ENSIAS

جامعة محمد الخامس بالرباط
Université Mohammed V de Rabat

Langage Python

Lecture5: Fonctions

Abderrahim MESBAH
a.mesbah@um5r.ac.ma

Plan

- Définition
- Passage des arguments
- Porté des objets

Définition

Définition

- **Une fonction est un ensemble d'instructions regroupées sous un nom et s'exécutant à la demande.**
  - Syntaxe de définition d'une fonction en python

```
def  nom_fonction(arg1, arg2, …,argN):

        Instruction1
        …
        InstructionN

        return val1, val2, …, valm
```

  - Syntaxe d'appel d'une fonction en python

```
a1, a2, …, am=nom_fonction(x1, x2, …,xN)
```

Définition

- **Exemple**

  - Définition d'une fonction

```
def maFonction(x, y, z):
        a = z / x
        b = '2' + y
        return a, b
```

  - L'appel de la fonction

```
c, d = maFonction(7, 'k', 1.618)
```

```
c, d = maFonction(x=7, y='k',z= 1.618)
```

Passage des arguments

Passage des arguments

- **Un ou plusieurs paramètres, pas de retour (Méthode)**

```
#Affiche la table de multiplication des <base> de <debut> à <fin>.

def table(base, debut, fin) :
        n = debut
        while n <= fin :
                print(n, 'x', base, '=', n * base)
                n += 1
```

  - L'appel de la fonction table (Deux méthodes d'appel)

```
table(7, 2, 4)
```

```
table(base=7, debut=2, fin=4)
```

```
2 x 7 = 14
3 x 7 = 21
4 x 7 = 28
```

Passage des arguments

- **Un ou plusieurs paramètres, un ou plusieurs retours**

```
from math import pi
def cube(x) :
        #Retourne le cube de largument.
         return x**3

def volumeSphere(r) :
        #Retourne le volume d'une sphère de rayon <r>
         return 4.0 * pi * cube(r) / 3.0
```

  - L'appel de la fonction volumeSphere

```
# Saisie du rayon et affichage du volume
rayon = float(input('Rayon : '))
print(''Volume de la sphère ='', volumeSphere(rayon))
```

Passage des arguments

- **Un ou plusieurs paramètres, un ou plusieurs retours**

Exemple avec utilisation d'un return multiple :

```
import math
def surfaceVolumeSphere(r) :
        surf = 4.0 * math.pi * r**2
        vol = surf * r/3
        return surf, vol
```

- L'appel de la fonction surfaceVolumeSphere

```
# programme principal
rayon = float(input('Rayon : '))
s, v = surfaceVolumeSphere(rayon)
print("Sphère de surface {: .3f} et de volume {:.3f}".format(s, v))
```

Passage des arguments

- **Paramètres avec valeur par défaut**

```
def accueil(nom, prenom, depart="MP", semestre="S2"):
    print(prenom, nom, "Département", depart, "semestre", semestre)
```

  - L'appel de la fonction accueil

```
# programme principal
accueil("BADR", "Ahmed")
>>>Ahmed BADR Département MP semestre S2

accueil("BADR", "Ahmed","Info")
>>>Ahmed BADR Département Info semestre S2

accueil("BADR", "Ahmed", semestre="S3")
>>>Ahmed BADR Département MP semestre S3
```

Passage des arguments

- **Nombre d'arguments arbitraire : passage d'un tuple de valeurs**

```
def somme(*args) :
        #Renvoie la somme du tuple <args>.
        resultat = 0
        for nombre in args :
                resultat += nombre
        return resultat
```

- L'appel de la fonction accueil

```
print(somme(23))
>>> 23
print(somme(23, 42, 13))
>>>78
```

Passage des arguments

- **Nombre d'arguments arbitraire : passage d'un tuple de valeurs**

```
def somme(a,b,c) :
        return a+b+c
```

  - L'appel de la fonction accueil

```
elements=(2,4,6)
print(somme(*elements))
>>> 12
```

Passage des arguments

- **Nombre d'arguments arbitraire : passage d'un dictionnaire**

```
def unDict(**kwargs) :
    return kwargs
```

  - L'appel de la fonction accueil

```
print(unDict(a=23, b=42))
>>> {'a': 23, 'b': 42}

mots = {'d' : 85, 'e' : 14, 'f' :9}
print(unDict(**mots))
>>> {'d': 85, 'e': 14, 'f': 9}
```

ENSIAS

Porté des objets

Porté des objets

- **La portée globale : celle du module ou du fichier script en cours. Un dictionnaire gère les objets globaux : l'instruction globals() fournit un dictionnaire contenant les couples nom:valeur ;**

- **La portée locale : les objets internes aux fonctions sont locaux. Les objets globaux ne sont pas modifiables dans les portées locales. L'instruction locals() fournit un dictionnaire contenant les couples nom:valeur.**

Passage des arguments

```
# x et fonc sont affectés dans le module => globaux
def fonc(y) : # y et z sont affectés dans fonc => locaux
    global x # permet de modifier x ligne suivante
    x = x + 2
    z = x + y
    return z
x = 99
print(fonc(1)) # 102
print(x) # 101
```

Passage des arguments

```
# x et fonc sont affectés dans le module => globaux
def fonc(y) : # y et z sont affectés dans fonc => locaux
        # dans fonc : portée locale
        z = x + y
        return z

x = 99
print(fonc(1)) # 100
print(x) # 99
```

Passage des arguments

```
def fonc(y) : # y, x et z sont affectés dans fonc => locaux
        x = 3 # ce nouvel x est local et masque le x global
        z = x + y
        return z
x = 99
print(fonc(1)) # 4
print(x) # 99
```

ENSIAS

Exercice: Quel serait le résultat des codes suivants ?

```
def something1(a, b):
    for i in range(a):
        print(b,end='')

something1(5, 8)
```

```
def something2(a, b):
    for i in range(a):
        b = b*b
    return b

print(something2(3,2))
```

```
def something3(a):
    return a-1, a+1

a, b = something3(5)
print(a)
```

```
def something4(a):
    sum = 0
    for b in a:
        if b < 0:
            sum -= b
        else:
            sum += b
    return sum

print(something4([2, -4, 3, -1, 7, -4]))
```

ENSIAS

Exercice: Quel serait le résultat des codes suivants ?

```
def something5(a):
 sum1 = 0
 sum2 = 0
 for i in range(len(a)):
    if i%2 == 0:
        sum1 += a[i]
    else:
        sum2 += a[i]
 return sum1, sum2

x, y = something5([1, 2, 3, 4, 5, 6])
print(x,y)
```