# Python
# Matplotlib

## MACbioIDi – February – March 2018

# Introduction

- **Matplotlib** is a Python 2D plotting library
  - Its origins was emulating the MATLAB graphics commands
  - It makes heavy use of NumPy
- Objective:
  - Create simple plots with just a few commands
    - If you want to see histogram of your data, you shouldn't need to instantiate object, call methods...

## Introduction

- The **Matplotlib** is divided into three parts:
  - ***Pylab interface***
    - Allow the user to create plots with code quite similar to MATLAB
  - **Matplotlib frontend**
    - Set of classes that allow you to create figures, text, lines…
  - **Backend**
    - Renderers, transformation, window...

# Firsts steps

- Three ways to use the library:
  - Using **PyLab** module:

```
from pylab import *

...
```

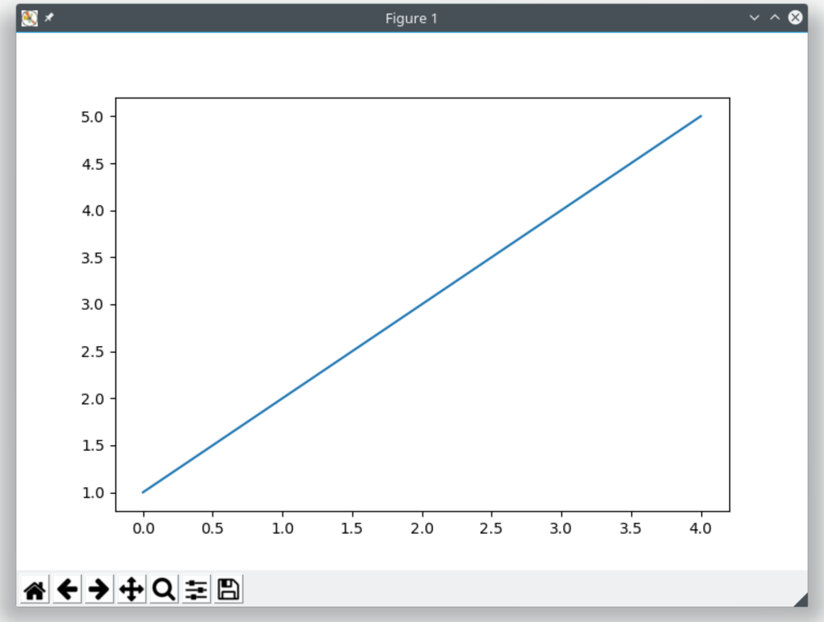
  - Using **PyPlot** module:

```
import matplotlib.pyplot as plt

...
```

  - Using **OOB interface**:
    - It allow to get a more control of the code but… it is **most difficult way**

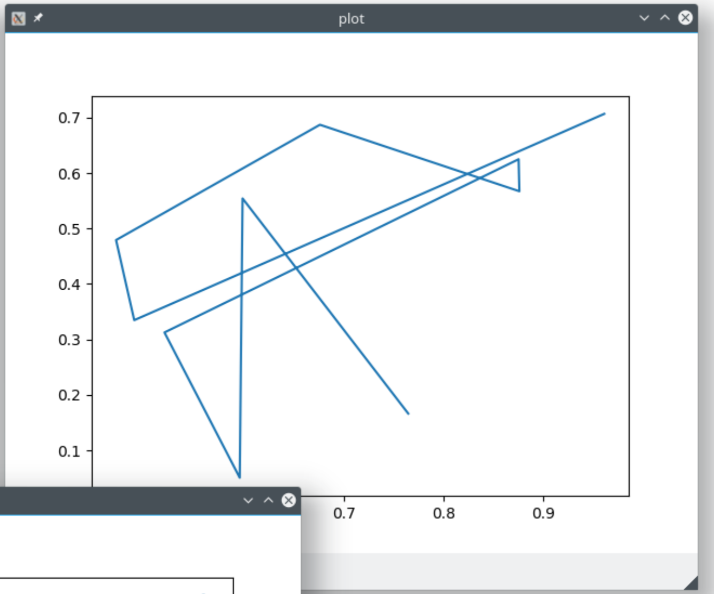
# First steps

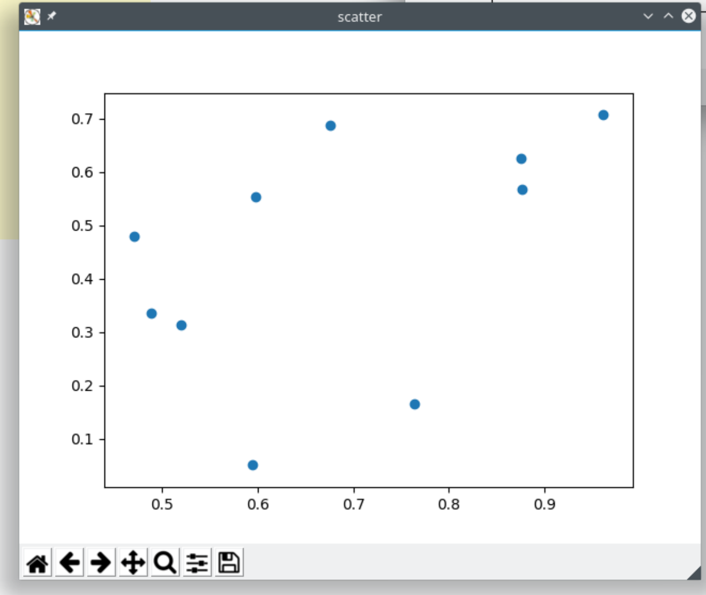
- There are two main zones where you will ‘draw’:
  - **Figure**: Contains all the plot elements.
  - **Axis:** Contains most of the figure elements and sets the coordinate system.



# First Steps

```python
import matplotlib.pyplot as plt
import numpy as np

plt.figure('scatter')
plt.figure('plot')

a = np.random.rand(10)
b = np.random.rand(10)

plt.figure('scatter')
plt.scatter(a,b)

plt.figure('plot')
plt.plot(a,b)
plt.show()
```





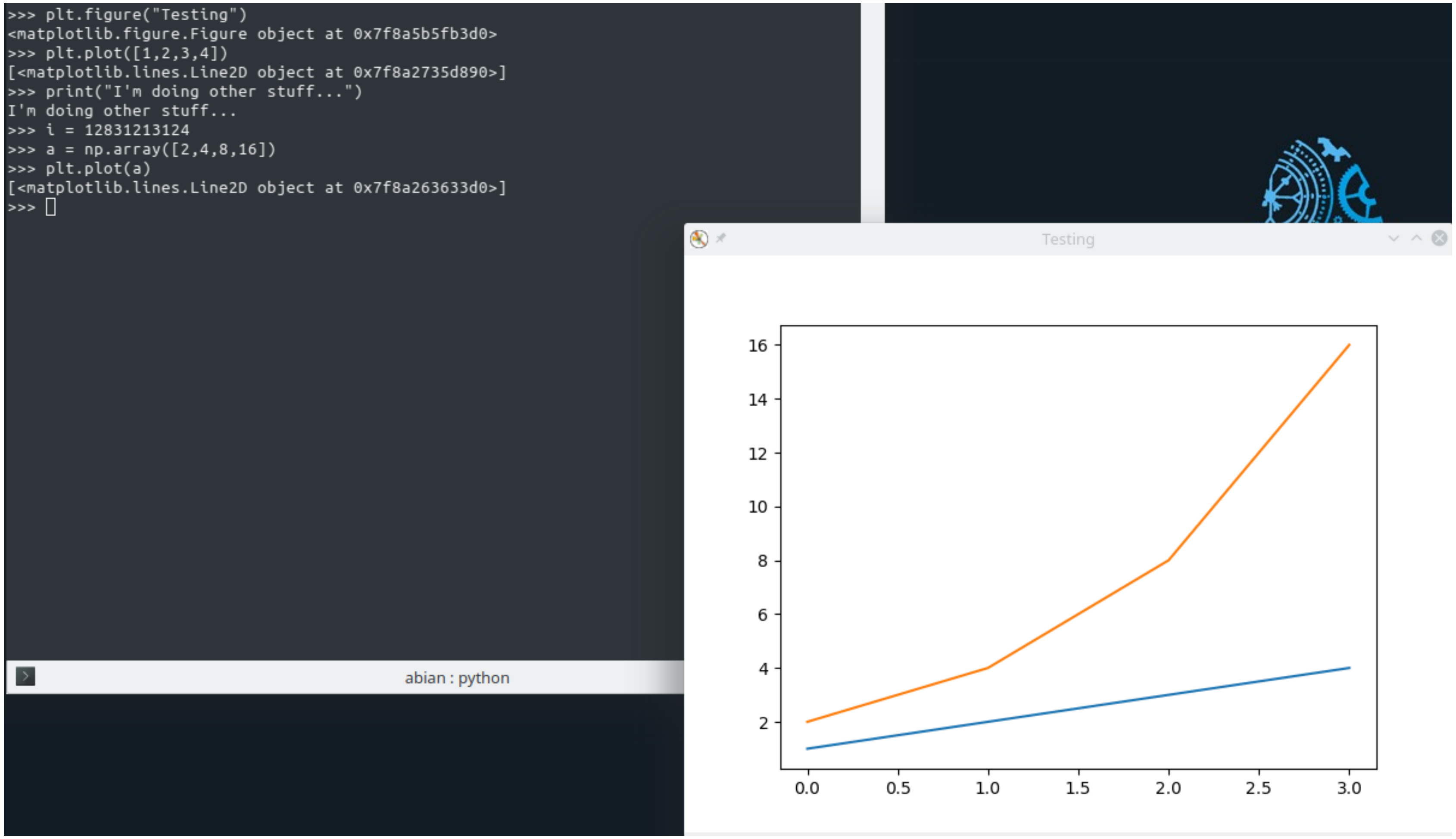
## Interactive Mode

- Allow you to add data in any moment

- By default, plots are non-interactive
  - You can switch that property using **plt.ion()** and **plt.ioff()**
  - **plt.isInteractive()** is used in order to check if iteractive mode is actived

- In interactive mode, you **don’t need to call plt.show()**

# Interactive Mode

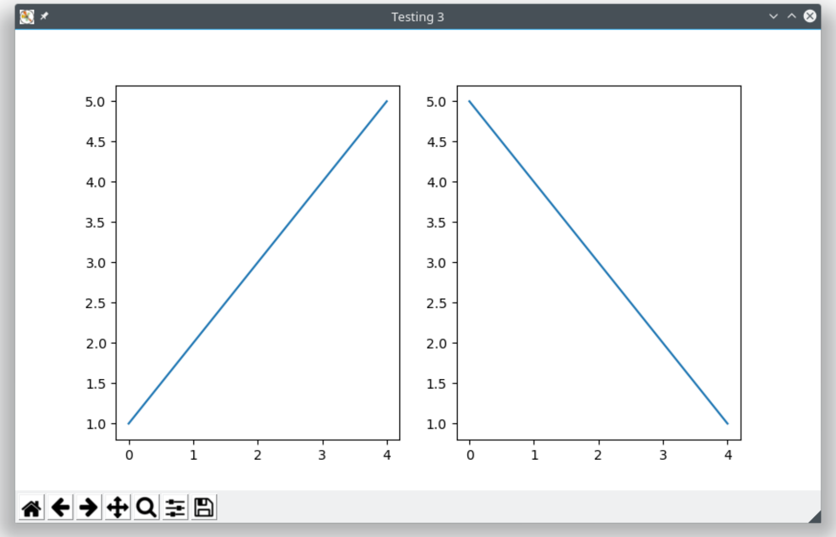


```
>>> plt.figure("Testing")
<matplotlib.figure.Figure object at 0x7f8a5b5fb3d0>
>>> plt.plot([1,2,3,4])
[<matplotlib.lines.Line2D object at 0x7f8a2735d890>]
>>> print("I'm doing other stuff...")
I'm doing other stuff...
>>> i = 12831213124
>>> a = np.array([2,4,8,16])
>>> plt.plot(a)
[<matplotlib.lines.Line2D object at 0x7f8a263633d0>]
>>> 
```

## Subplot

- With plt.subplot() you can configurate the figure in order to use various axis
  - Indicating the Layout

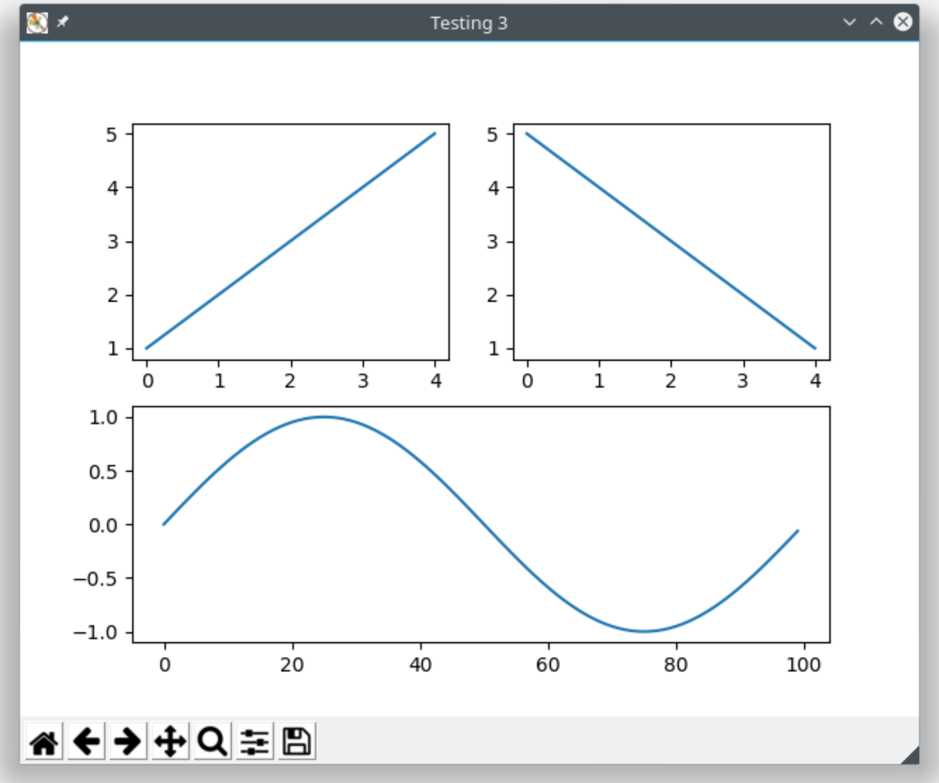
```
plt.figure("Testing 3")
plt.subplot(1,2,1) #1 row, 2 columns, index 1
plt.plot([1,2,3,4,5])
plt.subplot(1,2,2) #1 row, 2 columns, index 2
plt.plot([5,4,3,2,1])
plt.show()
```



# Subplot Exercise

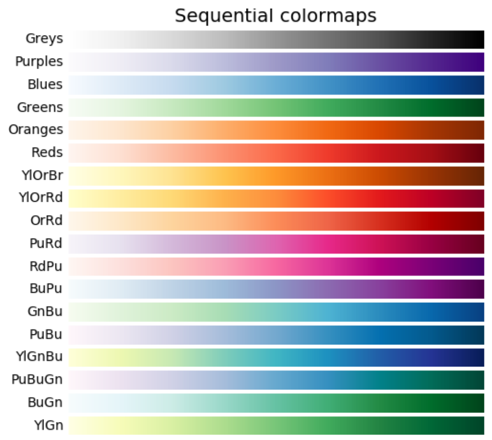
- Try to replicate the layout of the following image:

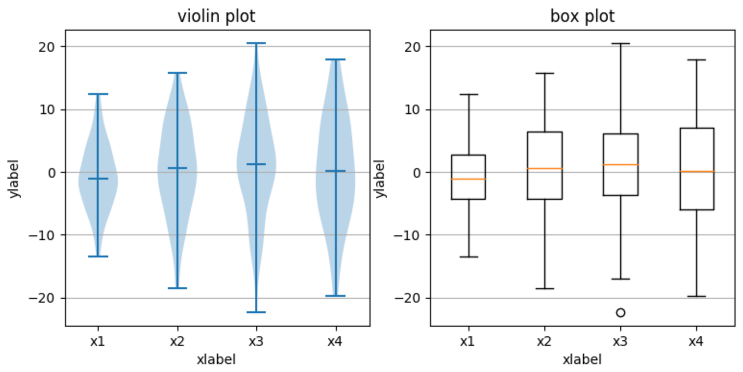
```
plt.figure("Testing 3")
[Subplot layout]
plt.plot([1,2,3,4,5])
[Subplot layout]
plt.plot([5,4,3,2,1])
[Subplot layout]
t = np.arange(0, 1, 0.01)
plt.plot(np.sin(2*np.pi*t))
plt.show()
```

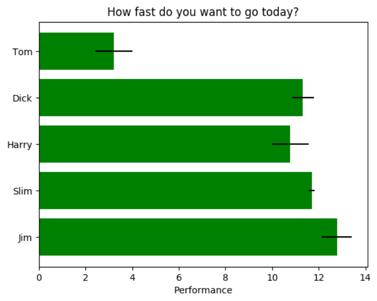


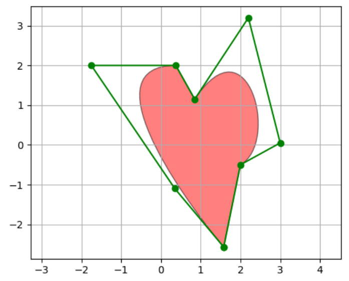
# Type of plots

- Matplotlib can apply multiple type of plots:
  - Lines, bars and markers
  - Shapes and collections
  - Statistical plots
  - Images, contours and fields
  - Color...







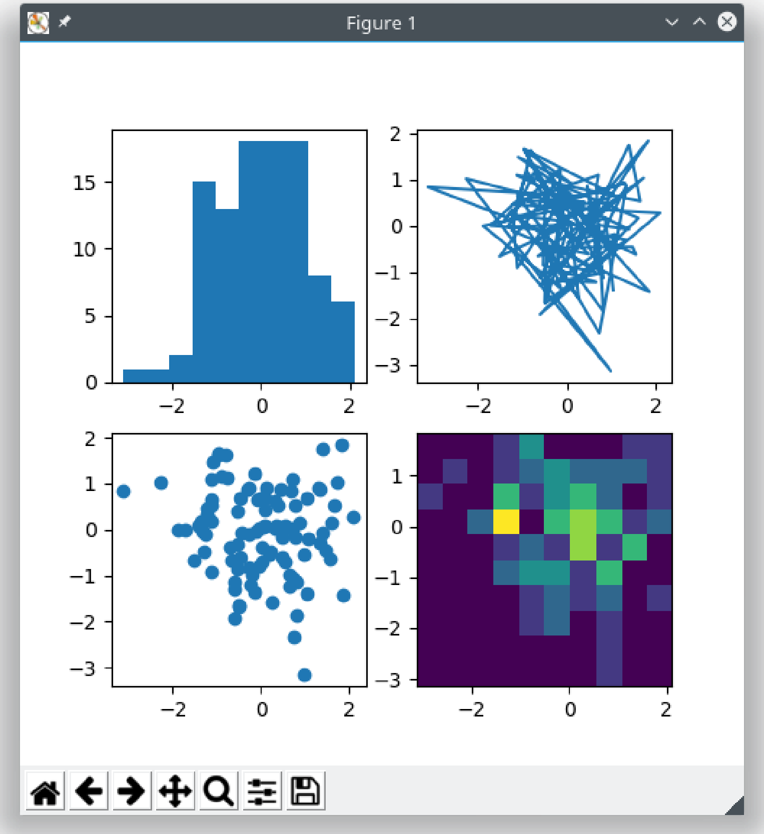


Reference: https://matplotlib.org/gallery.html

# Type of plots

- … and you can combinate in the same figure

```
np.random.seed(19680801)
data = np.random.randn(2, 100)
fig, axs = plt.subplots(2, 2, figsize=(5, 5))
axs[0, 0].hist(data[0])
axs[1, 0].scatter(data[0], data[1])
axs[0, 1].plot(data[0], data[1])
axs[1, 1].hist2d(data[0], data[1])
plt.show()
```



# Legend

- Matplot allows you to assign labels to plots in differents ways:

```
...
plt.plot([1,2,3,4,5], label='Line 1')
plt.legend()
...
Line2 = plt.plot([2,4,8,16,32])
plt.legend(Line2, [“Line2”])
```

- plt.legend() contains multiple parameters:
  - **loc**
  - **bbox_to_anchor**
  - **ncol**
  - **fontsize**

# Miscellaneus

- Assign labels to axis:

```
...
plt.xlabel("X label example")
plt.ylabel("Y label example")
...
```

- Set a title to the plot:

```
...
plt.title("Title, but…")
...
```

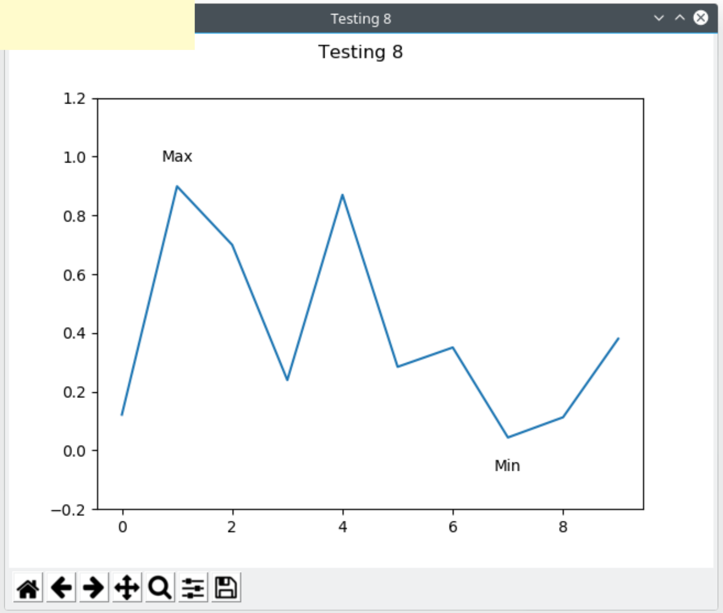
- Set a “real title” to the plot:

```
...
plt.suptitle("Real title")
plt.title("It is a subtitle")
...
```

# Miscellaneus

- Write down a text in the plot:

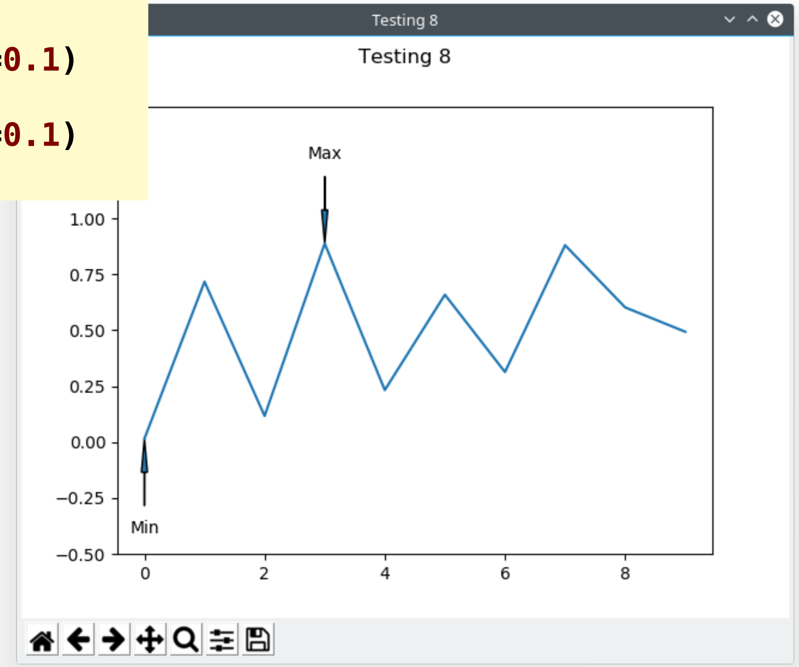
```
y = np.random.rand(10)
plt.plot(y)
plt.ylim(-0.2, 1.2) #define values range in Y axis
plt.text(np.argmin(y), np.min(y) - 0.1, u'Min', fontsize = 10,
         horizontalalignment='center', verticalalignment='center')
plt.text(np.argmax(y), np.max(y) + 0.1, u'Max', fontsize = 10,
          horizontalalignment='center', verticalalignment='center')
plt.show()
```



# Miscellaneus

- To include a arrow to relate the text and plot:

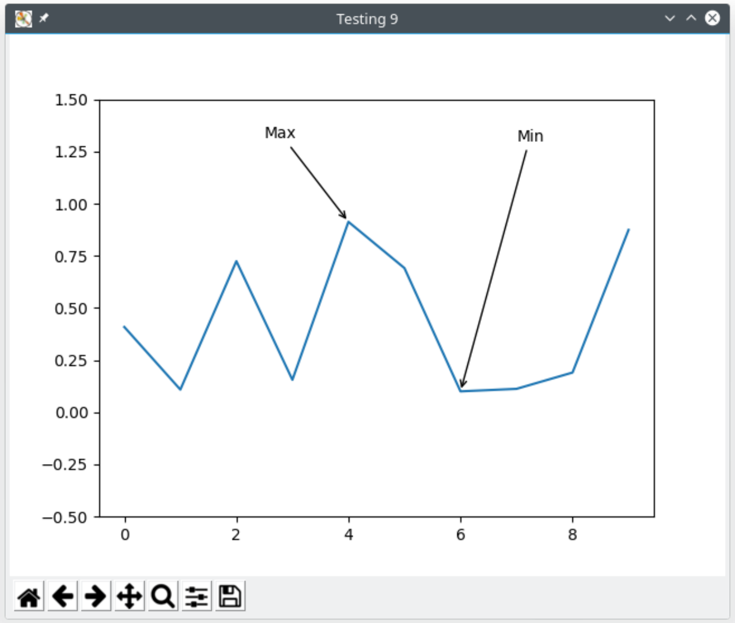
```
y = np.random.rand(10)
plt.plot(y)
plt.ylim(-0.5, 1.5)
plt.text(np.argmin(y), np.min(y) - 0.4, u'Min', fontsize = 10,
      horizontalalignment='center', verticalalignment='center')
plt.text(np.argmax(y), np.max(y) + 0.4, u'Max', fontsize = 10,
     horizontalalignment='center', verticalalignment='center')
plt.arrow(np.argmin(y), np.min(y) - 0.3, 0, 0.3,
     length_includes_head="True", shape = "full", head_width=0.1)
plt.arrow(np.argmax(y), np.max(y) + 0.3, 0, -0.3,
     length_includes_head="True", shape = "full", head_width=0.1)
plt.show()
```



# Miscellaneus

- With plt.annotate, we can replicate what we have done with plt.txt and plt.arrow:

```
...
plt.annotate(u'Max', xy = (np.argmax(y), np.max(y)),
     xycoords = 'data',
     xytext = (np.argmax(y) - 1.5, np.max(y) + 0.4),
      textcoords = 'data',
      arrowprops = dict(arrowstyle = "->"))
plt.annotate(u'Min', xy = (np.argmin(y), np.min(y)),
      xycoords = 'data',
      xytext = (np.argmin(y) + 1, np.min(y) + 1.2),
      textcoords = 'data',
      arrowprops = dict(arrowstyle = "→"))
...
```



# Exercise

- Using Titanic disaster dataset:
  - Plotting a histogram with the number of males and females survivor
- Complete the unfinished code:
  - Remember **NumPy Indexing**
  - Generates the histogram generating **two bar plots** (take a look to attribute width… and the x coordinate in the scale)
  - Assign a label to both plots and show a legend

# Exercise

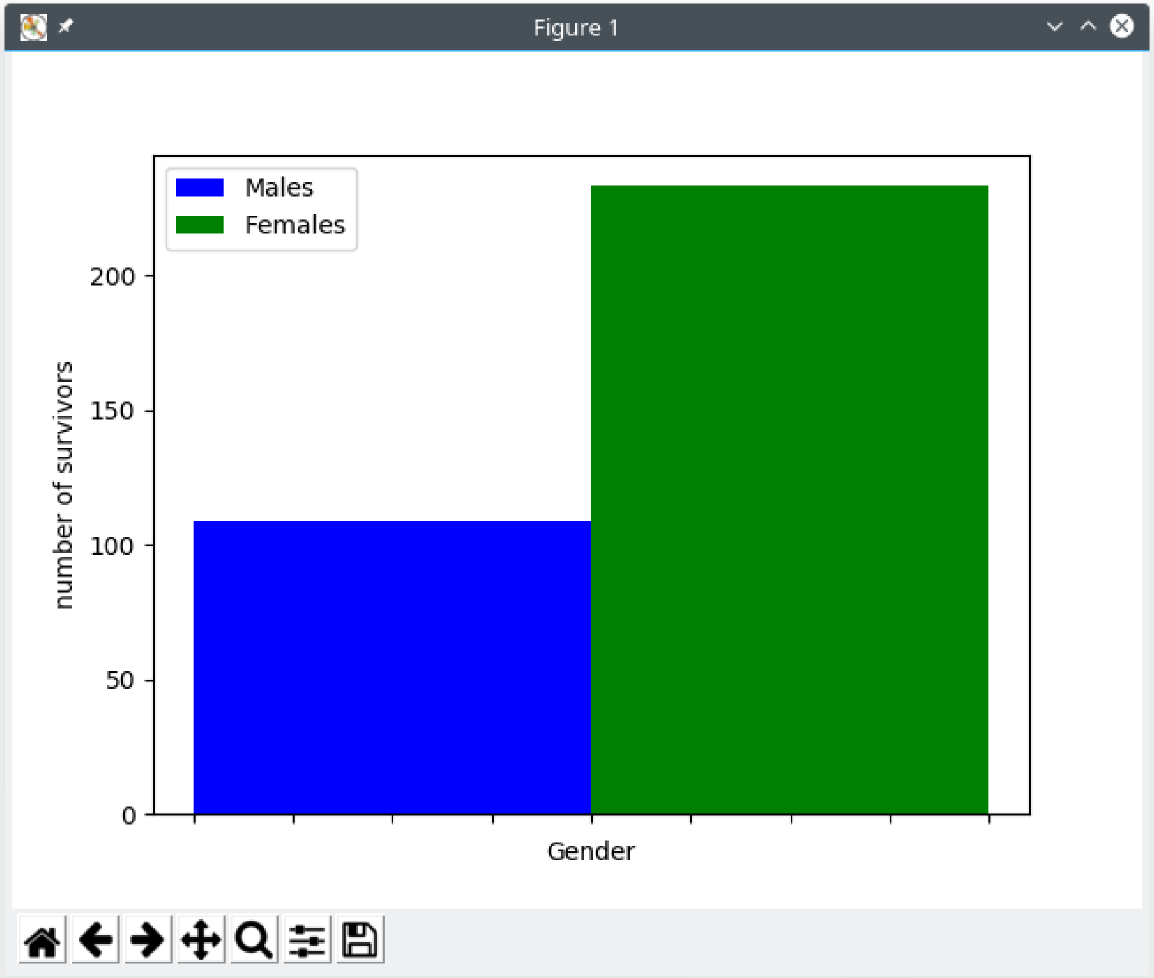


Figure 1
Males
Females
number of survivors
0
50
100
150
200
Gender

# Python Matplotlib

MACbioIDi – February – March 2018