UNIVERSITÉ JEAN MONNET SAINT-ÉTIENNE

Info. - TD : Feuille 1

Rappel : la fonction `split` permet de découper une chaine de caractères en sous-parties. Ainsi `str.split("a b c", " ")` (ou `"a b c".split(" ")`) retournent la liste `["a", "b", "c"]`.

Exercice 1 – Prix de gros (15 min.)

Supposons les règles suivantes pour le calcul du prix d'un produit :

- on paye 10€/kg pour le premier kilogramme,
- on paye 8€/kg pour les kilos de 1kg à 5kg,
- on paye 7/kg€ pour les kilos au delà de 5kg,

Par exemple, 900g coûtent 9€, 1kg coûte 10€, 1100g coûtent 10.80€, 5kg coûtent 42€, 10kg coûtent 77€.

Q1) Tracer approximativement, à la main, la courbe $prix = f(poids)$.

Q2) En conservant l'indentation, réorganiser les lignes ci dessous pour écrire un programme qui demande un poids (réel, en kilos), et calcule puis affiche le prix correspondant.

```
p = float(input("Entrer le poids en kg : "))
else:
if p <= 1:
    else: # p > 5
    prix = 10 * p
    if p <= 5:
        prix = 10*1 + 8*4 + 7*(p-5)
        prix = 10*1 + 8*(p-1)
print(prix)
```

Q3) Simplifier le programme grâce à `elif` (pour *else if*).

Exercice 2 – Points au tir à l'arc (25 min.)

Q4) Écrire un programme qui demande à l'utilisateur d'entrer les coordonnées (réelles, en cm) x et y d'une flèche dans une cible, à l'aide de 2 `input`.

Q5) En supposant que le centre de la cible a pour coordonnées $(0, 0)$, faire que le programme calcule et affiche la valeur de la flèche : 10 points si elle est dans le centre (2cm de rayon), 9 points ensuite (2cm suivants, …). Le programme peut utiliser `elif`.

Q6) En reprenant le programme précédent, écrire une fonction `points_fleche(d)` qui renvoie le nombre de point d'une flèche qui se trouve à une distance d du centre.

{t:} **Q7)** Ré-écrire le programme en utilisant cette nouvelle fonction.

Q8) Comment faut-il changer le programme pour qu'il ne fasse qu'un seul `input` et demande les deux coordonnées,n: séparées par un espace ? },

Q9) Écrire un programme qui demande maintenant les coordonnées des 3 flèches (donc 6 réels, séparés par des espaces) et affiche la somme des points des 3 flèches. Par exemple, si l'utilisateur rentre `1.2 -1 0 0 -0.1 3` cela affiche `29` (2 flèches à 10 points et 1 à 9).

Q10) Imaginer (sans forcément l'écrire) le programme avec 4, 5, 6, …, 20 flèches.

Q11) Si ce n'est pas déjà le cas, faites que le programme accepte un nombre n de flèches (donc $2n$ nombres réels, séparés par des espaces) et affiche la somme des points de ces flèches. Dans le cas de la question précédente, il doit avoir le même comportement, et si on lui entre par exemple `1.2 -1 0 0 -0.1 3 0 0 0 0` cela affiche `49` (4 flèches à 10 points et 1 à 9).

Exercice 3 – Opération sur « vecteurs » (30 min.)

Dans cette exercice on va utiliser des listes Python pour représenter des « vecteurs » (à 2 et plus de dimensions). Ainsi, un vecteur avec 5 dimensions sera représenté par une liste de 5 réels.

Q12) En réorganisant les lignes ci-dessous, écrire une fonction qui calcule la norme euclidienne d'un vecteur (c'est à dire la racine de la somme des carrés de ses éléments).

```
def norme(v): # version simple
    for elem in v:
    return s
    s = 0
    s = s**0.5
        s = s + elem**2
```

Python propose une syntaxe (notation) appelée « liste en compréhension ». Cette syntaxe est une sorte de mélange entre la notation pour créer une nouvelle liste `[ ... ]` et celle du `for` (il est possible d'ajouter en plus un `if`). Voici un exemple qui calcule une liste des `log` des nombres issus d'une liste `lll` :

```
resultat = [ math.log(elem) for elem in lll ]
```

Q13) En utilisant une « liste en compréhension » et la fonction `sum(...)` (qui calcule la somme d'une liste), ré-écrire la fonction `norme(v)`.

Q14) Écrire une fonction `somme(a,b)` qui calcule et renvoie la somme élément par élément de deux vecteurs a et b (de même dimension), telle que si $c = somme(a, b)$ alors $\forall i, c_i = a_i + b_i$. Écrire une version avec `append` et une version avec une liste en compréhension.

Q15) De même, écrire une fonction `produit(a,b)` qui calcule et renvoie le produit élément par élément de deux vecteurs a et b (de même dimension), tel que si $c = produit(a, b)$ alors $\forall i, c_i = a_i * b_i$

Q16) En réutilisant certaine(s) des fonctions précédentes, écrire une fonction `produit_scalaire(a, b)` qui calcule et renvoie le produit scalaire de deux vecteurs de même dimensions d, défini par : $\sum_{i=1}^{d} a_i * b_i$

Exercice 5 – Le retour du tir à l'arc (20 min.)

Q19) Le fait de saisir des paires de nombres séparés par des espaces est peu lisible et peu pratique. Écrire un programme qui s'attend à avoir les flèches séparées par `|`. Il voudrait par exemple recevoir `1.2 -1|0 0|-0.1 3|0 0|0 0`.

Q20) Faites que le programme affiche des détails de calcul sous cette forme (pour l'exemple précédent) :

```
Il y a 5 flèches, pour un total de 49 points :
- flèche 1 : 10
- flèche 2 : 10
- flèche 3 : 9
- flèche 4 : 10
- flèche 5 : 10
```

Q21) Faites que le programme affiche `49 points` mais `1 point` (quand il n'y en a qu'un), et de même pour le nombre de flèches.

Q22) Faites que le programme affiche des détails de calcul sous cette forme (pour l'exemple précédent) :

```
Il y a 5 flèches, pour un total de 49 points :
- 4 flèches à 10 points
- 1 flèche à 9 points
```

Exercice 6 – Poum Tchak (10 min.)

Q23) Écrire un programme qui affiche les nombres de 1 à 100 (chacun sur un ligne), sauf que :

- pour les multiples de 3, il affiche `Poum`,
- pour les multiples de 5, il affiche `Tchak`,
- pour les nombres qui sont multiples de 3 et de 5, il affiche `PoumTchak`.