

CORRIGÉ DU DS D'INFORMATIQUE — 8 JANVIER 2020

Durée : 1 heure — Calculatrices interdites

EXERCICE 1 — LISTE

- Proposer un programme qui demande à l'utilisateur de saisir un entier strictement positif N , puis une liste de N nombres entiers, et qui affiche le maximum de ces entiers.

```

1  print('Choisissez un entier strictement positif N:')
2  N = int(input())
3
4  L = []
5
6  for k in range(N):
7      print('Choisissez un entier :')
8      E = int(input())
9      L = L + [E]
10
11  MAX = L[0]
12  for k in range(1, N):
13      if L[k] > MAX:
14          MAX = L[k]
15  print('La valeur maximale est', MAX)
```

- Modifier le programme précédent pour qu'il précise si au moins l'un des entiers saisis par l'utilisateur est pair.

```

1  print('Choisissez un entier strictement positif N:')
2  N = int(input())
3
4  L = []
5  T = False
6
7  for k in range(N):
8      print('Choisissez un entier :')
9      E = int(input())
10     L = L + [E]
11     if E % 2 == 0:
12         T = True
13
14  MAX = L[0]
15  for k in range(1, N):
16      if L[k] > MAX:
17          MAX = L[k]
18  print('La valeur maximale est', MAX)
19  if T:
20      print('Au moins un entier saisi est pair')
21  else:
22      print('Tous les entiers saisis sont impairs')
```

3. Modifier le programme précédent pour qu'il affiche la valeur moyenne des entiers saisis par l'utilisateur.

```

1  print(' Choisissez un entier strictement positif N:')
2  N = int(input())
3
4  S = 0
5
6  for k in range(N):
7      print(' Choisissez un entier :')
8      E = int(input())
9      S = S + E
10
11 print(' La valeur moyenne est', S/N)

```

EXERCICE 2 — CHAÎNES DE CARACTÈRES

Proposer un programme qui demande à l'utilisateur de saisir une phrase, et qui supprime les espaces de cette phrase avant d'afficher le résultat.

```

1  print(' Saisissez une phrase:')
2  PH = input()
3
4  PH2 = ''
5
6  for CAR in PH:
7      if CAR != ' ':
8          PH2 = PH2 + CAR
9
10 print(PH2)

```

EXERCICE 3 — CHAÎNES DE CARACTÈRES

1. Ecrire un programme qui demande à l'utilisateur de saisir une chaîne de caractères (appelée MOT) et un caractère (appelé CAR); ce programme doit afficher **True** si le caractère CAR appartient à la chaîne MOT, et **False** sinon.

```

1
2  # SOLUTION 1
3  print(' Saisissez un mot:')
4  MOT = input()
5
6  print(' Saisissez un caractère:')
7  CAR = input()
8
9  T = False
10
11 for L in MOT:
12     if L == CAR:
13         T = True
14
15 print(T)

```

```

1  # SOLUTION 2 --- THE FASTER, THE BETTER!
2  print (' Saisissez un mot:')
3  MOT =input()
4
5  print (' Saisissez un caractère:')
6  CAR =input()
7
8  print (CAR in MOT)

```

2. On souhaite écrire un script qui demande à l'utilisateur de saisir une chaîne de caractères **MOT** et un caractère **CAR**, et qui affiche la liste des positions où apparaît le caractère **CAR** dans la chaîne **MOT**.

Par exemple, si l'utilisateur saisit la chaîne "Quel beau texte", et le caractère "u", l'affichage produit doit être : [1,8].

Compléter le code ci-dessous (recopier et compléter sur votre copie seulement les lignes 6-7-8) :

```

1  print (' Saisissez un mot')
2  MOT =input()
3  print (' Saisissez un caractère ')
4  CAR =input()
5  LPOS =[]
6  for k in range(len(MOT)): # A Compléter
7      if MOT[k] ==CAR: # A Compléter
8          LPOS =LPOS +[k] # A Compléter
9  print (LPOS)

```

3. Comment compléter (rapidement) le programme précédent pour qu'il affiche également le nombre de fois où le caractère **CAR** apparaît dans la chaîne **MOT** ?

```

1  print (' Saisissez un mot')
2  MOT =input()
3  print (' Saisissez un caractère ')
4  CAR =input()
5  NB =0
6  LPOS =[]
7  for k in range(len(MOT)):
8      if MOT[k] ==CAR:
9          LPOS =LPOS +[k]
10         NB =NB +1
11  print (LPOS, NB)

```

4. (*) Proposer un programme demandant à l'utilisateur de saisir une phrase au clavier, et qui affiche la voyelle (*a, e, i, o, u* ou *y*) qui a été la plus utilisée dans cette phrase, ainsi que le nombre de fois où elle apparaît.

```

1  print(' Saisissez un mot!')
2  MOT =input()
3
4  VOY ='aeiouy'
5
6  L=[0,0,0,0,0,0]
7
8  for CAR in MOT:
9      for k in range(6):
10         if CAR ==VOY[k]:
11             L[k] =L[k] +1
12
13  MAX =L[0]
14  PMAX =0
15  for k in range(1,6):
16      if L[k] > MAX:
17          MAX =L[k]
18          PMAX =k
19
20  print('La voyelle la plus utilisée est : ', VOY[PMAX], '. Elle apparaît ', MAX, ' fois.')
```

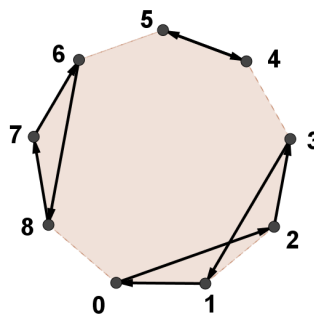
EXERCICE 4 — NOËL CANADIEN

Représentation et structure des données. Nous supposons que les individus sont numérotés de 0 à $(n-1)$, où n est le nombre total d'individus. Tout au long du problème, n désignera un entier supérieur ou égal à 2.

Les résultats du tirage au sort pourront être représentés sous la forme d'une liste, comme expliqué dans les deux exemples ci-dessous.

Exemple 2. On suppose cette fois que le groupe est constitué de 9 individus. Il se peut que le tirage au sort produise des sous-ensembles disjoints d'individus, comme l'illustre le résultat du tirage au sort suivant :

- 0 offre un cadeau à 2;
- 1 offre un cadeau à 0;
- 2 offre un cadeau à 3;
- 3 offre un cadeau à 1;
- 4 offre un cadeau à 5;
- 5 offre un cadeau à 4;
- 6 offre un cadeau à 8;
- 7 offre un cadeau à 6;
- 8 offre un cadeau à 7.



Ces résultats seront alors symbolisés par la **représentation** : $[0, 2, 3, 1, 'X', 4, 5, 'X', 6, 8, 7]$.

Les caractères 'X' permettront donc de séparer les trois sous-listes $[0, 2, 3, 1]$, $[4, 5]$ et $[6, 8, 7]$. En outre, de telles sous-listes seront appelés des **cycles**, et plus précisément nous dirons que :

$[0, 2, 3, 1]$ est un **cycle de longueur 4**, $[4, 5]$ un **cycle de longueur 2** et $[6, 8, 7]$ un **cycle de longueur 3**.

En résumé.

- Dans un groupe de n individus, chaque individu est représenté par un unique entier compris entre 0 et $(n - 1)$ (et réciproquement chaque entier de $\llbracket 0, n - 1 \rrbracket$ correspond à un unique individu). Dans ce problème, on assimilera donc un individu du groupe à un entier compris entre 0 et $(n - 1)$;
- Une **représentation** est une liste correspondant au résultat d'un tirage au sort ;
- La lecture de cette liste permet de savoir que l'entier (l'individu) situé à la i -ème place offre un cadeau à celui situé à la $(i + 1)$ -ème place (sauf dans quelques cas particuliers qui ne vous ont pas échappé).
- Un **cycle de longueur p** est une sous-liste de p entiers (de p individus) qui s'offrent des cadeaux entre eux.
- Si un tirage au sort produit plusieurs cycles, ceux-ci seront séparés par un ' X ' dans la **représentation**.

Partie I - Généralités sur les représentations

QUESTION 1 — On procède à un tirage au sort dans un groupe de n individus. Quelles sont les longueurs minimale et maximale d'un cycle dans la représentation de ce tirage au sort ?

La longueur minimale d'un cycle est 2 ("A offre un cadeau à B qui offre un cadeau à A"), puisqu'un individu ne peut s'offrir de cadeau à lui-même. La longueur maximale d'un cycle est n (comme dans la situation de l'exemple 1).

QUESTION 2 — Il apparaît assez clairement qu'il existe plusieurs représentations pour un même tirage au sort. Pour illustrer cette affirmation, une autre représentation correspondant à la situation de l'exemple 2 est :

- | | |
|---|--|
| A. <input type="checkbox"/> [0, 1, 3, 2, ' X ', 5, 4, ' X ', 8, 6, 7] | C. <input checked="" type="checkbox"/> [8, 7, 6, ' X ', 5, 4, ' X ', 3, 1, 0, 2] |
| B. <input type="checkbox"/> [4, 5, ' X ', 0, 2, 1, 3, ' X ', 8, 7, 6] | D. <input type="checkbox"/> [6, 7, 8, ' X ', 4, 5, ' X ', 0, 1, 2, 3] |

A ne convient pas (0 ne fait pas de cadeau à 1) ; B ne convient pas (2 ne fait pas de cadeau à 1) ; D ne convient pas (6 ne fait pas de cadeau à 7).

Partie II - Cas particulier d'une représentation à un seul cycle

Dans cette partie, on suppose que le tirage au sort a donné lieu à un unique cycle (comme dans l'exemple 1) au sein d'un groupe de n individus. On note RES la **représentation** de ce tirage au sort : RES est donc une liste de longueur n . Dans cette partie, on considèrera que la liste RES et l'entier n sont déjà définis.

QUESTION 3 — Parmi les instructions proposées ci-dessous, lequel (ou lesquels) permet(tent) de savoir qui est le premier individu K_0 de la représentation, ainsi que l'individu à qui K_0 doit faire un cadeau ?

A. $\begin{array}{|l} \square_1 \\ \square_2 \end{array} \left| \begin{array}{l} \text{print}(' \text{Le premier individu est}', \text{RES}[1]) \\ \text{print}(' \text{Il offre un cadeau à}', \text{RES}[2]) \end{array} \right.$

B. $\begin{array}{|l} \square_1 \\ \square_2 \end{array} \left| \begin{array}{l} \text{print}(' \text{Le premier individu est}', \text{RES}[0]) \\ \text{print}(' \text{Il offre un cadeau à}', \text{RES}[1]) \end{array} \right.$

C. $\begin{array}{|l} \square_1 \\ \square_2 \end{array} \left| \begin{array}{l} \text{print}(' \text{Le premier individu est}', \text{RES}[0]) \\ \text{print}(' \text{Il offre un cadeau à}', \text{RES}[0] + 1) \end{array} \right.$

D. $\begin{array}{|l} \square_1 \\ \square_2 \end{array} \left| \begin{array}{l} \text{print}(' \text{Le premier individu est}', \text{RES}[0]) \\ \text{print}(' \text{Il offre un cadeau à}', 'X') \end{array} \right.$

A convient “presque tout le temps”... mais ne convient pas lorsque l'individu K est en dernière position (de rang $\text{len}(\text{RES})-1$) de la représentation, puisqu'il doit alors faire un cadeau à $\text{RES}[0]$, et non à $\text{RES}[\text{len}(\text{RES})]$, qui n'est pas défini.

C ne convient pas, car il y a confusion entre l'entier K et son rang dans la représentation (K et $\text{RES}[K]$ n'ont aucune raison d'être égaux).

B convient : lorsque K n'est pas en dernière position dans la liste, c'est-à-dire lorsque son rang est compris entre 0 et $n-2$, les instructions qui suivent le “for” affectent à la variable “who” l'entier situé à la droite de K dans la représentation, et tout est au mieux. Lorsque K est en dernière position dans la liste, l'instruction qui suit le “if” n'est pas effectuée (puisque alors $K = \text{RES}[n-1]$, et que $i \leq n-2$ dans la boucle). A l'issue de la boucle, la variable who est donc toujours égale à $n+2$; étant strictement plus grande que n , l'avant-dernière ligne du programme affecte à who la valeur $\text{RES}[0]$.

QUESTION 4 — L'individu K a oublié à qui il doit offrir un cadeau... Quel est le programme (ou quels sont les programmes) lui permettant de retrouver de qui il s'agit ?

A. ☐

```
1 n = len(RES)
2 for i in range(n):
3     if RES[i] == K:
4         who = RES[i+1]
5     print(K, 'offre un cadeau à', who)
```

B. ☐

```
1 n = len(RES)
2 print(K, 'K offre un cadeau à', RES[K+1])
```

C. ☐

```
1 n = len(RES)
2 who = n+2
3 for i in range(n-1):
4     if RES[i] == K:
5         who = RES[i+1]
6     if who > n:
7         who = RES[0]
8     print(K, 'offre un cadeau à', who)
```

Partie III - Cas général

On revient au cas général d'un groupe de n individus. On **ne suppose plus** que le tirage au sort a donné lieu à un unique cycle. On note RES la **représentation** de ce tirage au sort : RES est donc une liste de longueur au moins égale à n . Dans cette partie, on considèrera encore une fois que la liste RES et l'entier n sont déjà définis.

QUESTION 5 — Parmi les programmes proposés ci-dessous, quel est celui (ou quels sont ceux) permettant de déterminer le nombre de cycles contenus dans RES ?

A. ☐

```
1 print('Il existe', len(RES) - n + 1, 'cycles')
```

B. ☐

```
1 NbCycles = 0
2 for i in range(len(RES)):
3     if RES[i] == 'X':
4         NbCycles = NbCycles + 1
5     print('Il existe', NbCycles, 'cycles')
```

C. ☐

```
1 NbCycles = 1
2 n = len(RES)
3 for i in range(n):
4     if RES[i] == 'X':
5         NbCycles = NbCycles + 1
6     print('Il existe', NbCycles, 'cycles')
```

QUESTION 6 — (Qui m'a offert ce magnifique cadeau?). Complétez-le programme ci-dessous qui demande à l'utilisateur de saisir un entier K , et qui retourne l'entier L qui a offert un cadeau à K .

```
1 print('Bonjour! Choisissez un entier:')
2 K = int(input())
3 L = -1
4
5
6
7
8
9
10
11
12
13
14 print(L, 'a offert un cadeau à', K)
```