Spécialité NSI

LYCÉE de l'ÉLORN
le choix de votre réussite

Cours -TD
Programmation en Python
La modularité

1- Rappel sur les fonctions

On crée des fonctions pour :

- Une meilleure organisation du programme (regrouper les tâches par blocs, lisibilité, maintenance)
- Eviter la redondance (pas de copier/coller, meilleure réutilisation du code)
- Rendre possible le partage des fonctions (via des modules)

Exemple :

```
def le_plus_petit(a, b):  # définition de la fonction
    if (a < b):
        résultat = a
    else:
        résultat = b
    return résultat     # le return renvoie la valeur et fait quitter immédiatement la fonction


# appels possibles de la fonction
print(le_plus_petit(10, 5))   # passage des paramètres par position
print(le_plus_petit(5, 10))
print(le_plus_petit(a=10, b=5))  # passage par les valeurs
print(le_plus_petit(b=10, a=5))
```

2- Les modules

2.1 Définition et modes d'importations

Un module est un fichier « .py » contenant un ensemble de variables, fonctions et classes que l'on peut importer et utiliser dans le programme principal (ou dans d'autres modules). Le mot clé **import** permet d'importer un module.

Exemples d'importations avec un module standard

```
import math            #on importe toutes les fonctions du module math
resultat=math.sqrt(2)  #le nom du module est obligatoire avec ce mode d'importation
print(resultat)
print(math.pi)


from math import sqrt  #on importe seulement la fonction sqrt() du module math
resultat=sqrt(2)       #le nom du module n'est plus obligatoire si from est utilisé
print(resultat)
print(pi)              #générera une erreur car pi n'est pas importé


from math import *     #on importe toutes les fonctions du module math
resultat=sqrt(2)
print(resultat)
print(pi)
```

2.2 Le chemin des importations

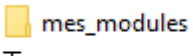
Python cherche automatiquement le(s) module(s) dans le « search path » c'est-à-dire :

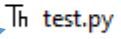
- le dossier courant (celui qui contient le script)
- les dossiers listés dans la variable d'environnement PYTHONPATH (configurable sous Windows)
- les dossiers automatiquement spécifiés à l'installation. On peut obtenir la liste avec la commande *sys.path* (il faut importer le module sys au préalable)

On peut définir un chemin particulier, notamment pour les modules personnels

Imaginons que le fichier « *tests_grandeurs.py* » contienne la fonction « *le_plus_petit()* » présentée au chapitre 1. Nous souhaitons l'utiliser comme un module. Si ce fichier est dans le dossier « *mes_modules* » l'importation sera :

Dossier contenant le fichier « tests_grandeurs.py » où est défini la fonction « le_plus_petit() »





Fichier utilisant le module personnel tests_grandeurs

Contenu du fichier test.py qui utilise le module tests_grandeurs :

```
import mes_modules.tests_grandeurs
print(mes_modules.tests_grandeurs.le_plus_petit(5,10))  #le chemin doit être mentionné avec la fonction
```

2.3 Les Alias définis pendant l'importation

L'alias permet d'utiliser des noms plus courts dans un programme.

```
#on peut importer plusieurs modules en les séparant par des virgules
import math as m, random as r   # m remplace le nom du module math, r celui du module random
resultat=m.sqrt(2)
nombre=r.random()
print(resultat, nombre)
```

2.4 Lister les fonctions contenues dans un module

dir() et help() permettent d'obtenir des informations sur un module.

```
import math
print(dir(math))   #ne fonctionne pas avec un from
print(help(math))
```

3- Travail demandé

En reprenant l'arborescence de l'exemple du chapitre 2.2 sur votre espace personnel, créer un module contenant une fonction capable de renvoyer la racine carrée du plus grand des deux nombres qui lui sont envoyés. Mettre en œuvre un Alias.

Exemple : racine_du_plus_grand(5, 100) devra renvoyer 10