# A GUIDE FOR THE PYTHON NEWBIE

## A GENTLE INTRODUCTION FOR NEWBIES

Juan M. Tirado

Jupyter notebook

Introduction

Python concepts

Exercises

PyPlot

Exercises

# JUPYTER NOTEBOOK

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text.

http://jupyter.org/

Install Jupyter Notebook on your operating system following the instructions you can find at

`https://jupyter.org/install.html`

# INTRODUCTION

# WHAT IS PYTHON?

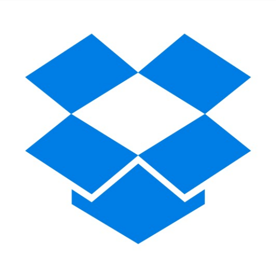
- Programming language developed by Guido Van Rossum in the late 80’s
- It was a Christmas project to develop a scripting language
- Its name comes from ”The Monty Python’s flying circus”
- Python 1.0 will be released in 1994

# WHAT IS PYTHON?

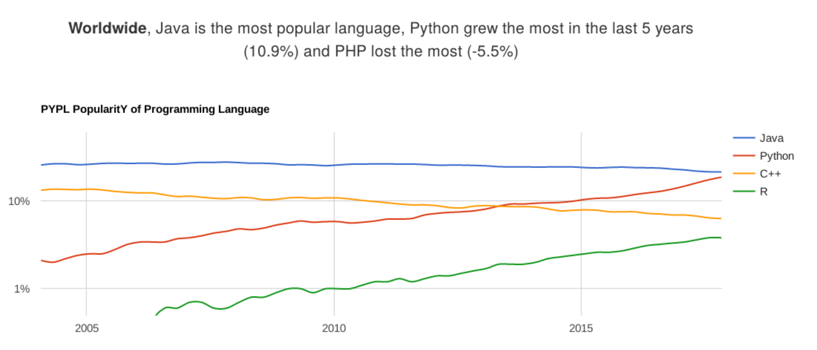
Python has many interesting features

- Open source maintained by the community
  - → Computer programming for everybody
- Designed to be simple
- Perfect for small tasks and prototypes
- Large amount of available libraries
- Easy to connect with C/C++/Java/etc
- Wrapper for more complex libraries

# ¿WHO USES PYTHON?



# PYTHON IS BECOMING THE MOST USED LANGUAGE



**Worldwide**, Java is the most popular language, Python grew the most in the last 5 years (10.9%) and PHP lost the most (-5.5%)

[1]

[1] Taken from `http://pypl.github.io/PYPL.html` on November 2017.

# ¿WHY IS PYTHON SO POPULAR?

- Easy syntax
- Object oriented
- Portable
- Interpreted
- Extensible

The perfect language for the newbie programmer

# MY FIRST HELLO WORLD!

A ”Hello world!” only requires one line:

```
print(’Hello world!’)
```

# PYTHON CONCEPTS

# PYTHON IN YOUR SHELL

- Python is an interpreted language
- We can interact without compilation
- Simply type `python` in your terminal

```
Python 3.4.2 (default, Oct  8 2014, 10:45:20)
[GCC 4.9.1] on linux
Type "help", "copyright", "credits" or "license"
for more information.
>>>
```

# PYTHON IN YOUR SHELL

- There are significative differences between Python 2.7.x and 3.4.x
- Check it using `python –version`
- The following examples use Python 3.4
- To install it in Ubuntu and similar operating systems:

```
sudo apt-get install python3.4
```

# OPERATIONS

```
>>> 1+2
3
>>> 3 * 3
9
>>> 4 / 2
2.0
>>> 1/2 #In Python 2.7 it returns 0
0.5
>>> 3 ** 2
9
>>> 10 % 4
2
```

# VARIABLES

- Python is not strongly-typed
- We do not have to define the type of the variable

```
>>> tax = 21
>>> price = 100
>>> to_pay = price + (price * (tax / 100))
>>> to_pay
121.0
```

- Why is the final result float?

# VARIABLES

- Casting may be required in certain situations

```
>>> int(to_pay) #force integer type
121
>>> 3/2 #by default it returns a float number
1.5
>>> int(3/2) #cast to integer
1
```

# STRINGS

```
>>> ’Hello world!’
’Hello world!’
>>> "Hello world!" #Also double quotes
’Hello world!’
>>> ’"Hello world!"’ #Better single quotes
’"Hello world!"’
>>> print(’Hello world!’)
Hello world!
>>> print("Hello world!")
Hello world!
>>> print(’"Hello world!"’)
"Hello world!"
```

# STRINGS

- We can concatenate strings

```
>>> part1 = ’To be or not to be. ’
>>> part2 = ’That is the question’
>>> question = part1 + part2
>>> question
’To be or not to be. That is the question’
```

- Get the string length

```
>>> len(question)
40
```

# STRINGS

- Analyse fragments

```
>>> question[0:5] #Start from zero
'To be'
>>> question[3:12]
'be or not'
>>> question[5:] #From index 5 till the end
' or not to be. That is the question'
```

# CONTROL STRUCTURES

- If... then... else...

```
>>> myvar = 10
>>> if myvar % 2 == 0:
...     print(’Even’)
... else:
...     print(’Odd’)
...
Even
```

# CONTROL STRUCTURES

- While

```
>>> myvar = 0
>>> while myvar < 5:
...     print(myvar)
...     myvar = myvar + 1
...
0
1
2
3
4
```

# CONTROL STRUCTURES

- For

```
>>> for i in range(5):
...     print(i)
...
0
1
2
3
4
```

# LISTS

- Python lists are extremely powerful

```
>>> primes = [1,2,3,5,7] #Declaration
>>> primes[0] #Selection
1
>>> primes[0:3] #Range selection
[1, 2, 3]
>>> primes[3:]
[5, 7]
>>> primes.append(11)
>>> primes
[1, 2, 3, 5, 7, 11]
>>> len(primes) #List length
6
```

# LISTS

- Lists can contain anything

```
>>> odd = [1,3,5] #Int
>>> even = ['two’,'four’,'six’] #Strings
>>> mezcla = [odd, even] #List of lists
>>> mezcla
[[1, 3, 5], ['two’, 'four’, 'six’]]
```

# DICTIONARIES

- Collection of <key, value> tuples
- Keys are unique

```
>>> mydict = {1: ’uno’, 2: ’dos’, 3: ’tres’}
>>> mydict[2] #Key access
’dos’
>>> mydict.keys() #All the keys
dict_keys([1, 2, 3])
>>> mydict.values() #All the values
dict_values([’uno’, ’dos’, ’tres’])
>>> mydict[4]=’cuatro’ #Set a new value
>>> len(mydict) #Collection length
4
```

# FUNCTIONS

- Reusable piece of code
- It takes from 0 to N arguments
- It returns from 0 to N values

```
>>> def is_there(key,the_dict):
...     if key in the_dict:
...        print("We found "+ str(key))
...     else:
...        print("We did not find "+ str(key))
...
>>> is_there(1,mydict)
We found 1
>>> is_there(10,mydict)
We did not find 10
```

# FUNCTIONS

```
>>> def is_there(key,the_dict):
...     if key in the_dict:
...        return True
...     else:
...        return False
...
>>> is_there(1,mydict)
True
>>> is_there(10,mydict)
False
```

# MODULES

- Modules make available operations not defined in the basic Python core
- Reserved word `import` makes the module accesible inside our code

```
>>> import math
>>> math.pi #Now we can use pi
3.141592653589793
>>> math.log2(64) #Use logarithms
6.0
>>> import random
>>> random.random() #Random numbers
0.3318419757324361
```

# INPUT/OUTPUT

- Read data from a file

```
>>> #Open the file in the given path
>>> with open("/tmp/test.txt") as reader:
...     for line in reader: #Read every line in the
...         print(line)
...
This is a test file

First line

Second line

...
```

# INPUT/OUTPUT

- Other files have a given format
- For example this CSV file:

```
user,name,email
jj16,juan,juan@laempresa.com
eli237,elisabeth,elisabeth@laempresa.com
luisb,luis,luisbarce@laempresa.com
```

# INPUT/OUTPUT

- We can easily access the content

```
>>> with open("/tmp/example.csv") as mycsv:
...     data = csv.reader(mycsv)
...     for row in data:
...         print(row)
...
['user', 'name', 'email']
['jj16', 'juan', 'juan@laempresa.com']
['eli237', 'elisabeth', 'elisabeth@laempresa.com']
['luisb', 'luis', 'luisbarce@laempresa.com']
```

# INPUT/OUTPUT

- The file header can be used to create a dictionary

```
>>> with open("/tmp/example.csv") as mycsv:
...     data = csv.DictReader(mycsv)
...     for row in data:
...        print(row[’user’]+’\t’+
           row[’name’]+’\t’+row[’email’])
...
jj16 juan juan@laempresa.com
eli237 elisabeth elisabeth@laempresa.com
luisb luis luisbarce@laempresa.com
```

# INPUT/OUTPUT

- Writing a file is similar

```
>>> data=[['jj16','juan','juan@laempresa.com'],
... ['eli237','elisa','elisa@laempresa.com'],
... ['luisb','luis','luisbarce@laempresa.com']]
>>> with open("/tmp/output.txt",'w') as writer:
...     for row in data:
...         writer.write('{0},{1},{2}\n'
...         .format(row[0],row[1],row[2]))
...
```

# INPUT/OUTPUT

- Similar solution using the CSV module

```
>>> data=[[’jj16’,’juan’,’juan@laempresa.com’],
... [’eli237’,’elisa’,’elisa@laempresa.com’],
... [’luisb’,’luis’,’luisbarce@laempresa.com’]]
>>> with open("/tmp/output.txt",’w’) as csv:
...     thewriter = csv.writer(csv)
...     for row in data:
...         thewriter.writerow(row)
...
```

# EXERCISES

MusicBrainz is an open music encyclopedia that collects music metadata and makes it available to the public

`https://musicbrainz.org/`

Download the prepared dataset you can find at `https://github.com/juanmanuel-tirado/musicbrainz/raw/master/recordings.csv`

```
BandName,          Country,  Genre,             SongName,                AlbumName,          Duration,  Score,  Year
Frank Zappa,       US,       rock,              "You Are What You Is",   "Teen-Age Wind,"    192000,    98,     1981
The White Stripes, US,       alternative rock,  "Icky Thump",            "Icky Thump",       254000,    100,    2007
...
```

Each line contains information about one song:

- **BandName**
- **Country**
- **Genre**
- **SongName**
- **AlbumName**
- **Duration**
- **Score**
- **Year**

Use Python to find:

- The number of songs in the dataset
- The number of bands
- The number of songs from The Beatles
- Average duration of songs per year
- How many rock songs were released per year since 1965

Some help:

- Load the file into memory using python I/O
- Make you store everything into a convenient data structure
- Use whatever loops or funcions you may need

# PYPLOT

# INTRODUCTION

PyPlot is a collection of Python functions inside the matplotlib library that simplifies the process of creating figures

Relevant examples can be found at the official tutorial page[2]

The following slides only cover basic examples

[2] https://matplotlib.org/users/pyplot_tutorial.html

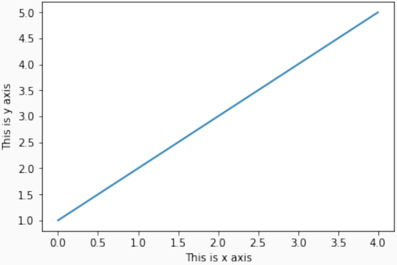
# INTRODUCTION

⚠ Before you start, make sure pyplot is installed in your environment

```
pip install pyplot
```

# INTRODUCTION

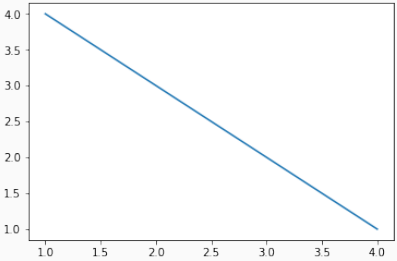
```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4,5])
plt.xlabel("This is x axis")
plt.ylabel("This is y axis")
plt.show()
```



# PLOTTING SERIES

*plot()* is a basic function that receives arrays of coordinates $x, y$ to be displayed

```
x=[1,2,3,4]
y=[4,3,2,1]
plt.plot(x,y)
plt.show()
```



# PLOTTING SERIES

By default points are linked using blue lines. The line style or marker can be specified using strings[3]:

| character | description |
|---|---|
| '-' | solid line |
| '- -' | dashed line |
| '-.' | dash-dot line |
| ':' | dotted line |
| 'o' | circle marker |
| 'v' | triangle down marker |

[3] https://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.plot

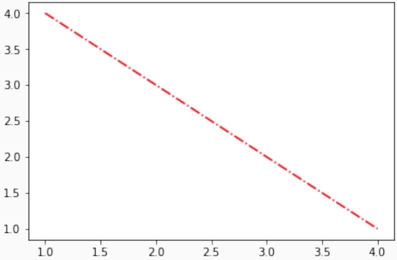
# PLOTTING SERIES

Additionally, we can specify the color to be used:

| character | color |
|---|---|
| ’b’ | blue |
| ’g’ | green |
| ’r’ | red |
| ’c’ | cyan |
| ’m’ | magenta |
| ’y’ | yellow |
| ’k’ | black |
| ’w’ | white |

# PLOTTING SERIES

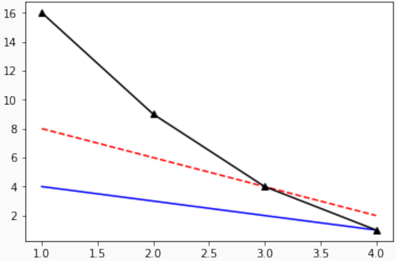
```
x=[1,2,3,4]
y=[4,3,2,1]
plt.plot(x,y,’r−.’)
plt.show()
```



# PLOTTING SERIES

Several data series can be displayed into the same plot

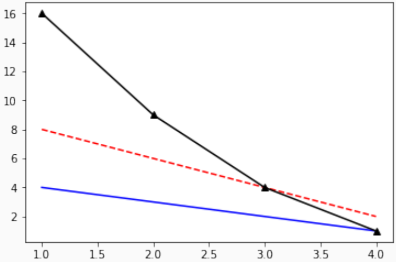
```
import numpy
x=[1,2,3,4]
y=numpy.array([4,3,2,1])
plt.plot(x,y,'b-',x,y*2,'r--',x,y**2,'k^-')
plt.show()
```



# PLOTTING SERIES

We can get a similar result calling plot() several times

```
x=[1,2,3,4]
y=numpy.array([4,3,2,1])
plt.plot(x,y,’b-’)
plt.plot(x,y*2,’r--’)
plt.plot(x,y**2,’k^-’)
plt.show()
```



# PLOTTING SERIES

Setting the label of every series, we can easily define a legend for the plot

```
import numpy
x=[1,2,3,4]
y=numpy.array([4,3,2,1])
plt.plot(x,y,'b-',label='series_1')
plt.plot(x,y*2,'r--',label='series_2')
plt.plot(x,y**2,'k^-',label='series_3')
plt.legend()
plt.show()
```

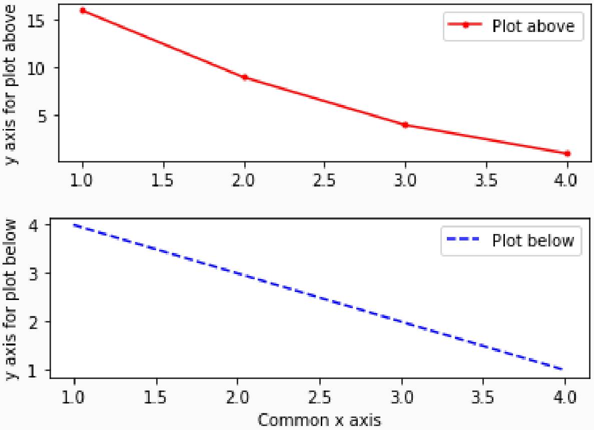
# PLOTTING SEVERAL FIGURES

We split a plot into subplots:

```
plt.figure(1)
plt.subplot(211)
plt.plot(x,y**2,’r.-’,label=’Plot above’)
plt.ylabel("y axis for plot above")
plt.legend()

plt.figure(2)
plt.subplot(212)
plt.plot(x,y,’b--’,label=’Plot below’)
plt.ylabel("y axis for plot below")
plt.legend()
plt.xlabel("Common x axis")
plt.show()
```

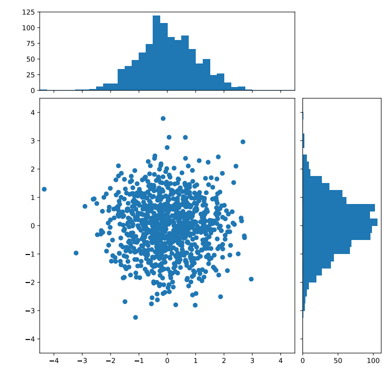
# PLOTTING SEVERAL FIGURES

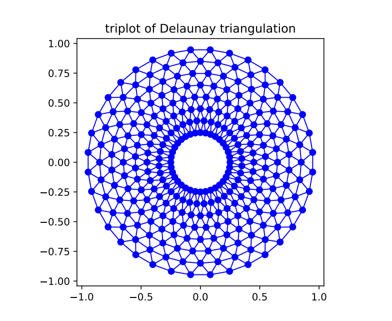


# EXAMPLES

Sometimes the best way to find how to plot our data is looking for existing examples

There is a complete gallery of examples available at `http://matplotlib.org/gallery.html`





# EXERCISES

- Using the musicbrainz dataset, plot how many rock songs were released every year since 1965
- Compare the number of rock songs release since 1965 with the number of pop songs in the same period