A Gentle Introduction to Python

5c22 Computational Methods

Sigmedia

Signal Processing and Media Applications

Darren Ramsook

Small history of Python

- Python is released by Guido Van Rossum (Jan 1989)
    - V1 release (1994)
    - V2 release (2000)
    - V3 release (2008) [*This is what we will be using!!*]
- Python is named after the hit comedy series Monty Python (link)
- Open source from day 1
- Due to its open codebase and its “relatively easy” code structure and levels of abstraction
    - Very popular for creating fast working prototypes
    - Loads of open source libraries for almost all domains of expertise
    - The go to tool in the “modern” era for data processing

What is Python?

- Python is a high level, general purpose development language
  - Great deal of libraries that are constantly updated to help with whatever task you have
    - Tensorflow / Pytorch - Deep Learning
    - Matplotlib / Seaborn - Data Visualization
    - Plotly - Dashboards
    - Scipy / Numpy - Matrix operations
    - PyGame - Making python based video games!

- Python is designed with emphasis on code readability
  - Very good for getting things working quickly, and allowing others to understand what you’ve done

- Python is an **interpreted** language, meaning that it is **much slower** than compiled languages (C++), so there are certain use-cases where Python will not be ideal

How can we use Python?

Recommended (Basic) Setup:

1) Python Interpreter
    - The lab machines *should* already has Python installed.
    - If you want to install Python on your own machine, follow the steps outlined: https://www.python.org/downloads/

2) Code editor
    - Visual studio code (VS Code) is a popular editor with loads of extensions available
    - VS Code *should* already installed on your lab machines
    - Link to install on your own machine: https://code.visualstudio.com/

3) Python Extension for VS Code (optional but recommended)
    - Visit the Extension Marketplace in VS Code
    - Search for Python and install the "*Python*" extension (verified by Microsoft)

Setting up a workspace

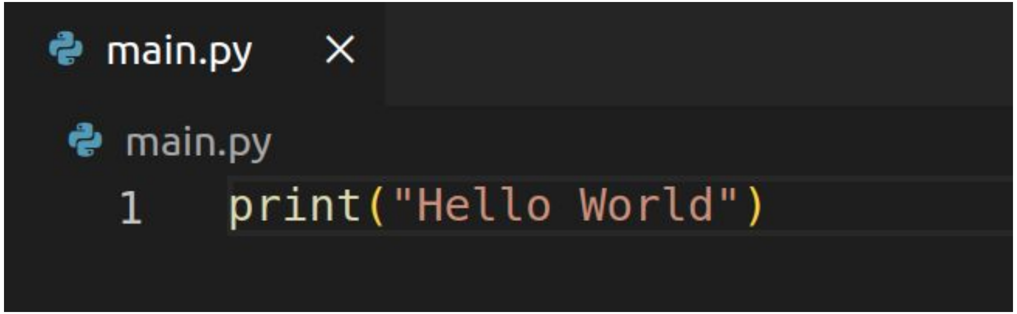
- Create a new folder somewhere in your machine to house all the files you need for this project
  - I recommend creating a “*PythonTutorial*” folder in your “*Documents*” folder (*“Documents/PythonTutorial”*)

- Open this folder using VS Code
  - File -> Open Folder -> Navigate to “*PythonTutorial*” and open

Running your first program

- Create a new file in VS Code and name it “main.py”
  - Right-click in the File Explorer in VSCode and choose new file

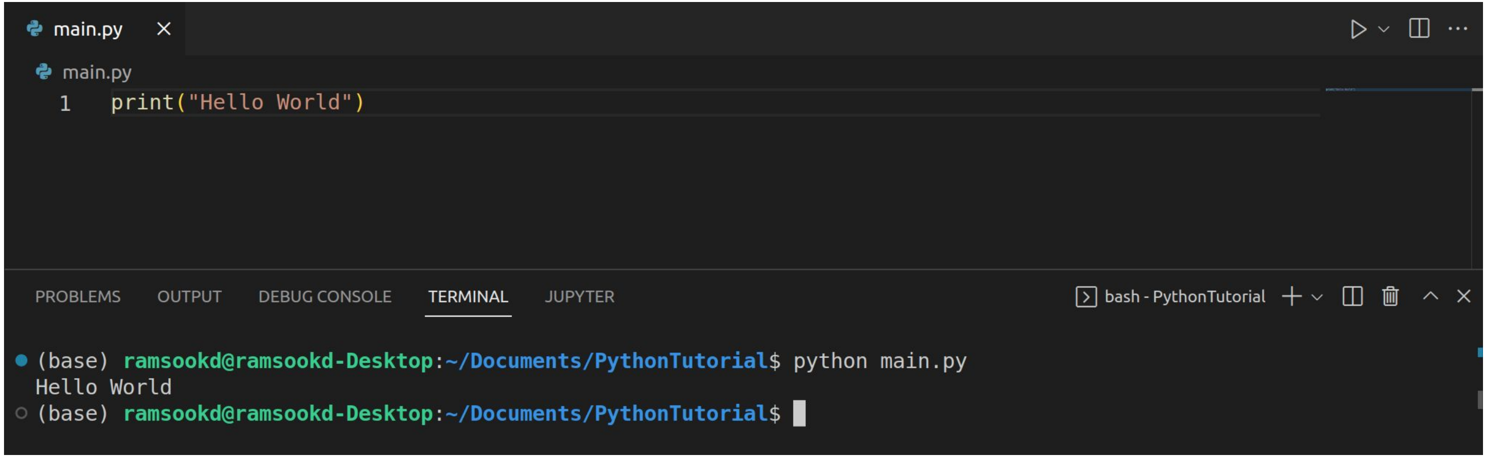
- Remember the “*Hello World*” Demo from your first day of programming?
  - It’s even easier in Python!
  - Print any text you want on the screen by using the print() command



```python
print("Hello World")
```

Running your first program - 2

- Now that we have a working program (“main.py”), we can run this through the Python Interpreter
- In VS Code, open a terminal window
  - In the toolbar click on Terminal -> New Terminal
- Execute “main.py” by running: python main.py



```python
print("Hello World")
```

```
(base) ramsookd@ramsookd-Desktop:~/Documents/PythonTutorial$ python main.py
Hello World
(base) ramsookd@ramsookd-Desktop:~/Documents/PythonTutorial$
```

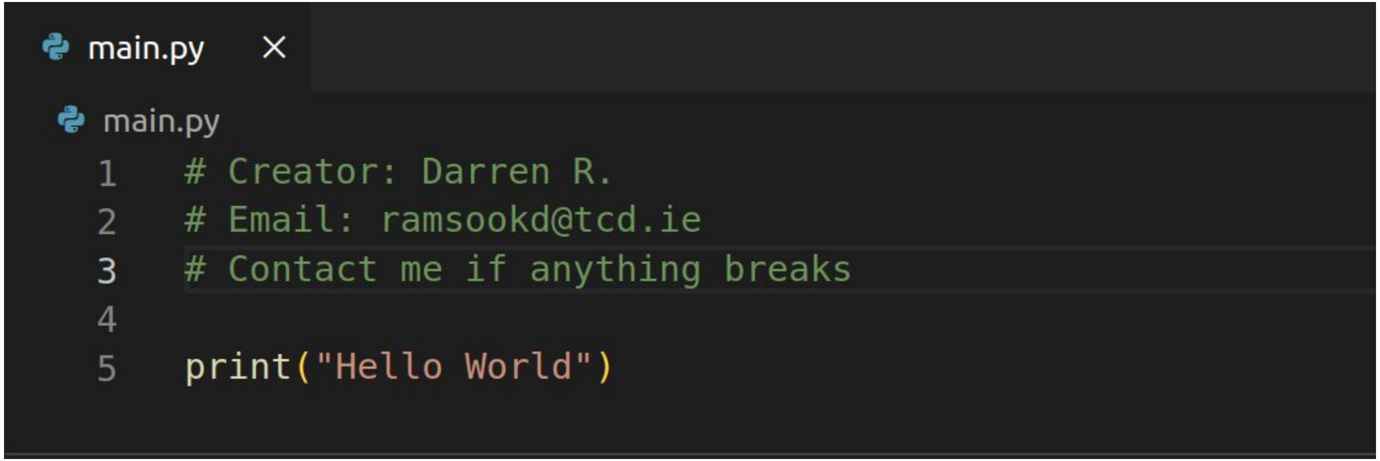
Congratulations!

- Congratulations on your first Python Program
    - You can now put “Python Programmer” on your CV

- But wait, all we did was use the print() function
    - There has to be more? Right?

Comments

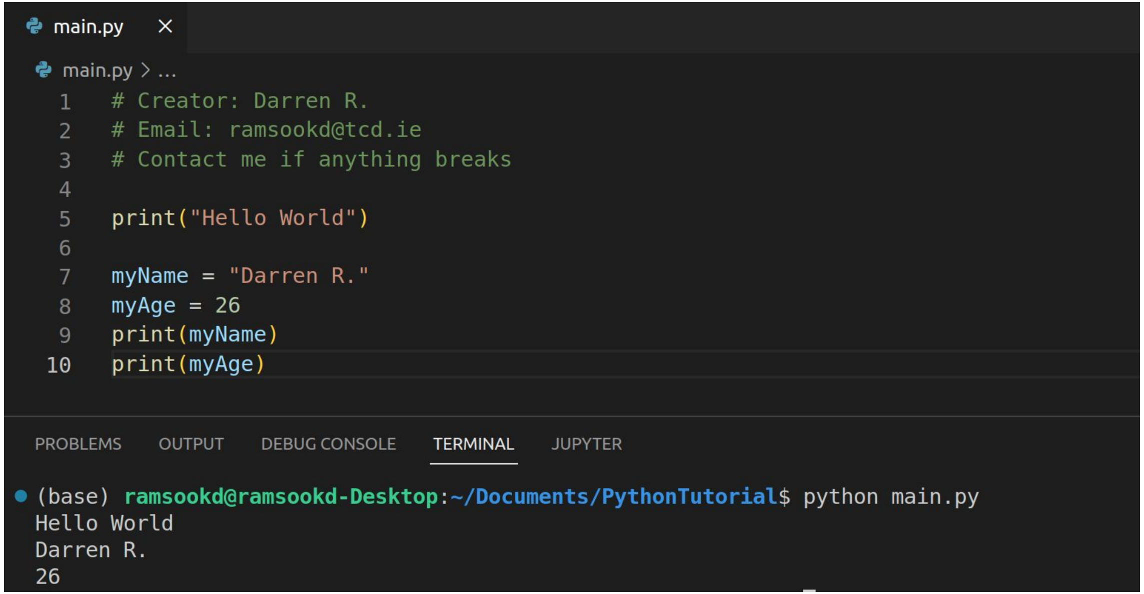
- Adding comments to code can help you (*and others!*) understand the general workflow of your program
  - Add comments by using the # symbol *WRITE WHAT CODE IS SUPPOSED TO DO*



```python
# Creator: Darren R.
# Email: ramsookd@tcd.ie
# Contact me if anything breaks

print("Hello World")
```

Variables

- Variables are containers used for storing data values
  - Python has no command for declaring variables, instead they are created the moment you assign a value to it
  - Use variables by: *variableName = variableValue*

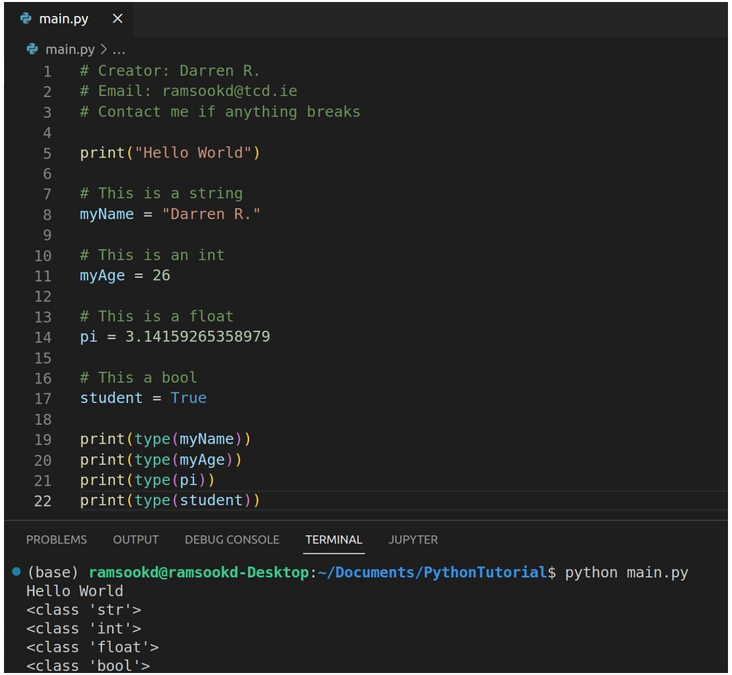


```python
# Creator: Darren R.
# Email: ramsookd@tcd.ie
# Contact me if anything breaks

print("Hello World")

myName = "Darren R."
myAge = 26
print(myName)
print(myAge)
```

```
(base) ramsookd@ramsookd-Desktop:~/Documents/PythonTutorial$ python main.py
Hello World
Darren R.
26
```

Data Types

- Python is stocked full with multiple different datatypes
  - Python will understand the data type you intend to use via your variable declaration
- Some of the basic data types are:
  - *string* - Used for storing text characters
  - *int* - Used for storing Integer values
  - *float* - Used to store 64 bit numbers
  - *bool* - True or False, binary type data

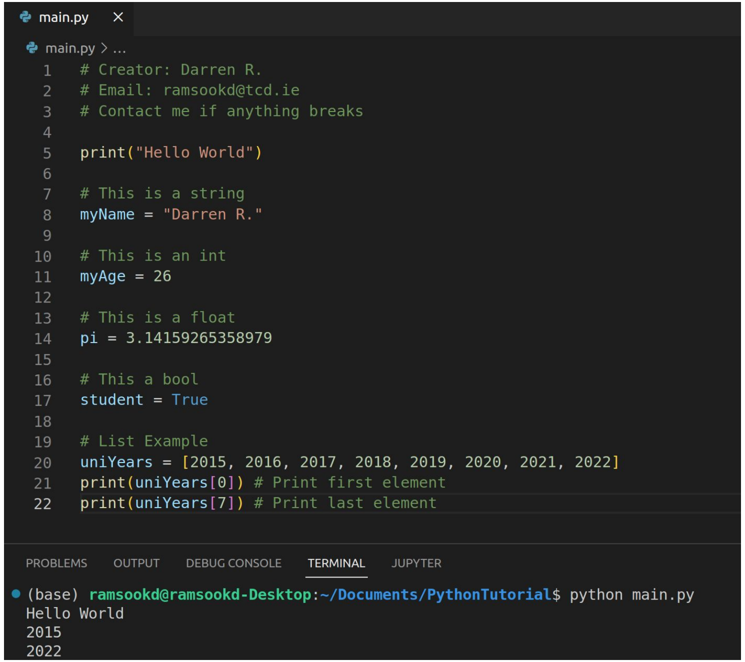


```python
# Creator: Darren R.
# Email: ramsookd@tcd.ie
# Contact me if anything breaks

print("Hello World")

# This is a string
myName = "Darren R."

# This is an int
myAge = 26

# This is a float
pi = 3.14159265358979

# This a bool
student = True

print(type(myName))
print(type(myAge))
print(type(pi))
print(type(student))
```

```
(base) ramsookd@ramsookd-Desktop:~/Documents/PythonTutorial$ python main.py
Hello World
<class 'str'>
<class 'int'>
<class 'float'>
<class 'bool'>
```

Data Structures in Python - List

- Lists are used in python to store a sequence of data elements
  - Lists are created using: *listName = [‘a’, ‘b’, ‘c’, ‘d’]*
  - Lists can be indexed by: *listName[x]*, where x is the index of the list
  - Lists indexing starts at *0* and goes to *N-1*, where *N* is the length of the list

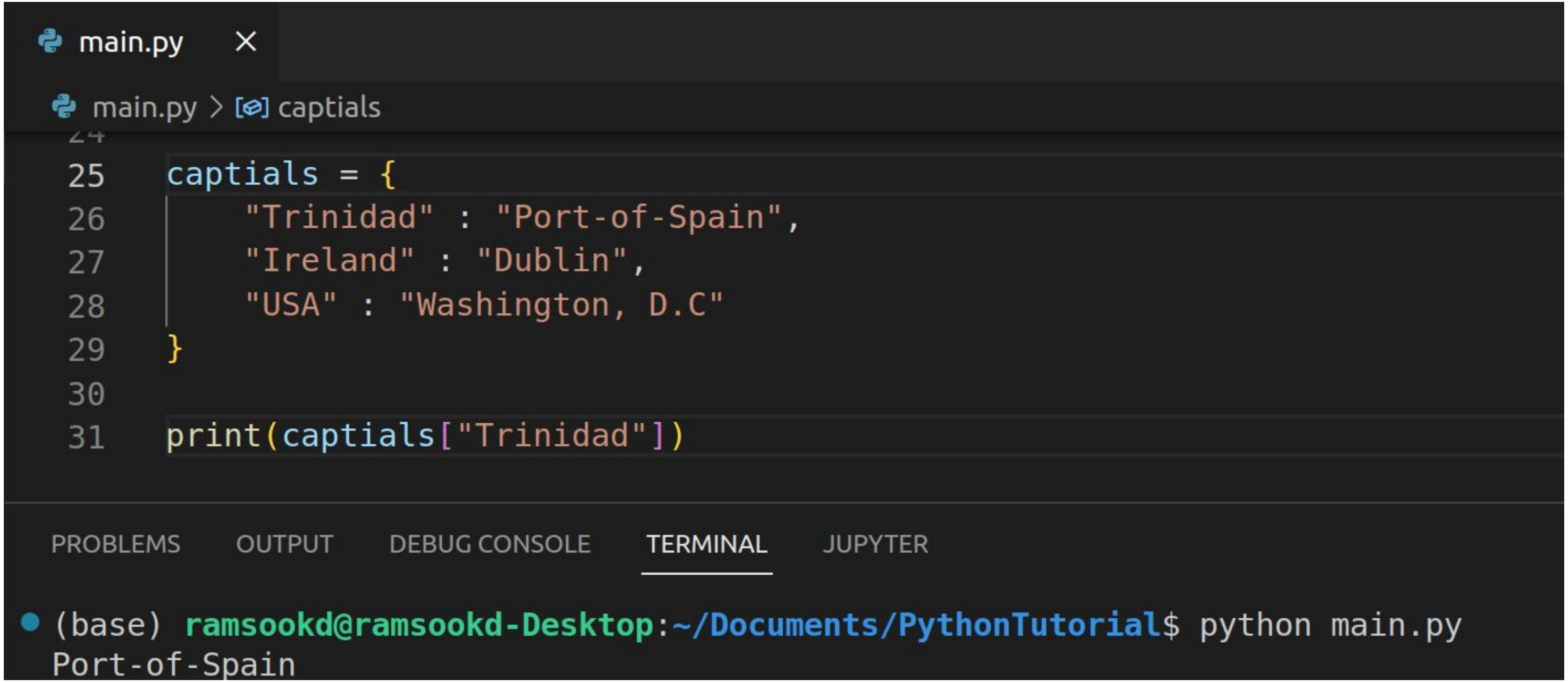


```python
# Creator: Darren R.
# Email: ramsookd@tcd.ie
# Contact me if anything breaks

print("Hello World")

# This is a string
myName = "Darren R."

# This is an int
myAge = 26

# This is a float
pi = 3.14159265358979

# This a bool
student = True

# List Example
uniYears = [2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022]
print(uniYears[0]) # Print first element
print(uniYears[7]) # Print last element
```

```
(base) ramsookd@ramsookd-Desktop:~/Documents/PythonTutorial$ python main.py
Hello World
2015
2022
```

Data Structures in Python - Dictionary

- Dictionaries are used to store data in key:value pairs
    - Note: Keys must be unique!

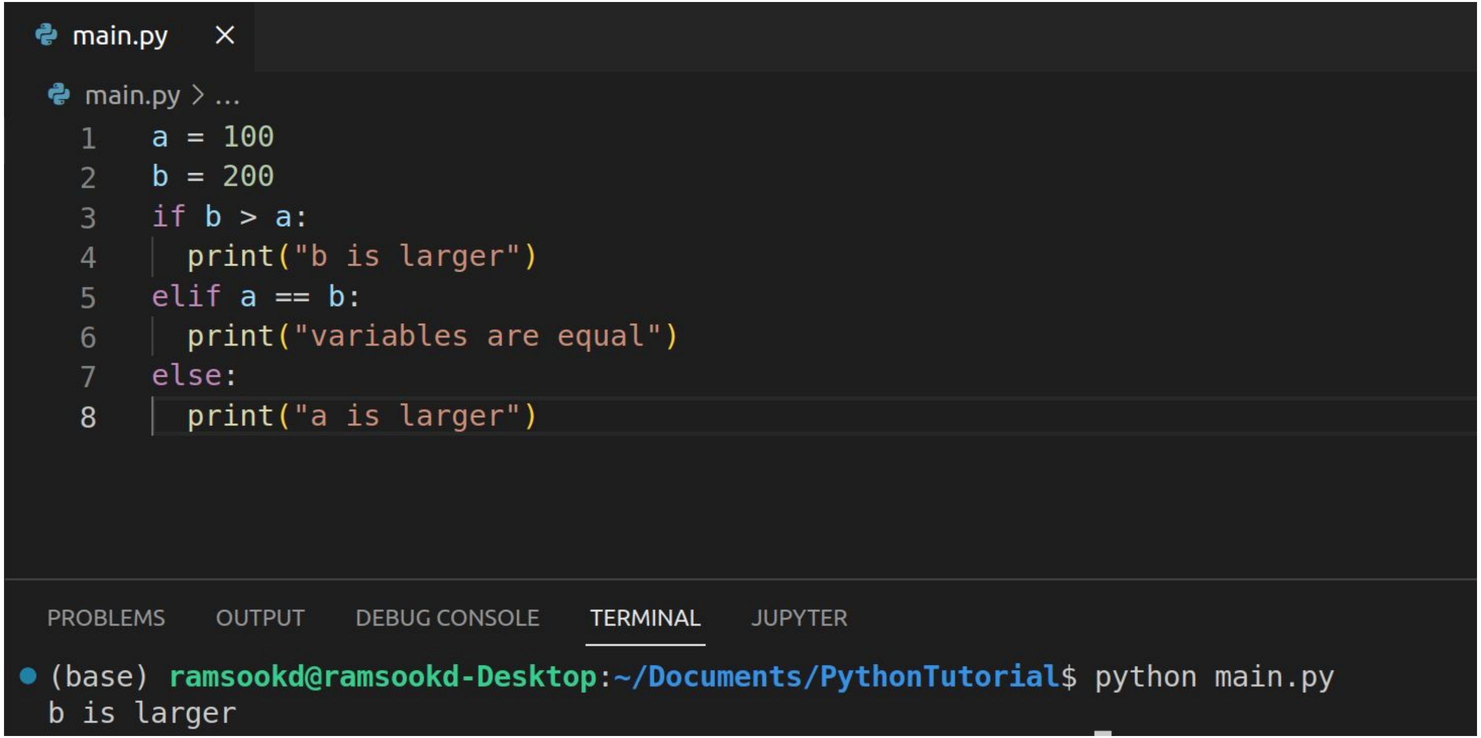


```python
captials = {
    "Trinidad" : "Port-of-Spain",
    "Ireland" : "Dublin",
    "USA" : "Washington, D.C"
}

print(captials["Trinidad"])
```

```
(base) ramsookd@ramsookd-Desktop:~/Documents/PythonTutorial$ python main.py
Port-of-Spain
```

Control Sequences

- If … else

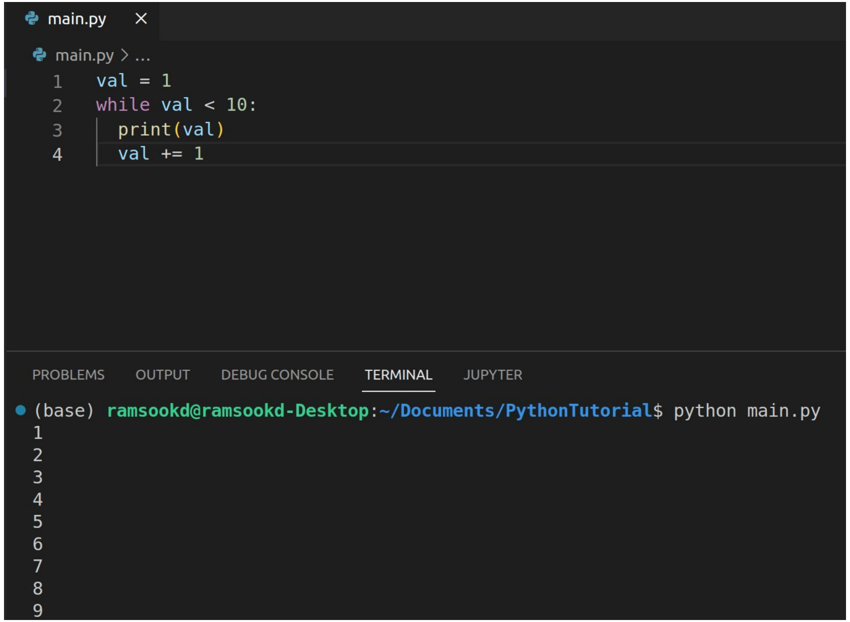


```python
a = 100
b = 200
if b > a:
  print("b is larger")
elif a == b:
  print("variables are equal")
else:
  print("a is larger")
```

```
(base) ramsookd@ramsookd-Desktop:~/Documents/PythonTutorial$ python main.py
b is larger
```

Control Sequences - While Loops

- Execute some code while a condition is True
  - Getting stuck in while loops can be very dangerous!
    - Use Ctrl+C to terminate program if this happens

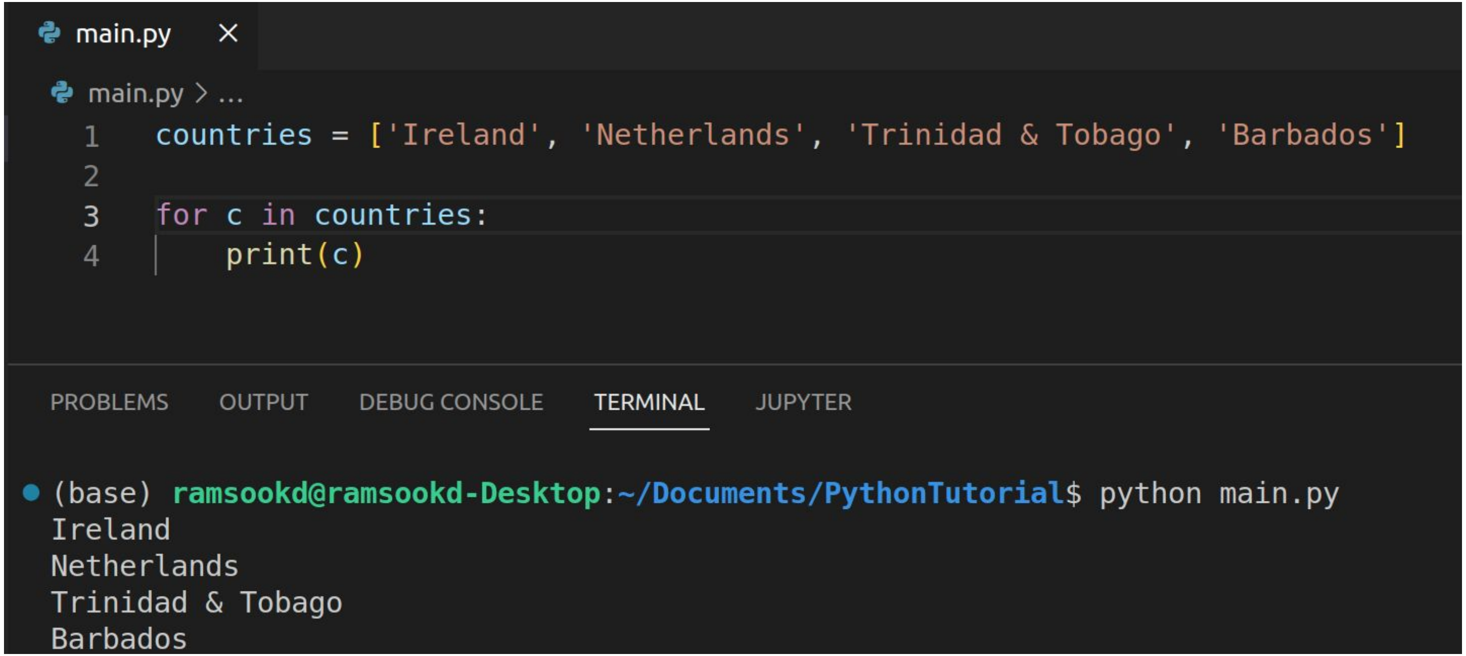


```python
val = 1
while val < 10:
  print(val)
  val += 1
```

```
(base) ramsookd@ramsookd-Desktop:~/Documents/PythonTutorial$ python main.py
1
2
3
4
5
6
7
8
9
```

Control Sequences - For Loops

- Use a for loop for iterating over a sequence

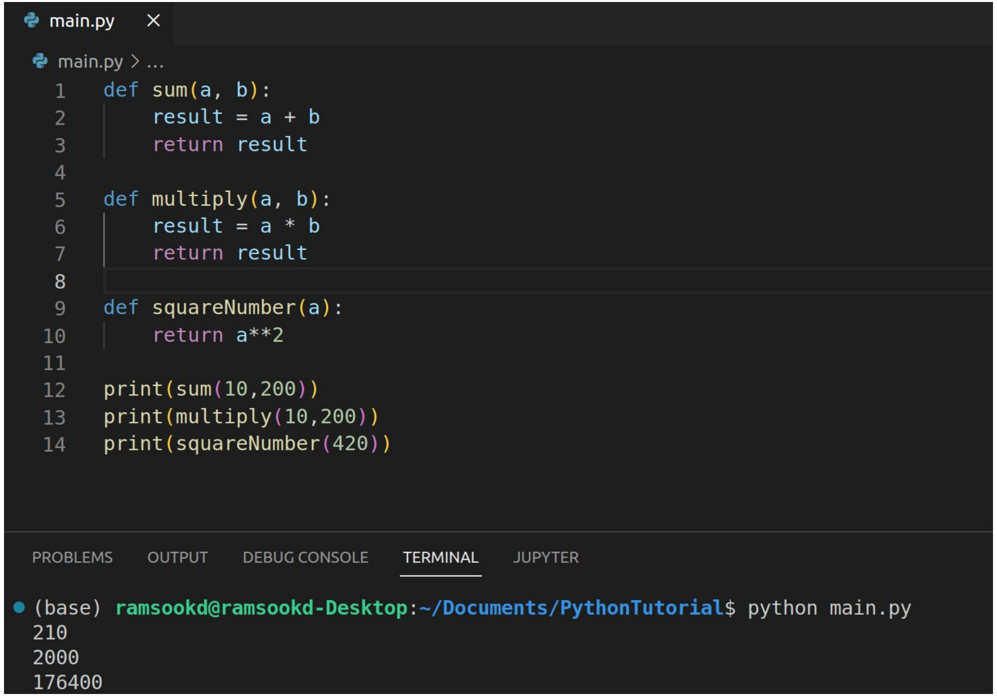


```python
countries = ['Ireland', 'Netherlands', 'Trinidad & Tobago', 'Barbados']

for c in countries:
    print(c)
```

```
(base) ramsookd@ramsookd-Desktop:~/Documents/PythonTutorial$ python main.py
Ireland
Netherlands
Trinidad & Tobago
Barbados
```

Creating Functions

- Functions are useful for creating code blocks that will be used in multiple scenarios

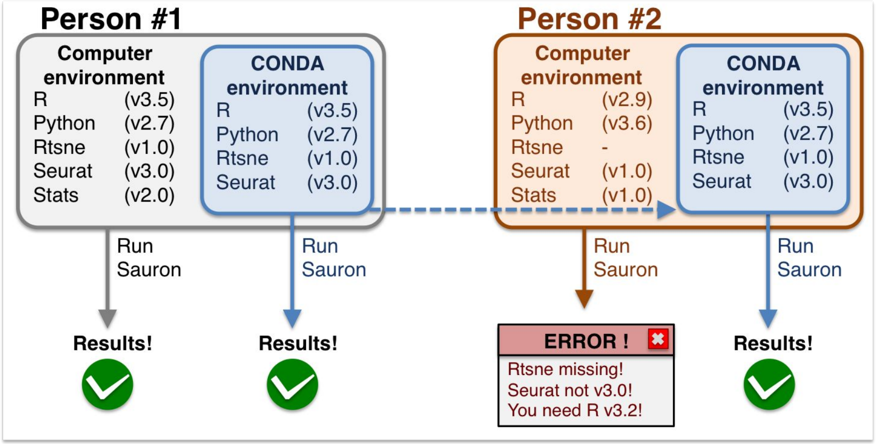


```python
def sum(a, b):
    result = a + b
    return result

def multiply(a, b):
    result = a * b
    return result

def squareNumber(a):
    return a**2

print(sum(10,200))
print(multiply(10,200))
print(squareNumber(420))
```

```
(base) ramsookd@ramsookd-Desktop:~/Documents/PythonTutorial$ python main.py
210
2000
176400
```

Revisiting our setup

- As stated before, Python’s real power comes from its open community the libraries available
- However things can get messy if we just start installing libraries randomly
- To help “containerize” this, we can use conda to help separate our projects



Conda

- Conda *should be* installed on your lab machines already
  - If not, you can install conda on your machine: https://docs.conda.io/projects/conda/en/latest/user-guide/install/index.html

- I also have this on standby whenever I use conda: https://docs.conda.io/projects/conda/en/4.6.0/_downloads/52a95608c49671267e40c689e0bc00ca/conda-cheatsheet.pdf
  - It’s a conda cheat sheet!

Creating an environment for our tutorial

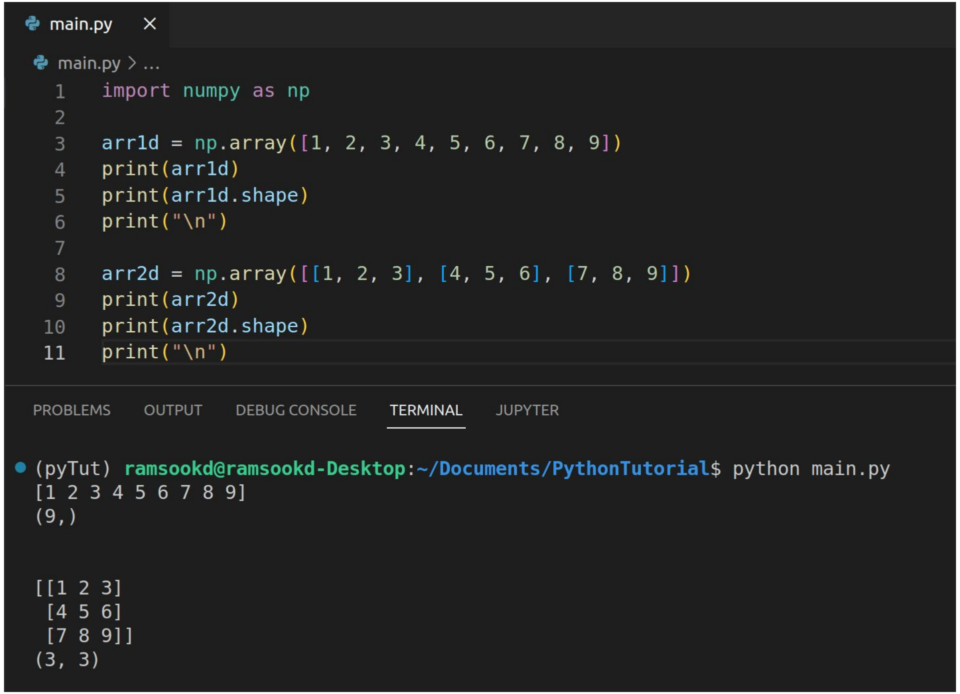
- In our next exercise, we want to use numpy
  - However we don’t want to install numpy for all Python projects, but just for our tutorial exercise

- We can solve this by using conda
  - First create a conda virtual environment: *conda create --name pyTut python=3*
  - Then activate this virtual environment by using: *conda activate pyTut*

- Once we have the virtual environment “pyTut” activated, all installed packages/libraries will be only installed to pyTut
  - Lets install numpy by running: *pip install numpy*

Using Numpy

- Numpy is a library created with the use of array mathematics in mind
  - Uses C/C++ backend for doing array calculations



```python
import numpy as np

arr1d = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9])
print(arr1d)
print(arr1d.shape)
print("\n")

arr2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print(arr2d)
print(arr2d.shape)
print("\n")
```

```
(pyTut) ramsookd@ramsookd-Desktop:~/Documents/PythonTutorial$ python main.py
[1 2 3 4 5 6 7 8 9]
(9,)


[[1 2 3]
 [4 5 6]
 [7 8 9]]
(3, 3)
```

We’ve finished our development

- But how can we be sure someone else can run our code?
- We installed NumPy, but not everyone has NumPy!
  - In a real project, you may have multiple different libraries
  - Ensuring that someone else is able to run your code is crucial
- A list of installed packages can be generated by pipreqs!

- Install pipreqs through: *pip install pipreqs*
- Now you generate a list of installed packages via:
    - *pipreqs .*

- This will generate a requirements.txt file in your working directory