FICHE DE L'ATELIER 11 : LES FONCTIONS (2 pages)

STRUCTURE GENERALE

Définition

```
def nom_fonction(x,y,z):
    …
    …
    …
    return(valeur_retournee)

Ici on revient sur la suite du code
```

Le nom de la fonction doit être entre le mot-clé def et le point double : .
Les instructions appartenant à la fonction doivent être tabulées.
Une fois le return rencontré, l'interpréteur sortira de la fonction : des lignes tabulées sous le return ne seront pas exécutées.

Le return permet de placer le résultat de la fonction dans le programme d'appel.
Exemple :

```
def plus_deux(x):
    resultat = x+2
    return(x)

z = 5
y = plus_deux(z)
print(y) # On imprime le contenu de y, 7 à l'écran
```

VOCABULAIRE

Procédure	Nom d'une fonction que ne possède pas de return() et ne renvoie donc rien vers le programme principal.

Méthode	Nom d'une fonction qui est contenue dans la classe d'un objet.
	On l'utilise en donnant le nom de l'objet suivi d'un point et du nom de la méthode :
	`mon_objet.ma_methode()`
	Certaines méthodes sont en réalité des fonctions qui renvoient un objet de la classe concernée. On note alors le nom de la classe (avec une majuscule), un point et le nom de la méthode.
	`mon_objet_image = Img.open("mon_image.jpg")`
	`fen_princ = Tk()`

Arguments et paramètres	Les **paramètres** désignent les variables qui vont contenir les données transmises lors de l'appel de la fonction : `def plus_deux(x):` ici, x est un paramètre. Les **arguments** désignent les variables qui sont transmises avec l'appel de la fonction : `Y = plus_deux(z)` ici, z est un argument.

Importation des modules

import ...	Pour importer les classes et méthodes d'une bibliothèque en gardant le nom de la bibliothèque lors de l'appel de la méthode (et donc d'éviter la confusion entre deux méthodes portant le même nom) : **import math** permet d'utiliser toutes les fonctions de math : Racine carre avec la fonction sqrt : `racine_de_x = math.sqrt(x)` Sin, cos ... avec la fonction sin : `sinus_de_x = math.sin(x)` Valeur approximative de pi avec pi : valeur_pi = **math.**pi .
import ... as ...	Pour importer les classes et méthodes d'une bibliothèque en changeant le nom de la bibliothèque lors de l'appel de la méthode (et donc d'éviter la confusion entre deux méthodes portant le même nom) : **import math as m** permet d'utiliser toutes les fonctions de math : Racine carre avec la fonction sqrt : `racine_de_x = m.sqrt(x)` Sin, cos ... avec la fonction sin : `sinus_de_x = m.sin(x)` . Valeur approximative de pi avec pi : valeur_pi = **m.**pi .
from .... import *	Pour importer intégralement les classes et méthodes d'une bibliothèque sans avoir à retaper le nom de la bibliothèque à chaque fois : `from math import *` permet d'utiliser toutes les fonctions de math : Racine carre avec la fonction sqrt : `racine_de_x = sqrt(x)`. Sin, cos ... avec la fonction sin : sinus_de_x = sin(x) . Valeur approximative de pi avec pi : valeur_pi = pi .
Composition des synthaxes précédentes	On peut importer un module d'une bibliothèque et modifier son nom d'appel : from PIL import Image as Img. On devrait alors utiliser Img.methode pour utiliser une méthode du module Image de PIL.