

[Source Image](#)

Comment rendre les logs utiles ?

En tant que développeurs, les logs constituent la partie expressive de notre application. Ils permettent de capturer l'activité d'une application.

En programmation informatique, un développeur peut décider d'afficher ou non les logs dans son application qui peut être une aide de traçabilité en cas de erreurs.

Le problème qui se pose maintenant est de savoir quoi afficher et où d'où l'objet de cet article.

Logs dans une fonction unique

Par définition, une fonction reçoit en entrée aucun ou plusieurs paramètres, ensuite effectue des opérations et retourne un résultat.



Dans une fonction simple les informations utiles à afficher dans les logs sont :

- Le(s) paramètre(s) que la fonction reçoit.

- Le résultat après opérations (optionnel utile quand on a une suite d'appel de fonctions).

Par exemple pour une fonction qui ne prend aucun paramètre et retourne le message 'Hello' on aurait en Python :

```
1 def hello():
2     print("hello()::est appelée")
3     return "Hello"
4
```

Supposons que la fonction prenne en paramètre un nom et retourne « Hello nom d'utilisateur », on aurait :

```
1 def hello(nom):
2     print(f"hello()::est appelée avec le paramètre nom='{nom}'")
3     return f"Hello {nom}"
```

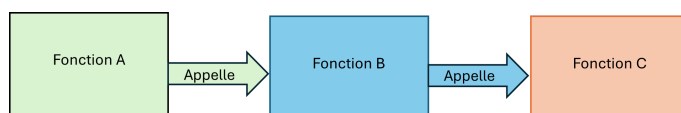
Si on veut afficher le résultat de la fonction dans les logs on aurait :

```
1 def hello(nom):
2     print(f"hello()::est appelée avec le paramètre nom='{nom}'")
3     reponse = f"Hello {nom}"
4     print(f"hello()::s'est exécutée avec le résultat: '{reponse}'")
5     return reponse
```

Quand les paramètres contiennent des informations confidentielles comme l'email et mot de passe, il ne faut pas afficher ces informations dans les logs il faut les remplacer par **.

Logs dans une suite de fonctions

Un programme informatique est une suite de fonctions et méthodes qui s'appellent entre elles. Supposons un programme constitué de 3 fonctions *fonction A*, *fonction B* et *fonction C* avec la pile suivante :



Quand on a plusieurs fonctions les logs doivent se présenter comme suit :

- La fonction appelante doit afficher ses paramètres lors de son appel.
- La fonction appelante après avoir exécuté une autre fonction doit afficher le résultat obtenu.
- La fonction appelante peut afficher le résultat renvoyé.

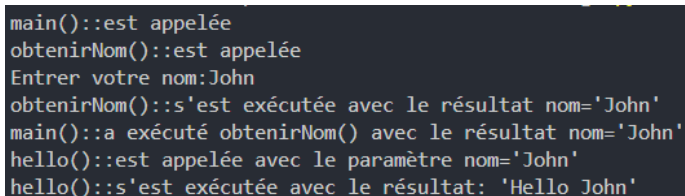
Soit une fonction main qui appelle une fonction obtenirNom et avec le nom saisi par l'utilisateur appelle une fonction hello pour afficher le message 'Hello nom d'utilisateur'.

Si on veut appliquer les logs dans cette suite de fonctions on aurait :



```
1 def main():
2     print("main():est appelée")
3     nom = obtenirNom()
4     print(f"main():a exécuté obtenirNom() avec le résultat nom='{nom}'")
5     hello(nom)
6
```

L'exécution donnerait :



```
main():est appelée
obtenirNom():est appelée
Entrer votre nom:John
obtenirNom():s'est exécutée avec le résultat nom='John'
main():a exécuté obtenirNom() avec le résultat nom='John'
hello():est appelée avec le paramètre nom='John'
hello():s'est exécutée avec le résultat: 'Hello John'
```

Dans les exemples précédents nous avons utilisé la fonction print pour afficher les logs. Dans vos projets il faut préférer des modules de logs de votre langage. En Python, vous pouvez utiliser [Logging](#).

Les techniques décrites plus haut reste adaptable selon vos besoins.

Quand les paramètres contiennent des informations confidentielles comme l'email et mot de passe, il ne faut pas afficher ces informations dans les logs il faut les remplacer par **.