# Introduction into *Python* programming

Parcours Progis

Etudes, Medias, communication, Marketing

Bahareh Afshinpour

- Introduction into *Python* programming with:
  - the traditional "hello world";
  - some data structures: lists, dictionaries;
  - If, loops;
  - Functions;

## WHY PYTHON

- Is a general-purpose language
- Python is one of the easier ones to learn
- There are lots of free tools out there you can use to code or learn Python
- Matured Community
- It has a large Libraries And Frameworks
- For AI, it has libraries like TensorFlow, PyTorch and Scikit-learn
- Python can also be used for Natural Language Processing using the NLTK

## SO HOW DO I GET STARTED?

## PYTHON LIBRARIES FOR DATA SCIENCE

Many popular Python toolboxes/libraries:

- NumPy
- SciPy
- Pandas
- SciKit-Learn

Visualization libraries

- matplotlib
- Seaborn

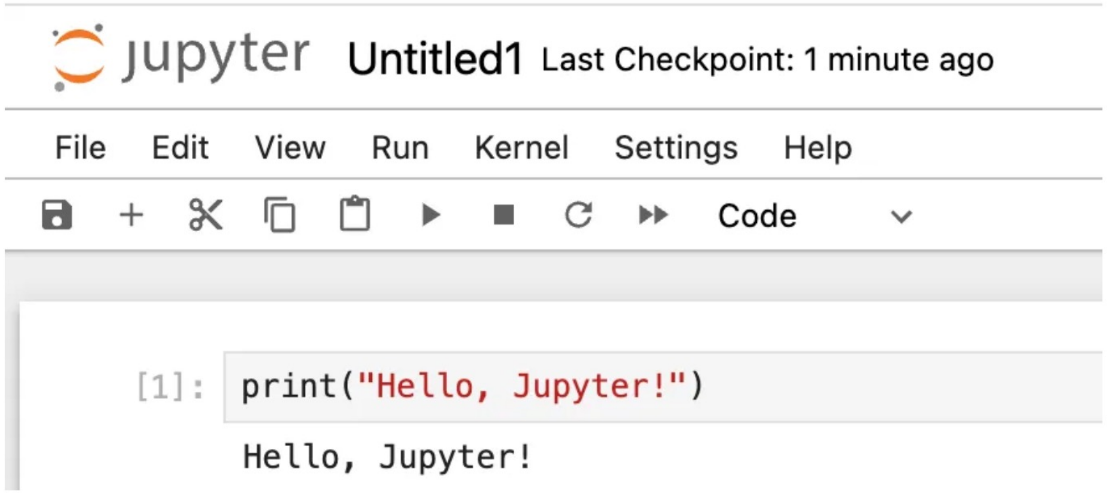
and many more …

## PYTHON SCIKIT-LEARN

- Popular machine learning toolkit in Python http://scikit-learn.org/stable/
  - Good documentation
  - Is realy easy to implement
  - Has most of the classification, regression and clustering algorithm
- Requirements
  - Anaconda
  - Available from https://www.continuum.io/downloads
  - Includes numpy, scipy, and scikit-learn (former two are necessary for scikit-learn)

# JUPYTER

- Jupyter is a freely available web application
- Jupyter promotes collaboration and reproducibility by allowing users to share their notebooks with others via email, GitHub, or the Jupyter Notebook Viewer.



```
[1]: print("Hello, Jupyter!")
     Hello, Jupyter!
```

## PICK UP GOOD HABITS RIGHT AWAY!

- Comments in your code help you or someone else understand
  - What your program does
  - What a particular line or section of code does
  - Why you chose to do something a particular way
  - Anything that might be helpful to know if I am looking at the code later and trying to understand it!

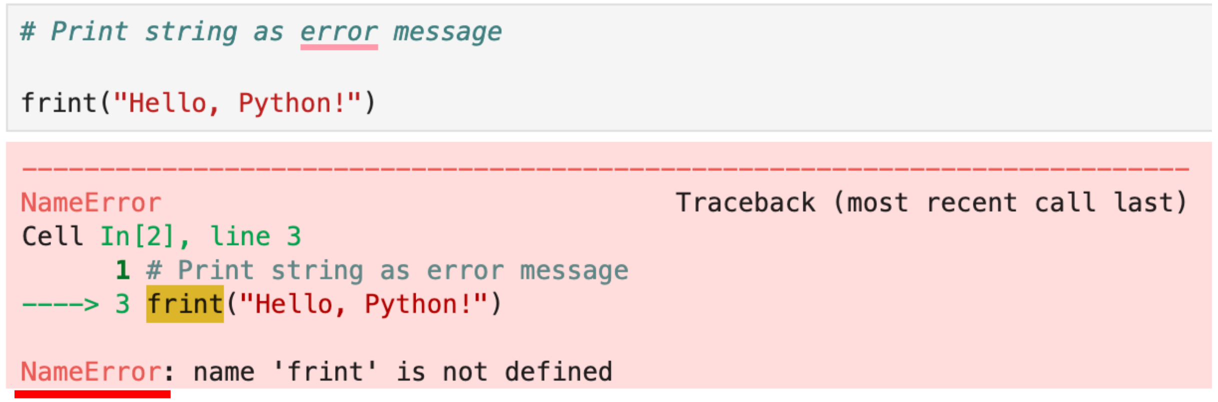
## IN PYTHON WE USE A # TO INDICATE COMMENTS

```
#My first Python Application
#Created by me!
#Print command displays a message on the screen
print('Hello World')
```

Did you notice the colors?

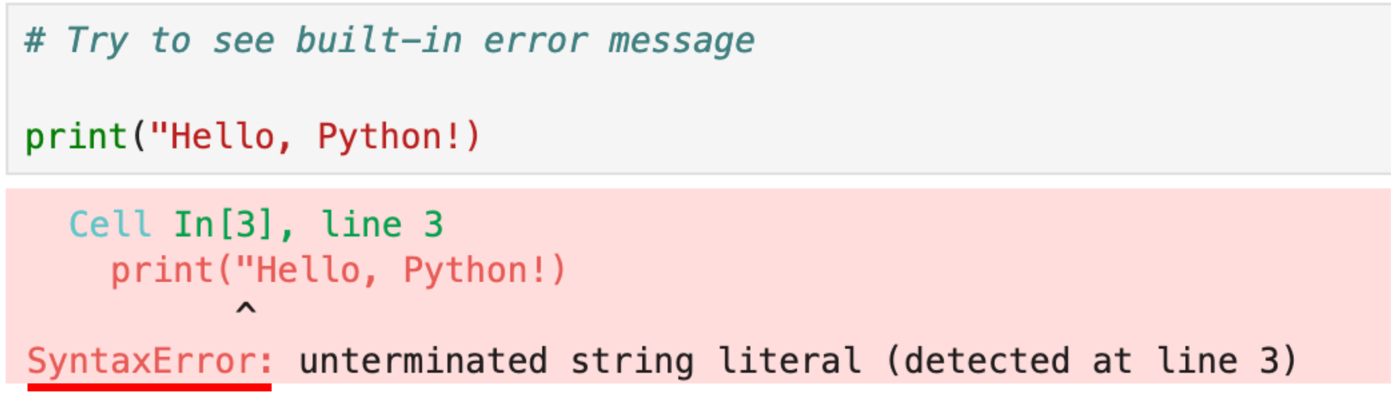
## ERRORS IN PYTHON

- It is important to read error messages carefully.



```
# Print string as error message

frint("Hello, Python!")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[2], line 3
      1 # Print string as error message
----> 3 frint("Hello, Python!")

NameError: name 'frint' is not defined
```



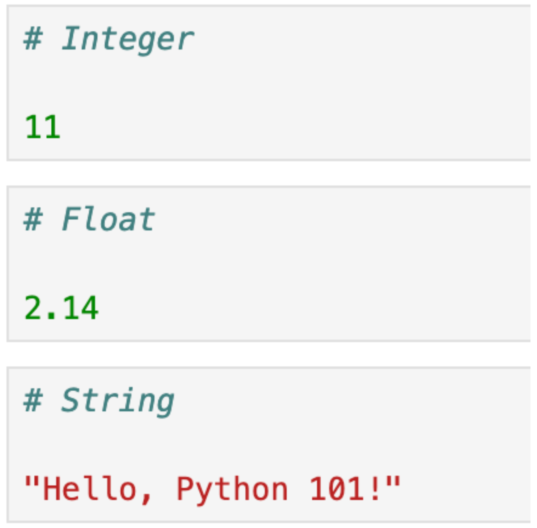
```
# Try to see built-in error message

print("Hello, Python!)
```

```
  Cell In[3], line 3
    print("Hello, Python!)
          ^
SyntaxError: unterminated string literal (detected at line 3)
```

Python interprets your script line by line as it executes it. Python will stop executing the entire program when it encounters an error

## DATA TYPES

- A type is how Python represents different types of data.
  - Integers, real numbers, string and boolean
- In Python, a string is a sequence of characters.
  - A string can be spaces or digits. A string can also be special characters.
- You can change the type of the expression in Python, this is called typecasting.



```
# Integer

11
```

```
# Float

2.14
```

```
# String

"Hello, Python 101!"
```

## VARIABLES

- We can use variables to store values.
- We assign a value to a variable using the assignment operator, i.e, the equal sign.
- We can then use the value somewhere else in the code by typing the exact name of the variable.
- We can use the type command in variables as well.

## LISTS

| **Index** | **0** | **1** | **2** | **3** | **4** |
|---|---|---|---|---|---|
| **List Data** | David | 4.12 | 6 | **[3,9]** | 657 |

**[David,4.12,6,[3,9],657]**

- Lists are a popular data structure in Python.
- Each box has a numerical reference called an index that is used to refer to the individual data item.
- A list is represented with square brackets.
- Lists can contain strings, floats, integers. Also, we can nest other lists.
- We also nest tuples and other data structures.
- Note that in Python the first element of the list shown here has an index of zero.

## LIST OPERATIONS

- Lists are mutable; can be changed in-place
- Lists are dynamic; size may be changed

```
>>> r = [1, 2.0, 3, 5]
>>> r[3] = 'word'                  # replace an item by index
>>> r
[1, 2.0, 3, 'word']
```

```
>>> len(r)                         # length of list; number of items
4
```

```
>>> 6 in r                         # membership test
True
```

## LIST METHODS, PART 1

- Lists have a set of built-in methods
- Some methods change the list in-place

```
>>> r = [1, 2.0, 3, 5]
>>> r.append('thing')                # add a single item to the end
>>> r
[1, 2.0, 3, 5, 'thing']
>>> r.append(['another', 'list'])  # list treated as a single item
>>> r
[1, 2.0, 3, 5, 'thing', ['another', 'list']]
```

```
>>> r = [2, 5, -1, 0, 20]
>>> r.sort()
>>> r
[-1, 0, 2, 5, 20]
```

```
>>> s = 'a few words'
>>> w = s.split()                 # splits at white-space (blank, newline)
>>> w
['a', 'few', 'words']
```

## DICTIONARY

- An unordered collection of key/value pairs
- Each key maps to a value

```
>>> h = {'key': 12, 'sara': 'word'}
>>> h['key']                                        # access by key
12
```

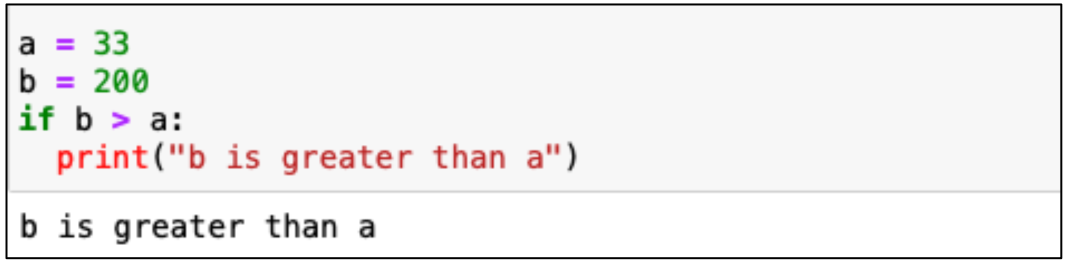
```
>>> h['Per'] = 'smith'                            # adding a key/value
>>> h
{‘sara': 'word', 'Per': ‘smith', 'key': 12}  # the output order is random
>>> h['Per'] = 'Johansson'                       # replaces the value
>>> h
{‘sara': 'word', 'Per': 'Johansson', 'key': 12}
```

- The key is
  - Usually an integer or a string
  - Should (must!) be an immutable object
  - Any key occurs at most once in a dictionary!
- The value may be any object
  - Values may occur many times

## PYTHON CONDITIONS AND IF STATEMENTS

Python supports the usual logical conditions from mathematics:

- Equals: a == b
- Not Equals: a != b
- Less than: a < b
- Less than or equal to: a <= b
- Greater than: a > b
- Greater than or equal to: a >= b



```
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

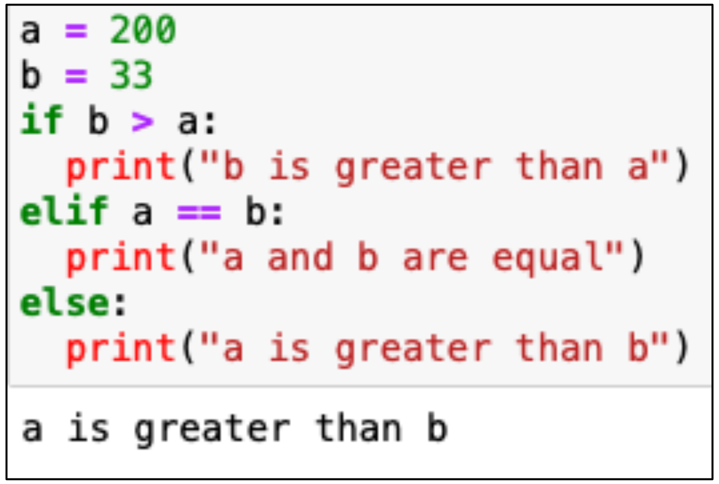
```
b is greater than a
```

An "if statement" is written by using the if keyword.

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

## ELIF AND ELSE

- The elif keyword is pythons' way of saying "if the previous conditions were not true, then try this condition".



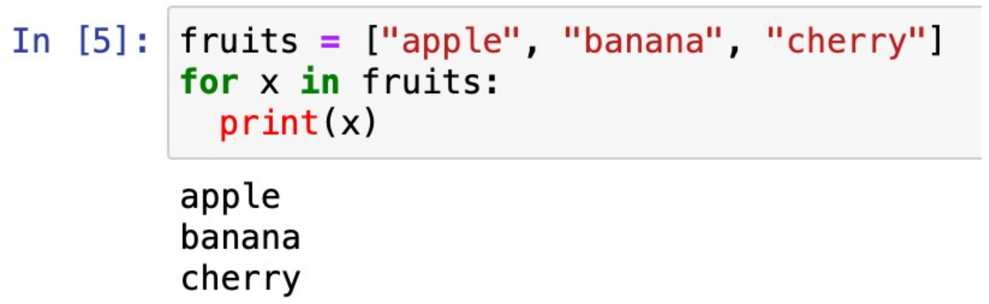
```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

```
a is greater than b
```

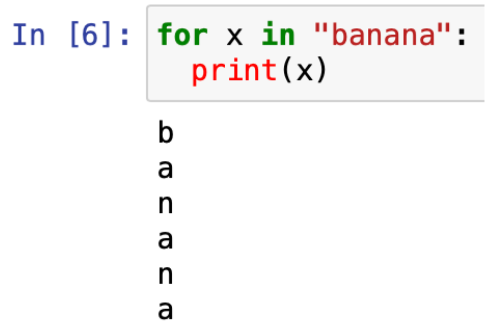
In this example a is equal to b, so the first condition is not true, but the elif condition is true, so we print to screen that "a and b are equal".

# FOR LOOP

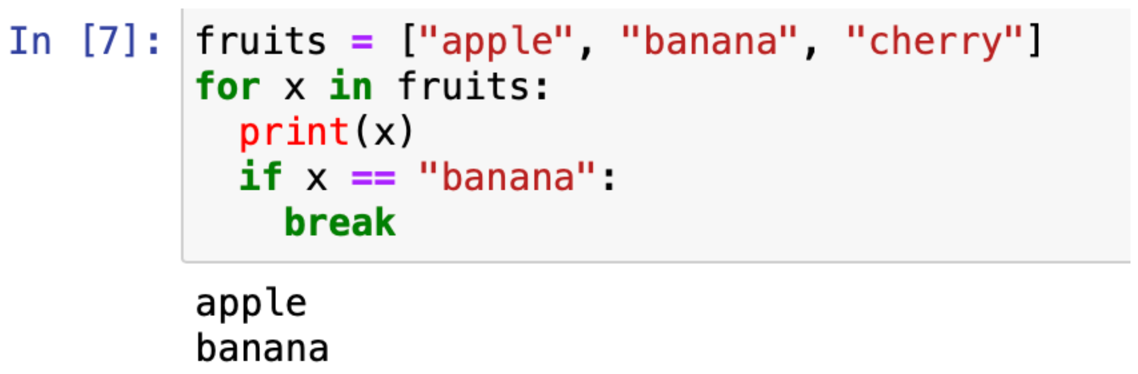
- A for loop is used for iterating over a sequence.
- The for loop does not require an indexing variable to set beforehand.
- With the break statement, we can stop the loop before it has looped through all the items:



```
In [5]: fruits = ["apple", "banana", "cherry"]
        for x in fruits:
          print(x)

        apple
        banana
        cherry
```



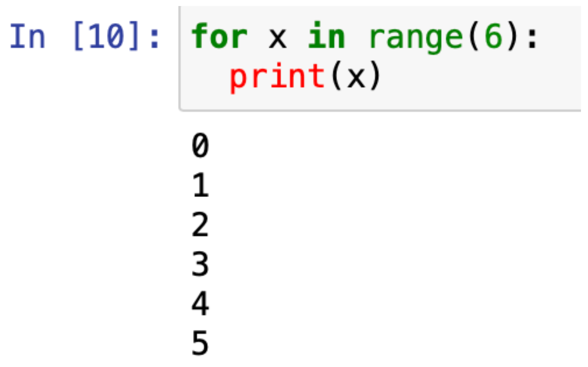
```
In [6]: for x in "banana":
          print(x)

        b
        a
        n
        a
        n
        a
```



```
In [7]: fruits = ["apple", "banana", "cherry"]
        for x in fruits:
          print(x)
          if x == "banana":
            break

        apple
        banana
```

## THE RANGE() FUNCTION

- To loop through a set of code a specified number of times, we can use the range() function,
- The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.
- Using the range() function:
- Note that range(6) is not the values of 0 to 6, but the values 0 to 5.
- The range() function defaults to 0 as a starting value, however it is possible to specify the starting value by adding a parameter: range(2, 6), which means values from 2 to 6 (but not including 6):



```
In [10]: for x in range(6):
           print(x)

         0
         1
         2
         3
         4
         5
```

## FUNCTION

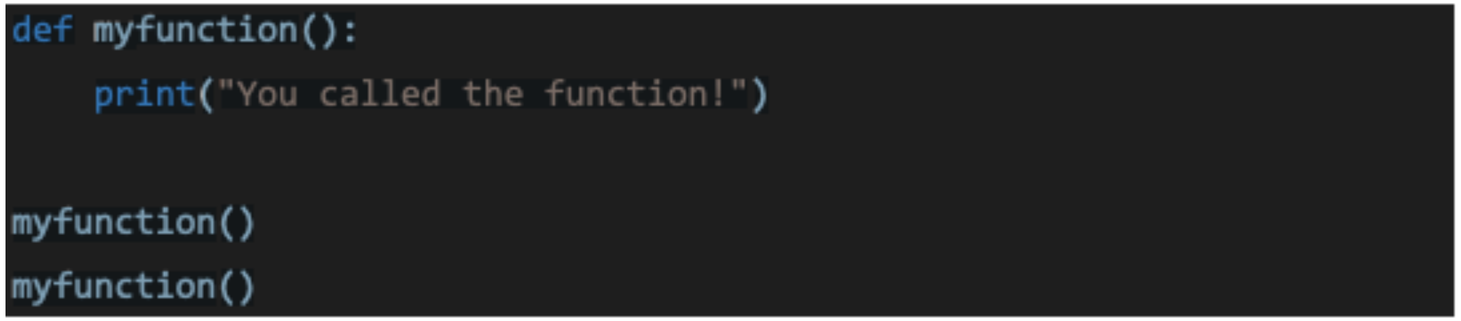
There are two kinds of functions in Python.

-Built-in functions that are provided as part of Python

- print(), input(), type(), float(), int() ...

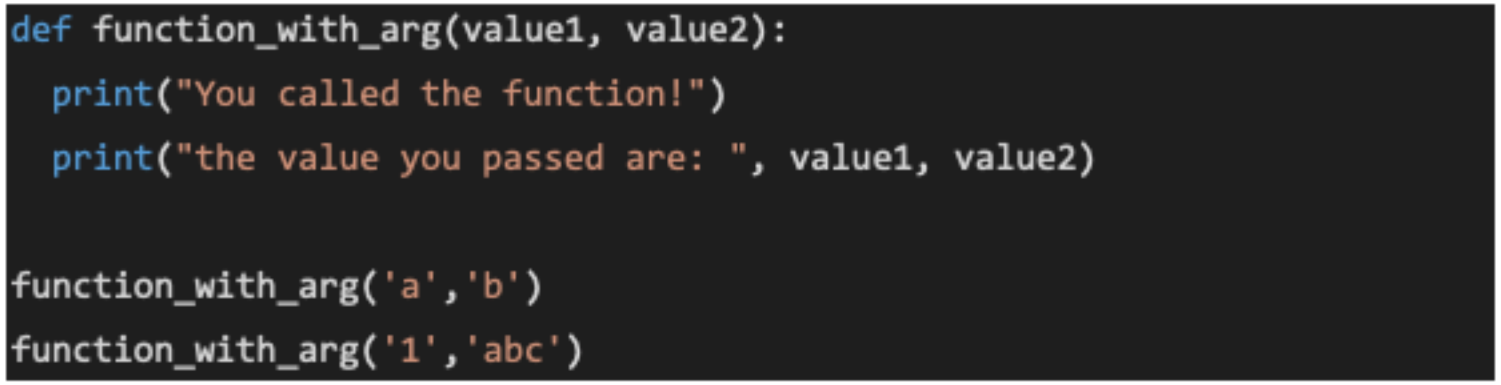
- Functions that we define ourselves and then use

## FUNCTION



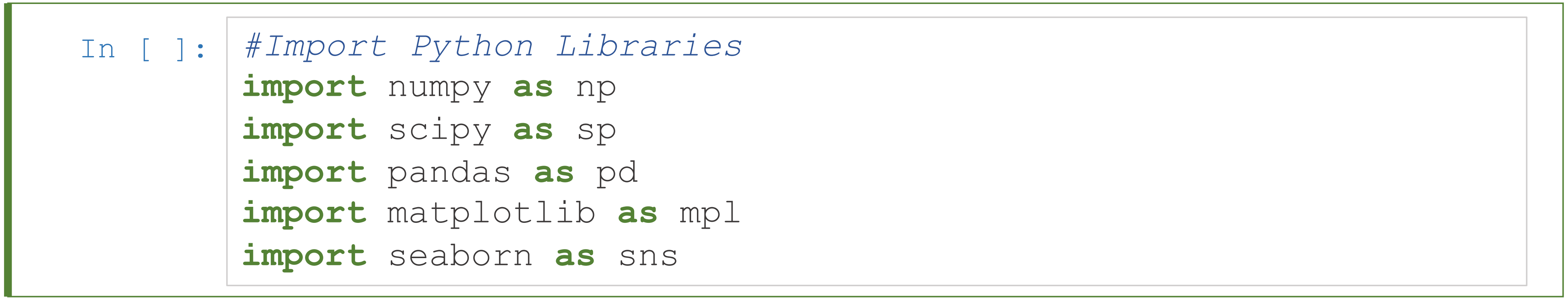
```
def myfunction():
    print("You called the function!")

myfunction()
myfunction()
```

What is the output of this program?



```
def function_with_arg(value1, value2):
  print("You called the function!")
  print("the value you passed are: ", value1, value2)

function_with_arg('a','b')
function_with_arg('1','abc')
```

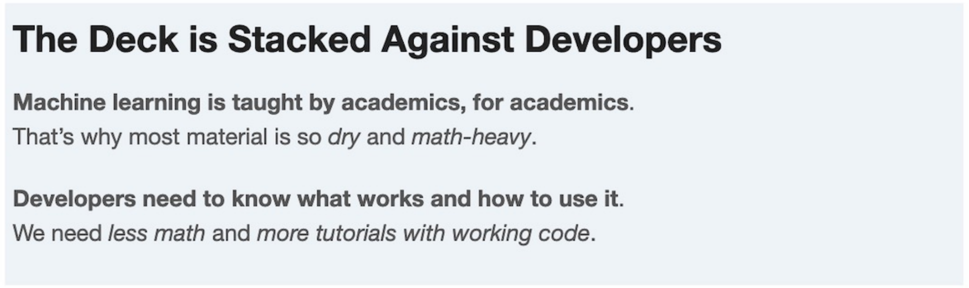
## LOADING PYTHON LIBRARIES



```python
#Import Python Libraries
import numpy as np
import scipy as sp
import pandas as pd
import matplotlib as mpl
import seaborn as sns
```

Press `Shift+Enter` to execute the *jupyter* cell

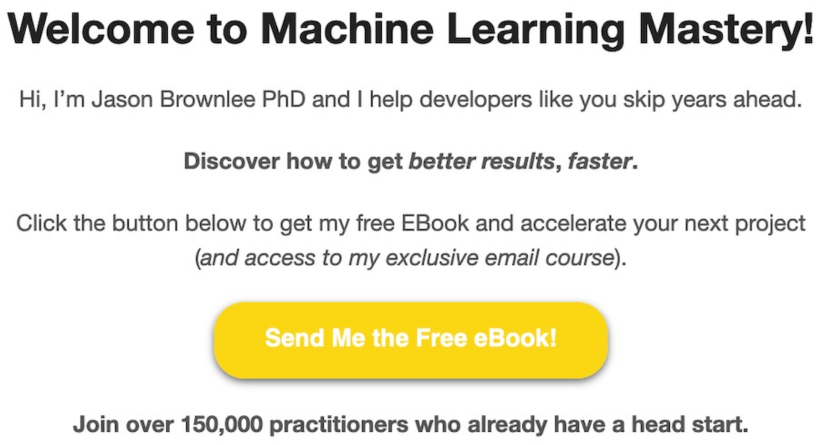
## Helpful Web Site in Python



### The Deck is Stacked Against Developers

**Machine learning is taught by academics, for academics.**
That's why most material is so *dry* and *math-heavy*.

**Developers need to know what works and how to use it.**
We need *less math* and *more tutorials with working code*.



### Welcome to Machine Learning Mastery!

Hi, I'm Jason Brownlee PhD and I help developers like you skip years ahead.

**Discover how to get *better results*, *faster*.**

Click the button below to get my free EBook and accelerate your next project (*and access to my exclusive email course*).

**Join over 150,000 practitioners who already have a head start.**

End