LANGAGE DE PROGRAMMATION PYTHON

ASSOHOUN E. Stanislas

- *L3 Actuariat IUA*

PLAN

- **CHI – DESCRIPTION GÉNÉRALE DU LANGAGE**
- **CH2 – FONCTIONS ET MODULES**
- **CH3 – STRUCTURES DE DONNEES ET CLASSE**

CHII

FONCTIONS ET MODULES

VII- FONCTIONS ET MODULES

VII-1- Fonctions

Définition de fonction :

```
def nomFonction(argument1, argument2, ... argumentk):
    instruction1
    instruction2
    ...
    instructionp
```

La ligne « def nomFonction(argument1, argument2, ... argument k) » est appelee l'en-tête de la fonction ; la séquence « instruction1, instruction2, ... instructionp » le corps de cette dernière. La fin du corps, et donc de la définition de la fonction, est indiquée par l'apparition de lignes ayant la même indentation que l'en-tête.

VIII- LES ENTREES ET SORTIES

VII-1- Fonctions

Définition de fonction :

```
def nomFonction(argument1, argument2, ... argumentk):
    instruction1
    instruction2
    ...
    instructionp
```

La ligne « def nomFonction(argument1, argument2, ... argument k) » est appelee l'en-tête de la fonction ; la séquence « instruction1, instruction2, ... instructionp » le corps de cette dernière. La fin du corps, et donc de la définition de la fonction, est indiquée par l'apparition de lignes ayant la même indentation que l'en-tête.

VII- FONCTIONS ET MODULES

VII-1- Fonctions

Appel de fonction

L'effet d'une définition de fonction n'est pas d'exécuter les instructions qui en composent le corps, mais uniquement de memoriser ces instructions en vue d'une (hypothétique) exécution ultérieure, provoquee par une expression, appelée appel de la fonction, qui prend la forme :

$$nomFonction(expression_1, expression_2, \ldots expression_k)$$

Les expressions expression1, expression2, ... expressionk sont appelees les arguments effectifs de l'appel. Leurs valeurs sont affectees à argument1, argument2, ... argument k, les arguments formels de la fonction, juste avant l'exécution de son corps, comme si cette exécution commencait par une série d'affectations.

VII- FONCTIONS ET MODULES

VII-1- Fonctions

Appel de fonction

Très souvent, le corps d'une fonction contient une ou plusieurs instructions de la forme.

return *expression*

dans ce cas, l'appel de la fonction, à l'endroit ou il est écrit, représente cette valeur renvoyée ; cet appel prend alors plutôt la forme:

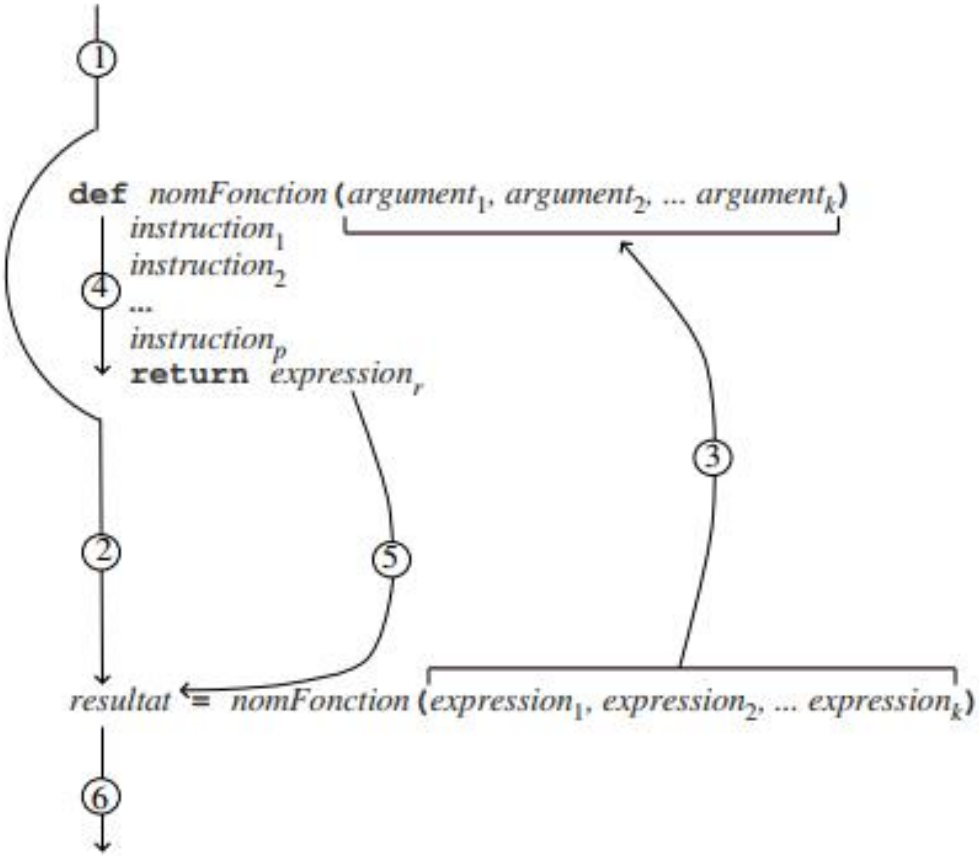
$$resultat = nomFonction(expression_1, expression_2, \ldots expression_k)$$

La figure suivante montre l'ordre chronologique d'exécution des instructions d'un programme comportant la définition et l'appel d'une fonction.

VII- FONCTIONS ET MODULES

VII-1- Fonctions

Appel de fonction



VII- FONCTIONS ET MODULES

VII-1- Fonctions

Application 1

Voici une fonction qui calcule la moyenne et l'écart-type des éléments d'une liste de nombres.

```
from math import sqrt
def moyenneEcartType(listeVal):
    nombre = 0
    sommeValeurs = 0.0
    sommeCarres = 0.0
for x in listeVal:
    nombre += 1
    sommeValeurs += x
    sommeCarres += x * x
moyenne = sommeValeurs / nombre
ecartt = sqrt(sommeCarres / nombre - moyenne * moyenne)
return (moyenne, ecartt)
# un appel de la fonction, pour la tester
valeurs = [ 1, 9, 4, 6, 8, 2, 5, 3, 7 ]
resultat = moyenneEcartType(valeurs)
print(resultat)
```

VII- FONCTIONS ET MODULES

VII-1- Fonctions

Application 2 (liste voir CH 3)

Recherche de l'indice de la première occurrence d'une valeur dans une liste, écrite sous forme de fonction. Dans le programme suivant, notez comment la fonction est abandonnée dès la rencontre de la valeur cherchée :

```
########
 def position(valeur, liste):
      for i in range(len(liste)):
                  if liste[i] == valeur:
                  return i
      return -1
# un appel de la fonction, pour la tester:
uneValeur = 64
uneListe = [1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]
p = position(uneValeur, uneListe)
if p >= 0:
      print( "la valeur est a la position", p)
else:
      print ("la valeur n'est pas présente")
```

VII- FONCTIONS ET MODULES

VII-1- Fonctions

✓ Variables locales et globales

A défaut d'une déclaration global (voir ci-après), toute variable apparaissant dans une fonction comme membre gauche d'une affectation est locale à cette fonction. Cela signifie qu'elle a une portée réduite à la fonction et, surtout, qu'elle est créée chaque fois que la fonction est appelée et détruite chaque fois que la fonction se termine.

A l'opposé de cela, les variables globales sont les variables créées à l'extérieur de toute fonction ; elles existent depuis le moment de leur création jusqu'à la fin de l'exécution du programme. Une variable globale peut, sans précaution particulière, être utilisée à l'intérieur d'une fonction si elle n'est pas le membre gauche d'une affectation et si elle n'est pas masquée par une variable locale de même nom. Pour que dans une fonction une variable globale puisse être le membre gauche d'une affectation, la variable en question doit faire l'objet d'une déclaration.

```
global nomDeVariable
```

VII- FONCTIONS ET MODULES

VII-1- Fonctions

✓ Récursivité

Une fonction est dite récursive lorsque son corps contient un ou plusieurs appels d'elle-même.

Par exemple, la factorielle d'un nombre n se définit « itérativement » par

$n! = n \times (n-1) \times (n-2) \times \cdots \times 2 \times 1.$

Il est évident qu'une définition récursive de la même quantité est

$$n! = \begin{cases} \text{si } n \leq 1 \text{ alors } 1 \\ \text{sinon } n \times (n-1)! \end{cases}$$

ce qui donne le code suivant :

```
def fact(n):
    if n <= 1:
        return 1
    else:
        return n * fact(n - 1)

# essai:
n = int(raw_input("nombre ? "))
print n, '! =', fact(n)
```

VII- FONCTIONS ET MODULES

VII-2- Modules

✓ Définition

Pour capitaliser les développements, Python propose la notion de module. Un module permet de fournir des bibliothèques de fonctions, structures de données, classes, à intégrer dans les programmes. Les définitions contenues dans un module sont utilisable globalement ou unitairement.

✓ Syntaxe de définition

Dans le cas de python, produire un module est identique à produire un programme : faire un fichier.

- Etape 1: définir le module avec ces procédures et fonctions

  « monmodule.py »

- Etape 2: importer le module dans votre code source

  from monmodule importe *

- Editer votre code en appelant les fonctions du module

.

VII- FONCTIONS ET MODULES

VII-2- Modules

- ✓ Application

```
## declaration du ficher « monmodule.py »
def perimetre(x):
    return 2*3.14*x
def aire(x):
    return 3.14*x*x

### importation du module
From monmodule.py import *
R=float(input(" Entrer le rayon du cercle : "))
vperi=perimetre(R)
vaire=aire(R)
print(" perimetre =", vperi)
print(" Aire =", vaire)
```