2.1 *Fonctions*

Une fonction est une sorte de sous-programme d'un programme Python dédié à réaliser une tâche particulière et auquel on peut faire appel dans le programme principal ou dans une autre fonction.

L'intérêt d'une fonction est d'éviter de réécrire plusieurs fois le même code d'instructions à différents endroits d'un programme. La syntaxe est la suivante.

```
def nom_de_la_fonction(liste de paramètres):
    instructions
```

Si on veut que la fonction renvoie un résultat, on utilise le mot-clé `return`. L'exécution d'une fonction s'arrête dès le premier `return` rencontré.

Remarque 1 Pour que la syntaxe soit correcte en Python, il est indispensable que :

1) la ligne commençant par `def` (l'identificateur de fonction) se termine par deux points (`:`)
2) le bloc d'instructions soit écrit en retrait par rapport aux autres lignes. Ce retrait est appelé l'indentation.

Voici un exemple. La fonction `somme` suivante possède deux paramètres a et b.

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

L'instruction `somme(3,5)` renvoie la somme de 3 et 5 c'est-à-dire l'entier 8. Dans ce cas, on dit qu'on a appelé la fonction somme en passant 3 et 5 en arguments. Ainsi, des arguments sont des valeurs données aux paramètres de la fonction (ici, a et b).

Il existe un certain nombre de fonctions pré-existantes en Python. Citons, en particulier, la fonction `print` qui permet d'afficher à l'écran les quantités passées en argument et `input` qui permet de demander à l'utilisateur de saisir une ou plusieurs valeurs.

Ainsi, si on veut voir afficher à l'écran la valeur d'une variable `A`, on écrit `print(A)` et, de même, si on veut afficher à l'écran la valeur renvoyée par la fonction `somme` lorsqu'on passe 3 et 5 en arguments, on écrit `print(somme(3,5))`.

2.2 *Instruction conditionnelle*

Une instruction conditionnelle est une instruction du type

Si (condition C) alors (bloc d'instructions 1) sinon (bloc d'instructions 2)

ce qui signifie : si la condition C est vraie, effectuer le bloc d'instructions 1 et sinon (si elle est fausse), effectuer le bloc d'instructions 2.

Remarque 2 La partie « sinon (bloc d'instructions 2) » n'est pas obligatoire. Si on l'omet alors aucune action n'est effectuée si la condition C est fausse.

La syntaxe d'une instruction conditionnelle est la suivante :

en langage naturel :

Si (condition C)
bloc d'instructions 1
Sinon
bloc d'instructions 2

en Python :

```
if (condition C):
    instructions 1
else :
    instructions 2
```

Voici un exemple. La fonction `lpg` prend en arguments deux nombres (entiers ou flottants) et renvoie le plus grand des deux.

```
def lpg(a,b):
    if a >= b:
        return a
    else :
        return b
```

Remarque 3 Comme pour les fonctions, les deux points à la fin de la ligne contenant `if` et après `else` ainsi que l'indentation pour les blocs d'instructions 1 et 2 sont indispensables.

Par ailleurs, lorsqu'on a plus de deux cas possibles dans une instruction conditionnelle, on utilise la syntaxe suivante :

```
if (condition 1):
    instructions 1
elif (condition 2):
    instructions 2
else :
    instructions 3
```

On peut ajouter autant de conditions intermédiaires que l'on veut en ajoutant `elif` devant chacune d'elles.

2.3 *Exercices*

♦ **PYT.1** Écrire une fonction `difference` telle que `difference(a,b)` renvoie $a-b$ et une fonction produit telle que `produit(a,b)` renvoie $a \times b$.

♦ **PYT.2** Écrire une fonction `est_isocele` qui prend en arguments trois nombres entiers `a`, `b` et `c` et qui renvoie `True` si le triangle de côtés `a`, `b` et `c` est isocèle et `False` sinon.

Le faire ensuite en deux lignes de code.

Écrire de même une fonction `est_rectangle` qui prend en arguments trois nombres entiers `a`, `b` et `c` et qui renvoie `True` si le triangle de côtés `a`, `b` et `c` est rectangle et `False` sinon, toujours en deux lignes de code.

♦ **PYT.3** Écrire une fonction `lpp` prenant en arguments deux nombres `a` et `b` et qui renvoie le plus petit des deux.

♦ **PYT.4** Écrire une fonction `valeur_absolue` prenant en argument un nombre `x` et qui renvoie la valeur absolue de `x`.

♦ **PYT.5** Écrire une fonction `est_entier` qui prend en argument un nombre (entier ou flottant) `x` et qui renvoie `True` si la valeur de `x` est entière (indépendamment du type) et `False` sinon. Ainsi, `est_entier(3)` ou `est_entier(3.0)` renvoie `True` et `est_entier(4.23)` renvoie `False`.

☞ **Indications exercice 5 :** En Python, `3 == 3.0` renvoie `True`.

♦ **PYT.6** Écrire une fonction `est_pair` prenant en argument un entier `n` et qui renvoie `True` si `n` est pair et `False` sinon.

♦ **PYT.7**

1) Écrire une fonction `intervalle1` prenant en argument un nombre x et qui renvoie `True` si x appartient à l'intervalle $]-2;3]$ et `False` sinon.
2) Écrire une fonction `intervalle2` prenant en argument un nombre x et qui renvoie `True` si x appartient à l'intervalle $]-\infty;-3] \cup [5;+\infty[$ et `False` sinon.
3) Écrire une fonction `intervalle3` prenant en argument un nombre x et qui renvoie `True` si x appartient à l'intervalle $]-5;-3] \cup [0;2[$ et `False` sinon.
4) Écrire une fonction `intervalle4` prenant en argument un nombre x et qui renvoie `True` si x est strictement positif et différent de 1 ou si x est strictement négatif et différent de -1 et `False` sinon.

♦ **PYT.8** Écrire une fonction `signe` prenant un nombre x en argument et qui affiche `positif` si le nombre est strictement positif, `nul` si le nombre est nul et `négatif` si le nombre est strictement négatif.

☞ **Indications exercice 8 :** Pour afficher `positif`, la syntaxe est `print("positif")`.

♦ **PYT.9** Une année est bissextile si elle est divisible par 4 mais pas par 100 ou bien si elle est divisible par 400. Par exemple, 2024 est bissextile car 2024 est divisible par 4 mais pas par 100. De même, 2000 est bissextile car 2000 est divisible par 400. En revanche, 2100 n'est pas bissextile car 2100 est divisible par 100 mais pas par 400.

Écrire une fonction Python `est_bissextile` qui prend en argument un entier naturel `n` et affiche `bissextile` si l'année `n` est bissextile et `non bissextile` sinon.