Exercices d'application

Les fonctions sous python

Exercice 1 : Appel de fonctions

La fonction vitesse suivante calcul la vitesse moyenne d'un véhicule en connaissant son temps de parcours et la distance parcourue.

```
def vitesse (temps, distance) :
    '''
    Calcul la vitesse moyenne d'un
    vehicule

    arguments :

    temps ->  reel positif     (secondes)
    distance -> reel positif (metres)

    retour :
    reel positif -> vitesse en m/s
    '''
    vitesse = distance/temps

    return vitesse
```

1. **Entourer** sur le code ci-dessus la spécification de la fonction et d'une couleur différente le corps de la fonction.
2. **Ecrire** l'appel de fonction à effectuer pour obtenir la vitesse moyenne d'un véhicule parcourant 1 km en 45 secondes.
3. **Indiquer** la valeur renvoyée par les instructions suivantes :

```
>>> vitesse(200 , 5000)
```

```
>>> vitesse(5000 , 200)
```

Exercice 2 : Définition et appel de fonctions

On souhaite créer une fonction max2 qui détermine la plus grande valeur de deux valeurs entières.

1. **Indiquer** le nombre d'arguments et de valeurs de retour que possédera cette fonction. **Décrire** chacun de ces arguments / retours.
2. **Définir** et **spécifier** cette fonction max2.
3. **Ecrire** l'appel de fonction à effectuer pour sélectionner la plus grande des valeurs entre 10 et 36.

On souhaite maintenant créer une fonction max3 qui renvoie le plus grand de trois valeurs entières passées en paramètre.

4. **Ecrire** la fonction max3 en se servant de la fonction max2 précédente.

Exercice 3 : portée des variables

1. **Ecrire** une fonction surface(r) qui calcule la surface d'un disque caractérisé par son rayon. La valeur PI sera définie comme une variable globale.

Exercice 4 : Appels multiples

En mathématiques, la factorielle d'un entier naturel n (qui s'écrit **n!**) est le produit des nombres entiers strictement positifs inférieurs ou égaux à n. Par exemple : ***5! = 5 x 4 x 3 x 2 x 1 = 120***
On considère le programme suivant :

```
1  def mult(a, b) :
2      return a * b
3
4  def fact(n) :
5      f = 1
6      for i in range(2,n+1) :
7          f = mult(f,i)
8      return f
```

La table d'exécution suivante représente l'évolution incomplète des variables lors de l'appel fact(4)

Ligne	Etat des variables	Commentaires
4	fact : n 4	Appel fact(4)
5	fact : n 4 f 1	
6	fact : n 4 f 1 i 2	
7	fact : n 4 f 1 i 2	Appel mult(1, 2)
1	mult : a 1 b 2	
2	mult : a 1 b 2	valeur 2 renvoyée
	fact : n f i	
6	fact : n f i	
7	fact : n f i	
1	mult : a b	
2	mult : a b	
	fact : n f i	
6	fact : n f i	
7	fact : n f i	
1	mult : a b	
2	mult : a b	
	fact : n f i	

2. **Compléter** la table d'exécution précédente et écrire la valeur renvoyée par fact(4)

Exercice 5 : Problèmes d'application

3. **Ecrire** une fonction pour vérifier si un nombre est compris entre 0 et 100. Cette fonction doit renvoyer un booléen.

4. **Ecrire** une fonction surface permettant de calculer la surface d'un rectangle définit par sa largeur et sa longueur. Cette fonction doit vérifier que les arguments sont positifs ou nuls.

Exercice 6 : Conversion de coordonnées GPS

Un récepteur GPS a reçu, à 16 h 00 min 0 s 15 ms, le signal émis par un satellite à 15 h 59 min 59 s 910 ms. Le signal se propage à environ 300 000 km/s.

1. **Ecrire** une fonction temps_en_secondes() qui prend en argument un horaire donné en heure, minute, seconde et milliseconde, et qui renvoie son équivalent en seconde.

2. **Écrire** une fonction distance() qui prend en arguments l'horaire d'arrivée du message et son horaire d'émission, en seconde, et qui renvoie la distance parcourue par le signal, en kilomètre.

3. **Interpréter** le résultat de l'instruction suivant :

```
>>>distance(temps_en_secondes(15,59,59
,910),temps_en_secondes(16,0,0,15))
31499.99999877764
```

On souhaite maintenant convertir les coordonnées GPS données en degré décimal (DD) en coordonnées données en degré, minute, seconde (DMS), et inversement.
Pour exprimer une coordonnée GPS, l'unité de base est le degré d'angle (1 tour complet = 360°), puis la minute d'angle (1° = 60'), et la seconde d'angle (1' = 60"). Par exemple, une latitude de 12,345 °(DD) correspond à 12° 20' 42" (DMS).

4. **Écrire** une fonction Python dms_a_decimale() qui prend en argument les valeurs en degré, minute, seconde d'une coordonnée DMS, et qui renvoie la coordonnée GPS en degré décimal.

5. **Écrire** une fonction Python decimale_a_dms() qui prend en argument les coordonnées GPS (uniquement positives) en degré décimal, et renvoie les valeurs correspondantes en degré, minute, seconde.

6. **Tester** ces deux fonctions dans la console à l'aide des deux exemples ci-dessous.

```
>>> decimale_a_dms(12.345)
(12.0, 20.0, 42.0)
>>> dms_a_decimale(12,20,42)
12.345
```