Fiche n°03 : Programmation des fonctions et procédures

Rédigée par Pascal Delahaye

Cette fiche contient des élèments de cours à assimiler pour le cours de la semaine suivante.
Vous devez impérativement :

- Travailler avec un ordinateur afin de vérifier une à une les différentes instructions qui sont présentées.
- Vérifier et confirmer votre assimilation en effectuant tous les exercices d'entrainement proposés à la fin du poly.
- Prévoir environ 1h30 de travail personnel.

Une évaluation de 10mn sera effectuée au prochain cours pour vérifier la qualité de votre travail d'auto-apprentissage.

I] Introduction

Nous pouvons distinguer 3 types de programmes :

1) Les séries d'instructions

On demande simplement à l'ordinateur d'effectuer un certain nombre d'opérations que l'on appelle aussi des instructions.

Python
```
instruction 1
instruction 2
.
.
.
instruction n
```

`Exemple 1.` Test de la parité d'un nombre.

Python
```
a = eval(input("Choisissez un nombre entier"))    # Le programme demande de choisir un entier
if a % 2 == 0 : print(a," est une nombre pair")   # Le programme affiche si l'entier choisi
else          : print(a," est une nombre impair") # est pair ou impair
```

2) Les procédures

Une *procédure* est une série d'instructions :

- qui porte un nom
- qui effectue un travail dont le résultat dépend de variables (appelées *arguments*)
- qui ne renvoie pas de résultat.

Pour définir une procédure, on utilise la commande `def`.

Python
```
def nom_de_la_procedure(var1, var2 ...):   # les variables sont juste des noms
    from *** import ***, ***               # importation des fonctions utilisées
    instruction 1                          # une indentation de 4 espaces est obligatoire
    instruction 2                          # une indentation de 4 espaces est obligatoire
    .                                      # une indentation de 4 espaces est obligatoire
    .                                      # une indentation de 4 espaces est obligatoire
    instruction n                          # une procédure ne renvoie rien
```

`Exemple 2.` Une procédure `ajout_taille()` qui ajoute la taille d'une liste au niveau de sa dernière composante :

Python
```
def ajout_taille(L):
    n = len(L)                          # n prend pour valeur la taille de la liste L
    L.append("taille = " + str(n))      # on ajoute une phrase à la fin de la liste L
```

On remarque que ce programme ne renvoie rien, mais modifie le contenu de la donnée L qui se trouve en argument.

`Exemple 3.` Une procédure intéractive `somme()` pour calculer la somme de deux nombres :

Python
```
def somme():
    print("Nous allons calculer la somme de deux nombres.")
    a = eval(input("Entrer une premiere valeur : "))      # l'odinateur interroge l'utilisateur
    b = eval(input("Entrer une deuxieme valeur : "))      # l'odinateur interroge l'utilisateur
    print("la somme des deux valeurs est : ", a+b)        # l'odinateur affiche à l'écran le résultat
```

On remarque que ce programme ne renvoie rien, mais se contente d'afficher un résultat à l'écran.

3) Les fonctions

Une *fonction* est une série d'instructions :

- qui porte un nom
- qui effectue un travail dont le résultat dépend de variables (appelées *arguments*)
- qui renvoie une valeur que l'on peut stocker, si on le souhaite, dans une variable.

Pour définir une fonction, on utilise la commande `def` et on renvoie une valeur grâce à la commande `return`.

Python
```
def nom_de_la_procedure(var1, var2 ...):    # var1, var2... sont des arguments (juste des noms)
    from *** import ***, ***                # importation des fonctions utilisées
    instruction 1                           # Bien respecter le décallage de 4 espaces
    instruction 2
    .
    .
    instruction n
    return valeur                           # Une fonction renvoie une valeur
```

Remarque 1. Les fonctions mathématiques vérifient les mêmes caractéristiques.
Ainsi par exemple, la fonction mathématique φ définie par $\varphi(f,a,b) = \int_a^b f(t)\,\mathrm{d}t$:

- Porte le nom φ
- Dépend de 3 variables : $\begin{cases} f \text{ une fonction continue sur } \mathbb{R} \\ a \text{ un réel} \\ b \text{ un réel} \end{cases}$
- Renvoie une valeur $\varphi(f,a,b)$ qui est l'intégrale de f entre a et b.

`Exemple 4.` Une fonction booléenne `pair()` qui teste la parité d'un entier :

Python
```
def pair(a) :
    if a%2 == 0 : return True
    else        : return False
```

L'instruuction `pair(4)` renvoie `True` tandis que `pair(5)` renvoie `False`.

`Exemple 5.` Une fonction booléenne `implique(P,Q)` qui renvoie "True" si $P \Rightarrow Q$:

Rappel : la proposition $A \Rightarrow B$ est vraie lorsque A est fausse ou lorsque A et B sont simultanément vraies.

Python
```
def implique(P,Q) :
    return (not P) or (P & Q)
```

L'instruction `implique(2 == 3, True)` renvoie `True` tandis que `implique(3 > 2, 1==2 )` renvoie `False`.

`Exemple 6.` Comme en mathématiques, il est possible de composer des fonctions entre elles.
Programmation des fonctions `somme()`, `produit()` et `calcul()`.

Python
```
##### Définition des sous-fonctions

def somme(x,y):             # Renvoie la somme de 2 nombres
    return x+y

def produit(x,y):           # Renvoie le produit de 2 nombres
    return x*y

##### Définition de la fonction principale

def calcul(x,y):            # Renvoie le double de la somme de 2 nombres
    produit(somme(x,y),2)

##### Utilisation de la fonction calcul()

calcul(2,3)                 # On appelle la fonction avec les arguments 2 et 3
```

L'instruction `calcul(2,3)` renvoie 10.

II] Les fonctions Lambda

Pour les fonctions ou les procédures ne contenant qu'une seule instruction, Python propose la syntaxe suivante.

Python
```
## DEFINITION de la fonction :

f = lambda var1, var2 ... : instruction   # f est une fonction d'arguments var1, var2...
                                          # Ici on n'utilise pas la commande "return"
## UTILISATION de la fonction

f(val1, val2...)                          # val1 et val2 sont des valeurs particulières de var1
                                          # et var2
```

1. La fonction f définie par $f(x) = x^2 - 1$:

Python
```
f = lambda x : x**2 -1
f(3)                        # Cela renvoie 8
```

2. La fonction f définie par $f(x, y) = x^y$:

Python
```
f = lambda x, y : x**y
f(3,2)                      # Cela renvoie 9
```

3. La fonction *inverse* qui inverse l'ordre des composantes d'une liste :

Python
```
inverse = lambda L : L.reverse()

L = [1,2,3]
inverse(L)
L                           # L'ordre des composantes de L a été inversé
```

4. La fonction *compose* qui renvoie $f \circ g(x)$:

Python
```
compose = lambda f, g, x : f(g(x))

compose(lambda x : x**2, lambda x : 1/x, 3)
```

5. La fonction `integr` qui renvoie $\int_a^b f(t)\,\mathrm{d}t$:

Python
```
from sympy import var, integrate              # importation depuis sympy
var('x')                                      # x devient ainsi une variable formelle

integr = lambda a,b,f : integrate(f(x),(x,a,b))

integr(0,4,lambda x : x**2)
```

III] La programmation en Python

De façon générale sous SPIDER, les programmes en Python se tapent dans la fenêtre "Editeur" et s'utilisent dans la fenêtre "console".

Structure d'un programme Python :

```
## REDACTION des PROGRAMMES : (dans l'éditeur)

def nom_fonction1(var_1, var_2, ...):       # les arguments sont juste des noms de variables
    from *** import ***, ***                # importation des fonctions utilisées
    instruction1
    instruction2
    .
    .

def nom_fonction2(var_1, var_2, ...):       # les arguments sont juste des noms de variables
    from *** import ***, ***                # importation des fonctions utilisées
    instruction1
    instruction2
    .
    .
```

```
## UTILISATION du PROGRAMME : (dans la console)

nom_fonction(valeur_1, valeur_2, ...)        # Les valeurs sont des données explicites
```

Avant d'exécuter un programme, il faut commencer par l'enregistrer. Vous pouvez le faire dans un répertoire personnel sur le disque dur de l'ordinateur ou sur une clé USB. On peut alors le lancer en tapant sur la touche F5.

Après avoir appuyé sur la touche F5, la fonction que l'on vient éventuellement de créer dans l'éditeur peut être utilisée depuis la console dédiée à celui-ci. On appelle alors la fonction de la façon suivante :

`nom_programme(valeur_1, valeur_2, ...)`

IV] Exercices

Exercice : 1

Construire les fonctions lambda :

1. d'arguments deux flottants x et y et qui renvoie $x\cos y$.
2. d'arguments deux listes L1 et L2 et qui renvoie true si les premiers éléments sont égaux.
3. d'arguments x et un entier n et qui renvoie la liste contenant n fois la valeur x
4. d'argument un entier n et qui renvoie true si n est un carré (importer `sqrt` et `floor`).
5. d'arguments une fonction f et un flottant x et qui renvoie $f\circ f(x)$.

Exercice : 2

Construire les procédures :

1. sans argument qui demande à l'utilisateur son nom, puis son prénom et qui affiche : "tu t'appelles ....".
2. d'argument une liste et qui échange les deux premières composantes.
3. d'arguments deux listes de 2 flottants représentant des vecteurs et qui dit si les vecteurs sont colinéaires.
4. d'argument une liste a trois composantes et qui effectue une permutation circulaire de celles-ci.

Exercice : 3

Construire les fonctions :

1. d'arguments deux listes à deux composantes et qui renvoie la liste obtenue par différence des deux listes précédentes.

2. sans argument qui choisit deux entiers au hasard entre 1 et 3 et qui renvoie True si ces entiers sont égaux et False sinon.
3. d'arguments f, x et y et qui demande à l'utilisateur de choisir un entier a puis renvoie $a \times f(x-y)$.
4. d'arguments deux propositions booléennes et renvoie True si ces propositions sont équivalentes (c'est à dire si elles ont même valeur de vérité).

Exercice : 4

Programmation :

1. Quels sont les 3 types de programmes que l'on peut distinguer ?
2. Quelle est la différence entre une fonction et une procédure ?
3. Quelle est la différence entre une fonction et une fonction lambda ?
4. Dans quelle fenêtre de SPIDER :
   (a) rédige-t-on les programmes ?
   (b) utilise-t-on les fonctions ou les procédures programmées ?