La fonction print en Python

Print en Python

La fonction print permet d'afficher le contenu d'un ou plusieurs objet(s) dans la console. La syntaxe de la fonction est print suivi de parenthèses : print (a, b, c). Le message sera transformé en chaîne de caractères peu importe le type des objets passés en paramètres.

Si le contenu d'un objet ne peux pas être affiché de manière lisible pour un humain, print affichera la référence de l'objet.

test.py

```
    print('commentcoder.com')  # commentcoder.com
    print('commentcoder.com', 42, [1, 2, 3])  # commentcoder.com 42 [1,
2, 3]

    def test():
       pass

print(test)  # <function test at 0x142424242>
```

La fonction print permet entre autres :

- D'afficher tout ce qu'on veut en Python
- D'afficher du texte en couleur
- De modifier son formatage avec l'aide d'autres paramètres comme sep et end

Commençons sans plus tarder à explorer toutes les capacités de la fonction print en Python 3 (et Python 2 en fin d'article) !

https://youtu.be/n2aAnQVZOfA?t=53

Comment utiliser la fonction print en Python ?

Pour utiliser la fonction print en Python, on lui passe des paramètres qui seront transformés en strings :

test002.py

```
# Afficher une chaîne de caractères :

print("Bienvenue sur commentcoder.com")
# Affichera "commentcoder.com" sur la sortie standard

# Afficher un nombre :
```

```
print(42)
# Affichera "42" sur la sortie standard

# Afficher le contenu d'une variable :

nombre = 42
print(nombre)
# Affichera "42" sur la sortie standard
```

Il est aussi possible d'afficher plusieurs objets Python avec la fonction print :

test003.py

```
## Afficher plusieurs objets :

print(1, 2, 3)
# Afficher "1 2 3"

# On peut aussi mixer les types d'objets :

print('École', 42)
# Afficher "École 42"

# En utilisant l'unpacking/destructuration

liste = [1, 2, 3, 4, 5]
print(liste[0], *liste[1:-1], liste[-1])


# Affichera 1 2 3 4 5
# Explications :
# liste[0] = 1 (le premier élément de la liste)
# liste[1:-1] = [2, 3, 4] (les éléments en partant du 1er non-inclus,
jusqu'au dernier non-inclus)
# L'opérateur * permet de déstructurer la liste [2, 3, 4] en 3 nombres
2, 3 et 4
# liste[-1] = 5 (le dernier élément de la liste)
```

Comment afficher des variables en plus du texte ?

Il est possible d'afficher le contenu de variables en plus de strings (chaines de caractères) avec la fonction print en Python. La méthode la plus couramment utilisée est probablement celles des chaînes de caractères littérales formatées (ou f-strings) mais il existe d'autres manières d'afficher des variables en plus du texte.

Voyons les 3 plus courantes ensemble.

La concaténation avec print

En Python et avec la fonction print, on peut concaténer (coller des strings ensemble) avec l'opérateur + ou bien en passant plusieurs paramètres à print.

test004.py

```
un = 1
deux = 2
trois = 3

print('Un : ' + str(un) + ' deux : ' + str(deux) + ' trois : ' +
str(trois))

print('Un', un, 'deux :', deux, 'trois :', trois)

##Tous les deux donnent Un 1 deux : 2 trois : 3 sur la sortie standard.
```

Explications :

Quand on utilise l'opérateur +, il faut que tous les éléments soient du même type pour finalement être castés (transformés) en strings.

Dans l'exemple si dessus, j'utilise sont la fonction str() pour transformer nos nombres en objets de type string.

Quand on utilise les virgules, le séparateur par défaut est , , on a donc pas besoin de mettre des espaces comme quand utilise le +.

La méthode format en Python

La méthode format permet d'afficher le contenu de variables directement dans une chaîne de caractères en Python en utilisant les accolades {} pour définir où placer la valeur des objets.

test005.py

```
# On peut passer des variables avec la méthode format
variable_1 = 'a'
variable_2 = 'b'
print('Alef = {}, Bet = {}'.format(variable_1, variable_2))

# On peut aussi directement utiliser des constantes
```

```
print('wahid = {}, ithnan = {}'.format(1, 2))
```

Les f-string en Python

Les chaînes de caractères littérales formatées ou f-string permettent d'afficher des variables en plus de texte avec la fonction print en Python.

test006.py

```
# On peut afficher le contenu d'une variable dans une f-string :
reponse =  42
print(f'Quelle est la réponse au sens de la vie ? {reponse}')
# Quelle est la réponse au sens de la vie ? 42

# On peut afficher autant d'éléments de tous types qu'on veut, comme
des constantes :
print(f'Un : {1}, deux : {2}, trois : {3}')
# Un : 1, deux : 2, trois : 3
```

Le paramètre sep pour la fonction print en Python

Le séparateur par défaut entre les arguments pour la fonction print en Python est un espace : ' '. On peut passer une valeur au paramètre sep à la fonction print() pour remplacer cet espace par n'importe quelle autre chaîne de caractères en Python 3.

Voici quelques exemples d'utilisation du paramètre sep pour la fonction print en Python.

test007.py

```
# Cas de base sans l'utilisation du paramètre sep :
print(1, 2, 3)  # 1 2 3

# Utiliser des virgules pour séparer des variables :
print(1, 2, 3, sep=',')  # 1,2,3
print(1, 2, 3, sep=', ')  # 1, 2, 3

# Enlever l'espace mis par défaut :
print(1, 2, 3, sep='')  # 123
```

Le paramètre end pour la fonction print en Python

La fonction print() fini son affichage par un retour à la ligne (\n) par défaut. Il est possible de modifier ce comportement de base en utilisant paramètre end pour afficher une autre chaîne de caractères à la place.

Voici quelques exemples d'utilisation du paramètre end pour la fonction print en Python.

test008.py

```
# Cas de base sans l'utilisation du paramètre end :
print(1)
print(2)
print(3)
"""
Affichera :
1
2
3
"""

# Enlever le retour à la ligne
print("1", end='')
print("2", end='')
print("3", end='')
# Donnera : 123

# Utiliser un espace en fin de print
print("1", end=' ')
print("2", end=' ')
print("3")
# Donnera : 1 2 3

# Utiliser un espace en fin de print
print("contact", end='@')
print("commentcoder", end='.')
print("com")
# Donnera : contact@commentcoder.com
```

Utiliser sep et end avec la fonction print en Python

On peut aussi utiliser les paramètres sep et end ensemble :

test009.py

```
# Pour afficher une date suivi d'un point d'exclamation et un retour à
la ligne
```

```
print('12', '08', '2023', sep="/", end=" !\n")

# Donnera : 12/08/2023 !
```

Comment changer la couleur de print en Python ?

Il est possible de changer la couleur du texte qu'on affiche dans le terminal avec Python.