Everything in Python is an object and has a truth value!

Each object in python has a truth value and it's **True** or **False**.

---

1. True Objects

- modules, packages and libraries are True

In [1]:
```
# modules, packages, libraries are True
import random # random module is True
import numpy as np # numpy module is True

print('modules(random is a built-in module) are', bool(random))
print('packages(numpy is a package) are', bool(np))
```

```
modules(random is a built-in module) are True
packages(numpy is a package) are True
```

- functions are True

In [2]:
```
# functions are True.
def func():
    pass

print('funcitons are', bool(func))
```

```
funcitons are True
```

- classes are True

In [3]:
```
# class is an object itself and is True.
class MyClass:
    pass

print('classes are', bool(MyClass))
```

```
classes are True
```

- methods are True

In [4]:
```
# methods are True
class MyClass:
    def __init__(self):
        self.value = 1
    # It's a method.
    def increase(self):
        self.value = self.value + 1

my_object = MyClass()
print('methods are',bool(my_object.increase))
```

```
methods are True
```

- True Built-in types:
  - True
  - non-zero int
  - non-zero float
  - non-empty str

In [5]:

```
print(1,'\t->\t', bool(1)) # non-zero int
print(-1,'\t->\t', bool(-1)) # non-zero int
print(complex('-1+j'),'->\t', bool(complex('-1+j'))) # non-zero complex
print(True, '\t->\t', bool(True)) # True bool
print(0.01,'\t->\t', bool(0.01)) # non-zero float
print(-0.01, '\t->\t', bool(-0.01)) # non-zero float
print('hi','\t->\t', bool("hi")) # non-empty string
```

```
1       ->      True
-1      ->      True
(-1+1j) ->      True
True    ->      True
0.01    ->      True
-0.01   ->      True
hi      ->      True
```

- **True** Built-in data structures:
  - non-empty **list**
  - non-empty **tuple**
  - non-empty **set**
  - non-empty **range**
  - non-empty **dict**

In [6]:

```
print([1,2,3], '\t->\t',bool([1,2,3])) # non-empty list
print((1,2,3), '\t->\t',bool((1,2,3))) # non-empty tuple
print({1,2,3}, '\t->\t',bool({1,2,3})) # non-empty set
print(range(2), '\t->\t',bool(range(2))) # non-empty range
print({'Jim': '+23453', 'Sara': '+79345'},'\t->\t', bool({'Jim': '+23453', 'Sara': '+79345'}
```

```
[1, 2, 3]       ->      True
(1, 2, 3)       ->      True
{1, 2, 3}       ->      True
range(0, 2)     ->      True
{'Jim': '+23453', 'Sara': '+79345'}     ->      True
```

- other objects
  - Elipsis

- NotImplemented

In [7]:
```
print(bool(...)) # Ellipsis object is True
print(bool(NotImplemented)) #NotImplemented object to show a function or code block which is
```

```
True
True
C:\Users\Reza\AppData\Local\Temp\ipykernel_6148\1587832201.py:2: DeprecationWarning: NotImp
lemented should not be used in a boolean context
  print(bool(NotImplemented)) #NotImplemented object to show a function or code block which
is not implemented!
```

---

2. False Objects

- False built-in types:
  - None
  - 0
  - 0j
  - 0.0
  - False
  - '' or ""

In [8]:
```
print(None,'\t->\t',bool(None))
print(0,'\t->\t',bool(0)) # int zero
print(complex('0'),'\t->\t', bool(complex('0'))) # zero complex
print(False,'\t->\t',bool(False)) # False bool
print(0.0,'\t->\t',bool(0.0)) # float zero
print(-0.000,'\t->\t',bool(-0.0000)) # float zero
print('""','\t->\t',bool("")) # empty strings
print("''",'\t->\t',bool('')) # empty strings
```

```
None     ->       False
0        ->       False
0j       ->       False
False    ->       False
0.0      ->       False
-0.0     ->       False
""       ->       False
''       ->       False
```

- False built-in data structures:
  - empty list
  - empty tuple
  - empty set
  - empty dict

In [9]:
```
print([],'->',bool([])) # empty list
print((),'->',bool(())) # empty tuple
print(set(),'->',bool(set())) # empty set
print({},'->',bool({})) # empty dictionary
print(range(0),'->',bool(range(0))) # empty range
```

```
[] -> False
() -> False
set() -> False
{} -> False
range(0, 0) -> False
```

---

3. Programmer specifies the Truth of object with __ bool __ method

---

Programmer objects

- You can specify, when the object is True or False with __ bool __ method.

- Example 1:
  - If student's grade was below 10 the student fails and becomes False, If its grade was above or equal to 10 the student passes the test.

In [10]:
```
class Student:
    def __init__(self, name, grade):
        self.name = name
        self.grade = grade

    def __bool__(self):
        if self.grade >= 10:
            return True
        else:
            return False

jim = Student('Jim', 14)
sara = Student('Sara', 7.9)
students = [sara, jim]

for student in students:
    if student: # Now student can be used as a condition of if clause.
        print(student.name, 'passed the test!')
    else:
        print(student.name, "didn't passed the test!")
```

```
Sara didn't passed the test!
Jim passed the test!
```

- Example 2:
  - If cache's size becomes more than five, Its truth value becomes False, otherwise it's True.

In [11]:
```python
class CacheOfFiveElements:
    def __init__(self):
        self.elements = []

    def add(self, element):
        self.elements.append(element)

    # when the object has less than 5 elements the object is True, otherwise its Flase.
    def __bool__(self):
        if len(self.elements) >= 5:
            return False
        else:
            return True

    def __repr__(self):
        return ', '.join(str(x) for x in self.elements)
```

In [12]:
```python
cacheof5 = CacheOfFiveElements()
for element in range(10):
    if cacheof5: # it becomes False if the numbers of elements exceeds 5 ,otherwise it's Tru
        cacheof5.add(element)

print(cacheof5)
```

```
0, 1, 2, 3, 4
```