1 Notion de fonctions

Supposons que nous voulions calculer les images de plusieurs nombres par la fonction f définie par :

$$f(x) = x^5 - 6x^4 + 3x$$

On pourrait procéder de la manière suivante :

```
>>> x = 4
>>> x**5 - 6 * x**4 + 3 * x   #calcul de f(4)
>>> x = -7
>>> x**5 - 6 * x**4 + 3 * x   #Calcul de f(-7)
```

On constate qu'il devient vite fastidieux de recopier à chaque fois la formule de calcul. Même en utilisant l'historique, cela s'avère compliqué si on doit manipuler plusieurs formules différentes. Pour y remédier, nous allons pouvoir définir une fonction informatique, c'est à dire un programme prenant des paramètres en entrée et fournissant des résultats en sortie à l'aide d'instructions utilisant ces paramètres. En Python, la syntaxe permettant de définir la fonction précédente est la suivante :

```
def f(x) :
    return  x**5 - 6 * x**4 + 3 * x
```

Remarques :
- L'indentation est, tout comme pour les conditions, primordiale ;
- `f` est ici le nom de la fonction ;
- `x` est le paramètre d'entrée, c'est-à-dire la variable dont il faudra préciser la valeur pour pouvoir utiliser la formule ;
- Attention à ne pas oublier les deux points « `:` » ;
- Attention à l'indentation devant le `return` ;
- `x**5 - 6 * x**4 + 3 * x` est le résultat attendu.

Pour utiliser une fonction enregistrée dans un fichier, on peut exécuter le fichier en question puis demander le calcul des valeurs souhaitées dans la console, comme ici par exemple :

```
>>> f(5)
```

Observez les fonctions suivantes. Que renvoient-elles comme résultat(s) ?

```
from math import *
def cercle(r) :
    return  2*pi*r, pi*r**2
```

```
def rectangle(a,b) :
    return  a*b
```

```
def absolue(x) :
    if (x>=0) :
        return  x
    else :
        return -x
```

On prendra dès maintenant l'habitude de documenter les fonctions à l'aide de commentaires expliquant les différentes étapes. Par exemple :

```
from math import *              # nécessaire pour utiliser pi
def cercle(r) :                 # r=rayon
    return  2*pi*r, pi*r**2     # périmètre, aire
```

Remarque : une fonction peut être utilisée dans une autre fonction ou dans un script. Par exemple :

```
from math import *              # nécessaire pour utiliser pi
def cercle(r) :                 # r=rayon
    return  2*pi*r, pi*r**2     # périmètre, aire


def volume_cylindre(r,h) :          # r=rayon, h=hauteur
    a,b=cercle(r)               # a = périmètre de la base, b=aire de la base
    return b*h                  # volume du cylindre
```

2 Variables locales et globales

Les variables définies à l'intérieur d'une fonction sont appelées variables locales. Elles n'existent que le temps que la fonction s'exécute. Une fois l'exécution terminée, les variables locales sont effacées de la mémoire.

Au contraire, les variables globales définies à l'extérieur des fonctions restent en mémoire.

Déterminons par exemple une fonction qui calcule Le prix TTC d'un article. On définit la TVA dans une variable à l'extérieur de la fonction.

```
>>> tva=20.0                            # On définit le taux de TVA à appliquer
>>> def ttc(P) :
        prix = P*tva/100+P               # On calcule le prixc TTC
        return  prix
>>> ttc(327)                        # Calcul du prix TTC pour un produit coutant 327 euros
>>> tva
>>> prix
```

Que remarque-t-on pour les variables `tva` et `prix` ?

3 Fonctions récursives

Une fonction peut dans certains cas s'appeler elle-même, on dit alors que c'est une fonction récursive.

Elle respecte alors les conditions suivantes :

— elle contient un cas de base ;

— elle doit pouvoir se ramener au cas de base en modifiant ses paramètres à chaque appel ;

— elle doit s'appeler elle-même.

Examinons un exemple de fonction récursive, celui de la fonction puissance qui calcule x^n en utilisant $x^n = x * x^{n-1}$:

```
def puissance(x,n) :      # On veut calculer x^n, x réel, n entier naturel
    if (n==0) :
        return 1
    else :
        return x*puissance(x,n-1)     #x^n = x * x^{n-1}
```

On a bien le cas de base, c'est à dire quand l'exposant vaut 0, on sait que $x^0 = 1$. Par ailleurs, cette fonction va forcément se ramener au cas de base si on choisit $n \in \mathbb{N}$ puisqu'on diminue l'exposant à chaque appel (on finira donc par atteindre un exposant égal à 0). Enfin, il est évident qu'elle s'appelle elle même.

Que se passe-t-il si on utilise ici un exposant négatif ?

4 Exercices

Exercice 1

Créer une fonction `maximum` qui prend en entrée deux nombres réels `a` et `b` et renvoie comme résultat le maximum de ces deux nombres.

Tester `maximum(5,8.6)` et `maximum(7,-3)`.

Exercice 2

Créer une fonction `minmax` qui prend en entrée trois nombres réels `a` et `b` et `c` et renvoie comme résultats le minimum et le maximum de ces trois nombres.

Tester `minmax(8,-2,4)`.

Exercice 3

Créer une fonction `parite` qui prend en entrée un entier naturel `n` et renvoie `True` si ce nombre est pair ou `False` si ce nombre est impair.

Tester `parite(18)` et `parite(19)`.

Exercice 4

Créer une fonction `carreParfait` qui en entrée un entier naturel `n` et renvoie `True` si ce nombre est un carré parfait ou `False` sinon.

Exercice 5

Créer une fonction récursive `factorielle` qui prend en entrée un nombre entier naturel `n` et qui renvoie le produit de tous les entiers inférieurs ou égaux à `n`.