Introduction à la Programmation en Python
Examen – **Durée : 2 heures**

Université de Paris – Samedi 8 Février 2020

- Aucun document ni aucune machine ne sont autorisés. Les téléphones doivent être rangés.
- Les exercices sont tous indépendants.
- Cet énoncé a 2 pages.
- Une réponse peut utiliser les réponses attendues à une question précédente même si elle est non traitée.
- Les fragments de code doivent être correctement indentés.

Exercice 1.

1. Quelle est la valeur affichée par ce programme :

```
def f(lis):
    i = len(lis) - 1
    while len(lis[i]) <= i:
        i = i - 1
    return i

lis = ["aa", "ab", "ab", "abc"]
print(f(lis))
```

2. Quelle est la valeur affichée par ce programme :

```
m = [[0, 1, 2, 3], [3, 2, 7, 0], [5, 3, 0, 1]]
x = 0
for i in range(0, len(m), 1):
    x = x + m[i][m[i][i]]
print(x)
```

Exercice 2. Dans cet exercice, on considère des textes où les mots sont séparés par des tirets "-" au lieu de les séparer par des espaces.

1. Écrire une fonction `prefix_tirets(s)` qui prend une chaîne de caractères `s` en paramètre, et qui renvoie le nombre de tirets initiaux de `s`. Par exemple `prefix_tirets("---bonjour-le-monde--")` renvoie 3 et `prefix_tirets("au-revoir")` renvoie 0.
2. Écrire une fonction `suffix_tirets(s)` qui prend une chaîne de caractères `s` en paramètre, et qui renvoie le nombre de tirets finaux de `s`. Par exemple `suffix_tirets("---bonjour-le-monde--")` renvoie 2 et `suffix_tirets("au-revoir")` renvoie 0.
3. Écrire une fonction `trim(s)` qui prend une chaîne de caractères `s` en paramètre et qui renvoie une nouvelle chaîne de caractères qui est comme `s` sans les tirets initiaux et finaux.
   Par exemple `trim("---bonjour-le-monde--")` renvoie `"bonjour-le-monde"`, `trim("au-revoir")` renvoie `"au-revoir"` et `trim("----")` renvoie la chaîne vide `""`.
4. Écrire une fonction `compact(s)` qui prend une chaîne de caractères `s` en paramètre et qui renvoie une nouvelle chaîne de caractères qui est comme `s` sauf que tous les tirets répétés ont été remplacés par un seul tiret. Par exemple `compact("--bonjour---le---monde--")` renvoie `"-bonjour-le-monde-"`.
5. Écrire une fonction `max_tirets(s)` qui prend une chaîne de caractères `s` en paramètre et qui renvoie le nombre maximal de tirets successifs de `s`. Par exemple `max_tirets("Tout--le----monde--")` renvoie 4.

Exercice 3. On s'intéresse ici à une numérotation des couples d'entiers. Dans cet exercice, tous les entiers manipulés seront supposés être positifs ou nuls. On représente les couples d'entiers par des listes Python de taille 2 contenant des entiers. L'idée est que les couples d'entiers sont énumérés dans l'ordre $[0,0]$ - $[1,0]$ - $[0,1]$ - $[2,0]$ - $[1,1]$ - $[0,2]$ - $[3,0]$ ...

1. Écrire une fonction `ensuite` qui prend en paramètre une liste de deux entiers `[x,y]` et qui renvoie la liste `[x-1,y+1]` sauf si `x` vaut déjà 0, auquel cas on renvoie `[y+1,0]`. Par exemple, `ensuite([1,1])` renvoie `[0,2]`, tandis que `ensuite([0,2])` renvoie `[3,0]`.
2. Écrire une fonction `niemeCouple` qui prend en paramètre un entier `n` et qui renvoie la liste obtenue après `n` répétition(s) de la fonction `ensuite` en partant de `[0,0]`. Par exemple `niemeCouple(0)` renvoie `[0,0]`, et `niemeCouple(4)` renvoie `[1,1]`, car `ensuite(ensuite(ensuite(ensuite([0,0]))))=[1,1]`.
3. Écrire une fonction `rangCouple(cpl)` qui prend en paramètre une liste `cpl` de deux entiers et renvoie le nombre de répétitions nécessaires de la fonction `ensuite` pour passer de `[0,0]` à la liste `cpl`. Par exemple `rangCouple([0,0])` renvoie 0 et `rangCouple([1,1])` renvoie 4 car `ensuite(ensuite(ensuite(ensuite([0,0])))) = [1,1]`.
4. Écrire une fonction `verifCouples(n)` qui prend en paramètre un entier `n` et renvoie `True` si pour tous les entiers `i` entre 0 et `n` inclus on a bien `rangCouple(niemeCouple(i))` qui vaut `i`. Cette fonction renvoie `False` dans le cas contraire.

Exercice 4. Le but de cet exercice est de déterminer le gagnant d'une élection selon le mode de scrutin de Condorcet.

Dans un scrutin de Condorcet, les candidats à l'élection sont numérotés de 0 à $n-1$ où n est le nombre de candidats. Un *bulletin* est une liste d'entiers. Un bulletin est *valide* pour un nombre n de candidats si tous les éléments dans la liste sont des valeurs différentes entre 0 et $n-1$ inclus, et si en plus tous les candidats ont été classés.

1. Écrire une fonction `differents(lis)` qui prend une liste `lis` en paramètre et qui renvoie `True` si tous les éléments de `lis` sont différents, et `False` sinon.
2. Écrire une fonction `domaine(lis,n)` qui prend une liste `lis` et un entier `n` en paramètres et qui renvoie `True` si tous les éléments de `lis` sont entre 0 inclus et $n-1$ inclus, et `False` sinon.
3. Écrire une fonction `valide(lis,n)` qui prend une liste `lis` et un entier n en argument, et qui renvoie `True` si `lis` est un bulletin valide pour n candidats, et `False` sinon. On remarque qu'un bulletin est valide si et seulement si la liste a la longueur n, tous les éléments sur la liste sont différents, et sont entre 0 et $n-1$ inclus.

Un bulletin exprime un ordre de *préférence* entre les candidats : un bulletin l préfère un candidat i sur un candidat j si i parait devant j dans l . Par exemple, le bulletin `[2,0,1]` pour 3 candidats préfère 2 sur 0 et aussi sur 1, et préfère 0 sur 1.

4 Écrire une fonction `prefere(lis,i,j)` qui prend un bulletin valide `lis` et deux candidats `i` et `j` en paramètres, et qui renvoie `True` si le bulletin préfère `i` sur `j`, et `False` sinon.

5 Écrire une fonction `tally(llis,n)` qui prend une *liste* de bulletins `llis` et un nombre `n` de candidats en paramètres. La fonction renvoie un tableau de dimension $n \times n$, sous forme d'une liste de listes, tel qu'on trouve dans la case à la ligne i et colonne j le nombre de bulletins valides qui préfèrent i sur j.

Un candidat i est le *gagnant* d'un scrutin de Condorcet si pour tout *autre* candidat j, le nombre de bulletins valides qui préfèrent i sur j est strictement plus grand que le nombre de bulletins valides qui préfèrent j sur i. Il est possible qu'un scrutin de Condorcet n'a pas de gagnant, mais quand il existe un gagnant il est unique.

6 Écrire une fonction `gagnant(llis,n)` qui prend une liste de bulletins `llis` et un nombre de candidats `n` en paramètre, et qui renvoie le gagnant du scrutin s'il existe, et `-1` sinon.