Python - Modules et packages - Généralités

[Mise à jour le :2/8/2021]

- **Sources**
  - **Documentation** sur Python.org : référence du langage, modules, packages, fonctions natives (built-in)
- **Mots-clés** : module, importation, package.

> Les mots ci-dessous sont dits “réservés”. Ils ne peuvent pas être utilisés comme nom de variable. Les mots soulignés sont une nouveauté de Python 3. Les mots en **gras** sont utilisés dans cette page.

and	continue	finally	is	raise
as	def	for	lambda	return
assert	del	**from**	None	True
async	elif	global	nonlocal	try
await	else	if	not	while
break	except	**import**	or	with
class	False	in	pass	yield

- Fonctions natives **(built-in)**[1] utilisées dans les exemples : **dir**(), **help**().

---

1. Les modules

> Un module est une **liste de fonctions et de variables** contenues dans un **fichier**. Pour travailler avec les fonctionnalités du module, il suffit de l'**importer**.

1.1 Utilisation

1.1.1 Méthode 1 : import

> L'importation d'un module se fait à l'aide du mot-clé **import**.

Exemples : importation des modules **math** et **random**

*.py

```
import math # avec cette déclaration, il faut préfixer les méthodes avec le nom du module (ici math)
a= math.sqrt(16) # la fonction utilisée est préfixée par le nom du module
print(a) # renvoie 4.0 dans la console
```

*.py

```
import random
print(random.randint(1,10)) # exemple de résultat : 3
```

1.1.2 Méthode 2 : from … import

> On peut éviter de préfixer les fonctions avec le nom du module ou limiter le nombre d'éléments importés en utilisant ***from … import*** à condition de ne pas importer des modules possédant des méthodes ayant le même nom.

Exemple 1

*.py

```
from math import * # on importe le module math, toutes les méthodes et variables sont accessibles
                   # sans se référer à math
a = sqrt(16)
print(a) # renvoie 4.0 dans la console
```

Exemple 2

```
from math import fabs, e, pi # seules les fonctions valeur absolue, e et pi sont importés

from random import * # on importe la bibliothèque complète
print(randint(1,10)) # exemple de résultat : 1

from datetime import datetime # On importe une seule fonction : datetime
datetime.now() # exemple de résultat : datetime.datetime(2019, 1, 3, 20, 6, 18, 490635)
```

> **dir**(*nom_module*) affiche la liste des fonctions.
> **help(“***nom_module***”)** donne une description de ces fonctions

Exemple

```
>>> import math
>>> dir(math)
['__doc__', '__loader__', '__name__',
, 'asin', 'asinh', 'atan', 'atan2', '
'degrees', 'e', 'erf', 'erfc', 'exp',
od', 'frexp', 'fsum', 'gamma', 'gcd',
```

1.1.3 Espace de noms

Lorsqu'on écrit *import math*, cela crée un **espace de noms** dénommé *math*, contenant les variables et les fonctions du module *math*. Lorsqu'on écrit *math.sqrt(25)*, on précise à l'interpréteur que l'on souhaite exécuter la fonction *sqrt* située dans l'espace de nom *math*. Il est ainsi possible d'utiliser une autre fonction sqrt située dans un autre module ou dans créer une dans l'espace de nom principal.

> Il est possible de modifier l'intitulé d'un espace de nom avec **as**.

Exemple

```
import math as calcul
a=calcul.sqrt(25)
print(a) # renvoie 5.0 dans la console
```

1.1.4 Quelques modules

- **Math** contient les fonctions mathématiques de base : sqrt (racine carrée), cos, sin, exp, etc. ainsi que quelques constantes comme Pi ou e.
- Matplotlib permet de tracer des graphes
- Numpy permet de manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux.
- Pandas est une bibliothèque écrite pour le langage de programmation Python permettant la manipulation et l'analyse des données. Elle propose en particulier des structures de données et des opérations de manipulation de tableaux numériques et de séries temporelles. Wikipédia
- Pillow contient des outils pour afficher des images numériques, appliquer des filtres, retoucher, enregistrer etc.
- **Time** permet de gérer le temps dans l'exécution d'un programme, en particulier la fonction *sleep(n)* stoppe l'exécution pendant n secondes.
- Turtle commande une tortue pour tracer des segments.
- **Random** génère des valeurs aléatoires.
- Scikit-learn est une bibliothèque destinée à l'apprentissage automatique. Wikipédia
- WxPython est une implémentation libre en Python de l'interface de programmation wxWidgets. Cette bibliothèque Python est utilisée pour la création d'interfaces graphiques, et est l'alternative de Tkinter la plus utilisée. Wikipédia.

> **ATTENTION**
> Si l'importation du module ne fonctionne pas, c'est certainement qu'il n'est pas installé. L'installation d'un module passe par un gestionnaire de paquets comme **pip** ou **conda**.

Voir le site sur lequel il se trouve pour la procédure.

Un module installé sous **VSCode** dans un terminal PowerShell avec la commande :

*.ps

```
python -m  pip install nomModule ou
pip install nomModule
```

est disponible **après son redémarrage**.

1.2 Création

- **Source** : les modules sur docs.python.org

Exemple

- **Création** du fichier *fibo.py* contenant les fonctions ci-dessous.

fibo.py

```
# Fibonacci numbers module

def fib(n):    # write Fibonacci series up to n
    a, b = 0, 1
    while a < n:
        print(a, end=' ')
        a, b = b, a+b
    print()

def fib2(n):   # return Fibonacci series up to n
    result = []
    a, b = 0, 1
    while a < n:
        result.append(a)
        a, b = b, a+b
    return result
```

- **Première forme** de l'appel de la fonction *fib* à partir du fichier *test1.py*

test1.py

```
import fibo # Importation de la totalité des fonctions

fibo.fib(50) # résultat : 0 1 1 2 3 5 8 13 21 34
```

- **Deuxième forme** de l'appel de la fonction *fib* à partir du fichier *test2.py*

test2.py

```
from fibo import fib # importation de la seule fonction fib

fib(50) # résultat : 0 1 1 2 3 5 8 13 21 34
```

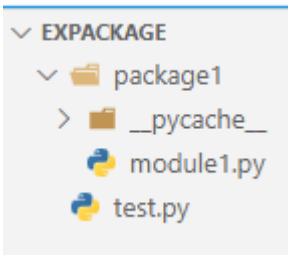
2. Les packages

- **Source** : les paquets sur docs.python.org

Lorsqu'on dispose d'un grand nombre de modules, il peut être intéressant de les organiser dans des répertoires. Un répertoire rassemblant des **modules** est appelé package.

> Un **package** sert à regrouper plusieurs modules. En pratique les packages sont des **répertoires**. Le nom du package est le nom du répertoire.

Exemple : le fichier module1.py contenant les fonctions carre et cube est placé dans le répertoire **package1**.



module1.py

```
def carre(a)
    return(a*a)

def cube(a)
    return(a*a*a)
```

Les fonctions carre et cube peuvent être utilisées en important package1.module1

test.py

```
from package1.module1 import carre

x=5
print(f"Le carré de {x} vaut {carre(x)}") # Le carré de 5 vaut 25
```

3. Fichier .py, script ou module?

En Python, lorsqu'on crée un fichier qui rassemble des instructions, on appelle ce fichier un **script** et on le sauvegarde avec une extension **.py**, par exemple *test.py*.

Si le fichier **.py** contient uniquement des définitions de fonction on l'appelle un **module**.

> Un même **fichier .py** peut jouer le rôle de **script** ou de **module** suivant son utilisation.

- **La variable global __name__**

> A l'intérieur d'un module, le nom du module (en tant que chaîne de caractères) peut être obtenu grâce à la **variable globale __name__**.

Quand on utilise un fichier en tant que script, la variable globale __name__ prend pour valeur '**__main__**'. Ceci permet d'avoir dans le fichier **un bloc d'instructions** qui sera exécuté uniquement lorsque le fichier est lancé en tant que script.

- **Exemple 1 - Exécution du module *module1* en tant que script**

module1.py

```
def carre(a)
    return(a*a)

def cube(a)
    return(a*a*a)

if __name__ == "__main__":                    # Si module1 est appelé
en tant que script :
    x = 3                                     # ces deux lignes sont
exécutées
    print(f"Le carré de {x} vaut {carre(x)}") # Résultat : Le carré de
3 vaut 9
```

- **Exemple 2 - Exécution du module *module1* en tant que module**

test.py

```
# Le module est appelé dans le fichier test.py
from package1.module1 import carre

x=5                                           # Le code sous if __name__ ==
```

```
"__main__" n'est pas exécuté
print(f"Le carré de {x} vaut {carre(x)}") # Résultat : Le carré de 5 vaut 25
```

> **Télécharger** le répertoire du projet **VSCode** ici

Résumé

- On écrit les programmes Python dans des fichiers portant l'extension **.py**
- On peut créer des fichiers contenant des **modules** pour séparer le code.
- On peut créer des répertoires contenant des **packages** pour hiérarchiser un programme.

Pour aller plus loin ...

- Python import: Advanced Techniques and Tips
- How to Publish an Open-Source Python Package to PyPI
- Python Packages: Five Real Python Favorites

1)
Fonctions toujours disponibles.