Listes/tuples : exercices du début

Exercice 1

a) On considère le programme suivant :

```
def cube(lon):
    """ calcul du volume et de l'aire """
    vol = lon**3
    surf = (lon**2)*6
    return (vol,surf)

c = int(input("longueur de l'arête du cube ? "))
(v,s)=cube(c)
print("volume : ",v,"   surface : ",s)
```

Comment appelle-t-on en Python `(vol,surf)` et `(v,s)` ?
Que fait la ligne `(v,s)=cube(c)`? Comment s'appelle cette opération ?
Qu'obtient-on lorsque l'utilisateur saisit 2 ?
Qu'obtiendrait-on avec `print(cube(c))` ?

b) On a ajouté une nouvelle fonction au programme précédent :

```
def affiche(n):
    for i in range(1,n+1):
        (v,s)=cube(i)
        print("cube de côté ",i," : volume ",v,"   surface",s)

p = int(input("Saisir un entier : "))
affiche(p)
```

Que fait cette fonction ? Qu'obtient-on lorsque l'utilisateur saisit 3 pour p ?

c) De façon analogue, créer un programme avec une fonction sphère de paramètre *r* le rayon qui retourne un tuple avec le volume et la surface, on rappelle les formules : $V = 4/3\pi r^3$ et $S = 4 \pi r^2$ (**pi** est dans le module **math**). Dans le corps principal du programme, on affichera les résultats en dépaquetant le tuple comme dans la question a).

d) Ajouter une fonction, de paramètre un entier n, qui réalise un affichage du volume et de la surface des sphères de rayon : 1, 2, …, n (analogue à la question b)).

Exercice 2

On considère les instructions suivantes :

```
numeros = [543,318,68,54,597]
print(len(numeros))
print(numeros[1])
for x in numeros :
     x+=1
print(numeros)
for i in range(len(numeros)) :
     numeros[i]+=1
print(numeros)
```

a) Comment s'appelle en Python le type de la variable `numeros` ?
b) A quoi correspond `len(numeros)` ?
c) Quelle est la valeur de `numeros[1]` ?

d) Quelles sont les valeurs successives de x lorsque l'on écrit `for x in numeros` ?
e) Il y a-t-il une différence entre les deux boucles for ? Le tester sous spyder.
f) Que peut-on écrire pour rallonger `numeros` avec l'entier 745 ? Donner deux possibilités.

Exercice 3 : Jeu

Ecrire un programme avec deux fonctions :

- une fonction **saisit**() qui demande à l'utilisateur de saisir un entier et retourne cet entier (on doit redemander tant que la saisie n'est pas valide,
- une fonction **joue**() qui appelle la fonction saisit(), puis crée une liste de cinq chiffres (entier entre 0 et 9, il peut y avoir des doublons), si l'entier saisi par l'utilisateur est présent dans la liste, il a gagné, sinon il a perdu (la fonction joue() affiche le résultat, elle ne retourne rien).
  Exemples d'exécution :

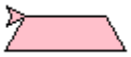
  Saisir un chiffre : 8
  8 est dans [8, 9, 4, 6, 3] : vous avez gagné :-)

  Ou encore :

  Saisir un chiffre : 6
  6 n'est pas dans [7, 2, 7, 4, 3] : vous avez perdu :-(

Exercice 4 : Triangle

a) Ecrire une fonction **trapeze**(*base, cote, sup, couleur*) qui dessine à l'aide du module **turtle** un trapèze symétrique de longueur de base *base*, de longueur de côté *cote*, le trait supérieur a pour longueur *sup*, les angles en bas à droite et à gauche sont de 60°, les angles en haut à droite et à gauche sont de 120° (les dimensions entrées sont liées pour que cela donne bien un trapèze, on suppose ici que les paramètres conviennent) . On remplit le trapèze avec *couleur* (utiliser **begin_fill**() , **fillcolor**(nomCouleur), **end_fill**()). On ne vérifie pas ici que les paramètres *base, coté, sup* donnent bien un trapèze, on pourra tester en prenant base=60, cote=20, sup=40. Par exemple trapeze(60,20,40,"pink") dessine :



b) Ajouter une fonction **triangle** avec deux paramètres : une longueur et une liste de trois couleurs, qui dessine un triangle équilatéral comme ci-contre, triangle obtenu ici avec triangle(100,["blue","green","yellow"])
On rappelle qu'en Python existe la possibilité de paramètres optionnels, c'est le cas ici, on mettra le paramètre *couleurs* optionnel, ses valeurs par défaut seront : ["orange","purple","red"]). Donc l'appel triangle(100) doit donner :