

## **Introduction to Coding in Python**

### **Overview of the workshop**

The goal of this workshop is to introduce the concepts of coding in Python. The python environment is first introduced with the different options available. Then, the python language basics are explained which are mainly the variables, arrays, operators, if/else statements, while loops, for loops, and functions. Finally, the essential libraries in Python for machine learning are introduced which are the NumPy, SciPy, Pandas, scikit-learn, and matplotlib.

### **Duration**

2.5 hours

### **Name of workshop facilitator**

Nour Droubi

Alaa Maarouf

### **Learning Objectives**

- 1- Learn what is computer programming and what are programming languages.
- 2- Get introduced to the python environment.
- 3- Learn the basics of coding with Python and be able to use concepts such as the arrays, if/else statements, for loops, while loops, and functions.
- 4- Get introduced to essential libraries for machine learning in Python.

### **Underlying theory**

#### **1- Computer Programming:**

Computer programming is the process of designing and building a computer program that executes a specific task through a set of commands that the computer understands. Such programs include creating video games, building websites, creating animated videos, and much more. People who design and build computer programs are called programmers. Learning the process of designing and building computer programs, allows the programmer to gain problem solving skills and to be able to think more clearly and accurately.

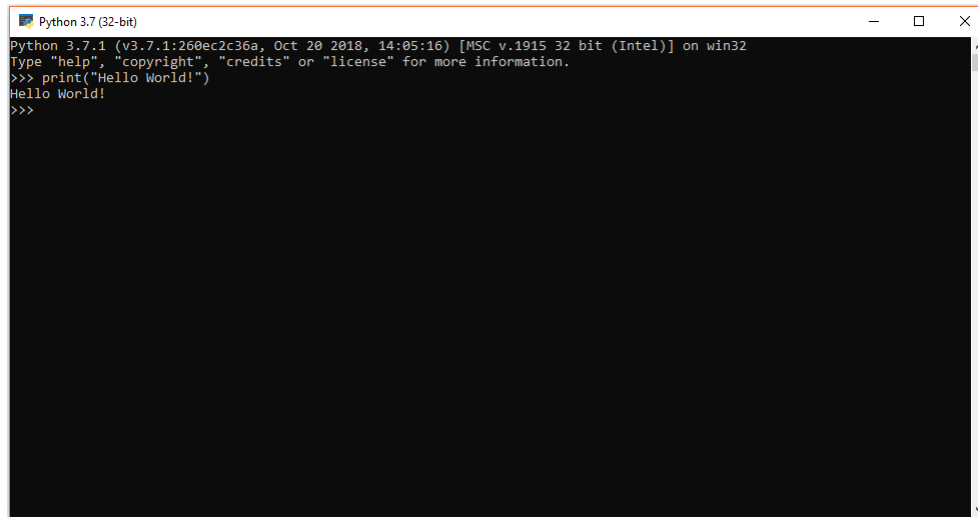
#### **2- What are programming languages?**

Computer programs are created through programming languages. A programming language is a set of rules that instruct a computer of the operations that it has to perform and the algorithm that it has to follow. Every programming language has its unique words, symbols, and grammatical rules. Many different programming languages exist and are used worldwide. The most popular programming languages currently used are Java, C++, and Python.

#### **3- Introduction to Python:**

Python is a high-level and a general-purpose programming language that was first released in 1991.

- Python environment:
  - Python Shell: is where the command and lines of code are entered and directly executed. For example, when the line of code `print("Hello World!")` is entered, this line of code is executed and the sentence *Hello World!* appears directly in the next line.

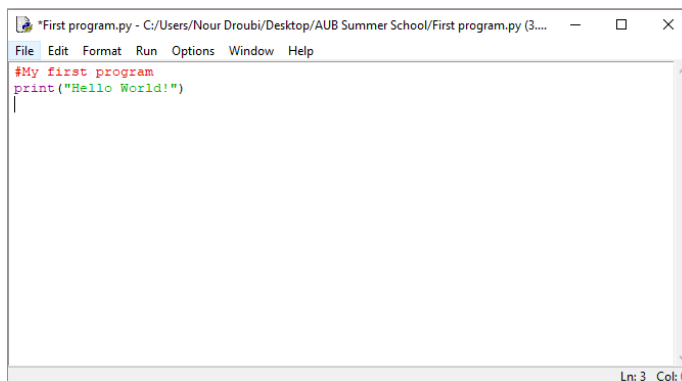


```
Python 3.7 (32-bit)
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:05:16) [MSC v.1915 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Hello World!")
Hello World!
>>>
```

- IDE: is the Integrated Development Environment where the entire code is written to be saved, used, and executed later on. IDEs are similar to text editors except that they are specific to writing and editing code. Many IDEs can be used for writing Python code such as Spyder, PyCharm, IDLE, and much more.

Examples:

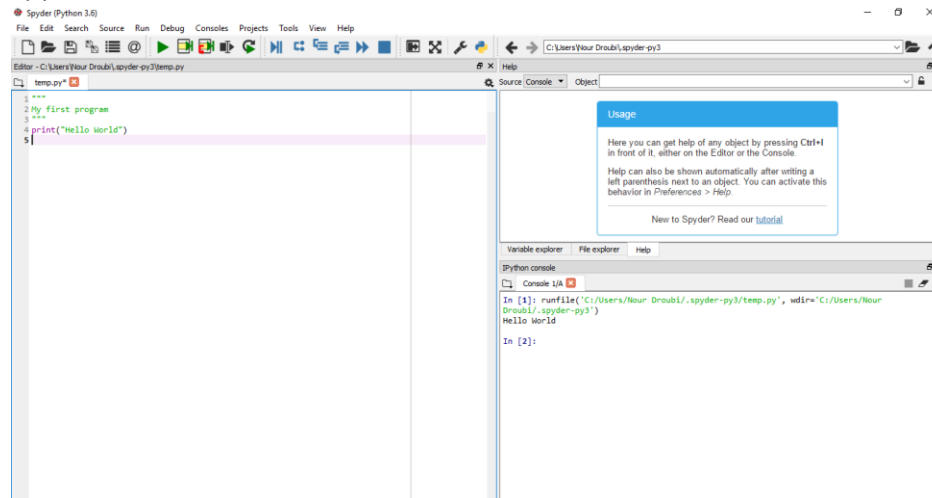
IDLE:



```
File Edit Format Run Options Window Help
#My first program
print("Hello World!")
|
```

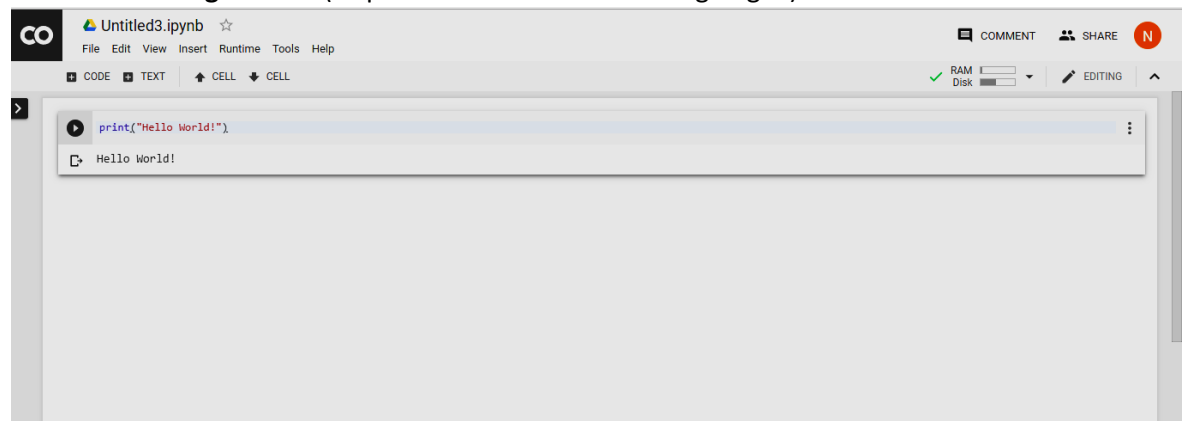
Ln: 3 Col: 0

## Spyder:

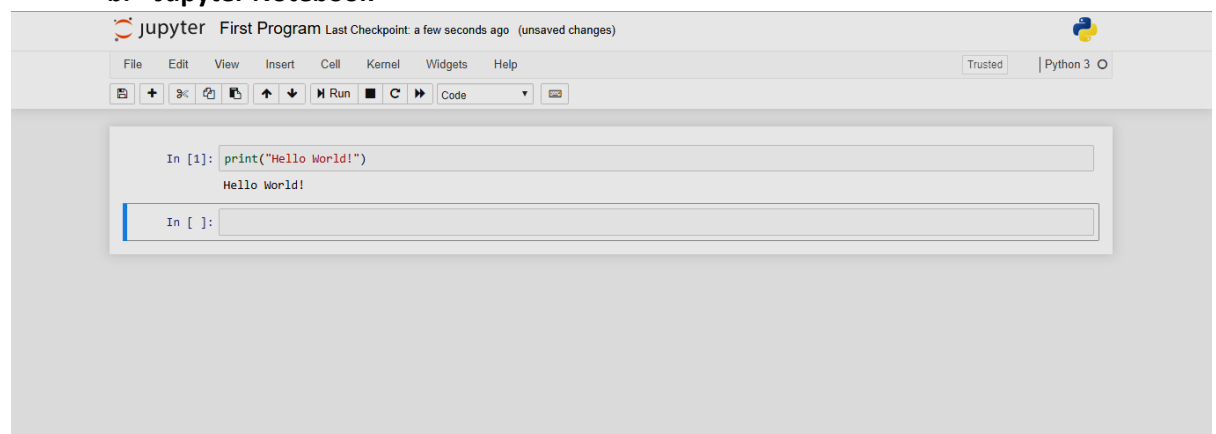


- Cloud alternatives: environments that run completely online on the cloud. They require no prior setup. Examples includes:

### a. Google Colab (requires a Gmail account for signing in)



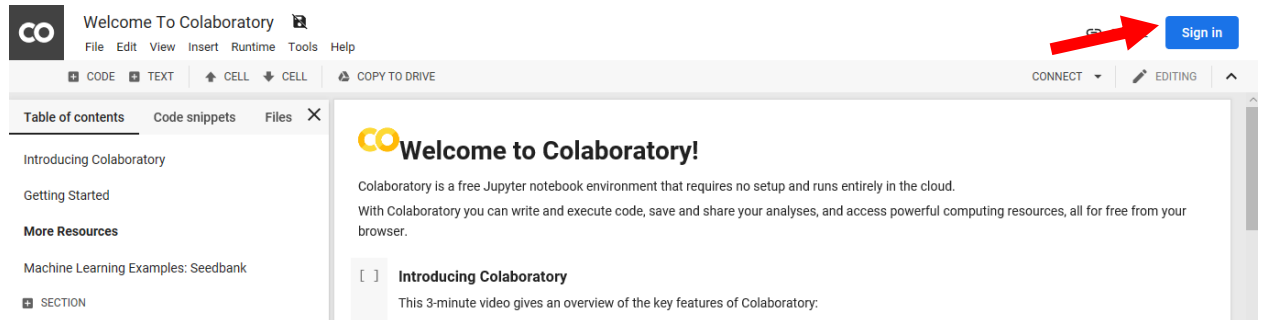
### b. Jupyter Notebook



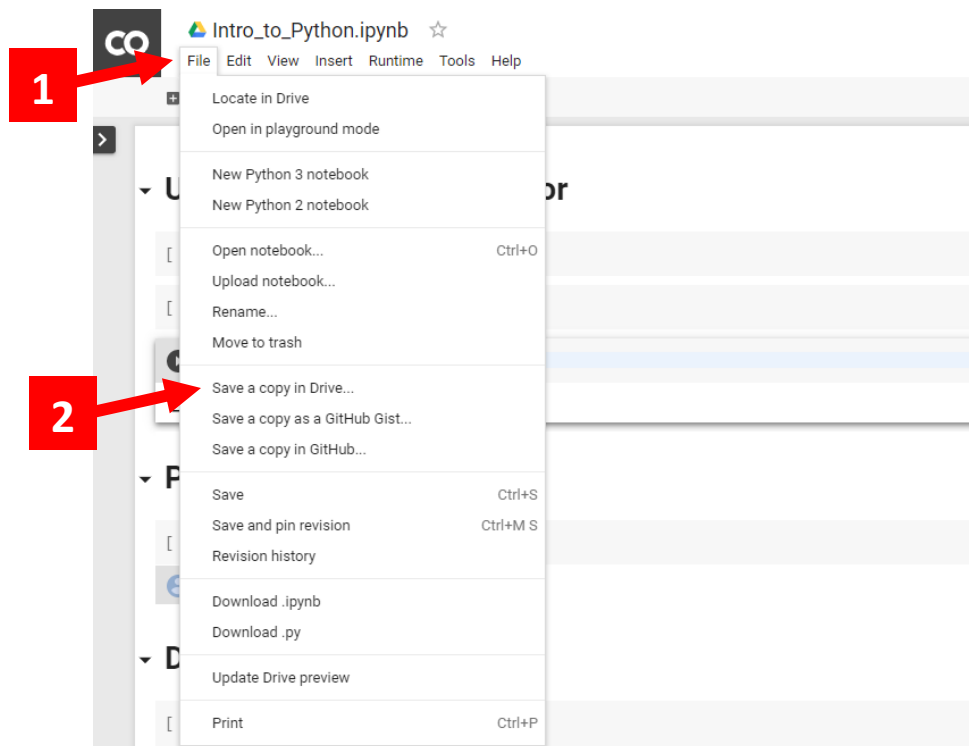
- Google Colab Tutorial:

The following steps have to be followed in order to sign in and use Google Colab:

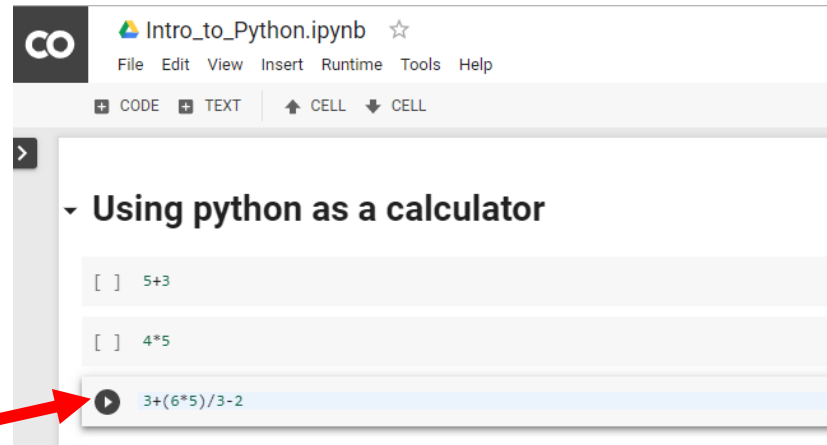
- 1- Click on the link of the Jupyter Notebook
- 2- Sign in with your Gmail account.



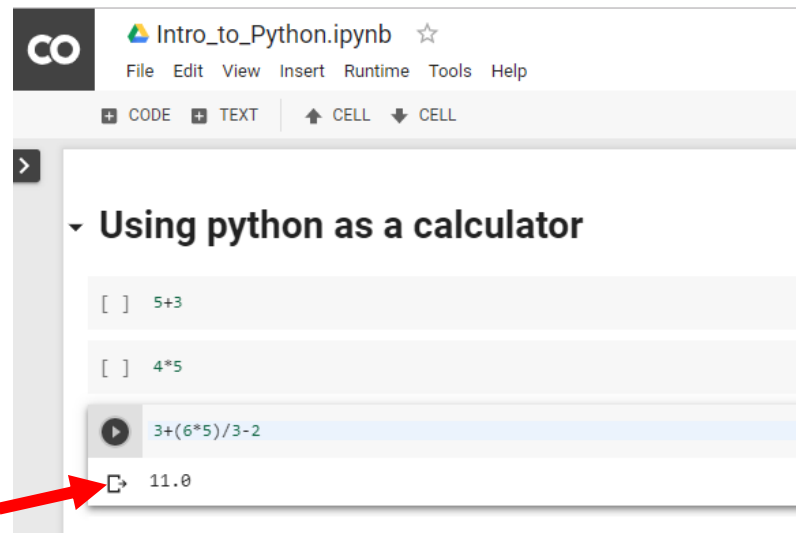
- 3- Make a copy of the file on your Google Drive by selecting File from the upper bar and then selecting "Save a copy in Drive". This will allow you to edit the file and save it in your Google Drive.



- 4- Run the code using the RUN button besides the code block.



- 5- The result is then executed under it



- Demos:

- 1- Using python as a calculator:

```
>>> 5+3
8
>>> 4*5
20
>>> 3+(6*5)/3-2
11.0
```

- 2- Simple print function:

```
>>> print("Hello World!")
Hello World!
```

- Language basics:
  - **Basic Data Types:** numbers (signed, unsigned, integers, floating point numbers), characters (a,b,A), Boolean (0 or 1, True or False)
  - **Complex Data Types:** Strings (combination of characters)
  - **Variable:** is a reserved memory location that stores values that can be used again. To create a variable, it has to be given a name and assigned to a value using the equals sign ( = ). Each variable has to be given a name that follows these rules:
    - The first character can't be a digit
    - They can be of any length
    - They can contain uppercase and lowercase letters, digits, and underscore ( \_ )

```
String = "This is a string"
number_integer = 4
number_decimal = 3.5
```

```
print(string)
print(number_integer)
print(number_decimal)
```

Output:

```
This is a string
4
3.5
```

- **Arrays:** a type of variable that holds many pieces of information. It can also hold heterogeneous information. Elements in the array are accessed using their index. The first element in an array has the index 0 in python.

```
#Initializing the array
array = [1, 4, "hello", 4.5]
```

```
#Printing the elements of the array
print(array[0])
print(array[1])
print(array[2])
print(array[3])
```

Output:

```
1
4
hello
4.5
```

Using slicing to access elements: slicing allows to access several elements from the array at once.

The notation is `array[ starting index : ending index ]`. The ending index is not accessed using this notation. The index (ending index - 1 ) is accessed.

Using the format: `array[ starting index : ]` accesses the array from the starting index till the last element.

Using the format: `array[ : ending index]` accesses the array from the first index till the ending index - 1

Using the format: `array[ : ]` accesses all the elements of the array

```
array = [1, 4, "hello", 4.5]
print( array[ : 3 ] )
print( array[ 2 : ] )
print( array[ 1 : 3 ] )
print( array[ : ] )
```

Output:

```
[1, 4, 'hello']
```

```
['hello', 4.5]
```

```
[4, 'hello']
```

```
[1, 4, 'hello', 4.5]
```

- **Operators:**

- Addition ( + )
- Subtraction ( - )
- Multiplication ( \* )
- Division ( / )
- Assignment operators ( = )
- Relational operators:
  - greater than: >
  - greater than or equal: >=
  - less than: <
  - less than or equal: <=
  - equal-to: ==
  - not-equal-to: !=
- Logical operators:
  - And: and
  - Or: or
  - Not: not
  - Boolean: True or False

- **Comments:** lines of codes that start with the symbol # are considered as comments and are not executed by the program. Comments are usually used to explain your written code to other programmers.

**#This is a comment**

- **If/else statements:** An if statement is a programming conditional statement.

If (condition) -> do this

Else -> do that

The program will check the condition. If the condition is satisfied and hence it is True, then the program will execute certain lines of code. Otherwise, other lines of code are executed.

**x = 0**

**if (x==0):**

**print("True")**

**else:**

**print("False")**

**Output:**

**True**

**x = 3**

**if (x==0):**

**print("True")**

**else:**

**print("False")**

**Output:**

**False**

- **While loops:**

A set of statements are executed as long as the condition is True.

In this example, the variable i is initialized to 1 and then it increments at every iteration of the while loop. The while loop will be entered as long as the variable i is less than 6.

**i = 1**

**while ( i < 6 ):**

**print( i )**

**i = i + 1**

**Initially: i = 1**

**Iteration 1:** the while loop checks if i is less than 6. i = 1 which is less than 6, so 1 printed and i is incremented by 1. => i = 2

**Iteration 2:** the while loop checks if i is less than 6. i = 2 which is less than 6, so 2 printed and i is incremented by 1. => i = 3

**Iteration 3:** the while loop checks if i is less than 6. i = 3 which is less than 6, so 3 printed and i is incremented by 1. => i = 4

**Iteration 4:** the while loop checks if i is less than 6. i = 4 which is less than 6, so 4 printed and i is incremented by 1. => i = 5



**Iteration 5:** the while loop checks if i is less than 6. i = 5 which is less than 6, so 5 printed and i is incremented by 1. => i = 6

**Iteration 6:** the while loop checks if i is less than 6. i = 6 is not less than 6, so the while loop is not entered, nothing is printed, and the program terminates.

Output:

1  
2  
3  
4  
5

#### - **For Loops:**

The for loop is used to iterate over a sequence. A set of statements are executed for every item in the sequence.

In this example, the for loop iterates over the numbers array and prints every item in it.

```
numbers = [1, 2, 3]
for x in numbers:
    print(x)
```

Output:

1  
2  
3

Using for loops with the range( ) function:

The range function returns a sequence of numbers, starting from 0 by default and increment by 1 by default and ends at the specified number.

In this example, the for loop iterates over a range of 3 starting at 0.

```
for x in range(3):
    print(x)
```

Output:

0  
1  
2

#### - **Functions**

A function is a block of code that only runs when it is called. It can return data as its result.

Functions are defined using the "def" keyword.

Naming a function is similar to naming variables and they must follow these rules:

- Start with a letter or underscore only ( \_ )
- Should be lowercase
- Can contain numbers except for the first character

- Shouldn't be the same as a Python Keyword
- Can have any length but they are preferred to be short

```
def my_function():  
    print("Hello World!")
```

This presents the definition of the function. However, the code will not run unless the function is called by its name as follows:

```
my_function()
```

Output:

Hello World!

A function can also take parameters as its input that can be used inside the function. In this example, the function takes a parameter number as its input which is used inside of the function to compute the result.

```
def my_function_with_parameter(number):  
    print (number + 2)
```

the function is then called as follows:

```
my_function_with_parameter(5)
```

Output:

7

Using the return() function inside the function definition will allow the function to return the result.

In this example, the same function defined above is used, however, the print() function is replaced with the return() function.

```
def my_function_with_parameter(number):  
    return (number + 2)
```

the function is then called as follows:

```
my_function_with_parameter(5)
```

However, calling the function as such will not print anything to the output since no print function is present. The result can be seen as follow:

```
print( my_function_with_parameter(5) )
```

Output:

7

#### - Indentation

The python indentation is very important, and the program will not run if the indentation is not correct.

For example, this code has a correct indentation and will run with no errors:

```
for x in range(3):  
    print(x)
```

However, this code will cause an error to appear and the program will not run:

```
for x in range(3):  
print(x)
```

Error:

**IndentationError: expected an indented block**

- **Types of errors:**

NameError: attempt to access an undeclared variable

SyntaxError: python interpreter syntax error

IndexError: request for an out-of-range index for sequence

ZeroDivisionError: division by any numeric zero

AttributeError: attempt to access an unknown object attribute

c- Python for Machine Learning:

- Libraries:

All libraries have to be imported before using them in the code as follows:

```
import library_name
```

The built-in functions of the library can then be accessed using the dot operand. For example, this line of code: `library_name.func1( )` uses the built-in function `func1` of the library.

- NumPy

A general-purpose array-processing package required for data analysis. It provides efficient and high-performance multi-dimensional array object and tools to work with these arrays.

- i. Creating NumPy Arrays:

`numpy.array( )` converts a list to a NumPy array.

```
import numpy as np  
#One-dimensional NumPy array  
list1 = [1 , 2 , 3]  
numpy_array= np.array( list1 )  
print("One-dimensional array: ", numpy_array)
```

```
#Two-dimensional NumPy array  
list2 = [ [1 , 2 , 3], [4, 5, 6] ]  
numpy_array2= np.array( list2 )  
print("Two-dimensional array: ", numpy_array2)
```

Output:

One-dimensional array: [1 2 3]

Two-dimensional array: [[1 2 3]  
[4 5 6]]

ii. Shape of NumPy array:

The shape function can be used to print the dimensions of a NumPy array.

```
import numpy as np
#One-dimensional NumPy array
list1 = [1 , 2 , 3]
numpy_array= np.array( list1 )
print("One-dimensional array: ", numpy_array)
print("Shape: ", numpy_array.shape)

#Two-dimensional NumPy array
list2 = [ [1 , 2 , 3], [4, 5, 6] ]
numpy_array2= np.array( list2 )
print("Two-dimensional array: ", numpy_array2)
print("Shape: ", numpy_array2.shape)
```

Output:

One-dimensional array: [1 2 3]

Shape: (3,)

Two-dimensional array: [[1 2 3]  
[4 5 6]]

Shape: (2, 3)

iii. Reshape function:

Used to change the dimensions of an array without changing its data. The reshape function takes as input the NumPy array to be reshaped with the new array dimensions.

```
import numpy as np
list1 = [1 , 2 , 3 , 4 , 5 , 6]
numpy_array= np.array( list1 )
print( np.reshape ( numpy_array , (3,2) ) )
```

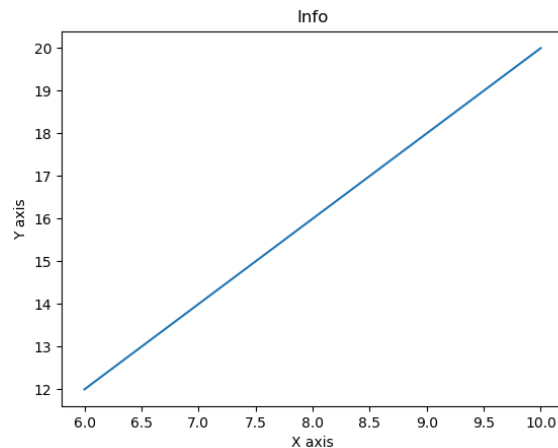
Output:

[[1 2]  
[3 4]  
[5 6]]

- SciPy  
The SciPy library depends on NumPy. It has a variety of high-level science and engineering modules together such as signal processing tools, optimization tools, and linear algebra routines.
- Pandas  
Relational data tool built on top of NumPy. Used for data analysis.  
Main functions:
  - i. Reading and writing data
  - ii. Data alignment
  - iii. Reshaping
  - iv. Slicing, subsetting
  - v. Merging and joining of datasets
- Matplotlib  
Matplotlib is a 2D and 3D graphic library used to plot scientific figures.  
Types of plots:
  - i. Bar graphs
  - ii. Histograms
  - iii. Scatter plot
  - iv. Pie Plot
  - v. Area Plot

Example of plotting a straight line:

```
from matplotlib import pyplot as plt  
x=[6,8,10]  
y=[12,16,20]  
plt.plot(x,y)  
plt.title("Info")  
plt.ylabel("Y axis")  
plt.xlabel("X axis")  
plt.show()
```



- Scikit-learn

A simple and efficient library for data mining and data analysis.

Main uses:

- i. Classification
- ii. Regression
- iii. Clustering
- iv. Preprocessing
- v. Model selection