# Chapitre 4 : Fonctions (Première partie) - TD

**Exercice 1 : Test de compréhension***

1. Quelle est la différence entre définition et appel d'une fonction ?
2. Qu'est-ce que l'en-tête d'une fonction ? le corps d'une fonction ?
3. Quelle est la différence entre paramètres et valeurs des paramètres ?
4. Qu'est-ce qu'un algorithme appelant ?
5. Les parenthèses sont-elles obligatoires dans la définition d'une fonction s'il n'y a pas de paramètres ? Sont-elles nécessaires dans un appel de fonction s'il n'y a pas de valeurs de paramètre ?
6. Combien de fois une fonction peut-elle être appelée au minimum ? Au maximum ? Donner deux exemples où une fonction est appelée 100 fois, un avec les mêmes valeurs de paramètre pour tous les appels, puis l'autre avec des différentes valeurs de paramètre à chaque appel.
7. Considérons le code :

```
In [ ]: def afficheRes(nb) : # 1
            print( "Le résultat est " , nb) # 2


        def somme2nb( a, b) : #3
            print("Premier nombre = " , a , ", deuxième nombre = " , b) #4
            afficheRes(a+b) #5

        somme2nb(1,2); #6
```

Donner, pour la fonction `somme2nb`, les numéros des lignes associées à l'en-tête, au corps de la fonction, à la définition et à l'appel de la fonction. Indiquer également l'algorithme appelant. Donner finalement ses paramètres et ses valeurs de paramètres lors de son appel. Répondre ensuite aux mêmes questions pour la fonction `afficheRes`.

### Exercice 2 : Nombre d'appels*

1. Considérons le code :

```
In [ ]: def f1() :
            print ("hello")
            print ("hello")


        # Algorithme principal
        print( "hello");
```

   - À l'exécution, combien de fois la fonction `f1` est-elle appelée ?
   - Combien de fois est affiché hello ?

2. Considérons le code :

```
In [ ]: def f2() :
            print ("hello")
            print ("hello")

        # Algorithme principal
        print("hello")
        f2()
        f2()
```

   - À l'exécution, combien de fois la fonction `f2` est-elle appelée ?
   - Combien de fois est affiché hello ?

3. Considérons le code :

```
In [ ]: def f3() :
            print ( "hello")
            print ( "hello")

        def f4() :
            f3()
            print("hello")
            f3()


        # Algorithme principal
        print("hello")
        f3()
        f4()
```

   - Combien de fois sont appelées `f3` et `f4` ?
   - Combien de fois est affiché hello ?

4. Considérons le code :

```
In [ ]: def f5() :
            print("hello")
            print ("hello")

        def f6(n) :
            i = 0
            while i<n :
                f5()
                i=i+1
                print("hello")


        # Algorithme principal
        print("hello")
        f5()
        f6(2)
        f6(4)
```

- Combien de fois sont appelées `f5` et `f6` ?
- Combien de fois est affiché hello ?

### Exercice 3 : Valeur de paramètres¨*

- Qu'affiche le code :

```
In [ ]: def f(a) :
            print( "a = ", a )

        # Algorithme principal
        a = 3
        b = 5
        a = 18
        f(b)
```

- Qu'affiche le code :

```
In [ ]: def f(a, b) :
            print( "a = " , a , ", b = " , b)

        # Algorithme principal
        a = 3
        b = 4
        f(a,b)
        f(b,a)
        f(a,a)
        f(b+2,b-3)
```

### Exercice 4 : Portée des variables*

- Qu'affiche le code :

```
In [ ]: def f() :
            a = 18
            print ("a = " , a)

        # Algorithme principal
        a = 5
        f()
        print( "a = " , a )
```

### Exercice 5 : Paramètres et variables locales*

- Qu'affiche le code :

```
In [ ]: def f(a) :
            a = 10
            a +=1
            print ( "a = " , a)

        # Algorithme principal
        f(3)
        f(6)
```

### Exercice 6 : Comparer deux nombres*

Ecrire la fonction `compare` qui reçoit deux nombres entiers en paramètre `a` et `b` et affiche un message adapté selon que `a` est supérieur, inférieur ou égal à `b`.