

EXAMEN DE FIN DU SEMESTRE 1

Matière : INFORMATIQUE

Classes : 1^{ère} Année MP – T – PC Durée : 2 h

N.B. :

- L'indentation dans les instructions Python sera prise en considération dans le barème.
- Vous trouvez, en **Annexe**, quelques méthodes prédéfinies sur les conteneurs.

Exercice 1 (5 points)

Ecrire le script Python permettant de :

1. Saisir la chaîne de caractères **ch** formée uniquement par des lettres alphabétiques en minuscule.
2. Créer la variable **fin** contenant la dernière lettre de la chaîne de caractères **ch**.
3. Créer et afficher le dictionnaire **D** dont les clés sont chacune des lettres qui suivent la lettre **fin** dans la chaîne **ch** et leurs valeurs correspondantes sont leurs nombres d'apparition juste après la lettre **fin** dans la chaîne **ch**. Par exemple, dans la chaîne **ch= 'abbaabbabbabb'**, on a le caractère **fin= 'b'** qui est placé avant le caractère 'b', 4 fois ; et avant le caractère 'a', 3 fois, alors le dictionnaire résultat est **D = {'b': 4, 'a': 3}**.
4. Créer et afficher la liste **L** contenant la ou les clés du dictionnaire **D** qui correspondent à la valeur maximale dans **D**.

Exemples : Pour **D = {'b' : 3, 'c' : 2, 'a' : 3}**, on a **L=['b', 'a']**.

Pour **D = {'b': 3, 'c' : 2, 'a': 3, 'd' : 5}**, on a **L=['d']**.

Exercice 2 (5 points)

L'objectif de cet exercice est de gérer l'ensemble des rendez-vous formant l'agenda d'une personne. On choisit alors de représenter un agenda par la structure **ensemble** en Python dont chaque élément est un **tuple** composé de trois valeurs : date, heure et nom ; tels que :

- La **date** est représentée par une chaîne de caractères au format suivant : **'aaaa-mm-jj'** ;
- L'**heure** est représentée par un entier (on suppose que tous les rendez-vous ont lieu à une heure pile, par exemple 10h sera représenté par l'entier 10) ;
- Le **nom** est représenté par une chaîne de caractères désignant le nom de la personne avec qui on a rendez-vous.

Exemple : **Agenda = { ('2024-11-17', 10, 'Youssef'), ('2024-11-17', 11, 'Ibrahim'), ('2024-11-18', 10, 'Paul'), ('2024-11-19', 14, 'Paul'), ('2024-11-19', 10, 'Sacha') }**

Travail demandé :

Etant donné l'ensemble des rendez-vous **Agenda**, donner les instructions Python permettant de :

1. Saisir, d'abord, avec les contrôles de saisie nécessaires :

- L'entier **a** formé de quatre chiffres et qui représente l'année de la date à créer.
- L'entier **m** qui représente le mois de la date à créer.
- L'entier **j** qui représente le jour de la date à créer. Cet entier doit être compris entre 1 et le nombre de jours du mois précédemment saisi. **Indication** : Le module Python **calendar** contient la fonction **monthlen(a,m)** qui renvoie le nombre de jours correspondant au mois **m** de l'année **a** étant passés en paramètres.

Exemples : `calendar.monthlen(2024,2)` renvoie 29.

`calendar.monthlen(2024,12)` renvoie 31.

2. Construire, ensuite, la chaîne de caractères **d** qui représente la date d'un rendez-vous au format '**aaaa-mm-jj**', à partir des trois variables saisies : **a**, **m** et **j**. **Indication** : Le module Python **datetime** contient la fonction **date(a,m,j)** qui renvoie une date au format '**aaaa-mm-jj**' à partir des trois entiers passés en paramètres **a**, **m** et **j** correspondant respectivement à l'année, le mois et le jour d'une date donnée.

Exemple : `str(datetime.date(2024,12,1))` renvoie '**2024-12-01**'

3. Saisir l'entier **h** qui représente l'heure d'un rendez-vous. Cet entier varie entre 9 et 19.

4. A partir de l'ensemble donné **Agenda**, créer et afficher l'ensemble des rendez-vous qui ont lieu à la date **d** et après l'heure **h** saisis précédemment.

Exemple :

```
d='2024-11-17'
```

```
h=9
```

```
Résultat={ ('2024-11-17', 10, 'Youssef'),  
            ('2024-11-17', 11, 'Ibrahim') }
```

Exercice 3 (5 points)

On s'intéresse, dans cet exercice, à une étude statistique des résultats partiels du concours pour un établissement donné disposant du classement de ses candidats à l'échelle nationale.

L'une des notions utilisées en statistique est la **médiane**. Pour une série de valeurs triées, la médiane est la valeur de **x** qui coupe la série en deux sous-séries de tailles égales, telles que la première contient les valeurs qui sont inférieures à **x** et la deuxième contient celles qui sont supérieures à **x**.

Ainsi, pour une série de taille paire, la médiane est la moyenne des deux valeurs au milieu et pour une série de taille impaire, la médiane est la valeur du milieu. La médiane est une valeur qui peut ne pas faire partie de la série étudiée.

Exemples :

- Pour **S1** = 98, 103, 105, 106, 113, 114, 117, 138, de taille 8, la valeur médiane est $\frac{(106 + 113)}{2} = 109.5$
- Pour **S2** = 83, 98, 103, 105, 106, 113, 114 de taille 7, la médiane est 105.

Par ailleurs, si on désigne par **Mo** et **Me** respectivement les valeurs de la moyenne et de la médiane, alors on dit qu'on a une distribution :

- *Symétrique*, si **Mo** = **Me**,
- *Asymétrique à gauche*, si **Mo** > **Me**,
- *Asymétrique à droite*, si **Mo** < **Me**.

Exemples :

- Pour **S1**, on a la moyenne **Mo** = 111.75 et la médiane **Me** = 109.5, alors la distribution est dite *asymétrique à gauche*.
- Pour **S2**, on a la moyenne étant **Mo** = 103.14 et la médiane **Me** = 105, alors la distribution est dite *asymétrique à droite*.

On suppose que le classement, à l'échelle nationale des candidats, soit enregistré dans un dictionnaire nommé **d_Rangs** où chaque :

- **clé** est le nom de la section (chaîne de caractères ayant pour valeur soit 'MP' ou 'PC' ou 'T' ou 'BG' uniquement) ;
- **valeur** est la liste des *tuples* contenant chacun l'identifiant (entier) et le rang (entier) d'un candidat.

Travail demandé :

Ecrire un programme Python permettant de :

1. Saisir le nom d'une section **s** et l'entier **rg** (faire les contrôles de saisie nécessaires), puis afficher l'identifiant du candidat de la section **s** ayant le rang **rg**.
2. Saisir le nom d'une section **s**, puis enregistrer, dans la liste **L**, les rangs des candidats de la section **s** étant triés par ordre croissant.
3. A partir de la liste **L**, calculer et afficher **Mo** qui représente la valeur moyenne de ses éléments.
4. A partir de la liste **L**, calculer et afficher **Me** qui représente la valeur de la médiane de ses éléments.
5. A partir de la liste **L**, afficher le message correspondant à la qualification de la distribution des rangs des différents candidats ('symétrique', 'asymétrique à gauche', 'asymétrique à droite').

Exercice 4 (5 points)

1. Donner la complexité, dans le **pire des cas**, de chacun des programmes Python suivants :

```
# Programme (P1) :
n=int(input("Donner n: "))
i,x=1,n
while x > 1 :
    i=i+1
    x=x//2

# Programme (P2) :
n=int(input("n="))
for i in range(n):
    for j in range(i):
        print('*', end=' ')
    print()
```

```
# Programme (P3) :
n=int(input("n="))
for i in range(1, n*n+1):
    j=i
    while j > 0:
        print('=', end=' ')
        j=j//2
    print()
```

2. Soit le programme Python suivant :

```
# Programme :
x=int(input("Donner x: "))
s=0
for i in range (1,x+1):
    p=1
    for j in range(1,i):
        p*=j
    s+=1/p
print(s)
```

- Donner sa complexité dans le pire des cas.
- Qu'est-ce qu'il permet de calculer ?
- Proposer une autre variante de ce programme qui améliore sa complexité au pire des cas.
- Donner la complexité de la solution proposée.

ANNEXE - Quelques fonctions Python

Sans aucune obligation, les fonctions suivantes pourraient vous être utiles.

Opérations sur les *itérables* (*str*, *tuple*, *list*, *dict*, etc.)

- **len(it)** : retourne le nombre d'éléments de l'*itérable* *it*.
- **sum(it)** : retourne la somme des éléments de l'*itérable* *it*.
- **x in it** : vérifie si *x* appartient à l'*itérable* *it*.
- **sorted(it)** : retourne une liste contenant les éléments de l'*itérable* *it* dans l'ordre croissant.
- **lst.sort()** : trie la liste *lst* dans l'ordre croissant.
- **lst.count(val)** : retourne le nombre d'occurrences de *val* dans la liste *lst*.
- **lst.append(val)** : ajoute *val* à la fin de la liste *lst*.
- **e.add(val)** : ajoute l'élément *val* à l'ensemble *e*.
- **d.values()** : retourne un *itérable* formé par les valeurs du dictionnaire *d*.
- **d.items()** : retourne un *itérable* de couples (*k,v*) où *k* est une *clé* du dictionnaire *d* et *v* est la *valeur* associée.