```python
# BOOLEAN OPERATOR
not a, and, or, ==, is, !=, True, False
3 == 3.0                    # True
3 is 3.0                    # False

# STRINGS
a = "Hello" + " World"      # "Hello World"
a[4]                        # 'o'
a[2:4]                      # 'll'
a[:5]                       # "Hello"
a[:-2]                      # "Hello Wor"
a[-3]                       # 'r'
a[::-1]                     # "dlroW olleH"
f"{a} dada"                 # "Hello World dada"
len(a)                      # 11
a.lower()                   # "hello world"
a.upper()                   # "HELLO WORLD"
a.split(" ")                # ["Hello","World"]

# CONVERTIONS
int(a)                      # convert "a" to integer
float(a)                    # convert "a" to float
str(a)                      # convert "a" to string

# FUNCTIONS
def addition(a, b):
    return a + b

# "WHILE" STRUCTURE
while True:
    print("Do you know the IGS ?")

# "FOR" STRUCTURE
for i in range(3):
    print(i)

# WHILE & FOR : BREAK & CONTINUE
while True:
   continue      # skip to next iteration
   break         # stop the loop

# CONDITION STRUCTURE
if x<42:
   print("Yup")
elif x==418:
   print("I'm a teapot :)")
else:
   print("euhh...")

# RANGE FUNCTION (start, stop, step)
range(3)              # [0,1,2]  (=range(0,stop,1))
range(2,5)            # [2,3,4]
range(0,10,2)         # [0,2,4,6,8]

# EVALUATION (use with care : it's dangerous !)
a=eval("2+2")                          # 4
a=eval("[i for i in range(3)]")        # [0,1,2]
```

```python
# GLOBAL VARIABLE
a = 0
def f():
    global a
    a=3
f()         # now a=3 everywhere

# DEBUG
type(a)     # Returns object(a) type
dir(a)      # Lists attributes of an object a
help(a)     # Show doc of a (~man a)

# IO
a=input("Type something")           # ask a String
print("%d %s %.2f" %(2,"ui",3.1415))# "2 ui 3.14"
print("%g" % x)     # print w/ shortest representation
print("x =",3,"; y =",2)            # "x = 3 ; y = 2"

# OPEN
with open("flag.txt", "r") as fich  # READ
    for i in fich:
       print(i)              # line by line
    print(fich.readline())   # read the first line
    print(fich.readline())   # read the second line..
    print(fich.readlines())  # list of all lines

with open("flag.txt", "w") as fich  # WRITE
    fich.write("TRACS2023")

# OS (import os)
os.mkdir("dev")     # create a new directory
os.chdir("dev")     # Enter in the "dev" directory
print(os.getcwd()) # "/home/.../dev" (~pwd in bash)
os.listdir("dev")  # List of files/dir in "dev" (~ls)

# EXCEPTIONS
IndexError          # Out-of-range index (ex : list)
NameError           # Undefined variable name
SyntaxError         # Invalid code syntax
TypeError           # Wrong object type ("a"+1.0)
ZeroDivisionError   # Division by 0
IOError             # File empty or Non-existent file

# RAISING ERROR
raise ZeroDivisionError

# ERROR HANDLING
try:
    print(x/y)
except ZeroDivisionError:
    print ("Division by zero")
else:
    print ("No exceptions were raised")
finally:
    print ("Always executed")
```

```python
# LISTS
a = ["clubinfo",3]
b = [3.1415,5.2,1,0]
a+b                 # ["clubinfo", 3, 3.1415,5.2,1,0]
3 in a              # True
"clubrobot" in a    # False
all(a)              # True if each element of a is True
any(a)              # True if 1 element of a is True
a.append(4)         # Add 4 to a
a.index(4)          # 2 (because third position)
a.remove(4)         # Remove 4 from a
a.pop()             # Remove the last (3) and return it
a.pop(0)            # Remove ("clubinfo) and return it
b[0]                # "clubinfo"
b[1:3]              # [5.2, 1]
sorted(b)           # return [0, 1, 3.1415, 5.2]
b.sort()            # b = [0, 1, 3.1415, 5.2]

# TUPLES
a = ("clubinfo", 3)
b = (3.1415, 5.2, 1)
a + b               # ("clubinfo", 3, 3.1415, 5.2, 1)
3 in a              # True
"clubrobot" in a    # False
a[0]                # "clubinfo"
b[1:3]              # (5.2, 1)
len(a)              # 2
a.count(3)          # 1
a.index("clubinfo") # 0
x,y=a               # x="clubinfo", y=3
```

⚠ WARNING !

Tuples are immutable ! Their elements cannot be changed once created, ensuring data consistency and allowing them to be used as dictionary keys.

```python
# DICTIONNARY
d = {"name": "Alice", "age": 25}
len(d)              # 2
d["age"]            # 25
d.keys()            # dict_keys(['name', 'age'])
d.values()          # dict_values(['Alice', 25])
"age" in d          # True
"address" in d      # False
del d["age"]        # Remove age
d.pop("name")       # return Alice, remove it from d
d.get("name")       # "Alice"
d.clear()           # d = {}

# LIST COMPREHENSION
[(key,value) for key,value in range d.items()]

# RANDOM (import random)
random.randint(3,5) # "random" number between 3 and 5
```

```python
# IDENTITY AND MEMORY
id(a)               # memory address of 'a'
sys.getrefcount(a)  # number of references to 'a'
sys.getsizeof(a)    # size in bytes of 'a'

# IMPORT STUFF
import chatbot
print(chatbot.greet())

import chatbot as chat
print(chat.greet())

from chatbot import greet
print(greet())

# POINTERS (only with LIST or DICT)
a = [1,2]     # it seems too complicated ;)
b = a         # a = b = [1,2]
b[1] = 3      # a = b = [1,3]
c = a.copy() # c = [1,3]
c[1] = 50     # c = [1,50] but a = [1,3]

# MATH (import math)
math.pow(3, 2)                          # 9
math.inf                                # Infinity
math.floor(5.5)                         # 5
math.sin(0)                             # 0.0
math.cos(0)                             # 1.0
math.sqrt(16)                           # 4.0
math.isclose(1.0, 0.9, rel_tol=1e-1)    # True
math.factorial(5)                       # 120
math.log(2, 10)          # ~0.3 (log of 2 in base 10)

# NUMPY (import numpy)
import numpy as np

a = np.array([1, 2, 3])         # Create a 1D array
b = np.array([[1, 2], [3, 4]]) # Create a 2D array
b.shape                 # (2, 2) - Shape
np.zeros((2, 3))        # Array of zeros
np.ones((3, 3))         # Array of ones
np.identity((2,2))      # Identity Matrice

a + 2                   # [3, 4, 5]
a * 2                   # [2, 4, 6]
np.matmul(a,b)          # a*b (matrix)
np.dot(a,b)             # a*b (matrix)
np.dot(a, np.array([4, 5, 6])) # 32 - Dot product
b.T                     # Transpose of the 2D array
np.mean(a)              # 2.0 - Mean
np.sum(a)               # 6 - Sum of the array
np.min(a)               # 1 - Minimum element
np.max(a)               # 3 - Maximum element
np.concatenate([a, a]) # [1, 2, 3, 1, 2, 3]
np.vstack([a, a])       # Stack arrays vertically
np.hstack([a, a])       # Stack arrays horizontally
```