python™

Modules et Packages

Samuel SORIN
contact@samuelsorin.fr

Les modules

Les modules - introduction

Un module est un fichier contenant des définitions et des instructions

Le nom de fichier est le nom du module suffixé de .py

Un module et ses définitions peuvent être importées dans un autre module

On peut récupérer la définition d'un module (son nom) avec la variable __name__

Convention de nommage : noms courts, lettres minuscules, pas d'espace (underscore à la place)

exemple : mon_module.py

Les modules - Importer des fonctions externes

Les import se positionne en début de module

Utiliser l'instruction import pour utiliser des modules externe

Importer des fonctions en particulier : *from mon_module import fonction1, fonction2*

[déconseillé] Importer toutes les fonctions d'un module *from mon_module import **

Utiliser un alias pour renommer la fonction importée : *from module import fonction1 as truc*

Les modules - Importer des fonctions externes

```
>>> import math
math.sqrt(4)
2

>>> from math import sqrt
>>> sqrt(4)
2


>>> from math import *
>>> sqrt(4)
2


>>> from math import sqrt as racine_carre
>>> fibonacci(4)
2
```

Les modules - python path

Liste des dossiers qui sont pris en compte pour l'import de modules
Les dossiers par défaut sont,
- le dossier de la librairie python en cours d'exécution
- le dossier du script en cours d'utilisation

```
>>> import sys
>>> sys.path
[
'/home/ssorin/Bureau/python',
'/usr/lib/python3.6',
'/usr/lib/python3.6/lib-dynload',
'/usr/lib/python3.6/dist-packages',
'/usr/lib/python3.6/site-packages',
]
```

Les modules - modules standard

random : fonctions permettant de travailler avec des valeurs aléatoires
math : toutes les fonctions utiles pour les opérations mathématiques *(cosinus,sinus,exp,etc.)*
sys : fonctions systèmes
os : fonctions permettant d'interagir avec le système d'exploitation
time : fonctions permettant de travailler avec le temps
calendar : fonctions de calendrier
profile : fonctions permettant d'analyser l'exécution des fonctions
urllib2 : fonctions permettant de récupérer des informations sur internet
re : fonctions permettant de travailler sur des expressions régulières

Les modules - fonction dir()

utilisée pour trouver quels noms sont définis par un module.

Sans arguments, donne la liste des noms dans l'espace de noms local.

```
>>> dir(puissance)
['__annotations__', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__',
'puissance', 'sys']
```

Avec un argument, elle essaye de donner une liste d'attributs valides pour cet objet.

```
>>> dir(puissance)
['__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', 'carre', 'cube', 'pi',
'pi_carre', 'racine_carre', 'sqrt']
```

Exercices - modules

1 - Créer un module appeler puissance.py comprenant les fonctions :

- carre() : calcul le carré du chiffre donné en paramètre
- cube() : calcul le cube du chiffre donné en paramètre
- pi_carre : calcul le carré du nombre pi
- racine_carree : calcul la racine carré de la valeur donnée

Note : pour les fonctions a et b, utiliser les modules ‘pi’ et ‘sqrt’ fournis par la librairie “math” de python

2 - Créer un nouveau module et utiliser le module puissance.py que vous venez de créer

Les packages

Les packages - introduction

Les packages permettent de regrouper des modules ou des paquages.

- diminue le risque de conflits de noms de fonctions
- permet de bien ordonner un projet

un package est un dossier avec des modules python.
Avant la version 3.6 de Python, il était obligatoire d'avoir un fichier nommé __init__.py dans le package.

Convention de nommage : noms courts, lettres minuscules, pas d'espace (underscore à la place) - *exemple : mon_package*

Les packages - __init__.py

Les fichiers __init__.py sont nécessaires pour que Python considère un dossier contenant ce fichier comme un package *(plus obligatoire depuis 3.6 mais fortement conseillé)*.

Les fichiers __init__.py peuvent être vide.

Le fichiers __init__ est automatiquement exécuté, une seule fois, lors de l'import du package (avant tous les autres modules / fonctions …)

Sert généralement à décrire globalement le projet, sa version, son auteur ...

Les packages - __init__.py

Exemple de __init__.py

```
"""
    Package de demo pour les cours Python à l'ISFAC
"""
__version__ = "1.2.1"
__author__ = "ssorin"
```

+d'info : https://sametmax2.com/vous-pouvez-mettre-du-code-dans-__init__-py/

Exercices - packages

3 - Reprendre le module puissance.py et le mettre dans un package dédié

4 - Ajouter un module init au package et y indiquer le nom de l'auteur et la version de l'app

5 - refaire l'exercice des modules en important le module puissance.py nouvellement créé

6 - Afficher le nom de l'auteur et la version de l'app