

REPUBLIQUE TUNISIENNE

Ministère de l'enseignement Supérieure, de la
recherche scientifique



الجمهورية التونسية

وزارة التعليم العالي

والبحث العلمي

Institut Préparatoire aux Etudes d'Ingénieurs de
Sfax

المعهد التحضيري للدراسات الهندسية بصفاقس

Examen d'informatique

1^{er} semestre AU : 2021 – 2022

Date : 03 Janvier 2022

;

Section : MP2, PC2, PT2;

Nombre de page : 4

L'utilisation des calculatrices n'est pas autorisée pour cette épreuve.

Le langage de programmation sera obligatoirement Python.

*

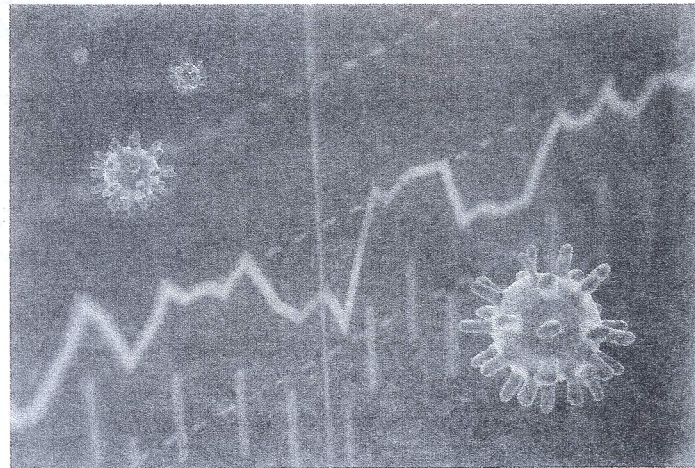
* *

Implémentation. Dans ce sujet, nous adopterons la syntaxe du langage Python. On rappelle qu'en Python, il importe de bien respecter les indentations car elles permettent de définir des blocs.

Covid 19 et Data Science

La pandémie de Covid-19 a provoqué une mobilisation générale des gens et institutions ayant vocation à sauver des vies.

Virologie, immunologie, pharmacologie, épidémiologie... les sciences de la vie ont été les premières à se mobiliser, avec pour toutes un besoin impérieux de quantifier, modéliser, simuler... Au XXI^e siècle, les sciences du vivant brassent des données. C'est pourquoi les *data sciences* sont montées au front. Mobilisée, la communauté des data sciences multiplie les initiatives dans la lutte contre le Covid-19. Elle participe à la compréhension du virus et de sa propagation en vue de fournir des outils à l'action publique et de concevoir un arsenal diagnostique et pharmacologique.



Dans notre sujet, nous allons mettre l'accent à ces types de données pour donner des outils d'analyse pour aider les experts à comprendre ces données afin de prendre les bonnes décisions.

Pour cela, on va se baser sur des données des patients infectés par le virus covid 19 enregistrées dans un fichier texte "patients_covid.csv".

Voici un extrait de ce fichier :

age	genre	douleur	pression	cholester	sucré	angine	coeur	date	city
53	feminin	D	140	203	B	oui	presence	2020-03-21	Sfax
67	masculin	A	110	211	A	oui	absence	2020-03-21	Monastir
53	feminin	C	140	199	A	oui	absence	2020-03-21	Tunis
67	masculin	D	120	229	A	oui	presence	2020-03-21	Sfax
67	masculin	B	130	245	A	non	absence	2020-03-21	Monastir

Partie I. Patient

Chaque ligne du fichier texte représente les valeurs des propriétés (features) d'un patient.

Question 1 :

Écrire une fonction **get_Feature(nomF, sep)** qui retourne la liste des noms des propriétés (features) des patients à partir du fichier **nomF**, sachant que les features existent dans la première ligne du fichier séparée par le paramètre **sep**.

```
>>> features = get_Feature("patients_covid2.csv", ';')
>>> print(features)
['age', 'genre', 'douleur', 'pression', 'cholester', 'sucré', 'angine', 'coeur', 'date', 'ville']
```

Question 2 :

Un patient sera représenté sous la forme d'un dictionnaire tels que :

- les clés sont les propriétés (**feature**) du patient, par exemple (age , genre , pression, ...)
- les valeurs du dictionnaire sont les valeurs des **features**.

Écrire une fonction qui retourne le dictionnaire représentant le patient dans la i^{ème} ligne du fichier nommé **nomF**, sachant que la première ligne d'indice 0 est celle des features.

```
>>> patient2 = get_patient(2, "patients_covid2.csv", ";")
>>> print(patient2)
{'age': '67', 'genre': 'masculin', 'douleur': 'A', 'pression': '110', 'cholester': '211', 'sucré': 'A', 'angine': 'oui', 'coeur': 'absence', 'date': '2020-03-21', 'ville': 'Monastir'}
```

Partie II. Serie

Chaque colonne dans le fichier texte sera représentée par la classe **Serie** dotée des attributs suivants :

- **label** : le nom de la propriété (feature)
- **data** : la liste des valeurs de cette feature.

```
class Serie :  
    def __init__(self, lab, L) :  
        self.label = lab  
        self.data = L
```

Question 3 :

Ajouter la méthode **average(self)** qui calcule et renvoie la moyenne de la **Serie** self ?

Question 4 :

Ajouter la méthode **variance(self)** qui calcule et renvoie la variance de la **Serie** self ?

Sachant que : $\text{variance}(s) = \frac{1}{n} \sum_{k=0}^{n-1} (v_k - \bar{v})^2$;

où v_k est la $k^{\text{ème}}$ valeur de la **Serie** s et $\bar{v}$ sa moyenne.

Question 5 :

Ajouter la méthode **value_counts(self)** qui renvoie un dictionnaire dont :

- les clés sont les différentes valeurs de l'attribut **data** de l'objet self.
- les valeurs sont les occurrences de chaque valeur de l'attribut **data** de l'objet self.

```
>>> l_age = [76, 40, 76, 55, 14, 76, 55, 68]  
>>> s_age = Serie('age', l_age)  
>>> print("value counts = ", s_age.value_counts())  
value counts = {76: 3, 40: 1, 55: 2, 14: 1, 68: 1}
```

Question 6 :

Ajouter la méthode **plot_counts(self)** qui trace la courbe tels que les abscisses sont les différents valeurs de l'attribut **data** de l'objet self et les ordonnées sont les occurrences de ces valeurs.

Question 7 :

Ajouter la méthode **__call__(self, v)** qui extrait et renvoie un nouveau objet de type **Serie** dont les attributs seront :

- label est égal à « index »
- data est égal à la liste des indices de la valeur **v** dans l'attribut **data** de l'objet self.

```
>>> s_age_index = s_age(76) # ou s_age.__call__(76)  
>>> s_age_index.data  
[0, 2, 5]
```

Partie III. DataSet

L'ensemble des données dans le fichier texte sera représenté sous la forme d'une classe **DataSet** dotée des attributs suivants :

- **file** : le nom du fichier texte
- **separator** : le caractère séparateur des colonnes du fichier
- **l_data** : une liste des objets de la classe **Serie**

```
class DataSet :  
    def __init__(self, nomF, sep) :  
        self.file = nomF  
        self.separator = sep  
        self.l_data = []
```

Question 8 :

Ajouter la méthode **get_Feature(self)** qui retourne la liste des features de l'attribut **file** séparé par l'attribut **separator**.

Note : utiliser la fonction **get_Feature** de la question 1.

Question 9 :

Ajouter la méthode **initialise_data(self)** qui remplit l'attribut **l_data** avec des objets de type **Serie** dont :

- l'attribut **label** est égal à **feature**
- et l'attribut **data** est égal à une liste vide.

```
>>> ds = DataSet("patients_covid2.csv", ";")  
>>> ds.initialise_data()  
>>> print('serie 0 du dataset, label = ', ds.l_data[0].label)  
serie 0 du dataset, label = age  
>>> print('serie 0 du dataset, data = ', ds.l_data[0].data)  
serie 0 du dataset, data = []
```

Question 10 :

Compléter la méthode **load_data(self)** qui remplit l'attribut **data** de chaque objet **Serie** dans la liste **l_data**.

```
def load_data(self) :  
    f = open(self.file, 'r')  
    ligne = f.readline() # lecture de la première ligne des features  
    features = self.get_Feature()  
    for ligne in f :  
        ligne = ligne.strip() # supprimer le \n  
  
        # ... à compléter ...  
  
    f.close()
```

Question 11 :

Ajouter la méthode **shape(self)** qui retourne un tuple de la forme (nombre de lignes , nombre de colonnes).