Chapitre 5 : Fonctions avec valeurs de retour - TD

Exercice 1 : Traces de programme*

Question 1. Combien de fois est appelée `f` ? Combien de fois est affiché hello ?

```
def f(a) :
    if a < 0 :
        return
    print("hello")

f(3)
f(-2)
```

Question 2. Quel est l'affichage produit ? Quelles sont les valeurs des variables à la fin ?

```
def f(x) :
    return 2 * x + 3

res = f(3)
print(res)
z = 2
res = f(z)
print(res)
print(f(res))
res = f(res)
print(res)
print(f(res))
```

Question 3. On considère les fonctions `somme1` et `somme2` définies par :

```
def somme1(a, b):
    print(a + b)

def somme2(a, b):
    return a + b
```

1. Quelle est la différence entre `somme1` et `somme2` ?
2. Utiliser la fonction `somme1` pour afficher le résultat de 2 + 7 (sans utiliser l'opérateur +). Même question avec la fonction `somme2`.
3. Utiliser la fonction `somme1` pour afficher le résultat de 2 + 7 + 18 (sans utiliser l'opérateur +). Même question avec la fonction somme2. Quelle est alors la fonction la mieux programmée ?

Exercice 2 : Test de parité*

Question 1. Écrire la fonction `estPair` qui retourne `True` si le nombre passé en paramètre est pair et `False` sinon.

Question 2. Définir une fonction de tests unitaires pour la fonction `estPair`.

Exercice 3 : Heure valide*

Question 1. Définir la fonction `estHeureValide` prenant en paramètre une heure (donnée par 3 entiers : le nombre d'heures, le nombre de minutes et le nombre de secondes) et retournant `True` si l'heure est correcte, et `False` sinon.
Remarque : Une heure est considérée comme valide si le nombre d'heures et compris entre 0 et 23 et le nombre de minutes et secondes sont compris entre 0 et 59.

Question 2. Définir une fonction de tests unitaires pour la fonction `estHeureValide`.

Exercice 4 : Nombre de secondes depuis minuit*

Question 1. Définir la fonction `nbs` qui étant donné une heure (donnée par 3 entiers : le nombre d'heures, le nombre de minutes et le nombre de secondes) passée en paramètre, retourne le nombre de secondes passées depuis minuit. À titre d'exemple, `nbs(1,1,1)` doit retourner 3661 (car une heure contient 3600 secondes et une minute 60 secondes).
Remarque : pour tester si l'heure passée en paramètre est valide, vous utiliserez la fonction `estHeureValide` de l'exercice précédent.

Question 2. Définir une fonction `test_nbs` qui correspond au test unitaire de la fonction `nbs`. Vérifier que votre fonction satisfait tous les tests.

Exercice 5 : Série harmonique**

La série harmonique est la série définie pour tout $n > 0$ par:

$$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{n}$$

Question 1. Définir la fonction `harmonique` prenant en paramètre un entier n et retournant la valeur H_n (la fonction retournera `-1` si H_n n'est pas défini).

Question 2. Définir une fonction de tests unitaire de la fonction `harmonique` sachant que $H_5 = 2.283333333333333$ et $H_{20} = 3.597739657143682$.

Pour aller plus loin

Pour les exercices suivants, des tests unitaires de chaque fonction à définir existent dans la version notebook de ce TD pour que vous puissiez tester vos fonctions.

Exercice 6 : Année bissextile**

Définir la fonction `estBissextile` qui retourne `True` si l'année passée en paramètre correspond à une année bissextile, et `False` sinon.

Remarque : une année est bissextile si elle est divisible par 4, mais pas par 100, sauf si elle est aussi divisible par 400. Ainsi 2008 était bissextile, 1900 n'était pas bissextile et 2000 était bissextile.

Exercice 7 : Nombre de jours dans un mois**

Définir la fonction `nbjours` prenant en paramètre un numéro de mois (entre 1 et 12), ainsi qu'une année et retournant le nombre de jours dans le mois.

Rappel :

- Janvier, mars, mai, juillet, août, octobre et décembre contiennent 31 jours.
- Février contient 29 jours pour les années bissextiles et 28 jours sinon.
- Avril, juin, septembre et novembre contiennent 30 jours.

Remarques :

- La fonction retournera 0 si le mois n'est pas valide.
- On utilisera la fonction `estBissextile` pour déterminer si l'année passée en paramètre est bissextile.

Exercice 8 : Date valide*

Définir la fonction `estDateValide` prenant en paramètre une date (définie par 3 entiers : jours, mois, années) et retournant `True` si la date est valide, et `False` sinon.

Remarque : on utilisera la fonction `nbjours` de l'exercice précédent pour vérifier si le mois passé en paramètre est correct.

Exercice 9 : Nombre de jours depuis le 1er janvier 1970***

Définir la fonction `jours1970` qui prend en paramètre une date et retourne le nombre de jours passés depuis le 1er janvier 1970. Si la date passée en paramètre n'est pas valide, la fonction retourne 0. Si la date est avant le 1er janvier 1970, alors la fonction renvoie un nombre négatif dont la valeur absolue est le nombre de jours depuis la date jusqu'au 01/01/1970.

Exercice 10 : Timestamp*

En informatique, pour comparer des dates, on utilise le *timestamp*. Un timestamp correspond au temps écoulé depuis un événement précis (qui sert alors de référence). Sur les systèmes UNIX, le timestamp correspond au nombre de secondes écoulées depuis le 1er janvier 1970 à minuit.

Définir la fonction `timestamp` qui prend en paramètre une date et une heure, et retourne le nombre de secondes écoulées par rapport au 1er janvier 1970.