PRACTICE 01 : THE INTRODUCTION TO PYTHON PROGRAMMING LANGUAGE

1. Aim of the practice

To develop data mining applications, we need to use a powerful programming language for mathematical calculations, data analysis, pattern recognition, signal processing, etc. Among these most commonly used languages: Matlab, Python and R. This practical work aims to learn the basic principles of the Python language, more precisely from the Anaconda distribution (another way is to Install Python individually).

2. Anaconda

Anaconda distribution platform is the most popular way to learn and use Python for scientific computing, data science, and machine learning. It is used by over thirty million people worldwide and is available for Windows, macOS, Linux platforms.

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® Distribution that allows you to launch applications and manage conda packages, environments, and channels without using command line interface (CLI) commands.

Home page of Anaconda displays all of the available applications which are already installed or are available to install: python editors (Spider, Jupyter Lab and Jupyter Notebook), Orange 3 for data mining, etc.

3. Python

Python is a popular object-oriented programming language, portable, easy to learn, free, open source, interpreted language (the source code of Python program would be first converted into bytecode and then executed by Python virtual machine), Availability of libraries, etc.

Python was developed by Guido van Rossum at Stichting Mathematisch Centrum in the Netherlands. The first version was released in 1991. It can be downloaded from www.python.org

The engine that translates and runs Python is called the Python Interpreter.

Python Interpreter is easily extended with new functions and data types implemented in C or C++.

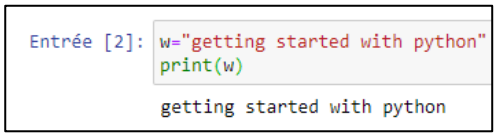
3.1. How to use python

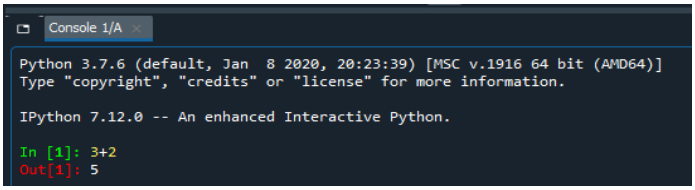
There are two ways to use it:

3.1.1. Immediate mode

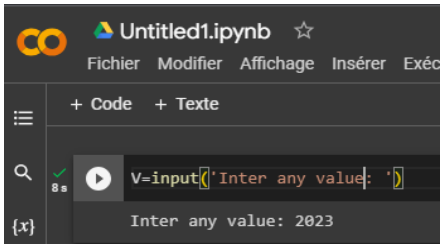
Immediate mode also called interactive mode. It aims to write the python expressions into python interpreter window, and the interpreter immediately shows the results. For example, Jupyter notebook (figure 1), editer of Spider (figure 2), Google Colab (for cloud usage) (figure 3).

(1)





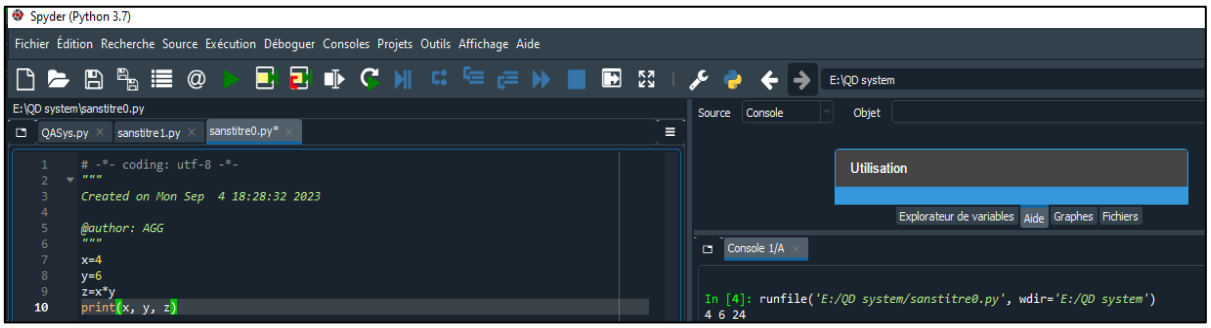
(2)



(3)

3.1.2. Script mode

Write a program in a file called Script (for example .py extension) and use the interpreter to execute the contents of the script.



4. Jupyter notebook

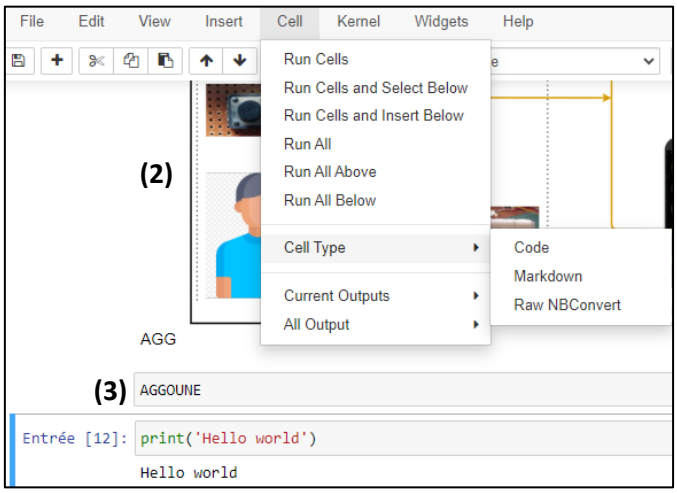
Jupyter notebooks basically provides an interactive computational environment for developing Python based Data Science applications. There are two ways to use Jupyter notebook: Anaconda Navigator then choose Jupyter notebook application or use Anaconda prompt and write jupyter notebook. After that, select python 3 and it will take you to the new notebook for start working in it.

The interface of Jupyter is a set of cells. There are three types of cells:

Code cells: As the name suggests, we can use these cells to write code, which is send to the kernel for the execution (see part 1 of the figure).

Markdown cells: they use for notating the computation process, which can be contain text, image, etc. (see part 2 of the figure).

Raw cells: The text written in them is displayed as it is. (see part 3 of the figure).



5. Algorithmic expressions

- **Basic mathematical operations**

Symbol	Signification
+, -, *, /	Addition, subtraction, multiplication, division
//	Floor division (return integer value)
%	Modulus (The remainder of the division)
**, =	Exponent, assign
+=, -=	Add and assign, subtract and assign
*=, /=	Multiply and assign, divide and assign
//=, %=	Floor divide and assign, modulus and assign
**=	Exponentiation and assign ex. a**=3

Symbol	Signification
>	Greater than
<	Less than
==	Equal to
!=	Not equal to
>=	Greater than or equal to
<=	Less than or equal to
and	Logical and
or	Logical or
not	Logical not
In, not in	finds a given object in (or not in) the sequence

- **Conditional expression**

```
if <expr>:
   <statement>
elif  <expr>:
  <statement>
else:
  <statement>
<statement>
```

Indentation to define new block

- **Loop expressions**

for x in range ([start], stop, [step]) # Generate numbers between start and stop-1 with step value. Default values for start and step are 0 and 1 respectively.

```
i=1
while condition :
    <statement>
    i += 1
```

- **Function**

```
def function_name(parameter1, parameter2):
    # function body
    # write some action
return value1, value2, ….
```

- **Input Function:** allows user input

  Input('enter new value for x')

- **Casting:** To perform a type casting, there are five built-in functions:
  1. **int():** convert any type variable to the integer type.
  2. **float():** convert any type variable to the float type.
  3. **complex():** convert any type variable to the complex type.
  4. **bool():** convert any type variable to the bool type.
  5. **str():** convert any type variable to the string type.

Exercises: during the Practical session