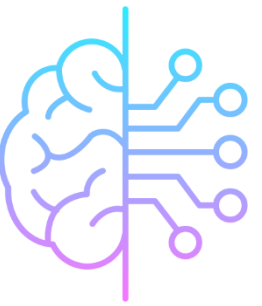




SESSION #01



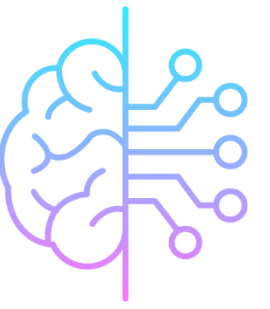
Python review from A2Z



WHAT IS PYTHON ?

Python is an **interpreted, high-level, general-purpose, object-oriented programming language.**

It is well known for its simple syntax, which brought him very close to normal human speech, and for its **large community.**



HOW TO USE PYTHON



ANACONDA

Anaconda is an open-source distribution of the Python and R programming languages for data science



PYTHON

instal python then use the ide that sweets you



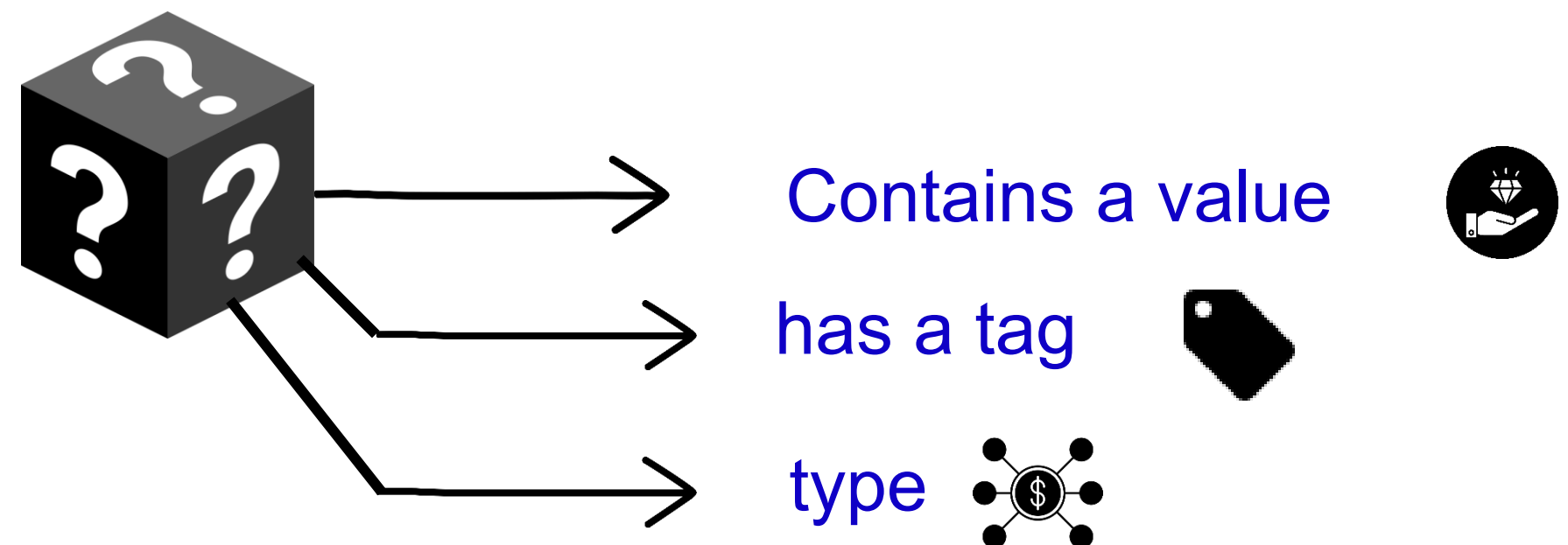
WHAT IS VARIABLE

S.Djellouli

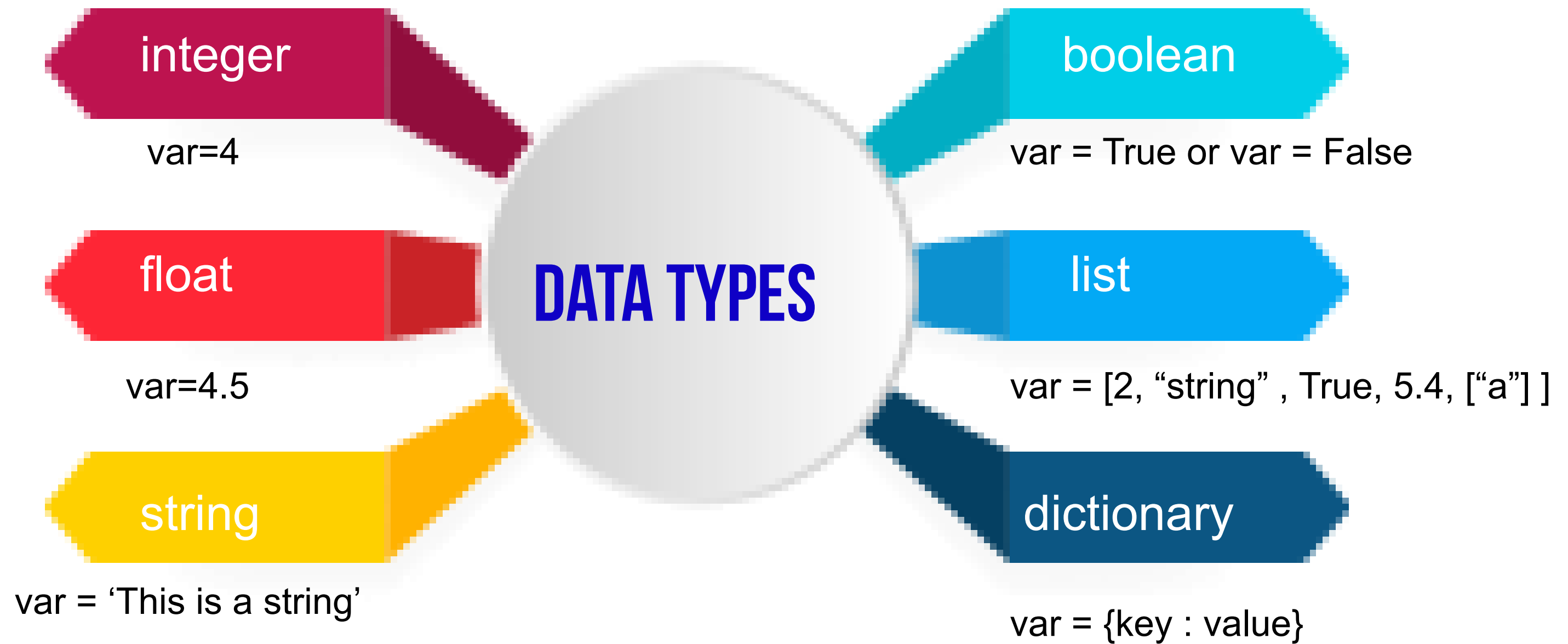


WHAT IS VARAIBEL

- In python, variables are defined the moment that a value has been assigned to them
- The type of the variable is determined by its content
- Everything is an object

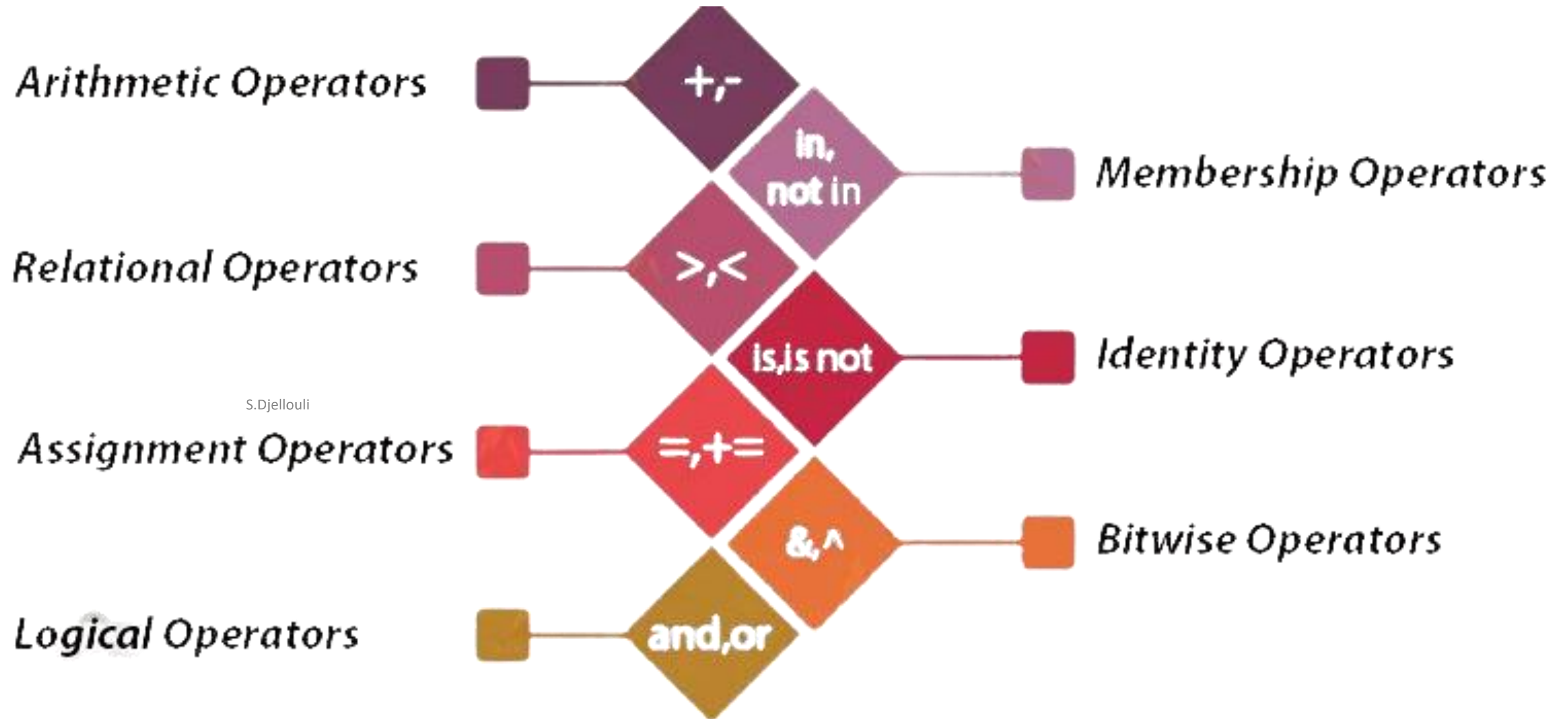


DATA TYPES

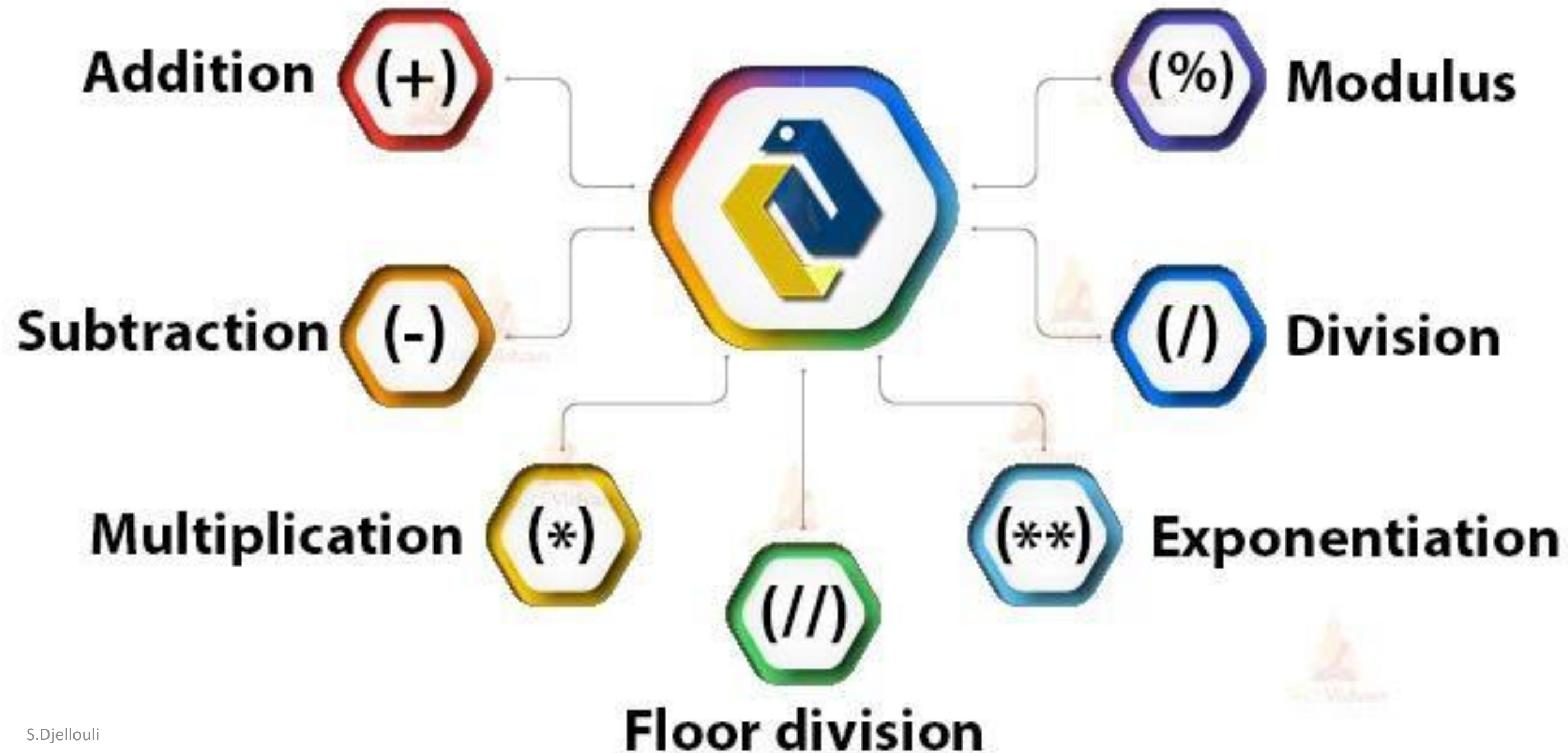


List	Dictionary
index value pair	key-value pairs.
List is initialized with [], and elements are separated by ','.	Dictionary is created by placing elements in {}, data is added as key-value pair, and each pair is separated by ','.
List values can be accessed by numeric indexes.	We can't edit the order of dictionary elements.
<small>S.Djellouli</small> The order of elements in the list can be modified	doesn't allow duplicate values with the same key in it.
allows duplicate values.	Dictionary items can be accessed by using key values. Key values can be of any data type.

PYTHON OPERATORS

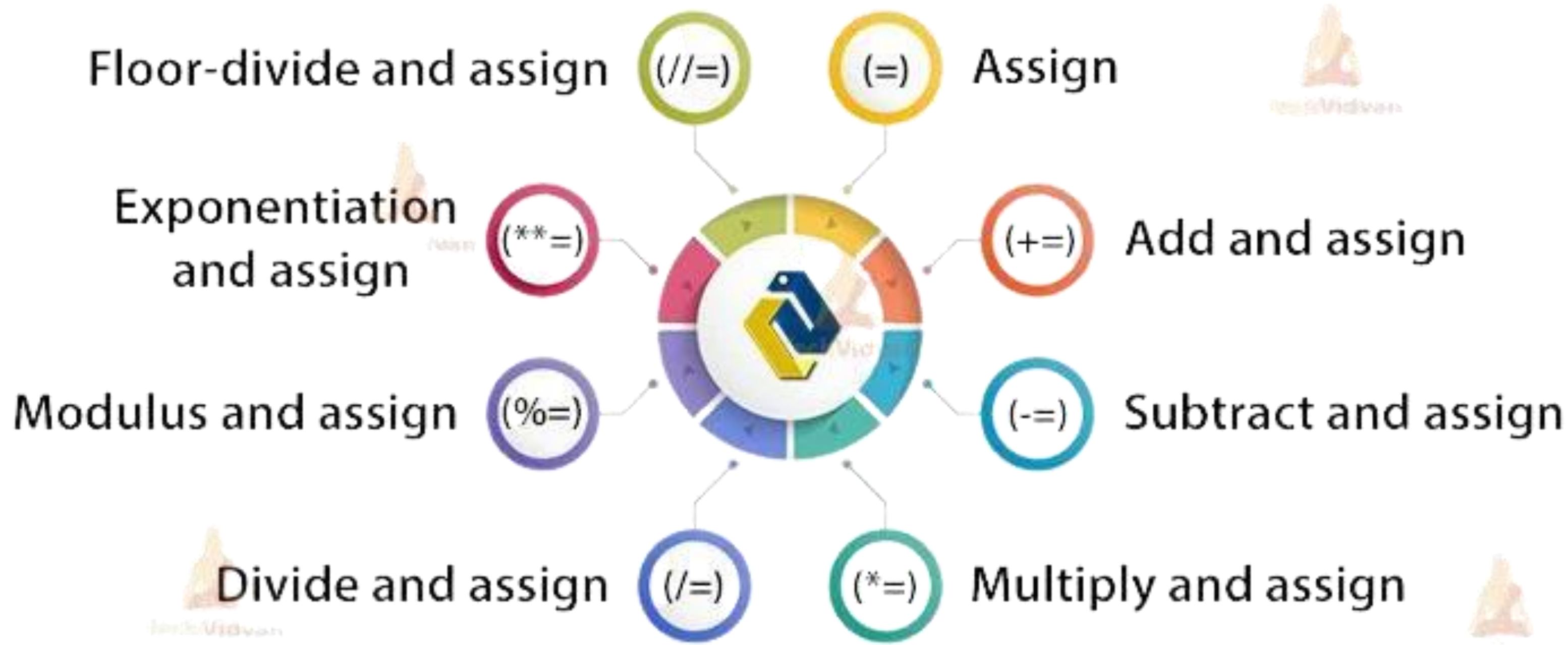


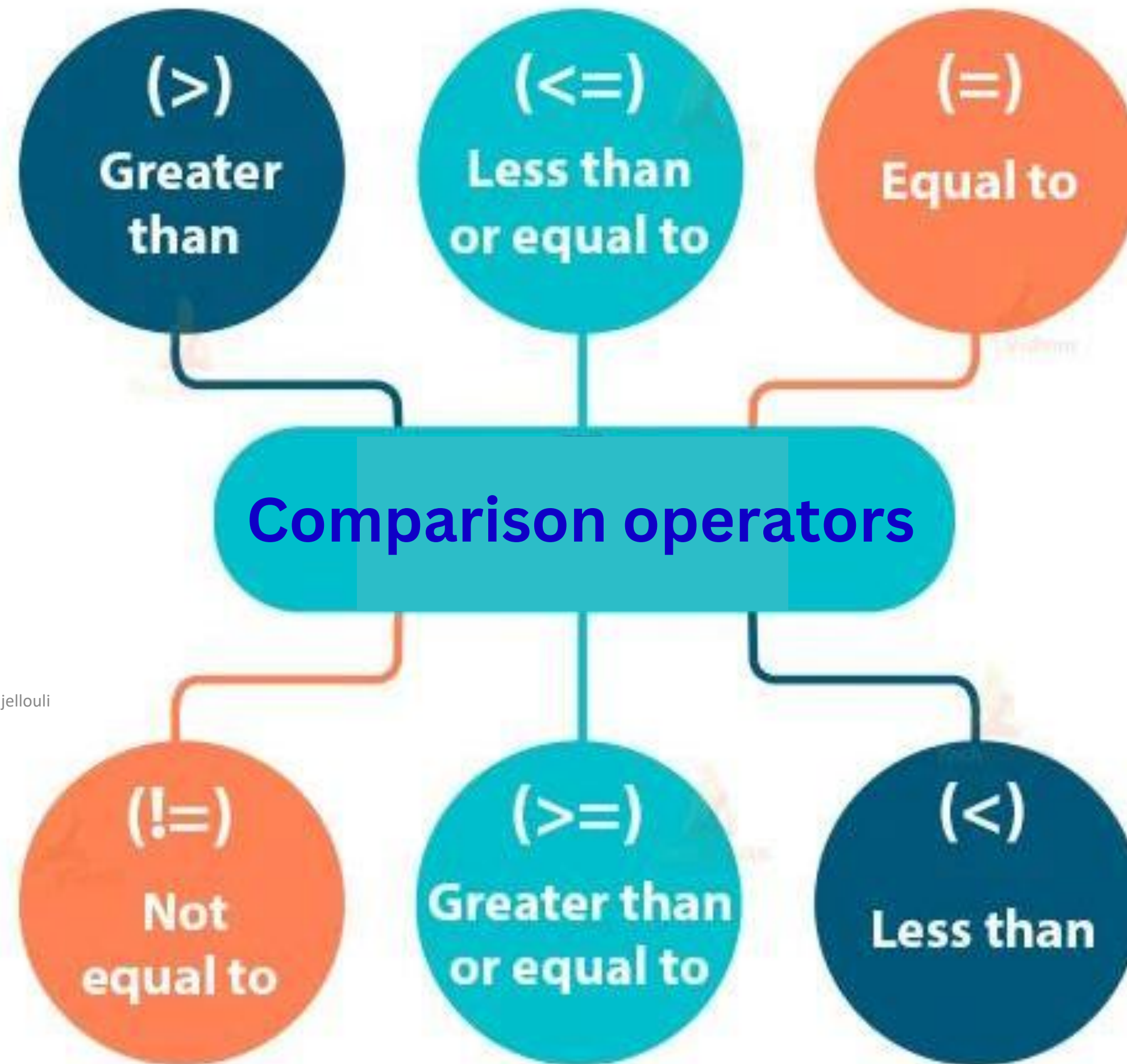
Python Arithmetic Operators



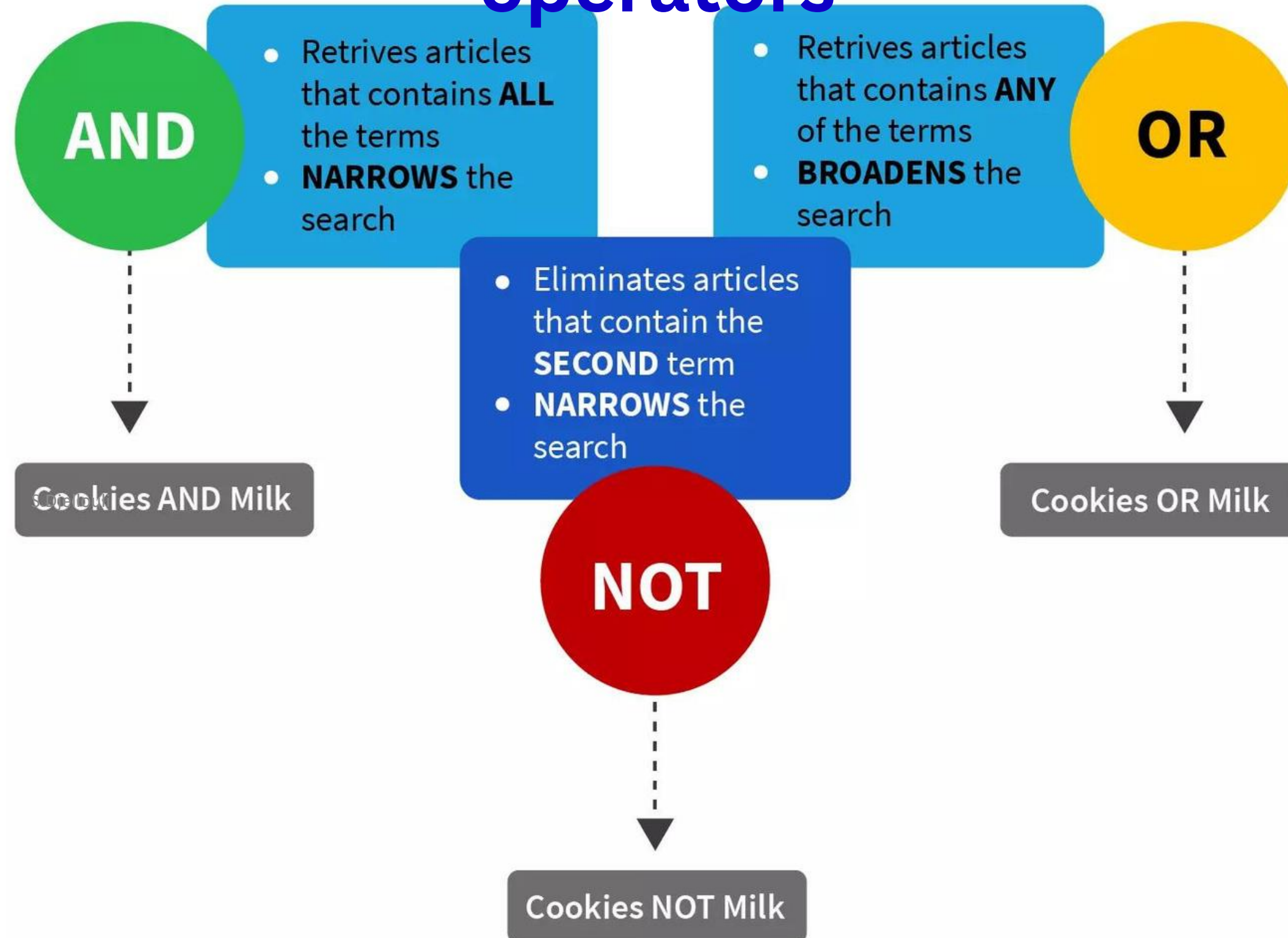


Python Assignment Operators



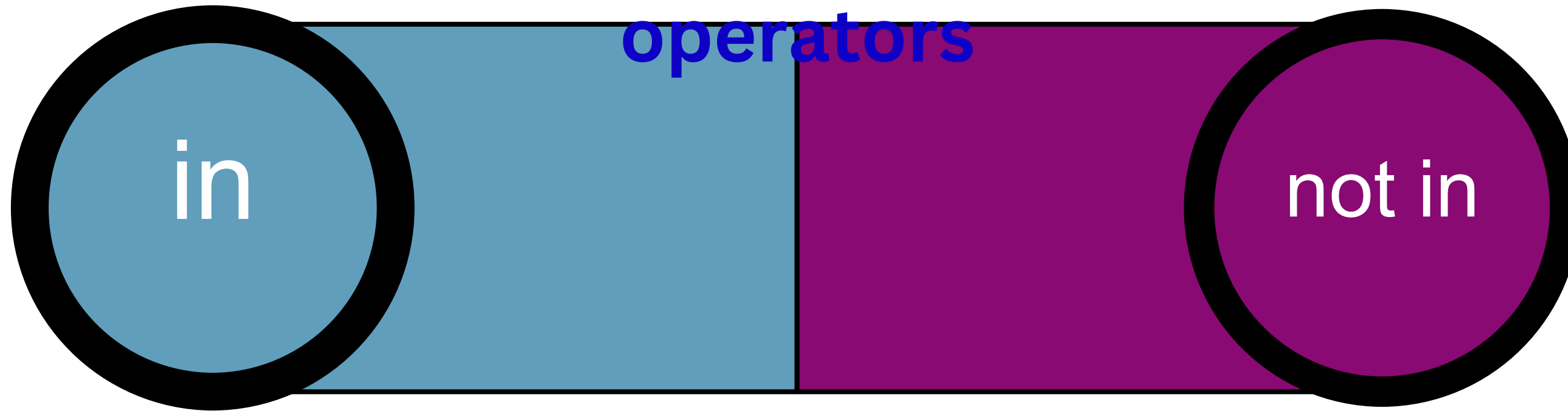


Logical operators



Membership

operators



The **in** membership operator in [Python](#) is used to check the existence of a given element in the given sequence like string, list, tuple, or dictionary. for "not in" it is the inverse of the "in".

PYTHON CONDITIONS STATEMENTS

IF STATEMENT

```
if condition :  
    # do stuff  
elif second condition :  
    # do other stuff  
elif third condition :  
    # do other other stuff  
else :  
    # do the left stuff
```

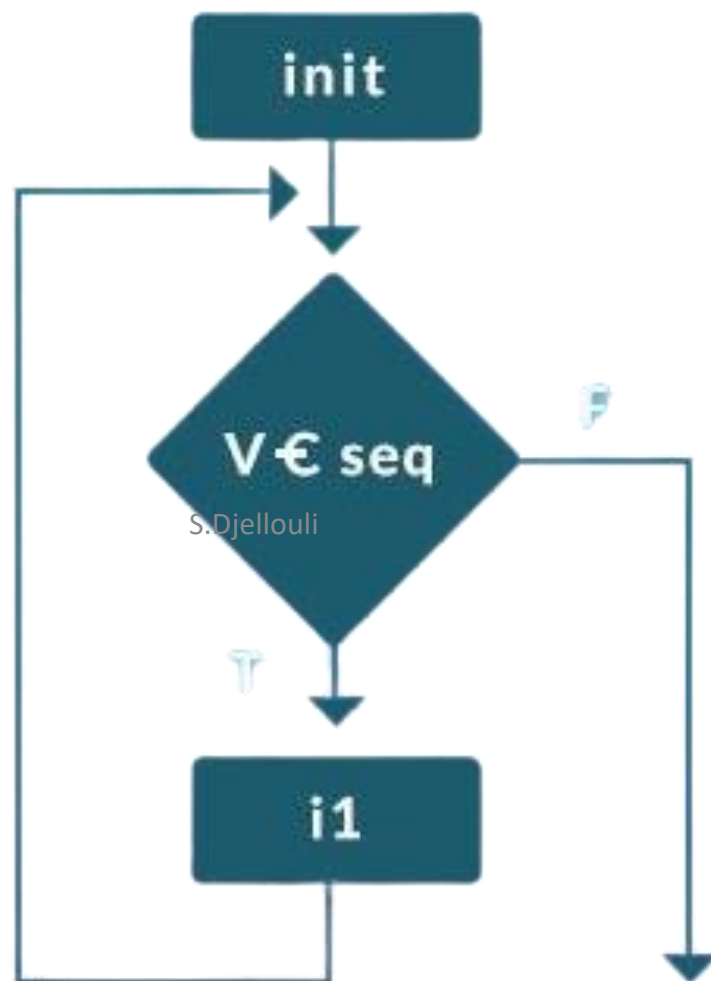
NOTE :

Indentation in Python is used to define blocks of code and control the flow of execution within the code.

It is a visual way of grouping statements and differentiating code blocks.

PYTHON LOOPS

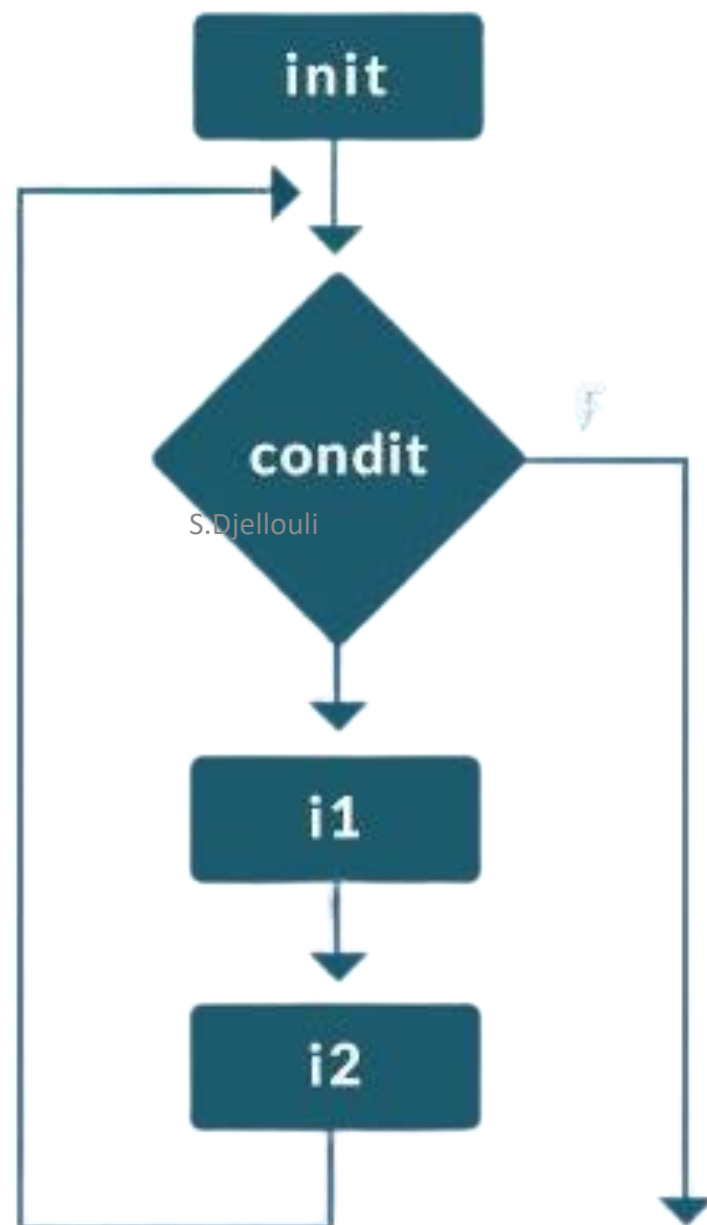
FOR:



```
for i in range(start, end, step):  
    # do some stuff here  
  
# default start = 0  
# default step = 1  
# the condition is : i < end
```

PYTHON LOOPS

WHILE:



```
while condition :  
    # do things
```

```
i = 0  
while i < 10 :  
    print(i)  
    i+=1
```

Scripts, Functions, Modules and Libraries

Python provides a variety of structures to help programmers write, organize, and reuse code effectively.

One of these tools is **scripts**, which are collections of code written in Python and executed line by line. **Functions** are another structure, which are blocks of code that can be called and reused. **Modules** are collections of functions and other objects in Python, and **libraries** are collections of modules that provide pre-written code for common tasks.

In next slides, we will discuss each of these concepts in detail.

Scripts

A script in Python is a basic and small unit of code and executed line by line. It is typically used to perform one-time operations, or test small pieces of code.

S.Djellouli

```
# Addition
x = 4
y = 7
print('Numbers are', x, 'and', y)
print('result:', x+y)
```


Functions

Functions are a way to organize code into reusable blocks and allow us to call it when we need to. Functions can be passed inputs, called arguments, and can return output.

```
# Addition
x = 4
y = 7
print('Numbers are', x, 'and', y)
print('result:', x+y)
```

S.Djellouli

The diagram shows a Python function definition with six numbered annotations in red text and arrows:

- 1. def keyword: points to the `def` keyword.
- 2. function name: points to the `add` function name.
- 3. function arguments inside (): points to the `x, y` arguments inside the parentheses.
- 4. colon ends the function definition: points to the colon `:` at the end of the first line.
- 5. function code: points to the `print` and `return` statements inside the function body.
- 6. function return statement: points to the `return` statement.

```
def add(x, y):
    print(f'arguments are {x} and {y}')
    return x + y
```

PYTHON FUNCTIONS

```
def function_name(arg1, arg2, arg3):  
    # do some actions here  
    # return something if necessary  
  
# to call a function  
function_name(arg1, arg2, arg3)
```

PYTHON FUNCTIONS

The diagram illustrates the components of a Python function definition. It shows the code: `def add(x, y):` on the first line, `print(f'arguments are {x} and {y}')` on the second line, and `return x + y` on the third line. Red arrows point from numbered labels to specific parts of the code: 1. 'def' keyword, 2. 'add' function name, 3. 'x, y' function arguments inside parentheses, 4. the colon at the end of the first line, 5. the 'print' statement (function code), and 6. the 'return' statement (function return statement).

```
def add(x, y):  
    print(f'arguments are {x} and {y}')
```

1. def keyword

2. function name

3. function arguments inside ()

4. colon ends the function definition

5. function code

6. function return statement

`def` `add` `(x, y):`
`print(f'arguments are {x} and {y}')`
`return x + y`

S.Djellouli

Modules

Even though the script is converted to a function, we still have to rewrite the function when we need it in another program.

For this we will get to know the **module**,

A module is a file containing Python definitions and statements. Modules can define **functions**, **classes**, **variables**, and any other **objects** in Python.

A module can be imported into another Python script or program, making its functions, classes, and objects available for use.

To make the idea clearer, let's take an example.

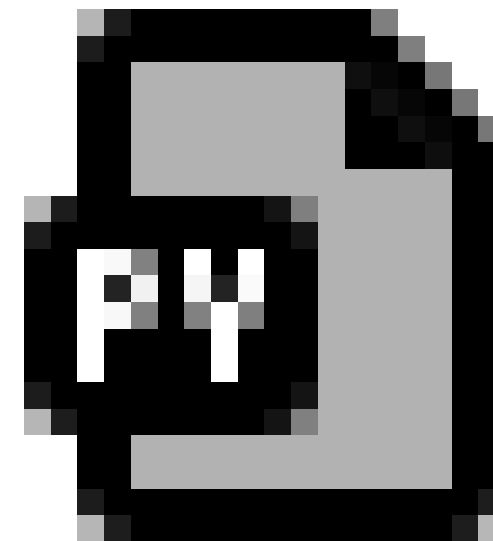
Let's collect the functions of performing arithmetic operations in a module.

Modules

```
def add(a, b):  
    return a + b  
  
def subtract(a, b):  
    return a - b  
  
def multiply(a, b):  
    return a * b  
  
def divide(a, b):  
    return a / b  
  
print("5 + 2 =", add(5, 2))  
print("5 - 2 =", subtract(5, 2))  
print("5 * 2 =", multiply(5, 2))  
print("5 / 2 =", divide(5, 2))
```

Here is an example of a simple Python script that implements functions for arithmetic operations:

To convert this script into a module, we can save it in a separate file with a **.py** extension, for example, **arithmetic.py**.



Modules

Then, in another script or program, we can import this module and use its functions.

```
import arithmetic

print("5 + 2 =", arithmetic.add(5, 2))
print("5 - 2 =", arithmetic.subtract(5, 2))
print("5 * 2 =", arithmetic.multiply(5, 2))
print("5 / 2 =", arithmetic.divide(5, 2))
```

S.Djenbouli

Libraries

A library is a collection of modules that provide additional functionality for use in a Python program.

PYTHON LIBRARIES

S.Djellouli



HOW TO INSTALL LIBRARIES

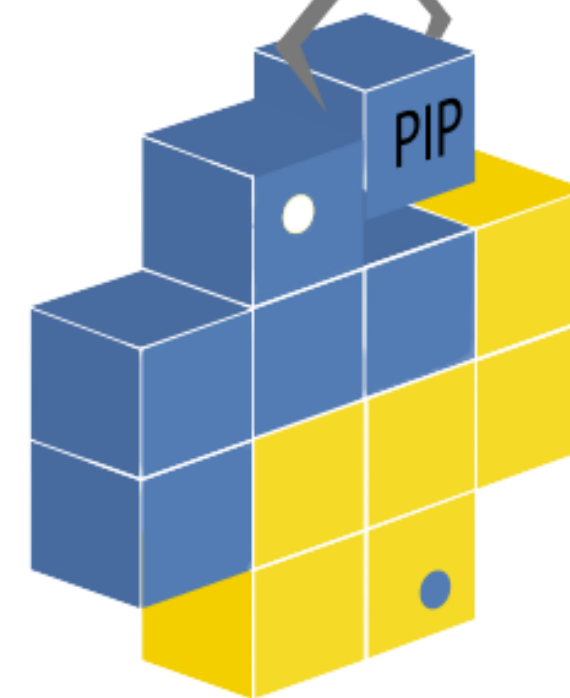


01 USING PIP

S.Djellouli

pip is the standard package manager for Python. It allows you to install and manage additional packages that are not part of the Python standard library.

pythonTM



pip
Installation

HOW TO INSTALL LIBRARIES



01 USING CONDA



S.Djellouli

Conda is an open-source, cross-platform, language-agnostic package manager and environment management system



HOW TO INSTALL LIBRARIES

COMPARISON:

<u>pip</u> 	 <u>conda</u> ANACONDA
pip search pyserial	conda search pyserial
<small>S.Djellouli</small> pip install pyserial	conda install pyserial
pip install pyserial --upgrade	conda update python
pip list	conda list

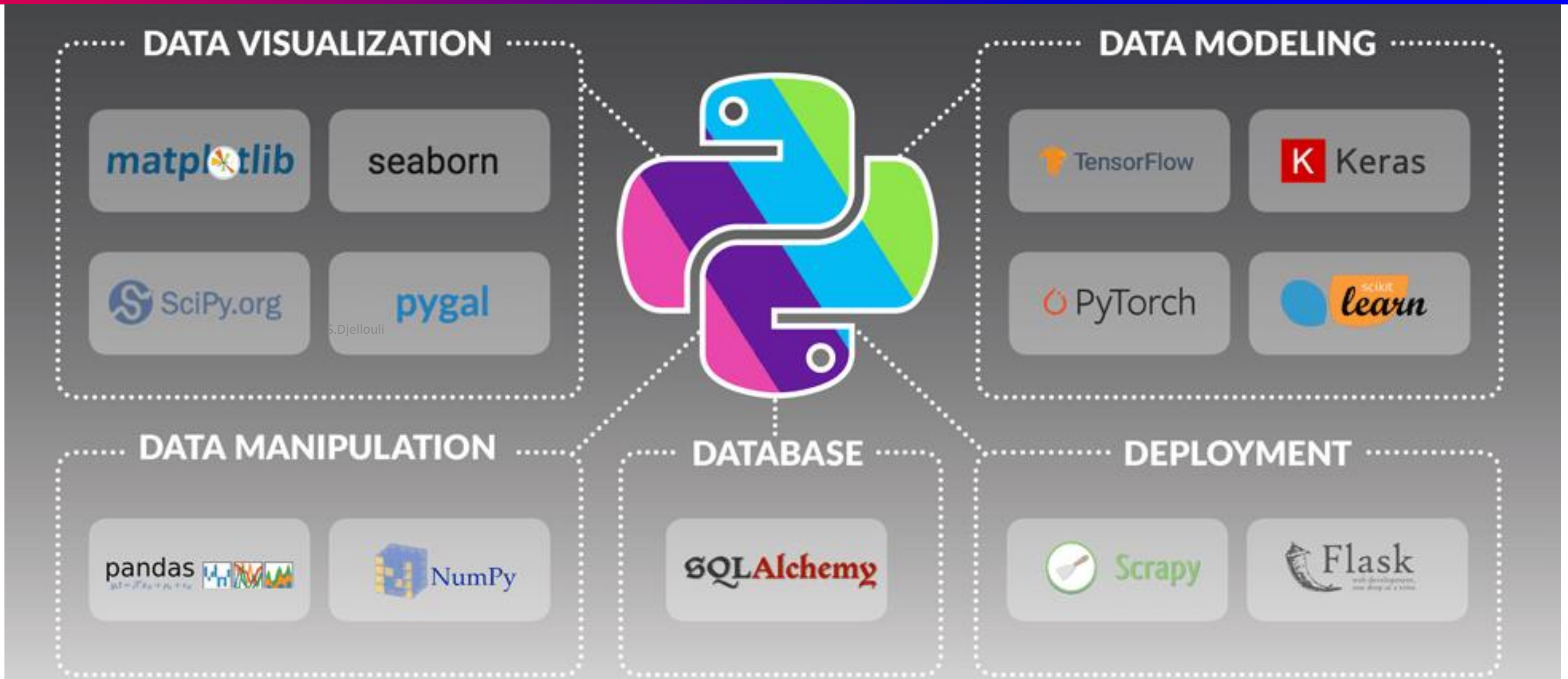
HOW TO IMPORT LIBRARIES

S.Djellouli

we can just use it by importing that library and calling the method of that library

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from matplotlib.pyplot import rcParams
rcParams['figure.figsize']= 20,5
```

SOME FAMOUS LIBRARIES



**THANK FOR YOUR
ATTENDANCE**

S.Djellouli