Introduction à la Programmation 1 PYTHON

51AE011F

Séance 7 de cours/TD

Université Paris-Diderot

Objectifs:

— Connaître ses classiques sur les listes.

— Apprendre à voir quand un programme est faux.

1 Devenir incollable sur les listes

Opérations à savoir faire sur les listes [COURS]

- **Rechercher**
  - Dire si un élément est présent dans une liste
  - Renvoyer la position d'un élément dans la liste (la première ou la derrière)
  - Compter le nombre de fois qu'un élément est présent dans une liste.
  - Savoir construire une liste des positions d'un élément dans une liste
- **Modifier**
  - Savoir échanger deux éléments de place dans une liste
  - Savoir renverser une liste
  - Savoir décaler tous les éléments d'une liste
  - Savoir appliquer une fonction à tous les éléments d'une liste

Exercice 1 (Présence dans une liste, ⋆)
On considère la fonction `isPresent (el, li)` *qui renvoie* `True` *si* `el` *est dans la liste* `li`.

```
def isPresent (el, li) :
    for i in range (0, 1, len(li)) :
        if (li[i] == el) :
            return (True)
        else :
            return (False)
    return (False)

def isPresent (el, li) :
    b = True
    for i in range (0, 1, len(li)) :
        if (li[i] == el) :
            b = True
        else :
            b = False
    return (b)

```

```python
def isPresent (el, li) :
    b = True
    for i in range (0, 1, len(li)) :
        return (li[i] == el)

def isPresent (el, li) :
    b = False
    i = 0
    while ( i < len (el) and b) :
        if (li[i] == el) :
            b = True
    return (b)

def isPresent (el, li) :
    b = False
    i = 0
    while ( i < len (el) and not(b)) :
        b = b and (li[i] == el)
    return (b)
```

code/exoPresent.py

1. *Pourquoi les propositions précédentes sont-elles fausses ? À chaque cas, donnez un exemple de listes d'entiers et d'éléments pour laquelle la fonction renvoie un mauvais résultat.*
2. *Proposez une correction pour chacun des cas.*

□

Exercice 2 (Occurences dans une liste, ⋆)
On considère la fonction `occ (el, li)` *qui renvoie le premier indice de* `el` *dans la liste* `el` *et* `-1` *si* `el` *n'est pas dans la liste.*

```python
def occ (el, li) :
    for i in range (0, 1, len(li)) :
        if (el == li[i]) :
            return (i)
        else :
            return (-1)
    return (-1)

def occ (el, li) :
    for i in range (0, 1, len(li)) :
        if (el == li[i]) :
            return (i)
    return (i)

def occ (el, li) :
    p = 0
    for i in range (0, 1, len(li)) :
        if (el == li[i]) :
            p = i
        else :
            p = -1
    return (p)

def occ (el, li) :
    p = 0
    for i in range (1, 1, len(li)) :
```

```
        if (el == li[i]) :
            p = i
    return (p)

def occ (el, li) :
    p = -1
    i = 0
    while ( i < len(li)) :
        if (el == li [i]) :
            p = i
            i = i + 1
    return (p)

def occ (el, li) :
    p = -1
    i = 0
    while ( i < len(li) and p != -1 ) :
        if (el == li [i]) :
            p = i
        i = i + 1
    return (p)
```

code/exoOccurence.py

.

1. *Pourquoi les propositions précédentes sont-elles fausses ? À chaque cas, donnez une exemple de listes d'entiers et d'éléments pour laquelle la fonction renvoie un mauvais résultat.*
2. *Proposez une correction pour chacun des cas.*

□

Exercice 3 (Compter dans une liste, ⋆)

On considère la fonction `count (el, li)` *qui renvoie le nombre de fois que l'élément* `el` *apparaît dans la liste* `li`.

```
def count (el, li) :
    c = 0
    for i in range (0, 1, len(li)) :
        if (li[i] == el) :
            c = c + 1
            return (c)
    return (c)

def count (el, li) :
    c = 0
    for i in range (0, 1, len(li)) :
        if (li[i] == el) :
            c = c + 1
        else :
            c = c - 1
    return (c)
```

code/exoCounting.py

.

1. *Pourquoi les propositions précédentes sont-elles fausses ? À chaque cas, donnez une exemple de listes d'entiers et d'éléments pour laquelle la fonction renvoie un mauvais résultat.*
2. *Proposez une correction pour chacun des cas.*

□

Exercice 4 (Construire la liste des positions, ⋆)
On considère la fonction `pos (el, li)` *qui renvoie une liste contenant les positions de* `el` *apparaît dans la liste* `li`.

1. *Que valent les variables* `li1`, `li2` *et* `li3` *après exécution du programme suivant (en supposant que la fonction* `pos` *est correctement définie).*

```
li1 = pos (2, [3, 4, 5, 8])
li2 = pos (3, [3, 3, 4, 3, 4, 5, 6, 4, 3])
li3 = pos (4, [3, 3, 4, 3, 4, 5, 6, 4, 3])
```

2. *On considère les propositions suivantes pour la fonction* `pos (el, li)` *(on suppose que la fonction* `count` *est celle de l'exercice précédent et qu'elle est correcte).*

```
def pos (el, li) :
    lret = []
    for i in range (0, 1, len(li)) :
        if (li[i] == el) :
            lret = [i]
    return lret

def pos (el, li) :
    n = count (el, li)
    lret = [0] * n
    for i in range (0, 1, len(li)) :
        if (li[i] == el) :
            lret[i] = i
    return lret

def pos (el, li) :
    n = count (el, li)
    lret = [0] * n
    for i in range (0, 1, len(li)) :
        if (li[i] == el) :
            lret[el] = i
    return lret

def pos (el, li) :
    n = count (el, li)
    lret = [0] * n
    j = 0
    for i in range (0, 1, len(li)) :
        if (li[i] == el) :
            lret[j] = i
    return lret
```

code/exoPos.py

   a *Pourquoi les propositions précédentes sont-elles fausses ? À chaque cas, donnez une exemple de listes d'entiers et d'éléments pour laquelle la fonction renvoie un mauvais résultat.*

   b *Proposez une correction.*

□

Exercice 5 (Échanger deux éléments dans une liste, ⋆)
Dans cet exercice, on considère la fonction `swap (i , j, li)` *qui échangent les valeurs des indices* `i` *et* `j` *de la liste* `li`.

1. *Pour comprendre ce qui se passe, échauffons-nous et dîtes ce qu'affiche le programme suivant :*

```python
def modif (l) :
    l[0] = 5

li = [1, 2, 4, 8, 16, 31]
modif (li)
print (li)
```

code/exoSwap1.py

2. *En déduire ce que doit retourner la fonction* `swap`.
3. *Pourquoi la définition suivante de* `swap` *n'est pas correcte ? Proposez une correction.*

```python
def swap (i , j, li) :
    if (i < len (li) and j < len(li)) :
        li[i] = li[j]
        li[j] = li[i]
```

code/exoSwap2.py

□

Exercice 6 (Inverser une liste d'éléments, ⋆)

Dans cet exercice, on considère la fonction `reverse (li)` *qui prend en paramètre une liste* `li` *et renvoie la liste où l'ordre des valeurs est inversée.*

1. *Que renvoie donc* `reverse ([0, 1, 2, 3, 4, 5])`.

```python
def reverse (li) :
    for i in range (0, 1, len (li)) :
        li[i] = li [len - 1 - i]
    return li

def reverse (li) :
    for i in range (0, 1, len (li)) :
        temp = li[i]
        li[i] = li [len - 1 - i]
        li [len - 1 - i] = temp
    return li

def reverse (li) :
    l = []
    for i in range (0, 1, len (li)) :
        l [i] = li [len - 1 - i]
    return li
```

code/exoReverse.py

2. a *Pourquoi les propositions précédentes sont-elles fausses ? À chaque cas, donnez une exemple de listes d'entiers et d'éléments pour laquelle la fonction renvoie un mauvais résultat.*

   b *Proposez une correction.*

   c *Parmi les corrections proposées, quelles sont celles qui changent la liste* `li` *passée en paramètre et quelles sont celles qui la laissent inchangée ?*

□

Exercice 7 (Décaler les éléments d'une liste, ⋆⋆)

Dans cet exercice, on considère la fonction `shift (li, p)` *qui prend en paramètre une liste* `li` *et modifie cette liste en décalant (de façon cyclique vers la droite) ces éléments de* `p` *positions (en supposant que* `p` *est positif).*

1. *Que valent les variables* `li1`, `li2` *et* `li3` *après exécution du programme suivant (en supposant que la fonction* `swap` *est correctement définie).*

```
li1 = [3, 4, 5, 8]
swap (li1, 1)
li2 = [3, 3, 4, 3, 4, 5]
swap (li2, 2 )
li3 = [1, 2, 3, 4, 5, 6]
swap (li3, 7)
```

```
def shift (li, p) :
    for i in range (0, 1, len (li)) :
        li [i] = li [i - p]

def shift (li, p) :
    for i in range (0, 1, len (li)) :
        temp = li [i]
        li [i - p] = temp

def shift (li, p) :
    for i in range (0, 1, len (li)) :
        li [(i + p) % len(li)] = li [i]
```

code/exoShift.py

2. a *Pourquoi les propositions précédentes sont-elles fausses ? À chaque cas, donnez une exemple de listes d'entiers et d'éléments pour laquelle la fonction renvoie un mauvais résultat.*

   b *Proposez une correction.*

□

2 DIY

Pour certains exercices, un "contrat" est proposé : ce sont des tests que nous vous fournissons pour vérifier votre réponse. Si votre réponse ne passe pas ces tests, il y a une erreur ; si votre réponse passe les tests, rien n'est garanti, mais c'est vraisemblablement proche de la réponse. Quand aucun contrat n'est spécifié, à vous de trouver des tests à faire.

Exercice 8 (Primalité, ⋆⋆)
Un entier p est premier si $p \geq 2$ et s'il n'a pas d'autres diviseurs que 1 et lui-même.

1. *Écrire une fonction* `prime` *qui prend en paramètre un entier n et qui renvoie* `True` *si n est premier, ou* `False` *sinon.*
2. *Écrire une fonction* `next` *qui prend en entrée un entier x et qui renvoie le plus petit nombre premier $p \geq x$. On pourra bien sûr se servir de la fonction* `prime` *précédente.*
3. *Écrire une fonction* `number` *qui prend en entrée un entier y et qui renvoie le nombre de nombres premiers $p \leq y$. On pourra bien sûr se servir de la fonction* `prime`.

Contrat:
Pour la fonction `next` *:*

$x = 2 \quad \to \quad 2$
$x = 10 \quad \to \quad 11$
$x = 20 \quad \to \quad 23$

Pour la fonction `number` *:*

$x = 10 \quad \to \quad 4$
$x = 20 \quad \to \quad 8$

□

Exercice 9 (Multiplication de matrices, $\star\star$)
Dans cet exercice, on supposera qu'une matrice A à m lignes et n colonnes sera encodé par une liste de listes contenant m listes de taille n. On rappelle qu'étant donné deux matrices A (à m lignes et n colonnes) et B (à n lignes et p colonnes), le produit $A \cdot B$ est une matrice à m ligne s et p colonnes telle que pour tout $1 \leq i \leq m$ et pour tout $1 \leq j \leq p$, le coefficient $C_{i,j}$ de C se trouvant à la i-ème ligne et la j-ième colonne est égal à :

$$C_{i,j} = \Sigma_{1 \leq k \leq} A_{i,k} B_{k,j}$$

1. *Écrire une fonction* `isMatrix (A)` *qui prend en paramètre une liste de listes d'entiers et vérifie qu'il s'agit de l'encodage d'une matrice (c'est à dire que chaque sous-liste à la même longueur), elle renvoie* `True` *dans ce cas et* `False` *sinon.*
2. *Écrire une fonction* `nbLines (A)` *qui prend en paramètre une liste de listes d'entiers qui encode une matrice et renvoie son nombre de lignes.*
3. *Écrire une fonction* `nbColumns (A)` *qui prend en paramètre une liste de listes d'entiers qui encode une matrice et renvoie son nombre de colonnes.*
4. *Écrire une fonction*`matriProx (A, B)` *qui prend en paramètre deux listes de listes d'entiers, vérifie qu'il s'agit de deux matrices, vérifie que le nombre de colonnes de* `A` *et égal au nombre de lignes de* `B`*, si ces conditions ne sont pas satisfaites, la fonction renvoie* `[]` *et sinon elle renvoie une liste de listes contenant la matrice correspondant au produit matriciel de* `A` *par* `B`*.*

□

Exercice 10 (Addition, $\star\star\star$)
Le but de cet exercice est de programmer l'addition décimale. Les deux nombres à additionner sont donnés sous forme de listes.

Par exemple, `x = [7, 4, 3]` *et* `y = [1, 9]`*, dont la somme doit être retournée sous forme de liste, dans cet exemple,* `[7, 6, 2]`*.*

Écrire une fonction `add` *qui prend en paramètre deux listes et fait l'addition des deux nombres représentés par les listes.*

Par exemple, si les entrées sont les listes `t1 = [3, 4, 7], t2 =[9, 1]`*, cela représente la somme* $743+19$*. On calcule d'abord* $9+3$ *ce qui fait* 2 *avec une retenue de* 1*. Puis on calcule* $4+1$*, plus la retenue, ce qui donne* 6*, avec une retenue de* 0*. Enfin, le dernier chiffre est* 7*. À la fin, on doit renvoyer la liste* `[0, 7, 6, 2]`*.*
Remarque : *Le premier chiffre dans la liste (celui le plus à gauche) peut-être* 0*.*

Contrat:

`t1 = [2, 4, 3, 5]`	`t2 = [4, 3, 6]`	→	*renvoie :* `[0, 2, 8, 7, 1]`
`t1 = [1, 2]`	`t2 = [1, 3]`	→	*renvoie :* `[0, 2, 5]`
`t1 = [6, 1, 2]`	`t2 = [0, 6, 3, 0]`	→	*renvoie :* `[1, 2, 4, 2]`

□

Exercice 11 (Pascal, $\star\star\star$)
Écrire un programme qui affiche le triangle de Pascal jusqu'au rang n où n est un entier demandé à l'utilisateur. Si l'utilisateur rentre 4, le programme affichera :

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
```

Le programme doit faire appel de deux fonctions auxiliaire la première qui, étant donné n, renvoie le triangle de Pascal sous forme d'une liste de listes d'entiers de n + 1 lignes, la i-ème ligne contenant les coefficients de la i-ème puissance du binôme $(a+b)$ et la deuxième qui sert à afficher une liste de liste en affichant sur chaque ligne le contenu de chacune des listes (en séparant les différentes valeurs par des espaces). □