Tutorial: Python fundamentals concepts for data science

In this tutorial, we introduce python fundamentals concepts for data science

Points to cover in this tutorial:

- Intallation of python
- Variables
- Types
- Operations
- Lists, dictionaries and tuples
- Loops
- Functions

Installation of python

On Windows

1. Go to the Python website python.org/downloads. The site will automatically detect that you are running Windows and the version you are using: it will then display the appropriate Windows installer.
2. Choose the version to install. We recommend the version version that start with 3.x.x(At the moment the latest version is 3.10.1)
3. Click on the button **Download python 3.x.X** to start the download in your local machine.
4. Run the installer once the download is complete. Check the Add Python 3.x.x to PATH box. This will allow you to run Python directly from the command prompt.
5. Once the installation is complete, open the command line and tipe **python** as a command to check if the installation was successful.

On Linux

- Most of Linux distribution have a python version installed by default. To check the actual installed version, open your prompt and type `python --version`.
- To install the lastest version, open the terminal and type the command `sudo apt-get install python`.
- To install the most recent version in Red Hat and Fedora, open the terminal and type `sudo yum install python`.

On Mac

Most of Mac have a default version of python installed. Check the actual installed version(python 2.7). If you don't need a newer version, you don't have to do anything special. Otherwise, you can also install a 3.x.x version.

1. Download the Python 3.x.x files from the Python website python.org/downloads. Your browser should automatically detect the type of OS et redirect you the the python page for Mac but if it is not the case, click the Mac link.
2. Double click the **PKG archive** to install Python. Follow the installation instructions that appear. In the large majority of cases, the default settings are sufficient.
3. In order to verify that the installation went perfectly, open the terminal and type python3. The Python 3.x.x interface should appear and mention the version.

I- Variables

A variable can be understood as a container that holds the elements manipulated in a program. Example:

```
>>>a = 2
>>> a
2
>>> prenom = "Pierre"
>>> prenom
"Pierre"
>>>adresse = "Rue 18, angle 6 Dakar"
>>> adresse
"Rue 18, angle 6 Dakar"
```

II- Types in python

In python, variables are not typed in advance. Their type is defined according to the value they contain. The following table presents some types that we can find in python.

Types	Definition	Exemp
Int(Integer)	Natural number	2,4, 45, 567, 1998, 334445555
Float	Real number. Floating point integer	3.1; 58.7; 45.0906488
Str(String)	It is a sequence of characters delimited either by apostrophe or by double quoting	"Orange", "python", "Data science"
bool(boolean)	Used for boolean values	True, False
List	collection of elements separated by a comma. A list is delimited by square brackets.	[2, 4,"Orange", 'Rue 12", 'ths@af.com", 34.5]

Types	Definition	Exemp
Dict	Collection of elements consisting of key/value. A dictionary is delimited by "braces".	{"fruit": ["orange","Mango"], "age": 23, "succeed":
Tuple	A tuple is delimited by "parentheses". The elements of a tuple are not modifiable unlike those of a list. They are said to be immutable	(2, 4,"Orange", 'Rue 12', 'ths@af.com", 34.5)

The *type()* function is used to check the type of a variable/object. Example:

```
>>>type(2)
int
>>> a = "Rose"
>>> type(a)
str
```

III- Operations in Python

Several operations can be performed on elements of the same or different type.

A- Addition - Rules

```
# int + int = int
>>> 2 +4
6
# int + float = float
>>> 4 + 5.9
9.9
# float + float = float
>>> 8.4 + 1.2 = 9.6
# str + int = error
>>> 'Joel' + 5
error: cannot concatenate string and int
# str + float = error
>>> 'voiture' + 4.6
error: cannot concatenate string and float
# str + str = str
>>> 'machine ' + 'learning'
machine learning
```

Substraction - Rules

```
# int - int = int
>>> 2 -4
-2
# int - float = float
>>> 6 - 5.9
1.9
# float - float = float
>>> 8.4 - 1.2
6.2
# str - int = error
>>> 'Joel' - 5
error: cannot concatenate string and int
# str - float = error
>>> 'voiture' - 4.6
error: cannot concatenate string and float
# str - str = error
>>> 'data science ' - 'machine learning'
error
```

Multiplication- Rules

```
# int * int = int
>>> 2*4
8
# int * float = float
>>> 6 * 2.0
12.0
# float * float = float
>>> 8.0 *  1.2 = 9.6
# str * int = str
>>> 'Joel' * 3
'JoelJoelJoel
# str * float = error
>>> 'voiture' * 4.6
error: cannot concatenate string and float
# str * str = error
>>> 'data science ' * 'machine learning'
error
```

C- Division - Rules

The division operator in Python is `/`. The integer division operator is `//`.

```
# int / int = int or float
>>> 2/4
0.5
# int / float = float
>>> 6 / 5.2
1.1538461538461537
# float / float = float
>>> 8.4 / 1.2
7.000

# int // int = int
>>> 2//4
0
# int // float = int
>>> 6 / 5.2
1
# float // float = float
>>> 8.4 / 1.2
7.0

# str / int = error
>>> 'Joel' / 5
error: cannot concatenate string and int
# str / float = error
>>> 'voiture' / 4.6
error: cannot concatenate string and float
# str / str = error
>>> 'data science ' / 'machine learning'
error
```

D- Comparison operators

They are used to compare two objects. The formula is `obj1 op obj2`. The comparison operators return an array (`True or False`). They can be used to set a condition.

operator in maths	Equivalent in python	signification	Example in python
>	>	greater than	5>2
<	<	less than	3<1.4
$\geq$	>=	greater or equal	2>=2
$\leq$	<=	less or equal	6<=8

operator in maths	Equivalent in python	signification	Example in python
=	=	equal	1= 1; 'learn' = 'er; "3 big" = "3 big"
$\neq$	!=	different	1!= 4; 'learn' != 'learn; "3 big" != "3 big"
$\in$	in	belong	4 in [2,4,9,1]; 'test' in 'this is a smart'
$\notin$	not in	does not belong to	"learning" not in ["machine", "Big data"]; 'machine' not in "machine

E-modulo

The modulo operator in python is "%". It is the remainder of the division of an integer a by an integer b.

```
>>>5%2
1
```

F- Power

The power operator in python is `**`. x to the power y is noted `x**y`. where x and y are either Int or Float. Example:

```
>>> 2**3
8
>>> 4.5**3
91.125
```

G- Incremetation

Incrementation is used to simply some operation like addition, substraction,concatenation,etc. It is apply to int, string, float, and list(concatenation of list). Incrementation is done with the syntax: `op=` where `op= {+,-, /, **, *}` Example:

```
>>> v = 5
>>> v += 3
8
>>> v -= 3
2
>>> v *= 4
20
>>> s = "machine "
```

```
>>> s += "learning"
"machine learnig
```

IV- List, dictionnary and tuple

A- List

A list in python is a collection of elements separated by a comma. The elements of a list are not necessarily of the same type.

```
# initialize a list
a = []
my_list = [1, 1, 3.12, False, "Hi"]
```

Operations on lists

```
# Add a list of elements in a list
[2, 4].extend(["Orange",2.6])
Result : [2,4, "Orange"]


# Concatenate two lists
new_list = [2, 4] + ["Orange"]
Result: [2, 4, "Orange"]

# or simplily
 new_list = [2,4]
 new_list += ["Orange"]
 result: [2, 4, "Orange"]


# add/append a single element in a list
my_list.append('learning')


# List are indexed from 0. indexes are used to get an element at a specific position(list_name[inc
>>>my_list[3]
False
```

```
# Clone elements of a list( slicing)
>>> my_list = [2,4,"root", "name", True, 3, ["Orange","Banana"],"python"]

>>> my_list [:]  # return all the element of the list
```

```
[2,4,"root", "name", True]

>>> my_list[:3]  # return the first 3 elements of a list
[2,4,"root", "name"]
>>> my_list[:-1] # retrurn the last element of the list
"python"

>>> my_list'[-3:] # return the last 3 elements of the list
[3, ["Orange","Banana"],"python"]

>>> my_list'[1:4] # return list element between postion 1 and 4.
["root", "name", True, 3, ["Orange","Banana"]]


# Modify an element at a specific position using index:
>>> my_list[0] = 6 #modify the element at position 0
[0,4,"root", "name", True, 3, ["Orange","Banana"],"python"]
```

B- Dictionnary

A dictionary is a collection of elements separated by commas. Each element is in key/value format. The key allows access to a value.

```
# Initialization of a dictionnary
my_dict = {}
my_dictio = {"Name": "Peter", "Age": 25, "Skills":["python", "Big data", "Machine learning", "data
```

Operations on dictionnary

```
# Get a value of a dictionnary with the key: valeur = dict_name[key]
>>> my_dictio["Age"]
25


# Get all the values of the dictionnary: dict_name.values()
>>>my_dictio.values()
["Peter", 25, ["python", "Big data", "Machine learning", "data science", "Analytics"],'Douala']


# Get all the keys of the dictionnary: dict_name.keys()
>>>my_dictio.keys()
["Name", "Age", "Skills",'city']
```

C- Tuples

A tuple is a sequence of elements separated by commas. A tuple is delimited by `()` parentheses. Unlike lists, the elements of a tuple cannot be modified. We say that tuples are immutable.

```
# Initialization of a tuple
>>>my_tuple = () # Inititialization with empty tuple
>>>the_tuple = (3,"Orange",2.7, False) # Initialization with not empty tuple
>>> t = "orange", 4,True # Initialization without bracket and with values
>>> t
("orange", 4,True)
```

Remark: All the operations that can be performed on the lists except the modification are also performed on the tuples.

IV The structures of control

These statements allow a program or portion of a program to be executed under a condition.

The `If...` statement is used when there is only one option to check

```
>>> if condition:
        Instruction 1
        Instruction 2
           ...
        Instruction n
```

Example :

```
# if x >2 then add 3 to x
if x>2:
   p = x+3
```

Condition `if ... else...` is used when the first condition is not true and we want to handle the other case.

```
if condition:
    Instruction 1
    Instruction 2
       ...
    Instruction n

else:
```

```
    Instruction 1
    Instruction 2
        ...
    Instruction m
```

Example:

```
# If p is greater than 2, then sum x +3 and affect to p else initialize i, and print " x is small'
if x>2:
   p = x+3
else:
    i=0
    while i<5:
       print("x is small")
```

Condition `if...elif` is used when one would like to manage several alternatives.

```
if condition 1:
    intruction1
elif condition 2:
    instruction 2
```

Double-cliquez (ou appuyez sur Entrée) pour modifier

VI- Loop

Loops are used to execute a part or portion of a program several times.

`For` Loop

It is used when you know how many times you must have the operation. Example:

```
A = [4,5,8,7,0,1]
B= []
for i in  A:
      # increment each element of the list A and append the result in the list B
      B.append(i+1)
```

```
Result:  B= [5,6,9,8,1,2]
```

While loop

The `while` loop is used when you don't know how many times the loop will be performed.

```
i = 9
# while i is greater than 5, execute the program
while i >= 5:
      print("the value of i is: ",i-1)

Result:
The value of i is: 8
The value of i is: 7
The value of i is: 6
The value of i is: 5
```

Double-cliquez (ou appuyez sur Entrée) pour modifier

VI- Functions

Functions are used when one wants to isolate a set of instructions that must be performed several times by a program with only different parameters. A function is created with the keyword `def` and its skeleton is presented as follows

```
def Nom_de_la_fonction():
      Instructions
```

A function can take no parameters or take one or more parameters. exemple:

```
# Function without parameters
def Incrementation():
      i=0
      my_list = []
      while i > 5:
          #Add i to the list
          my_list.append(i)
          # i = i +1
          i+=1
```

```
     return my_list
>>>Incrementation()
[0,1,2,3,4]


# Fonction with parameter
def  adding(number):
          i = 5
          i = 5 + number
          return i

>>> adding(3) # Execution of the function
8
```

To execute a function you just have to call the function with its parameters like this:

`function_name(parameters)`

The python language has a set of functions already defined (predefined function). We used some of them previously in this tutorial. Examples:

print() : print result

```
>>> print("I am learning data science")
I am learning data science
>>> print(2)
2
>>> b = 4 + 5
>>> print(b)
9
```

range(n, m, inc): creates an ordered sequence of numbers between n (by default n=0) and m-1 increment by inc(by default inc=1). Where n, and m and inc are float?

```
>>> range(4):
[0,1,2,3]
>>> range(2, 8)
[2,3,4,5,6,7]
# Returns the list of even numbers between 2 and 9
>>> range(2, 10, 2)
[2, 4, 6,8]
```

len(objet): return the number of element of an object(liste, str, tuple, dict)

```
>>> a = [3,23.4, 5]
>>> len(a)
3
```

```
>>> len("machine learning")
16
>>> len((5,90,33, 0))
4
```

sum(objet): Sum all element of the object(list,tuple, array)

```
>>> my_list = [3,4,6,1]
>>> sum(my_list)
14

>>>sum((3,7,9))
19
```

Proposed by *Cedric Lontsi S.*

Email : cedriclontsi@gmail.com

github : https://github.com/cedricnguen