

PYTHON N° 11 / PRÉSENTATION D'UN RÉSULTAT

1 print : une fonction sous-estimée

séparé par un espace et va à la ligne

```
1 print("oui", "non")
2 print("hello")
```

ce qui donne :

oui non
hello

séparé par * et va à la ligne

```
1 print("oui", "non", sep = "*")
2 print("hello")
```

ce qui donne :

oui*non
hello

séparé par * et ne va pas à la ligne

```
1 print("oui", "non", sep = "*", end = "")
2 print("hello")
```

ce qui donne :

oui*nonhello

2 présentation d'1 résultat : écriture formatée via les f-strings

- la fonction print est intéressante pour des affichages simples
- lorsque l'on veut travailler sérieusement (présentation spécifique pour une base de données ou pour faire un rapport à la direction), il faut utiliser les f-strings via format qui sont beaucoup plus puissants et pas si compliqués à manipuler
- voyons quelques exemples
- il est totalement inutile de connaître par coeur l'ensemble de ces exemples ; il faut juste savoir qu'il existe et pouvoir les retrouver rapidement
- nous allons droit au but, vous trouverez plus d'explication là : [page_1](#) et [page_2](#)

2.1 affectation

[affectation avant le f-string pour affichage](#)

```
1 x = 32
2 nom = "John"
3 print(f"{nom} a {x} ans")
```

ce qui donne : John a 32 ans

[affectation dans le f-string pour affichage](#)

```
1 print(f"J'affiche l'entier {10} et le float {3.14}")
2 print(f"J'affiche la chaîne {'Python'}")
```

ce qui donne :

J'affiche l'entier 10 et le float 3.14

J'affiche la chaîne Python

remarque : obligation dans ce cas des guillemets simples si le f-string est définie avec des guillemets doubles (sinon il va y avoir une confusion pour python)

2.2 format de nombre

[pas de format](#)

```
1 prop_GC = (4500 + 2575) / 14800
2 print("La proportion de GC est", prop_GC)
```

ce qui donne : La proportion de GC est 0.4780405405405405

[format float à 2 ou 3 décimales](#)

```
1 print(f"La proportion de GC est {prop_GC:.2f}")
2 print(f"La proportion de GC est {prop_GC:.3f}")
```

ce qui donne :

La proportion de GC est 0.48

La proportion de GC est 0.478

[format decimal integer](#)

```
1 nb_G = 4500
2 nb_C = 2575
3 print(f"Ce génome contient {nb_G:d} G et {nb_C:d} C, soit une prop↵
    de GC de {prop_GC:.2f}")
4
5 perc_GC = prop_GC * 100
6 print(f"Ce génome contient {nb_G:d} G et {nb_C:d} C, soit un %GC ↵
    de {perc_GC:.2f} %")
```

ce qui donne :

Ce génome contient 4500 G et 2575 C, soit une prop de GC de 0.48

Ce génome contient 4500 G et 2575 C, soit un %GC de 47.80 %

format écriture scientifique

```
1 avogadro_number = 6.022_140_76e23
2 print(f"{avogadro_number:.0e}")
3
4 print(f"{avogadro_number:.3e}")
5 print(f"{avogadro_number:.6e}")
```

ce qui donne :

6e+23

6.022141e+23

2.3 format de position

format alignement : important pour la présentation ou la préparation d'une BDD

```
1 >>> print(10) ; print(1000)
2 10
3 1000
4 >>> print(f"{10:>6d}") ; print(f"{1000:>6d}")
5      10
6     1000
7 >>> print(f"{10:<6d}") ; print(f"{1000:<6d}")
8 10
9 1000
10 >>> print(f"{10:^6d}") ; print(f"{1000:^6d}")
11  10
12 1000
13 >>> print(f"{10:*^6d}") ; print(f"{1000:*^6d}")
14 **10**
15 *1000*
16 >>> print(f"{10:0>6d}") ; print(f"{1000:0>6d}")
17 000010
18 001000
```

format alignement : atomes d'une molécule au format PDB

```
1 >>> print("atom HN") ; print("atom HDE1")
2 atom HN
3 atom HDE1
4 >>> print(f"atom {'HN':>4s}") ; print(f"atom {'HDE1':>4s}")
5 atom    HN
6 atom    HDE1
```

2.4 2 astuces

[affichage des accolades dans un f-string](#)

```
1 print(f"Accolades littérales {{ ... }} et pour le formatage {10}")
```

ce qui donne : Accolades littérales { ... } et pour le formatage 10

[accolades vides => erreur](#)

```
1 print(f"accolades sans variable {}")
```

ce qui donne :

File "<stdin>", line 1

SyntaxError: f-string: empty expression not allowed

2.5 expression dans un f-string

[tout l'intérêt du f-string](#)

```
1 print(f"Le résultat de 5 * 5 vaut {5 * 5}")
2 entier = 2
3 print(f"Le type de {entier} est {type(entier)}")
```

ce qui donne :

Le résultat de 5 * 5 vaut 25

Le type de 2 est <class 'int'>

2.6 complément sur les listes : enumerate

[tout l'intérêt du f-string](#)

```
1 my_list = ['apple', 'banana', 'grapes', 'pear']
2 for counter, value in enumerate(my_list):
3     print counter, value
4
5 # Output:
6 # 0 apple
7 # 1 banana
8 # 2 grapes
9 # 3 pear
```

on peut aussi démarrer à un index choisi

2.7 complément sur les listes : zip - unzip

[ex 1:](#)

```
1 first_name = ['Joe', 'Earnst', 'Thomas', 'Martin', 'Charles']
2 last_name = ['Schmoe', 'Ehlmann', 'Fischer', 'Walter', 'Rogan', 'Green'↵
    ]
3 age = [23, 65, 11, 36, 83]
4
5 print(list(zip(first_name, last_name, age)))
6
7 # Output
8 # [('Joe', 'Schmoe', 23), ('Earnst', 'Ehlmann', 65), ('Thomas', '↵
    Fischer', 11), ('Martin', 'Walter', 36), ('Charles', 'Rogan', ↵
    83)]
```

[ex 2:](#)

```
1 first_name = ['Joe', 'Earnst', 'Thomas', 'Martin', 'Charles']
2 last_name = ['Schmoe', 'Ehlmann', 'Fischer', 'Walter', 'Rogan', 'Green'↵
    ]
3 age = [23, 65, 11, 36, 83]
4
5 for first_name, last_name, age in zip(first_name, last_name, age):
6     print(f"{first_name} {last_name} is {age} years old")
7
8 # Output
9 #
10 # Joe Schmoe is 23 years old
11 # Earnst Ehlmann is 65 years old
12 # Thomas Fischer is 11 years old
13 # Martin Walter is 36 years old
14 # Charles Rogan is 83 years old
```

[ex 3:](#)

```
1 full_name_list = [('Joe', 'Schmoe', 23),
2                   ('Earnst', 'Ehlmann', 65),
3                   ('Thomas', 'Fischer', 11),
4                   ('Martin', 'Walter', 36),
5                   ('Charles', 'Rogan', 83)]
6
7 first_name, last_name, age = list(zip(*full_name_list))
8 print(f"first name: {first_name}\nlast name: {last_name} \nage: {↵
    age}")
```

```
9
10 # Output
11
12 # first name: ('Joe', 'Earnst', 'Thomas', 'Martin', 'Charles')
13 # last name: ('Schmoe', 'Ehlmann', 'Fischer', 'Walter', 'Rogan')
14 # age: (23, 65, 11, 36, 83)
```

[ex 4:](#)

```
1 names = ['nik', 'katie', 'james']
2 ages = [32, 31, 34]
3 dictionary = dict(zip(names, ages))
4 print(dictionary)
5 # Returns: {'nik': 32, 'katie': 31, 'james': 34}
```

3 Exercice

Ex 1 : passer en revue l'ensemble des exemples

Ex 2 : écrire sur 5 lignes les nombres de 0 à 49 espacés proprement ; recommencer avec des nombres à virgules (éventuellement arrondis)