TP noté
Durée : 1h30

I Exercices sur feuille. Durée : 30 min

Exercice 1 : Lecture

Question 1. Que vaut k ?

```
x,y = 2,3
d,c = 5,4
if x>0 and x<d :
    if y>0 and y<c :
        k = y
    else :
        k = x
else :
    k = 1
```

Question 2. Qu'affiche Python ?

```
a = 2
b = 5
for i in range (3,14) :
    print(a)
    c = a+b
    a = b
    b = c
```

Question 3. Qu'affiche Python ?

```
i = 7
while i<=21 :
    i = i+1
    if i%3 == 0 :
        i = i+1
    else :
        print(i)
```

Exercice 2 : Fonction puissance 3

Ecrire une fonction Python `cube(x)` prenant en argument un nombre (entier ou flottant) `x` et renvoyant la valeur de son cube x^3.

Exercice 3 : Produit

Ecrire une fonction Python `produit(L)` prenant en argument une liste d'entiers `L` et renvoyant un entier égal au produit de ses éléments.

Exercice 4 : Map

Ecrire une fonction Python `map(L,f)` prenant en arguments une liste d'entiers `L` et une fonction `f` définie sur les entiers. Cette fonction devra construire puis renvoyer la liste des éléments de `L` auxquels on a appliqué la fonction `f`.

Par exemple, `map([1,2,3],cube)` devra renvoyer `[1,8,27]`.

Exercice 5 : Appartenance

1. Ecrire une fonction Python `appartient(L,x)` prenant en arguments une liste d'entiers `L` et un entier `x`. Cette fonction devra renvoyer le booléen `True` si l'entier `x` apparaît dans la liste `L` et le booléen `False` sinon.
2. Ecrire une fonction Python `appartient_tab(L,x)` prenant en argument une liste de listes d'entiers `L` et un entier `x`. Cette fonction devra renvoyer le booléen `True` si l'entier `x` apparaît dans l'une des sous listes de la liste `L` et le booléen `False` sinon.

II Exercices sur machine. Durée : 1h

Exercice 1 : Somme à calculer

Ecrire une fonction `sommation(n)` qui prend en argument un entier naturel `n` et qui renvoie le calcul de la somme suivante $\sum_{1 \leq i < j \leq n} \frac{1}{i+j}$.

Exercice 2 : Somme de fractions

On représente une fraction par une liste de deux entiers, le premier élément de la liste étant le numérateur. Par exemple, la fraction $\frac{3}{2}$ est représentée par la liste `[3,2]`

Ecrire une fonction Python `somme_frac(p,q)` prenant en en arguments deux listes `p` et `q` représentants des fractions et renvoyant une liste représentant la somme de ces deux fractions.

On ne demande pas de simplifier la fraction obtenue.

Exercice 3 : Nombre d'occurences dans une chaîne

Ecrire une fonction Python `occurence(chaine)` prenant en argument une chaine de caractères `chaine` que l'on pourra supposer non vide, et renvoyant le nombre d'occurences dans cette chaine de son premier caractère (la première apparition incluse).

Par exemple, l'appel `occurence('baobab')` devra renvoyer `3`.

Exercice 4 : Liste triée ?

Ecrire une fonction Python `croissant(L)` prenant en argument une liste d'entiers `L` et renvoyant le booléen `True` si cette liste est triée dans l'ordre croissant et le booléen `False` sinon.

Exercice 5 : Insertion d'élément

Ecrire une fonction Python `inserer(L,x)` prenant en argument une liste `L` triée dans l'ordre croissant et un élément `e`, et renvoyant une nouvelle liste triée dans l'ordre croissant et contenant les éléments de `L` plus la valeur `e`.

Exercice 6 : Supprimmer les doublons

Écrire une fonction Python `supprimeDoublons(L)` prenant en paramètre une liste d'entiers `L` et retournant une nouvelle liste sans éléments dupliqués. Par exemple, l'appel `supprimeDoublons([6,1,0,6,7,9,1,8,8,5])` retourne `[6,1,0,7,9,8,5]`.

Exercice 7 : Triangle de pascal

Écrire une fonction Python `triangleDePascal(n)` prenant en paramètre un entier naturel `n` et retournant une liste de constituée de `n` sous-listes. La $i^{\text{ième}}$ sous-liste devra représenter la $i^{\text{ième}}$ ligne du triangle de Pascal. Par exemple, l'appel `triangleDePascal(4)` renvoie `[[1],[1,1],[1,2,1],[1,3,3,1]]`.

Exercice 8 : Polynômes

Dans ce problème, on représente un polynôme par une liste contenant ses coefficients. Par convention, le premier élément de la liste correspondra au terme constant et le dernier élément de la liste correspondra au terme de plus haut degré.Il faudra, d'une part, faire attention à ne pas oublier les zéros correspondants à l'absence de certains termes et, d'autre part, considérer la non unicité de la représentation d'un polynôme. En effet, rajouter des zéros en fin de liste consisterait à expliciter l'absence de coefficient de degré supérieur et donc ne changerait pas le polynôme.

Par exemple, le polynôme $6+5X-2X^3+7X^5$ peut être représenté par la liste `[6,5,0,-2,0,7]` ou encore par la liste `[6,5,0,-2,0,7,0,0]`.

On ne considérera, dans ce problème, que des coefficients entiers.

1. **Evaluation :** Ecrire une fonction Python `Evaluation(P,x)` prenant en arguments une liste d'entiers `P` représentant un polynôme et un entier `x`. Cette fonction devra renvoyer le résultat de l'évaluation de `P` au point `x`.

   Par exemple, `Evaluation([5,3,2],10)` devra renvoyer $5+3\times 10+2\times 10^2=235$.

2. **Degré d'un polynôme :** Ecrire une fonction Python `Degre(P)` prenant en argument une liste d'entiers `P` représentant un polynôme et renvoyant le degré du polynôme représenté. Attention au cas ou la liste finit par des zéros.

   Par exemple, l'appel `Degre([0,1,0,4,0,0])` devra renvoyer `3`.

3. **Somme :** Ecrire une fonction Python `Somme(P1,P2)` prenant en arguments deux listes d'entiers `P1` et `P2` représentant chacune un polynôme et renvoyant une liste représentant la somme de ces deux polynômes.