TD - Fonctions

Première générale - NSI - Lycée Bergson - 2021/2022

Objectif Ce TD a pour objectif de mettre en application vos connaissances sur les fonctions.

Pré-requis Pour réaliser ce TD il faut:

- Avoir suivi le cours/TP sur les fonctions.

1 Rappels

Avant d'écrire une fonction... Il faut :

- déterminer ce que la fonction attend comme **paramètres** ;
- attribuer un nom de variable à chacun des paramètres ;
- déterminer le **type** de chacun des paramètres ; (entier, booléen, chaîne de caractères, etc.) ;
- déterminer ce que la fonction doit renvoyer et le type de ce qu'elle renvoie le cas échéant.

- Les paramètres se placent dans **l'en-tête** de la fonction, entre parenthèses après le nom de la fonction.
- Pour renvoyer une valeur on utilise le mot clé `return`.

```
def ma_fonction(x):
    # corps de la fonction
    return valeur
```

- Lors de l'appel d'une fonction, lorsque l'instruction `return` est rencontrée on sort immédiatement de la fonction.

2 Analyse

Observer les programmes ci-dessous et répondre aux question suivantes.

- Quels sont les paramètres de la fonction ?
- Quel est le type des paramètres ?
- Qu'affiche le programme ?

1.
```
def une_fonction(prenom):
    reponse = "Je m'appelle " + prenom + "."
    return reponse
print(une_fonction("Pierre"))
```

2.
```
def une_autre_fonction(i,j):
    difference = i - j
    return difference
print(une_autre_fonction(3))
```

3.
```
def encore_une_autre(prenom):
    return "Hello world!"
    reponse = "Je m'appelle " + prenom + "."
    return reponse
print(encore_une_autre("Pierre"))
```

4.
```
def division(i):
    return i / 3
print(division("6"))
```

5.
```
def produit(i,j):
    return i * j
print(produit(2,3))
print(j)
```

6.
```
def concat(a,b):
    return a + b + c
a = "Hello"
b = "world"
c = "!"
print(concat(a,b))
```

3 Documentation

Pour qu'une fonction soit plus compréhensible il est très fortement recommandé d'écrire une documentation qui va donner des détails sur le comportement de la fonction.

La documentation s'écrit au tout début de la fonction et en utilisant les triples guillemets `'''`.

```
def ma_fonction(parametre1, parametre2):
    '''
    Phrase qui explique ce que fait la fonction
    parametre1 : (type) à quoi sert parametre1
    parametre2 : (type) à quoi sert paramètre2
    return : (type) ce que la fonction renvoie
    '''
    # corps de la fonction
    return valeur
```

Remarques

- On doit trouver au miminum dans la documentation ce que fait la fonction.
- Une bonne documentation indique également le type et l'utilisation des paramètres ainsi qu'une description de la valeur renvoyée.
- Cette documentation peut ensuite être utilisée par n'importe quel utilisateur dans l'interpréteur python avec la commande `help(nom_de_la_fonction)`.

Exemple

```
def fact(n):
    '''
    Calcule la factorielle d'un entier n.
    n : (int) un entier pour lequel la fonction calcule la factorielle
    return : (int) la factorielle de n
    '''
    f = 1
    for i in range(2, n+1):
        f *= i
    return f
```

Exercice 1
On donne la fonction suivante :

```
def est_diviseur(n,d):
    '''
    Indique si d est un diviseur de n.
    n : l'entier à tester
    '''
    if n % d == 0:
        return True
    else:
        return False
```

Pourquoi cette documentation n'est-elle pas satisfaisante ? Corriger la documentation.

Exercice 2
On donne la fonction suivante qui calcule un entier n à la puissance p :

```
def puissance(n,p):
    return n**d
```

Écrire la documentation de cette fonction.

4 Exercices

Pour chacun des exercices suivants :

- Écrire les fonctions et/ou programmes demandés.
- Écrire la documentation des fonctions.
- Tester la fonction (ou programme le cas échéant) avec un exemple de votre choix.

Exercice 3
Écrire une fonction `somme` qui prend en paramètres deux entiers x et y et retourne leur somme.

Exercice 4
Écrire une fonction `difference` qui prend en paramètres deux entiers x et y et renvoie la soustraction de y à x.

Exercice 5
Écrire une fonction `produit` qui prend en paramètres deux entiers x et y et retourne leur produit.

Exercice 6
Écrire une fonction `cube` qui prend en paramètre un entier n et renvoie le cube de n.

Exercice 7
Écrire une fonction `sdpc` qui prend en paramètre deux entiers x et y. La fonction calcule la somme de la différence de y à x et du produit de x et y, puis renvoie le cube du résultat. On utilisera les fonctions `somme`, `difference`, `produit` et `cube` définies précedemment.

Exercice 8
Écrire une fonction `est_longueur_paire` qui prend en paramètre une chaine de caractères s et renvoie `True` si sa longueur est paire, `False` sinon.

Exercice 9
Écrire une fonction `est_premier` qui prend en paramètre un entier n et renvoie `True` si c'est un nombre premier, `False` sinon. On rappelle qu'un nombre premier est un nombre qui n'est divisible que par deux nombres : 1 et lui-même. Par exemple, 2 est un nombre premier.

Exercice 10
Écrire une fonction `nombres_premiers` qui prend en paramètre un entier n et affiche tous les nombres premiers jusqu'à n. On utilisera la fonction `est_premier` définie précédemment.

Exercice 11
Écrire une fonction `maximum` qui prend en paramètre trois entiers x, y, z et renvoie le maximum de ces trois entiers. Écrire une fonction `minimum` qui cette fois-ci renvoie le minimum.
Écrire un programme qui demande à l'utilisateur de saisir trois entiers puis lui demande de saisir "maximum" ou "minimum" et lui affiche le maximum/minimum des trois entiers saisies selon le choix de l'utilisateur.