Table of Contents

This is a Jupyter notebook!

The **Jupyter Notebook** (https://jupyter.org/) is an open-source web application that allows to create and share documents that can integrate code, text, images, multimedia, data, mathematics:

- live, pretty-printed **python code**
- **visualizations** (e.g, images)
- **formatted text**
- **equations.**

A Jupyter Notebook is organized in Input **cells**, where code, text, images, or equations can be written.

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

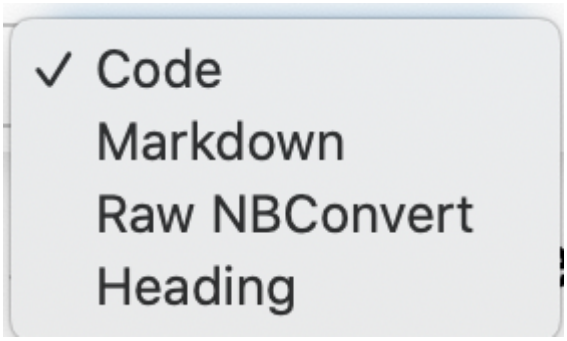
When a cell is executed with `Run` or `Shift-Enter` , its content is ***interpreted*** on the fly and the output is displayed (`Out` part of the cell, below it)

Two alternatives to run / execute a cell:

→ Click on the run button in the **toolbar**

→ Type the `Shift` and `Enter` keys together

Each cell can be of one of the following four types, that can be selected from the menu in the toolbar:



- **Code**
- **Markdown**
- Raw NBConvert
- Heading

We are only interested in the first two types.

For instance this cell is of type `Markdown`, which is the type to use to produce **formatted text.**

Instead, if `Code` is selected, the cell is meant to include **Pyhton code** that can be executed locally to the cell.

By default a new cell has type `Code`.

You can **change the type to Markdown** using the drop-down menu or by typing `Esc m`.

If a cell is a Markdown one, **double click** on the cell to access its *raw* (i.e., unformatted) content

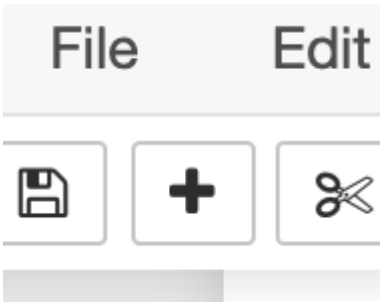
---

This is a **markdown** cell and this is *italic* text

This is a newline

You can add an **empty cell** *below* the current cell in two ways:

→ By using the `+` button in the toolbar



→ By pressing the key `Esc` and then the key `b`

→ By pressing the key `Esc` and then the key `a` the empty cell is added *before* the current one

---

Remove a cell by using

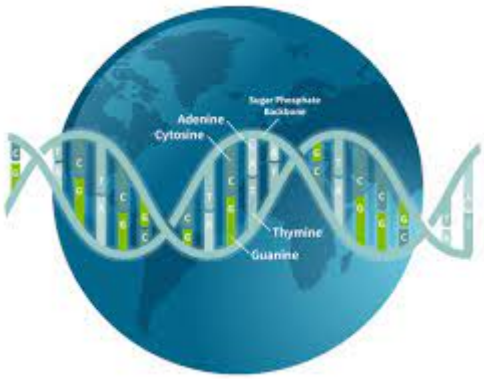
Or using the `Esc x` shortcut.

This is a blue text

To change color you need to use `HTML` tags!

In [ ]:

You can copy an image from the Internet or from your computer and **paste it** (e.g., using `Command-V` on a Mac) in a Markdown cell. The code below will be generated automatically



Let's see what we can do!

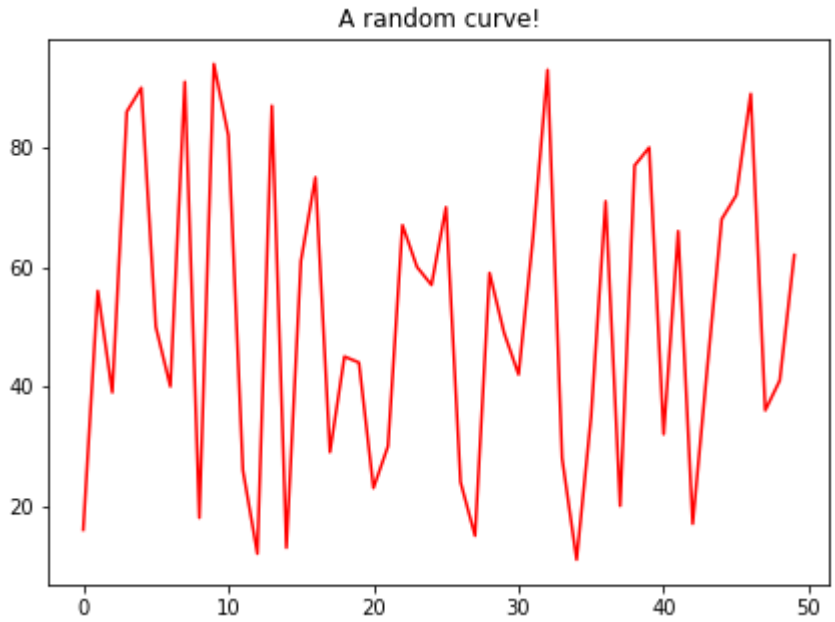
→ Live, pretty-printed **python code**

In [6]:
```
x = 0
for i in range(6):
    x += 1
    print('Helloooo from Jupyter!', x)
```

```
Helloooo from Jupyter! 1
Helloooo from Jupyter! 2
Helloooo from Jupyter! 3
Helloooo from Jupyter! 4
Helloooo from Jupyter! 5
Helloooo from Jupyter! 6
```

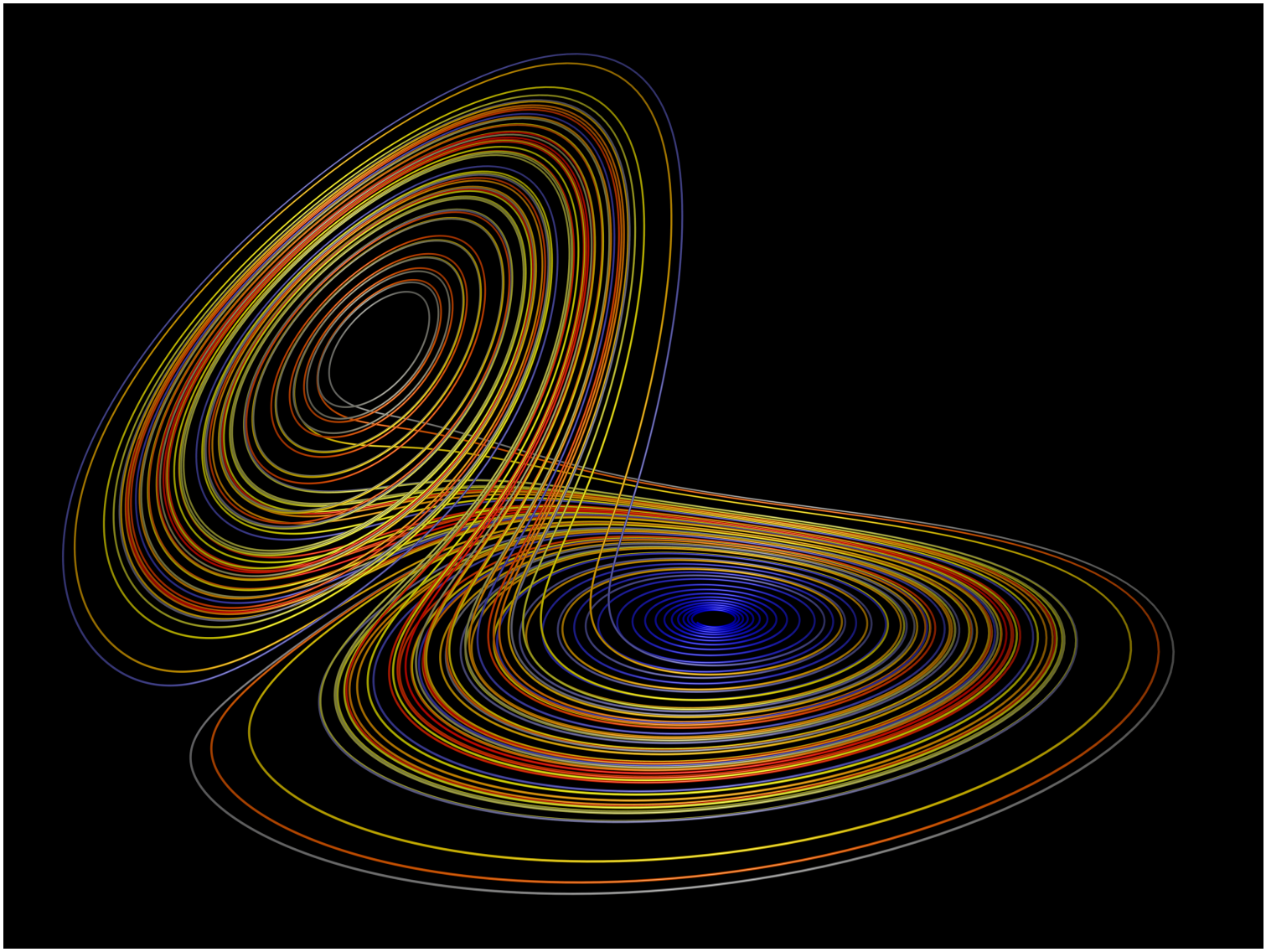
Display plots made with python

In [7]:
```
import matplotlib.pyplot as plt
import random

y = random.sample(range(10, 100), 50)

plt.figure(figsize = (7,5))
plt.title('A random curve!')

plt.plot(y, color='r')

plt.show()
```



→ **Visualizations** (e.g, images)

A Lorenz's strange attractor



To **resize** an image:

```
<div>
  <img src="attachment:image.png" width="450" height="276">
</div>
```

→ **Formatted text including mathematical symbols** using Markdown + HTML + LaTeX

This is a (complete) example of use of Markdown:

Lorenz's (Lorenz, 1963) (https://en.wikipedia.org/wiki/Lorenz_system) equations were the first example of a dynamical system showing

deterministic chaos.

The equations are the following:

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = -\beta z + xy$$

The following choices for the values of the parameters σ, ρ, β results in the strange attractor dynamics shown in the figure below:

- σ = 10
- ρ = 28 (Rayleigh number)
- β = 8/3

Simple math expression can be made using LaTeX syntax:

$$x^2 + y^2$$

→ **Shell commands** to interact with OS resources

In [8]:
```
%ls
```

```
IO_Files-Copy1.ipynb      L14.py*                TurtleCode/
Jupyter_notebooks.ipynb   L15.py*                Untitled.ipynb
L0.py*                    L17.py*                data.txt
L1.py*                    L1_1.py*               hw05.py*
L10.py*                   L8.py*                 scratch.py*
L11.py*                   L9.py*                 untitled3.py*
L12.py*                   Lab-1.py               untitled4.py*
```

Documentation

Extensive and well accessible documentation to start with Jupyter notebooks:

https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/index.html (https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/index.html)

To know more:

https://jupyter-notebook.readthedocs.io/en/stable/index.html (https://jupyter-notebook.readthedocs.io/en/stable/index.html)

https://jupyter-notebook.readthedocs.io/en/stable/notebook.html (https://jupyter-notebook.readthedocs.io/en/stable/notebook.html)

Markdown for easily writing formatted text. It can be combined with HTML tags and LaTeX for mathematical expressions!

→ A nice tutorial: https://www.datacamp.com/community/tutorials/markdown-in-jupyter-notebook (https://www.datacamp.com/community/tutorials/markdown-in-jupyter-notebook)

→ Useful cheatsheet: https://wordpress.com/support/markdown-quick-reference/ (https://wordpress.com/support/markdown-quick-reference/)

Basic formatting with Markdow

Headers

```
# Header 1
## Header 2
### Header 3
#### Header 4 ####
##### Header 5 #####
###### Header 6 ######

Closing hash marks are optional on all levels
```

Header 1

Header 2

Header 3

Header 4

Header 5

Header 6

This is header 1

This is normal text (not a header!)

This header 2

I'm smaller, but still boldface!

I like to use a line separator between cells, using `***` , to improve readability

---

Emphasis

```
*Emphasize* _emphasize_
**Strong** __Strong__
```

Emphasize
Strong

In this text I want to emphasize the word *python*, displaying it in *italic*, and I want really put the attention on **computer science**, displaying it using a **boldface** character.

Making **lists of items** is very useful!

Bullet Lists

```
* Item
* Item
- Item
- Item
```

- Item
- Item
- Item
- Item

The animals that I like the most are:

- cats
- wombats
- rabbits
- dogs
- ...

Sometimes I need have **lists with numbers:**

1. The is my first priority
2. This is my second priority
3. This is my third priority
4. whatever ...

Nice, I don't need to keep track of the numbers!

... can I add some more space between lines / items? Yes, but need to use a newline and `HTML` tags: `<p>`

1. The is my first priority with more space and larger font
2. This is my second priority with more space and larger font
3. This is my third priority with more space and larger font

Writing code in the text is important to explain how things have been done, or which commands need to be execute:

Code	`` `This is code` ``	`This is code`
Code block	~~~~ This is a piece of code in a block ~~~~ ``` This too ```	1 This is a 2 piece of code 3 in a block 1 This too

This is one line of code:

```
for i in range(10):


   def my_function(a,b):
       while True:
           print('Looping forever!')
```

Add a **link** to some document / web page:

```
A [link](http://example.com "Title").
```

This is a link to Google (http://www.google.com)

Be careful when using python coding in notebooks!

All the instructions that have been executed before *matter*: all variables and functions defined before executing a cell, do exist!

The order of the cells in the notebook doesn't (necessarily) matter, what matters is the order of executing them, that defines the *history* of the notebook's program.

In case, you can clear up all the history (100%) by going in the Header, and click:

`Kernel -> Restart and clear output.`

In [1]:
```
x = 4
```

In [2]:
```
L = [1,3,4,0,5]
```

In [3]:
```
print( list_sum(LL) )
```

```
---------------------------------------------------------------------
-------
NameError                                 Traceback (most recent cal
l last)
<ipython-input-3-cdb4376e2cf9> in <module>
----> 1 print( list_sum(LL) )

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

In [ ]:
```
LL = [0, 6, 7]
```

In [ ]:
```
print('Hello!')
```

In [ ]:
```
print(LL)
```

In [10]:
```
def list_sum(L):
    if x > 5:
        return sum(L)
    else:
        return -1
```

In [11]:
```
print( list_sum(L) )
```

```
13
```