**Institut National des Télécommunications et des Technologies de l'Information et de la Communication INT-TIC d'Oran Algérie**

# Analyse de Données

**3ème année ingénieur**

## Fiche de TP N° 1

## Langage Python : prise en main

Cette introduction à Python aborde, à travers des exemples, les notions de base du langage, à savoir les variables, les types, les opérateurs, l'affichage et la saisie, les chaines de caractères, les structures de contrôle et les fonctions. Les exemples sont présentés ici sous forme de lignes de commandes (à tester avec l'interpréteur python).

### 1. Variables, types (int, float et str) et opérateurs

```
>>> x = 2
>>> x
2
>>> y = 3.14
>>> y
3.14
>>> a = "Hello"
>>> a
'Hello'
>>> b = 'Hello'
>>> b
'Hello'
>>> c = """Hello"""
>>> c
'Hello'
>>> d = '''Hello'''
>>> d
'Hello'
```

Les symboles +, -, *, **, /, // et % sont respectivement les opérateurs d'addition, de soustraction, de multiplication, de puissance, de division réelle et entière et de modulo.

L'opérateur += effectue une addition puis affecte le résultat à la même variable. L'incrémentation en python se fait en utilisant cet opérateur (pas d'opérateur ++). Les opérateurs -=, *= et /= se comportent de manière similaire pour la soustraction, la multiplication et la division.

```
# Opérations sur les chaines de caractères
>>> chaine = "Salut"
>>> chaine
'Salut'
>>> chaine + " Python"
'Salut Python'
>>> chaine * 3
'SalutSalutSalut
```

```
# Opérations incorrectes
>>> chaine * 1.3
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can't multiply sequence by non-int of type 'float'
>>> chaine + 2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can only concatenate str (not "int") to str

# Fonction type(), et conversions avec les fonctions int(), float() et str()
>>> x = 5
>>> type(x)
<class 'int'>
>>> y = 3.0
>>> type(y)
<class 'float'>
>>> z = '2'
>>> type(z)
<class 'str'>
>>> str(x)
'5'
>>> int(z)
2
>>> float(z)
2.0
>>> z = '3.1416'
>>> float(i)
3.1416
```

## 2. Affichage avec la fonction `print`

```
>>> print("Hello world!")
Hello world!
>>> print("Hello world!", end="")
Hello world!>>>

>>> print("Hello") ; print("World")
Hello
World
>>> print("Hello", end="") ; print("World")
HelloWorld

>>> prix = 60000
>>> article = "PC"
>>> print(artcile, ":", prix, "DA")
PC : 60000 DA

# Affichage avec Séparateur
>>> prix = 60000
>>> article = "PC"
>>> print(artcile, ":", prix, "DA", sep="")
PC:60000DA
>>> print(artcile, ":", prix, "DA", sep="-")
PC-:-60000-DA

# Affichage formaté
>>> prix = 60000
>>> article = "PC"
>>> print("{} : {} DA".format(article, prix))
PC : 60000 DA
>>> print("{0} : {1} DA".format(article, prix))
PC : 60000 DA
>>> print("{1} : {0} DA".format(article, prix))
60000 : PC DA
```

```
>>> import math     # Mot clé "import" pour importer un module
>>> pi = math.pi
>>> print("La valeur de PI est", pi)
La valeur de PI est 3.141592653589793

>>> print("La valeur de PI est {:.2f}".format(pi))
La valeur de PI est 3.14
>>> print("La valeur de PI est {:.3f}".format(pi))
La valeur de PI est 3.142

>>> print("PI = {0:.2f}, E(exponentiel) = {1:.2f}".format(pi,math.e))
PI = 3.14, E(exponetiel) = 2.72

# Affichage avec Nombre de Positions
>>> print(10) ; print(1000)
10
1000
>>> print("{:>6d}".format(10)) ; print("{:>6d}".format(1000))
    10
  1000
>>> print("{:<6d}".format(10)) ; print("{:<6d}".format(1000))
10
1000
>>> print("{:^6d}".format(10)) ; print("{:^6d}".format(1000))
  10
 1000
>>> print("{:*^6d}".format(10)) ; print("{:*^6d}".format(1000))
**10**
*1000*
>>> print("{:0>6d}".format(10)) ; print("{:0>6d}".format(1000))
000010
001000

>>> print("Article {:>7s}".format("PC")) ; print("Article \
... {:>7s}".format("CLAVIER"))
Article      PC
Article CLAVIER

>>> print("{:7.3f}".format(math.pi))
  3.142
>>> print("{:10.3f}".format(math.pi))
     3.142
```

Remarque : la contre-barre « \ » et les trois points « … » permettent de continuer l'instruction sur la ligne suivante.

```
# Pour afficher les accolades
>>> print("Accolades : {{}}")
Accolades : {}
```

### 3. Saisie avec la fonction `input`

```
>>> a = float(input('Veuillez entrer un nombre quelconque : '))
... print ('Le carré de', a, 'vaut', a**2)
Veuillez entrer un nombre quelconque : 3
Le carré de 3.0 vaut 9.0
```

### 4. Chaines de caractères

Il est possible de délimiter une chaine de caractères en python en utilisant les apostrophes, les guillemets, ou encore 3 apostrophes.

```
>>> print("Une chaîne avec des 'apostrophes'.")
Une chaîne avec des 'apostrophes'.
```

```
>>> print('Une chaîne avec des "guillemets".')
Une chaîne avec des "guillemets".
>>> print('''Chaîne contenant des 'apostrophes' et des "guillemets". ''')
Chaîne contenant des 'apostrophes' et des "guillemets".
>>> z = '''hello
... world!'''
>>> print(z)
Hello
world!
>>> z
'hello\nworld!'

# Caractères spéciaux
>>> print("Une chaîne\n\tsur plusieurs lignes\n\t\tavec tabulations")
Une chaîne
	sur plusieurs lignes
		avec tabulations
>>> print("Une chaîne avec des \"guillemets\".")
Une chaîne avec des "guillemets".
>>> print('Une chaîne avec des \'apostrophes\'.')
Une chaîne avec des 'apostrophes'.
>>> print('Une chaîne avec trois contre-barres \\\\\\.')
Une chaîne avec trois contre-barres \\\.

# Longueur
>>> x = "Chaîne de caractères"
>>> len(x)
20
```

La comparaison entre chaines de caractères se fait avec les opérateurs ==, !=, >, <, >= et <=.

```
# Accès aux caractères : notation tableau ordinaire
>>> ch = "Informatique"
>>> ch[2]
'f'
>>> print(ch[0], ch[len(ch) - 1])
I e
>>> ch[-1]
'e'
>>> ch[-2]
'u'
>>> ch[-len(ch)]
'I'
>>> ch[12]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: string index out of range
```

## 5. Structures de contrôle

```
# Structure conditionnelle
>>> a = 10
>>> b = 5
>>> if b > a:       # Possibilité d'utiliser les opérateurs booléens and, or et not
...  print("b est plus grand que a")
... elif a == b:
...  print("a et b sont égaux")
... else:
...  print("a est plus grand que b")
...
a est plus grand que b
```

```
# Boucle for
>>> s = "Parcours de chaîne"
>>> for c in s : \
...  print(c, end=" ")
...
P a r c o u r s  d ' u n e  c h a i n e
# Fonction range
>>> for k in range(5): # 5 n'est pas inclus dans la séquence générée
...  print (k, end=' ')
...
0 1 2 3 4
>>> for k in range(5, 10): # avec deux paramètres
...  print (k, end=' ')
...
5 6 7 8 9
>>> for k in range(1, 10, 2): # avec trois paramètres
...  print (k, end=' ')
...
1 3 5 7 9

# Boucle while
>>> i = 0
>>> while i < 5: # 5 n'est pas inclus dans la séquence générée
...  print (2*i, end=' ')
...  i += 1
0 2 4 8
```

## 6. Fonctions

```
# Définition d'une fonction
>>> def compteur (deb,fin,pas) :
...  i = deb
...  while i < fin:
...   print (i, end=' ')
...   i += pas
...
# Appel de la fonction
>>> compteur (1,10,2)
1 3 5 7 9

# Même fonction avec des paramètres par défaut
>>> def compteur (deb=1,fin=10,pas=1) :
...  i = deb
...  while i < fin:
...   print (i, end=' ')
...   i += pas
...
# Appels de la fonction
>>> compteur ()
1 2 3 4 5 6 7 8 9
>>> compteur (20,25)
20 21 22 23 24
>>> compteur (pas=2)
1 3 5 7 9
```