Modules du jour-12

Qu'est-ce qu'un module

Un module est un fichier contenant un ensemble de codes ou un ensemble de fonctions qui peuvent être incluses dans une application. Un module peut être un fichier contenant une s eule variable, une fonction ou une grande base de code.

Création d'un module

Pour créer un module, nous écrivons nos codes dans un script Python et nous l'enregistrons en tant que fichier .py. Créez un fichier nommé mymodule.py dans votre dossier de projet. Écriv ons du code dans ce fichier.

```
# MyModule.py File Def Generate_full_name (FirstName, LastNa
me):
return FirstName + " + LastName
```

Créez un fichier main.py dans votre répertoire de projet et importez le fichier mymodule.py.

Importation d'un module

Pour importer le fichier, nous utilisons le mot-clé *import* et le nom du fichier uniquement.

```
# Main.py Fichier Import Mymodule Print (mymodule.generate_full_name ('Asabeneh', 'y
etayeh')) # asabeneh yetayeh
```

Les fonctions d'importation à partir d'un module

Nous pouvons avoir de nombreuses fonctions dans un fichier et nous pouvons importer toutes les fonctions dif

```
# fichier main.py de MyModule Import Generate_full_name, sum_two_nums, personne, gravit
é print (generate_full_name ('asabneh', 'encoreayeh')) print (sum_two_nums (1,9)) mass = 100;
poids = masse * print de gravité (poids) imprimer (personne ['premier nom']))
```

Import Functions from a Module and Renaming

Pendant l'importation, nous pouvons renommer le nom du module.

```
# fichier main.py de MyModule Import Generate_full_name comme Fullname,
sum_two_nums as total, personne comme p, gravity as g print (fullname ('asabn
eh', 'yetayeh')) print (total (1, 9)) mass = 100;  poids = masse * g imprimer (poid
s) print (p) print (p ['premier nom']))
```

Importer des modules intégrés

Comme d'autres langages de programmation, nous pouvons également importer des modules en important le fichier / fonction à l'aide du mot clé *import*. Importons le module commun qu e nous utiliserons la plupart du temps. Certains des modules intégrés communs: *math*, *datetime*, *os*, *sys*, *random*, *statistics*, *collections*, *json*, *re*

Module OS

En utilisant le module Python *os*, il est possible d'effectuer automatiquement de nombreuse s tâches de système d'exploitation. Le module OS dans Python fournit des fonctions pour cr éer, modifier le répertoire de travail actuel et supprimer un répertoire (dossier), récupérer s on contenu, modifier et identifier le répertoire actuel.

```
# Importer le module Import OS # Création d'un réper
toire os.mkdir ('Directory_name') # Modification du ré
pertoire actuel os.chdir ('path') # Obtenir le répertoire
de travail actuel os.getcwd () # Suppression du réperto
ire OS.rmdir ()
```

Module sys

Le module SYS fournit des fonctions et des variables utilisées pour manipuler différentes part ies de l'environnement d'exécution Python. Fonction sys.argv renvoie une liste des arguments de ligne de commande transmis à un script python. L'élément à l'index 0 dans cette liste est to ujours le nom du script, à l'index 1 est l'argument passé à partir de la ligne de commande.

Exemple d'un fichier script.py:

```
import sys
#print(sys.argv[0], argv[1],sys.argv[2])  # this line would
print out: filename argument1 argument2
```

```
print ('bienvenue {}. Profitez du défi {}!'. Format (sys.argv [1], sys.argv [2]))
```

Maintenant, pour vérifier comment ce script fonctionne, j'ai écrit dans la ligne de commande:

```
python script.py asabeneh 30daysofpython
```

Le résultat:

```
Bienvenue Asabeneh. Profitez d'un défi de 30 jours de fin de forme!
```

Quelques commandes SYS utiles:

```
# Pour quitter sys sys.exit () # pour connaître la plus grande variable enti
ère, il faut sys.maxsize # pour connaître l'environnement Path Sys.Path #
pour connaître la version de Python que vous utilisez SYS.Version
```

Module statistique

Le module statistique fournit des fonctions pour les statistiques mathématiques des données n umériques. Les fonctions statistiques populaires qui sont définies dans ce

module: *mean*, *median*, *mode*, *stdev* etc.

```
from statistics import * # importing all the statistics
modules
ages = [20, 20, 4, 24, 25, 22, 26, 20, 23, 22, 26]
print(mean(ages))       # ~22.9
print(median(ages))     # 23
print(mode(ages))       # 20
print(stdev(ages))      # ~2.3
```

Module mathématique

Module contenant de nombreuses opérations et constantes mathématiques.

```
import math
print(math.pi)           # 3.141592653589793, pi constant
print(math.sqrt(2))      # 1.4142135623730951, square root
print(math.pow(2, 3))    # 8.0, exponential function
print(math.floor(9.81))  # 9, rounding to the lowest
print(math.ceil(9.81))   # 10, rounding to the highest
print(math.log10(100))   # 2, logarithm with 10 as base
```

Maintenant, nous avons importé le module *math* qui contient beaucoup de fonction qui peut n ous aider à effectuer des calculs mathématiques. Pour vérifier les fonctions du module, nous p ouvons utiliser *help(math)*, ou *dir(math)*. Cela affichera les fonctions disponibles dans le mo dule. Si nous voulons importer uniquement une fonction spécifique du module, nous l'importo ns comme suit:

```
from math import pi
print(pi)
```

It moi

s également possible d'importer plusieurs fonctions à la fois

```
from math import pi, sqrt, pow, floor, ceil, log10
print(pi)                  # 3.141592653589793
print(sqrt(2))             # 1.4142135623730951
print(pow(2, 3))           # 8.0
print(floor(9.81))         # 9
print(ceil(9.81))          # 10
print(math.log10(100))     # 2
```

Mais si nous voulons importer toute la fonction dans le module mathématique, nous pouvons utiliser *.

```
from math import *
print(pi)                   # 3.141592653589793, pi constant
print(sqrt(2))              # 1.4142135623730951, square root
print(pow(2, 3))            # 8.0, exponential
print(floor(9.81))          # 9, rounding to the lowest
print(ceil(9.81))           # 10, rounding to the highest
print(math.log10(100))      # 2
```

Lorsque nous importons, nous pouvons également renommer le nom de la fonction.

```
de Math Import Pi as Pi Print (PI) # 3.141592
653589793
```

Module de chaîne

Un module de chaîne est un module utile à de nombreuses fins. L'exemple ci-dessous mont re une utilisation du module de chaîne.

```
Import String print (string.ascii_letters) # ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEF
GHIJKLMNOPQRSTUVWXYZ PRINT (String.Digits) # 0123456789 PRINT (String.Punctua
tion) # ! "# $% & '() * +, -. /:; <=>? @ [\] ^_ `{ | } ~
```

Module aléatoire

Vous connaissez maintenant les modules d'importation. Faisons une autre importation pour la familiariser avec elle. Importons le module *random* qui nous donne un nombre aléatoire entre 0 et 0,9999 .... Le module *random* a beaucoup de fonctions mais dans cette section, nous n'util iserons que *random* et *randint*.

```
à partir d'importation aléatoire aléatoire, RANDINT PRINT (random ()) # Il ne prend aucun a rgument; Il renvoie une valeur comprise entre 0 et 0,9999 imprimé (Rannt (5, 20)) # Il renvoi e un numéro entier aléatoire entre [5, 20] inclus
```

Vous allez loin. Continue! Vous venez de terminer les défis du jour 12 et vous êtes à 12 é tapes de la grandeur à la grandeur. Faites maintenant quelques exercices pour votre cerveau et vos muscles.

Exercices: Jour 12

Exercices: niveau 1

1. Écrivez une fonction qui génère un six chiffres / caractère Random_User_ID.

```
print (random_user_id ());
'1EEE33D'
```

2. Modifiez la tâche précédente. Déclarez une fonction nommée User_ID_GEN_BY_US ER. Il ne prend aucun paramètre mais il faut deux entrées à l'aide de l'entrée (). L'une des entrées est le nombre de caractères et la deuxième entrée est le nombre d'ID qui sont cens és être générés.

```
print (user_id_gen_by_user ()) # entrée utilisateur: 5 5 #Output: # kcsy
2 #smfyb #bwmeq #zxoyh # 2rgxf




print (user_id_gen_by_user ()) # 16 5 # 1gcsgplmabav
qz26 # yd7efwnqkns7qxat # ycarc5yrrupyg00s # ubgx
ofi7uxswaykn # div0sSutgadkwstr
```

3. Écrivez une fonction nommée RGB_COLOR_GEN. Il générera des couleurs RVB (3 valeurs allant de 0 à 255 chacune).

```
print (rgb_color_gen ()) # RGB (125 244.255) - La sortie doit être sous cette forme
```

Exercices: niveau 2

1. Écrivez une fonction list_of_hexa_colors qui renvoie n'importe quel nombre de couleurs hexadécimales dans un tableau (six nombres hexadécimaux écrits après #. Le système de ch iffres hexadécimaux est composé de 16 symboles, 0-9 et 6 premières lettres de l'alphabet, A -F. Vérifiez la tâche 6 pour les exemples de sortie).

2. Écrivez une fonction list_of_rgb_colors qui renvoie n'importe quel nombre de couleurs RVB dans un tableau.

3. Écrivez une fonction Generate_Colors qui peut générer n'importe quel nombre de coul eurs hexa ou RVB.

```
Generate_Colors ('Hexa', 3) # ['# a3e12f', '# 03ed55', '# eb3d2b'] g
enerate_colors ('hexa', 1) # ['# b334ef']
```

```
Generate_Colors ('RGB', 3) # ['RGB (5, 55, 175', 'RGB (50, 105, 100', 'RGB (15, 26, 80']
generate_colors ('rgb', 1) # ['rgb (33,79, 176)']
```

Exercices: niveau 3

1. Appelez votre fonction shuffle_list, il prend une liste comme paramètre et il renvoie une liste mélangée

2. Écrivez une fonction qui renvoie un tableau de sept nombres aléatoires dans une plage de 0-9. Tous les chiffres doivent être uniques.

Félicitations!