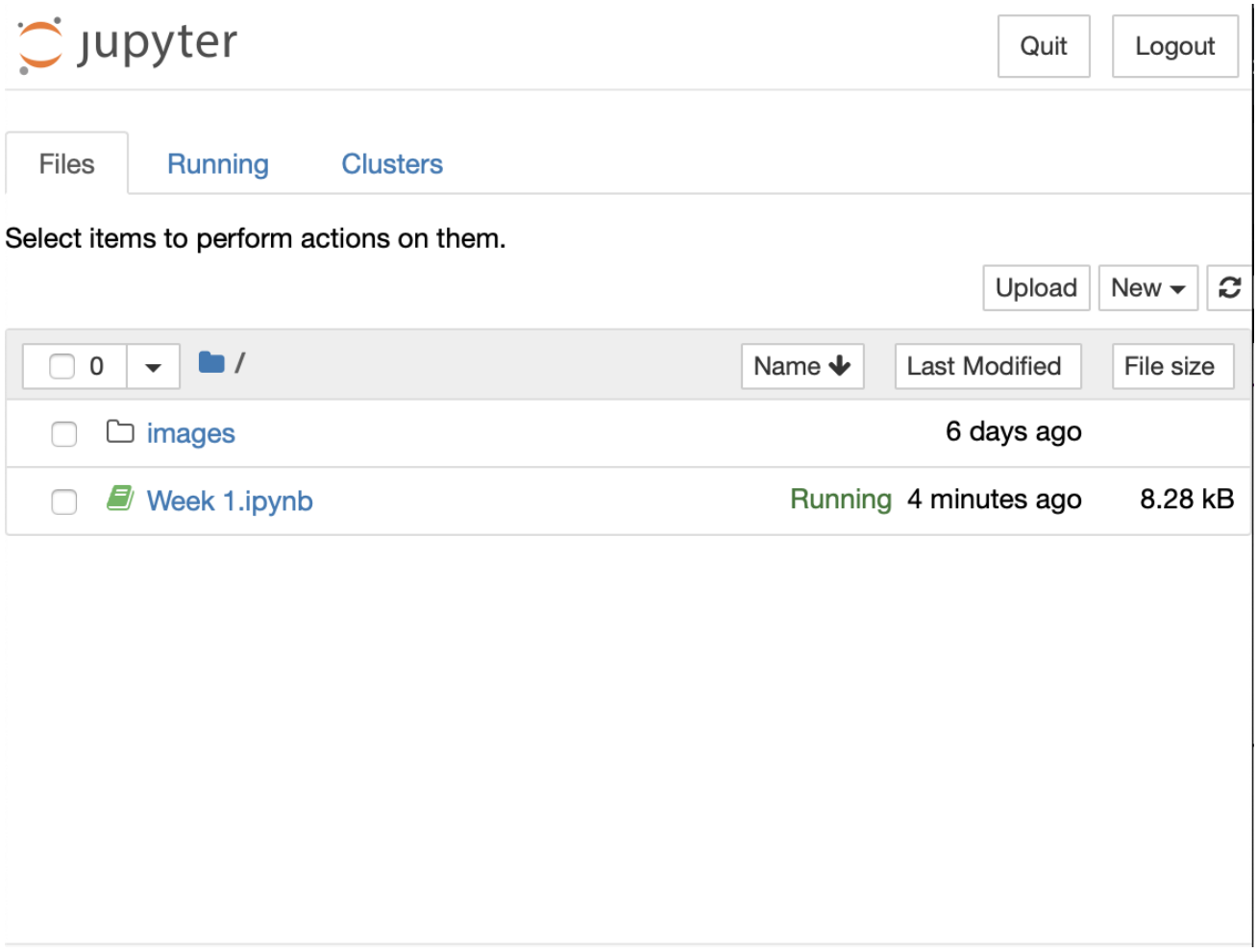
**DID YOU KNOW?** The Python language is named after Monty Python, a comedy group from the 70s.

## Enviroment Installation

1. Go to: https://www.anaconda.com/distribution/#download-section (https://www.anaconda.com/distribution/#download-section)
2. Download `Python 3.7 version`
3. Install Executable

# Jupyter Notebook Overview

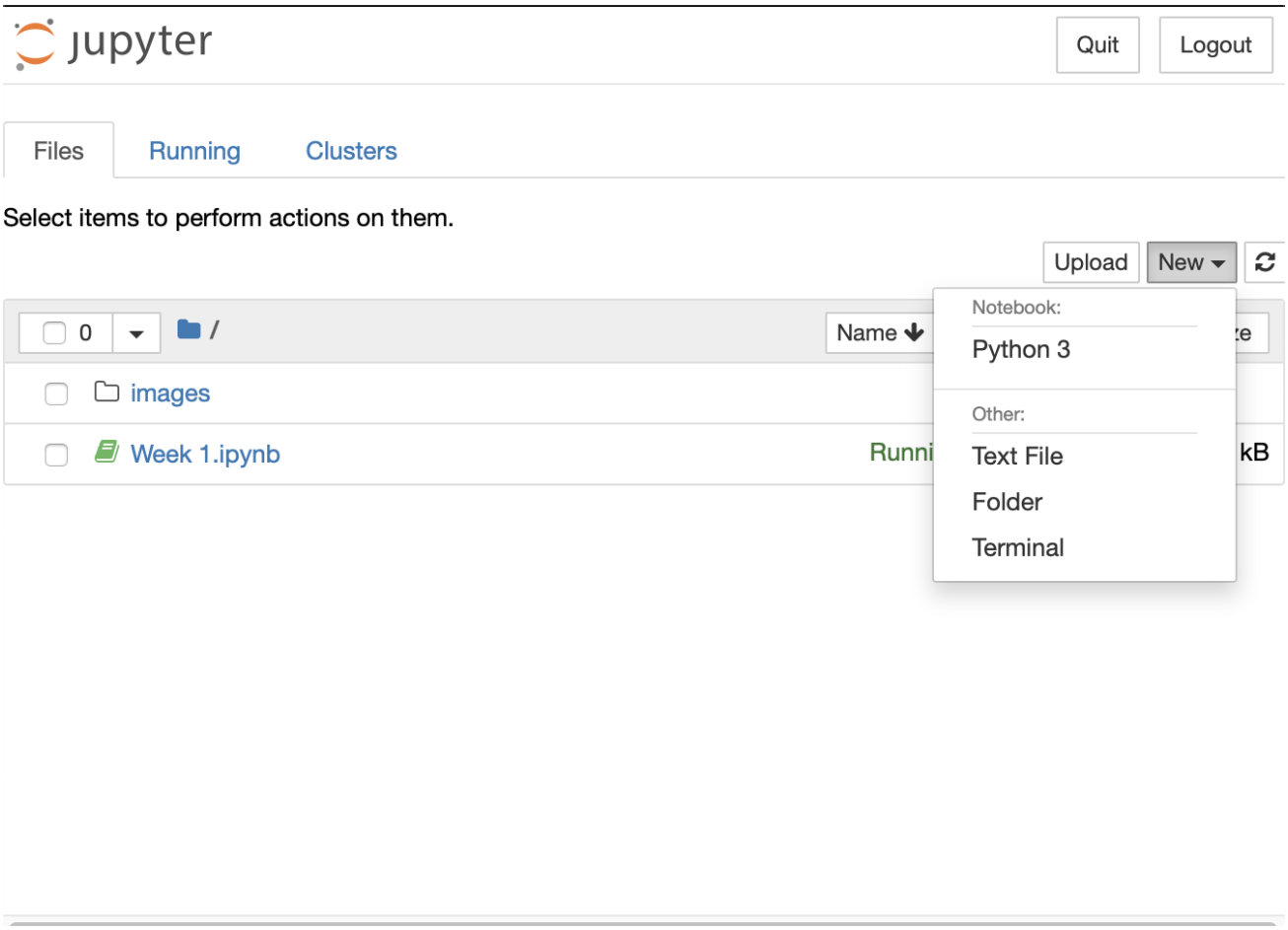
## Jupyter Notebook Interface: Dashboard




## Jupyter Notebook Interface: New Dropdown

- You can create
  - Python 3 Notebooks (.ipynb)
  - Text Files like normal .txt extensions or Python scripts (.py)
  - Folder directories
  - Terminal windows for executing commands

# Jupyter Notebook Interface: New Notebook

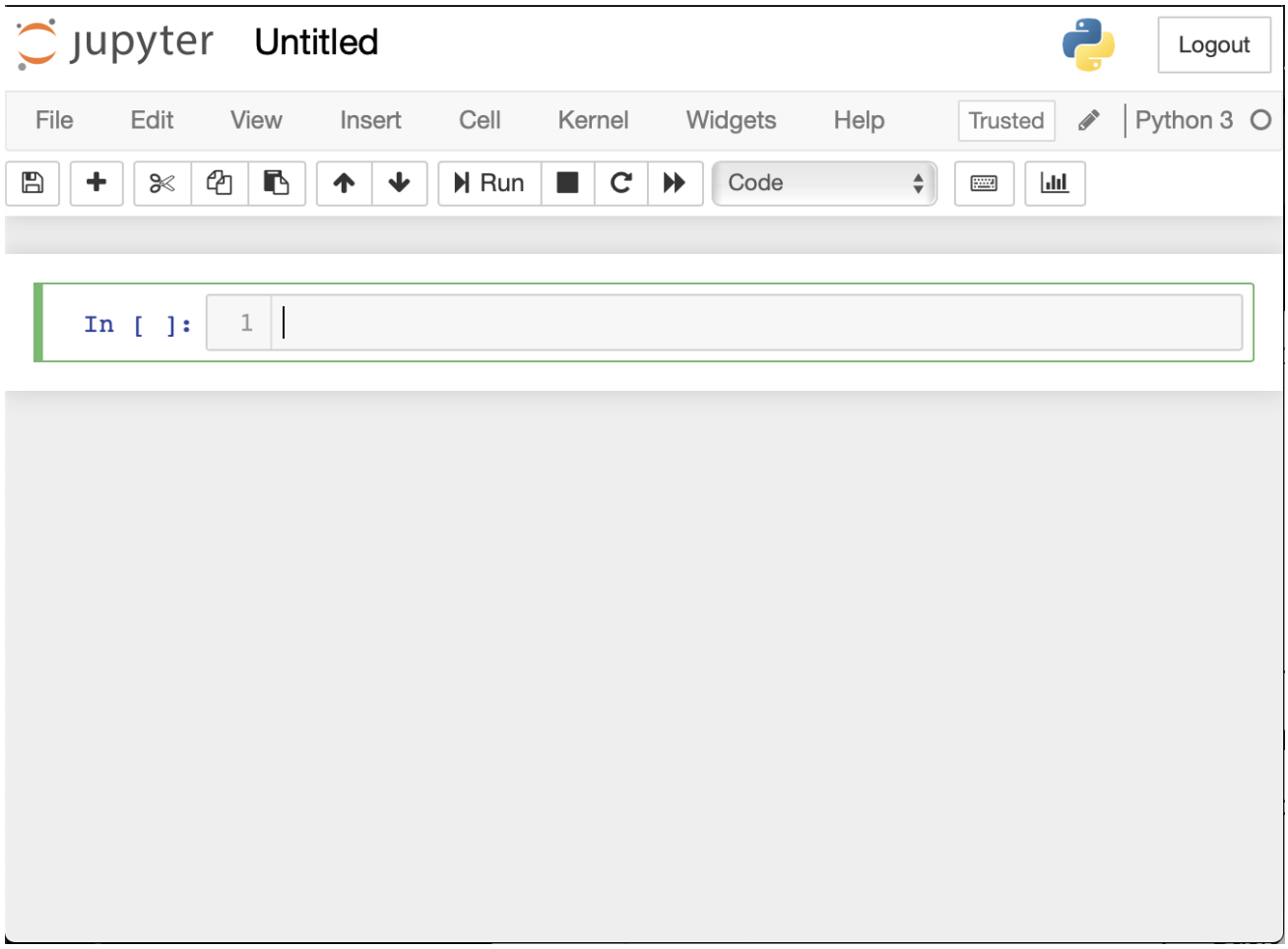



# Jupyter Notebook

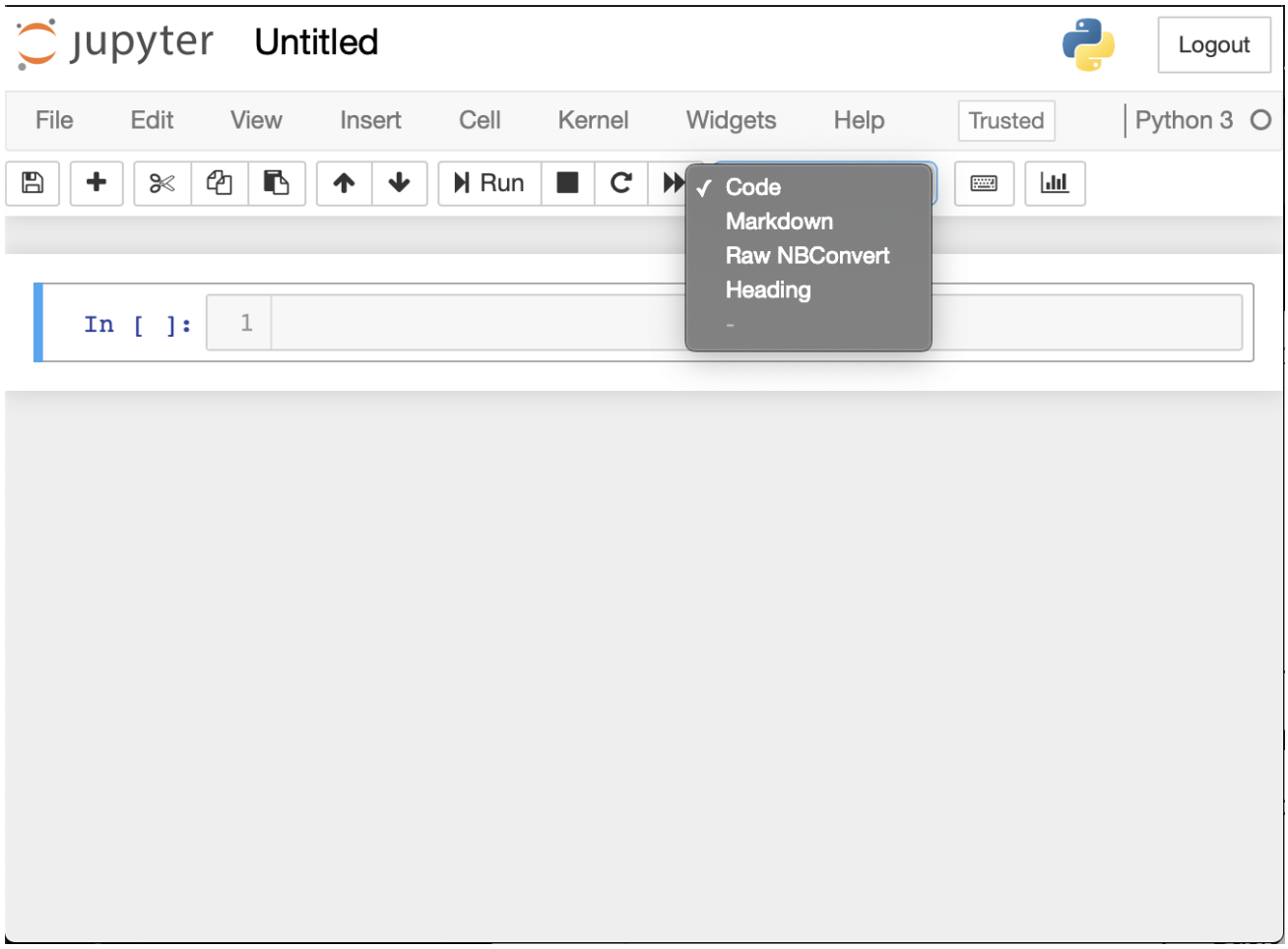
# Jupyter Notebook

- a browser-based Read-Evaluate-Print Loop (REPL) environment
- enables interactive computing
- used heavily in data science analysis
- this presentation is a jupyter notebook!

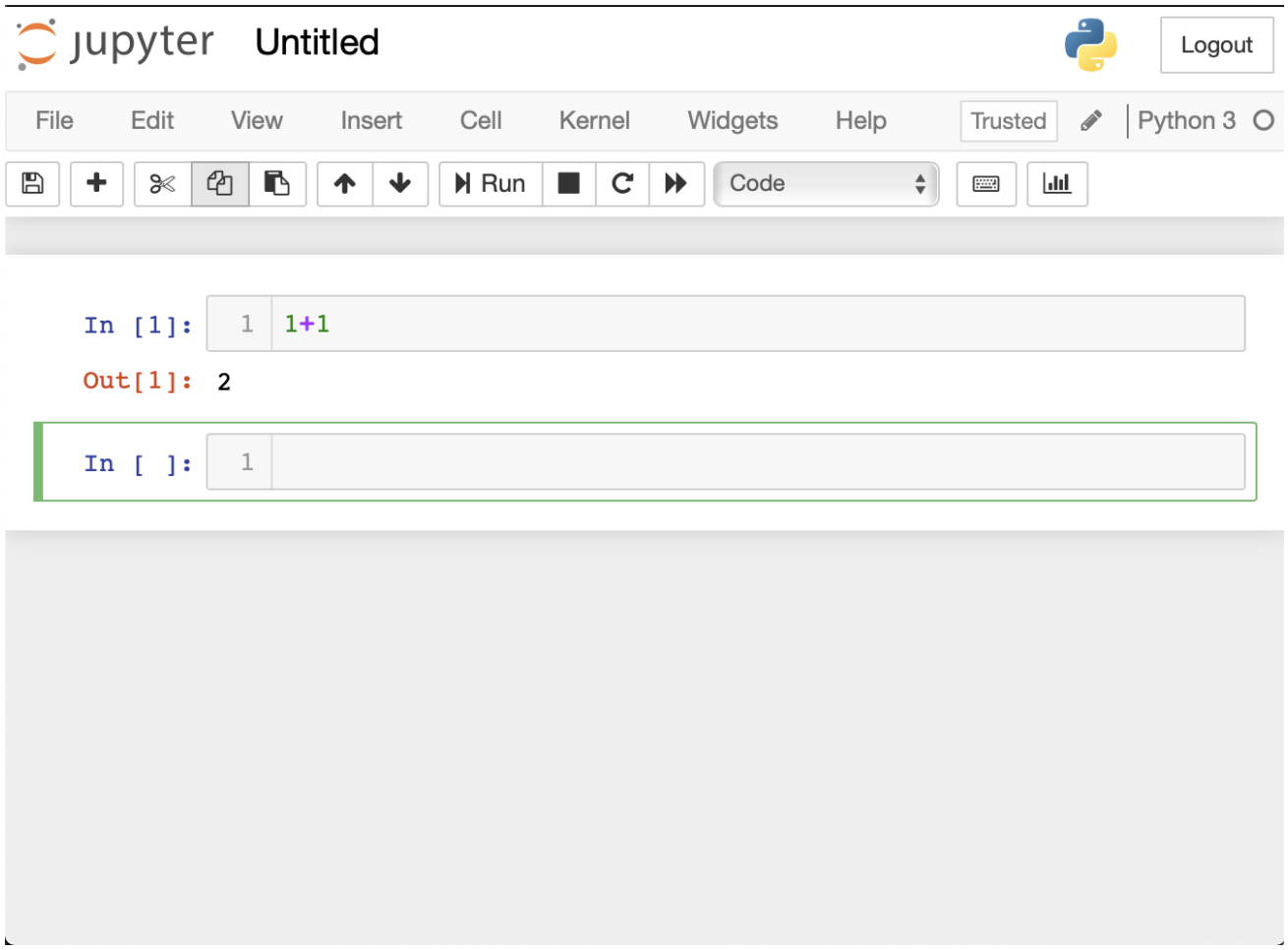
# Jupyter Notebook Interface: Notebook Interface



# Jupyter Notebook Interface: Cell-type



## Jupyter Notebook Interface: Running Cell




```
In [ ]: #!/usr/bin/env python3

        def main():
            """ Main entry point of the app """
            print("Hello EIT!")

        if __name__ == "__main__":
            # This is executed when run from the command line
            main()
```

## Common User Mistakes and Misconceptions (Jupyter Notebook vs Python Script)

- Python scripts always run sequentially, only halting to wait for user input (if any)
- You can run your Jupyter Notebook sequentially, but Jupyter Notebook allows you to run cells out of order

## Common User Mistakes and Misconceptions (Jupyter Notebook vs Python Script)

- Because Python scripts always execute sequentially, it is safer to overwrite variables
- You have to be careful with overwriting variables in different cells since you can run cells non-sequentially

## Common User Mistakes and Misconceptions (Jupyter Notebook vs Python Script)

- It takes time to get used to the programming paradigm of Jupyter Notebook
- If your logic looks good but you are still getting confusing errors, try restarting kernel then run cells sequentially again

# Let's code!

## Syntax: Line Endings

- No need for semi-colon at the end
  - In C:

    ```
    printf("Hello world!");
    ```

  - In Python:

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

In [ ]:
```
print("Hello world!")
```

## Syntax: Basic Operators

| Operator | Operation (Ordered by Precedence) |
|---|---|
| '(' and ')' | For specifying Precedence |
| ** | Exponent |
| * | Multiplication |
| / | Division |
| // | Floor Division |
| % | Modulo |
| + | Addition |
| - | Subtraction |

In [ ]:
```
# insert code here

print(10+6)
print(7*3.14)
print(7%3)
print(16+4**2/8-10)
```

## Syntax: Variables

- Naming Rules
  - Case-sensitive
  - Can contain letters, numbers, and underscores
  - Cannot start with a number
  - e.g. myVar, my_var, _myvar, number1

# Syntax: Variables

- Variables are *dynamically* typed
  - They don't have a pre-defined type
    - In C:

```c
int num1 = 75;
float pi = 3.14159;
```

    - In Python:

```python
num1 = 75
pi = 3.14159
```

In [ ]:
```python
# insert code here
mass = 15
mystr = "The mass of the object is " + str(mass) + " kg."
print(mystr)
```

# Syntax: Basic Input & Output

- `input(prompt)`
  - Prompts user for keyboard input
  - Any input from the keyboard before new line (Enter Key) is returned as a string
- `print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)`
  - first converts arbitrary number of objects to strings that are joined by `sep` and ends in `end`
  - concatinated string is printed in `file`

In [ ]:
```python
mass = input("Input mass (kg): ")
acc = input("Input acceleration (m/s): ")

force = int(mass)*int(acc)

print("The force of the object is " + str(force) + "N")
```

# Syntax: Comments

- `#<string>` - Single line comments
- `"""<string>"""` - Multi-line comments

```python
num1 = 75     # This is a comment.
    pi = 3.14159

    # This is also a comment.
    """This is a docstring. You can place this
    before a new function or class to explain
    what it does."""
```

## Syntax: Nesting

- Use indentations instead of braces
  - In C:

```
if(condition){
      //do stuff
      //here
  }
```

  - In Python:

```
if condition:
      #do stuff
      #here
```

# Simple Python Script Template

main.py

```python
#!/usr/bin/env python3

def main():
    """ Main entry point of the app """
    print("Hello World")

if __name__ == "__main__":
    # This is executed when run from the command line
    main()
```

## Reference Semantics

- If we do a = b
  - This makes a **reference** the object that b references
  - Assignment creates references, **not copies**

In [4]:
```
# insert code here
a = [1, 2, 3, 4, 5] #mutable objects
b = a # b is not a copy of a, it references the object in a
a.append(6)
print(b)

# immutable objects
a = 1
b = a
a = 2
print(a) # 1
print(b) # 2
```

```
[1, 2, 3, 4, 5, 6]
2
1
```

In [ ]:
```
#!/usr/bin/env python3

def main():
    """ Write a program that takes in a name and age, then output age in 5 years """


if __name__ == "__main__":
    # This is executed when run from the command line
    main()
```