Fiche 4 : Fonctions

1 Les fonctions : introduction

Jusqu'à présent, les programmes utilisés ne sont pas exploitables : ils affichent un résultat mais on ne peut pas les utiliser directement. De plus, on demande toujours à l'utilisateur de saisir les données, ce qui empêche une automatisation des tâches. L'avantage des fonctions est de plusieurs ordres :

— d'une part, si la fonction renvoie une donnée, il est possible de la récupérer pour effectuer un nouveau traitement ;

— d'autre part, il est possible d'appliquer une fonction à un nombre conséquent de données et de faire du traitement automatique.

Commençons par un exemple.

Exemple 1. La fonction ci-dessous

```
def SommePuissance(n,p) :
    S=0
    for i in range(1,n+1) :
        S=S+i**p
    return S
```

prend en argument deux entiers : n et p, et renvoie la somme des n premières puissances p^{e} entières. Expliquons les différents mots-clés de cette fonction.

1. le mot `def` permet de définir une fonction. Ici, on a donné comme nom à la fonction `SommePuissance`.
2. Le couple `(n,p)` correspond aux arguments de la fonction.
3. Le mot `return` qui correspond à la valeur de retour de la fonction.

Il est possible d'utiliser la fonction directement dans l'interpréteur de la manière suivante :

```
SommePuissance(10,2)
```

2 Fonctions à connaître

Voici une liste de quelques fonctions que nous avons déjà vu et qui sont à connaître (d'autres fonctions seront étudiées ultérieurement) :

Nom fonction	Effet	Exemple
`print`	affiche ce qui est en argument	print(“toto”)
`input`	laisse à l'utilisateur de saisir au clavier	a=input()
`str`	Convertit l'argument de la fonction en chaîne	v=str(1)
`int`	Convertit l'argument en entier si possible	v=int(“12”)
`float`	Convertit l'argument en réel si possible	v=float(input())

3 Syntaxe pour définir une fonction

Donnons quelques avantages des fonctions :
— possibilité d'automatisation de tâches ;
— possibilité de réutilisation ;
— valeur de retour stockable dans une variable.

fonction(arg1,$\cdots$,argN) Instructions retourner valeur	`def` NomFonction(arg1,$\cdots$,argN) : Instructions `return` valeur

Parfois, on ne veut pas de valeur de retour et seulement sortir de la fonction. Dans ce cas, on peut simplement écrire `return`.

4 Exemples

Exemple 2. La fonction min :

```
def minimum(x,y) :
    if x<=y :
        return x
    else :
        return y
```

Exemple 3. La fonction valeur absolue :

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

Exemple 4. La fonction ci-dessous vérifie si la suite $(u_n)_{n\in\mathbb{N}}$ définie par

$$u_0 = q, \forall n \in \mathbb{N}, u_{n+1} = u_n^2 - 4u_n + 1$$

est croissante jusqu'à un rang n donné :

```
def est_croissant(q,n) :
    u=q
    v=u**2 -4u +1
    for i in range(n) :
        if u>v :
            return False
        u=v
        v=u**2-4u+1
    return True
```