CM 2 - Bases (suite)

Programmation Python – Master 1 MAS

Romain Tavenard

2026

romain.tavenard@univ-rennes2.fr

UNIVERSITÉ RENNES 2

COMPLÉMENTS SUR LES BOUCLES & FONCTIONS

1

Il est possible de “perturber” le comportement d’une boucle (**for** ou **while**) à l’aide d’une des instructions **break** ou **continue** :

```
for i in range(10):
  if i == 5:
    break
  print(i)

for i in range(10):
  if i == 5:
    continue
  print(i)
```

Contexte :

- On doit coder deux fonctions
- L'une des deux est un cas particulier de l'autre

```
def fonction_generale(truc, machin, chose):
  # Ici le corps de la fonction
  return quelque_chose

def fonction_cas_particulier(truc)
  return fonction_generale(truc=truc, machin=32, chose="abc")
```

EXERCICE

1. *Écrivez une fonction qui prend en entrée une chaîne de caractères* `chaine` *et un entier* `n` *et affiche les mots de la chaîne* `chaine` *les uns après les autres.*
   Attention : si un mot de longueur exactement égale à `n` *est rencontré, il faudra afficher ce mot puis s'arrêter (et donc ne pas afficher les mots suivants).*
2. *En utilisant la fonction précédemment codée, écrivez une fonction qui prend en entrée une chaîne de caractères* `chaine` *et affiche les mots de cette chaîne les uns après les autres.*
   Attention : si un mot de longueur exactement égale à **5** *est rencontré, il faudra afficher ce mot puis s'arrêter (et donc ne pas afficher les mots suivants).*

UNIVERSITÉ RENNES 2

LES MODULES EN PYTHON

2

IMPORTER UN MODULE

Variantes de syntaxe pour l'import :

```
import mon_module
[...]
print(mon_module.machin)

import mon_module as truc  # Ici, on renomme
[...]
print(truc.machin)

from mon_module import machin  # Ici, on n'importe que `machin`
[...]
print(machin)
```

UN MODULE, C'EST QUOI ?

UN MODULE, C'EST QUOI ?

- Un (ensemble de) fichier(s) Python
- Stocké où ?

Un module, c'est quoi ?

- Un (ensemble de) fichier(s) Python
- Stocké où ?
  - Dans un dossier qui stocke tous les modules installés (`pip install ...`)
  - Dans le dossier courant (fichier `mon_module.py` $\Rightarrow$ module `mon_module`)
- Que se passe-t-il lors de l'import ?

Un module, c'est quoi ?

- Un (ensemble de) fichier(s) Python
- Stocké où ?
  - Dans un dossier qui stocke tous les modules installés (`pip install ...`)
  - Dans le dossier courant (fichier `mon_module.py` ⇒ module `mon_module`)
- Que se passe-t-il lors de l'import ?
  - Le fichier correspondant au module est exécuté (presque équivalent à *"Run Python File"* dans VS Code)

EXERCICE

```python
import math

print(math.pi)
```

“Qu’est-ce qui pourrait selon vous expliquer que, en exécutant le fichier “cm2.py” ci-dessus, on obtienne l’erreur :

```
Traceback (most recent call last):
  File "cm2.py", line 3, in <module>
    print(math.pi)
AttributeError: module 'math' has no attribute 'pi'
```

”

Structurer son fichier

`if __name__ == "__main__"`

Lors de l'import d'un module, tout le code au niveau du module est exécuté.

```python
def ma_fonction(x):
    return x * 2

print(ma_fonction(5))  # Exécuté à l'import ! (indésirable)
```

STRUCTURER SON FICHIER

if __name__ == "__main__"

Lors de l'import d'un module, tout le code au niveau du module est exécuté.

```python
def ma_fonction(x):
    return x * 2

print(ma_fonction(5))  # Exécuté à l'import ! (indésirable)
```

Solution : protéger les appels avec `if __name__ == "__main__"` :

```python
def ma_fonction(x):
    return x * 2

if __name__ == "__main__":
    print(ma_fonction(5))  # Exécuté seulement si lancé directement
```

Structurer son fichier

`if __name__ == "__main__"`

- Quand on *lance* le fichier : `__name__` vaut `"__main__"`
- Quand on *importe* le fichier : `__name__` vaut le nom du module

⇒ **Bonne pratique :** toujours placer les appels/tests dans ce bloc.