REPUBLIQUE TUNISIENNE

MINISTERE DE L'EDUCATION

---------- ❖ ----------

Tutoriel

Table des matières

A / Analyse des données avec pandas

A.1 / Installation de la bibliothèque pandas

```
python -m pip install pandas
```

A.2 / Importation de la bibliothèque pandas

```
import pandas as alias
ou bien
import pandas
```

A.3 / Création d'un dataframe à partir des sources de données externes

A.3.a) Importation des fichiers txt et csv

L'importation des fichiers **txt** ou **csv** se fait avec la fonction **read_csv()** du module pandas.

❖ **Syntaxe :**

```
df = pandas.read_csv("Chemin\Nom_Fichier.extension", sep = "séparateur", index_col = N°_Colonne, header = 0
, encoding="Code_codage")
```

❖ **Explication détaillée :**

Commande	Description
read_csv	Lire un fichier **txt** ou **csv**
Chemin\Nom_Fichier.extension	**Chemin :** Le chemin physique du fichier sur le disque dur. **Nom_Fichier :** Le nom du fichier. **Extension : txt** pour les fichiers textes ou **csv** pour les fichiers de type tableur dont les valeurs sont séparées par des virgules.
sep = "séparateur"	Elle permet d'indiquer le séparateur qui sépare les données. Les séparateurs les plus utilisés sont : ◈ **"\t"** : Les données seront séparées par une tabulation (séparateur par défaut en cas d'omission). ◈ **","** : Les données seront séparées par une virgule. ◈ **";"** : Les données seront séparées par un point-virgule.
index_col = N°_Colonne	Elle permet d'indiquer le Numéro de la colonne par le quel on débutera l'importation des données (En cas d'omission, on commencera par la colonne **0**).
header = 0	Le paramètre **header = 0** indique que les libellés des colonnes (variable) se trouvent sur la ligne **0**. Si les données ne contiennent pas des libellés, on mettra **header = None** (En cas d'omission, c'est **header = 0**).
encoding="Code_codage"	Un codage de caractères est un code qui associe un jeu de caractères du fichier. Code_codage= **"UTF-8" / "ANSI" / "Latin-1"** (En cas d'omission, Code_codage= **"UTF-8"**)

❖ **Exemple :**

```
import pandas as ps
df=ps.read_csv("pays.csv",sep=";")
```

A.3.b) Importation des fichiers Excel

L'importation des fichiers Excel se fait avec la fonction **read_excel ()** du module pandas. Mais avant d'utiliser cette fonction, il faut installer la bibliothèque **xlrd** (qui permet la lecture d'un fichier Excel en utilisant la commande : python -m pip install xlrd).

❖ **Syntaxe :**

```
df = pandas. read_excel ("Chemin\Nom_Fichier.xls", "Nom_Feuille" , index_col = N°_Colonne , header = 0)
```

❖ **Explication détaillée :**

Commande	Description
read_excel	Lecture du fichier Excel
Chemin\Nom_Fichier.xls ou bien Chemin\Nom_Fichier.xlsx	**Chemin :** Le chemin physique du fichier sur le lecteur **Nom_Fichier :** Le nom du fichier **L'extension :** xls ou xlsx
Nom_Feuille	Nom de la feuille de calcul à manipuler.
index_col = N°_Colonne	**Cette Commande est facultative** Permet d'indiquer le Numéro de la colonne par le quel on débutera l'importation des données (En cas d'omission, on commencera par la colonne **0**)
header = 0	Le paramètre **header = 0** indique que les libellés des colonnes (variable) se trouvent sur la ligne **0**. Si les données ne contiennent pas des libellés, on mettra **header = None** (En cas d'omission, c'est **header = 0**).

❖ **Exemple :**

```
import pandas
df = pandas.read_excel("Resultat02.xlsx","Feuil01")
```

A.4 / Manipulation d'un dataframe

A.4.a) Affichage des informations concernant un dataframe

Pour afficher les données d'un dataframe, on utilise l'instruction **info()** :

❖ **Syntaxe :**

```
Nom_dataframe.info()
```

❖ **Exemple :**

Le code

```
import pandas as ps
df=ps.read_csv ("pays.csv" , sep= ";")
print (df.info())
```

L'affichage

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex :  6 entries   ,  0 to 5
Data columns (total 3 columns):
Pays          6 non-null object
Superficie    6 non-null int64
Habitant      6 non-null int64
dtypes : int64(2)   , object(1)
memory usage  : 184.0  +  bytes
None
```

A.4.b) Affichage des informations sur les colonnes dans un dataframe

Pour afficher les informations sur les colonnes, on peut utiliser les instructions suivantes :

Syntaxe	Description	Exemple	Résultat
dataframe.columns	Retourne les noms des colonnes	import pandas as ps df=ps.read_csv("pays.csv",sep=";") print (df. columns)	Index(['Pays', 'Superficie', 'Habitant'], dtype='object')
dataframe.columns.values	Retourne les noms des colonnes sous forme d'un tableau	import pandas as ps df=ps.read_csv("pays.csv",sep=";") print (df. columns.values)	['Pays' 'Superficie' 'Habitant']
dataframe.dtypes	Retourne les noms des colonnes suivies de leurs types	import pandas as ps df=ps.read_csv("pays.csv",sep=";") print (df.dtypes)	Pays object Superficie int64 Habitant int64 dtype: object

A.4.c) Ajout d'une colonne dans un dataframe

Pour ajouter une colonne remplie par des données dans un dataframe, on utilise l'instruction :

```
Dataframe["Nom_Colonne"]=valeur
```

❖ **Exemple :**

Ajouter la colonne Densité pour les pays sachant que Densité = Nombre d'habitant/Superficie.

Le code :

```
import pandas as ps
df=ps.read_csv("pays.csv",sep=";")
df["Densité"]=df.Habitant/df.Superficie
print (df)
```

L'affichage :

	Pays	Superficie	Habitant	Densité
0	Russ	17098242	146781095	8.584572
1	Canada	998467	37560207	37.617875
2	USA	9826675	331883986	33.773783
3	Chine	9596961	1394112547	145.266043
4	Brésil	8514877	210301591	24.698136
5	Australie	774122	25105503	32.430939

A.4.d) Renommage des libellés d'une ou de plusieurs colonnes

Pour renommer les libelles d'une colonne ou plusieurs colonnes, on utilise l'instruction :

❖ **Syntaxe :**

```
df.rename(columns={"Colonne1": "Nouveau_Colonne1","Colonne2": "Nouveau_Colonne2"},inplace=True/False)
```

❖ **Explication détaillée :**

Commande	Description
inplace	**inplace = True** : signifie que les modifications sont appliquées sur le dataframe. **inplace = False** : signifie que les modifications ne sont pas appliquées sur le dataframe, il faut créer un nouveau dataframe pour récupérer les modifications.

❖ **Exemple :**

Renommer la colonne « Pays » par « Nom_Pays », il faut :

Le code :

```
import pandas as ps
df=ps.read_csv ("Pays.csv",sep=";")
df.rename(columns={"Pays": 'Nom_Pays'},inplace=True)
print(df)
```

dataframe avant exécution du code

	Pays	Superficie	Habitant
0	Russ	17098242	146781095
1	Canada	998467	37560207
2	USA	9826675	331883986
3	Chine	9596961	1394112547
4	Brésil	8514877	210301591
5	Australie	774122	25105503

Affichage après exécution du code

	Nom_Pays	Superficie	Habitant
0	Russ	17098242	146781095
1	Canada	998467	37560207
2	USA	9826675	331883986
3	Chine	9596961	1394112547
4	Brésil	8514877	210301591
5	Australie	774122	25105503

A.4.e) Suppression d'une colonne

Pour supprimer une colonne, on utilise la commande :

```
del dataframe["Nom_Colonne"]
```

❖ **Exemple :**

Pour supprimer la colonne « Habitant », il faut :

Le code :

```
import pandas as ps
df=ps.read_csv ("pays.csv",sep=";")
del df['Habitant']
print (df)
```

dataframe avant exécution du code

	Pays	Superficie	Habitant
0	Russ	17098242	146781095
1	Canada	998467	37560207
2	USA	9826675	331883986
3	Chine	9596961	1394112547
4	Brésil	8514877	210301591
5	Australie	774122	25105503

Affichage après exécution du code

	Pays	Superficie
0	Russ	17098242
1	Canada	998467
2	USA	9826675
3	Chine	9596961
4	Brésil	8514877
5	Australie	774122

A.4.f) Accès aux données d'un dataframe

Pour accéder aux données d'un dataframe, on peut utiliser les instructions suivantes :

Syntaxe :	dataframe	
Description	**Exemple**	**Résultat**
Retourne les données d'un dataframe	import pandas as ps df=ps.read_csv("pays.csv",sep= ";") print (df)	Pays Superficie Habitant 0 Russ 17098242 146781095 1 Canada 998467 37560207 2 USA 9826675 331883986 3 Chine 9596961 1394112547 4 Brésil 8514877 210301591 5 Australie 774122 25105503
Syntaxe :	dataframe.head()	
Description	**Exemple**	**Résultat**
Retourne les 5 premières lignes	import pandas as ps df=ps.read_csv("pays.csv",sep=";") print (df.head())	Pays Superficie Habitant 0 Russ 17098242 146781095 1 Canada 998467 37560207 2 USA 9826675 331883986 3 Chine 9596961 1394112547 4 Brésil 8514877 210301591
Syntaxe :	dataframe.tail()	
Description	**Exemple**	**Résultat**
Retourne les 5 dernières lignes	import pandas as ps df=ps.read_csv("pays.csv",sep=";") print (df.tail())	Pays Superficie Habitant 1 Canada 998467 37560207 2 USA 9826675 331883986 3 Chine 9596961 1394112547 4 Brésil 8514877 210301591 5 Australie 774122 25105503
Syntaxe :	dataframe.Nom_Colonne ou bien dataframe["Nom_Colonne"]	
Description	**Exemple**	**Résultat**
Retourne les données d'une colonne	import pandas as ps df=ps.read_csv("pays.csv",sep=";") print(df.Pays)	0 Russ 1 Canada 2 USA 3 Chine 4 Brésil 5 Australie Name: Pays, dtype: object

Syntaxe :	dataframe[["Nom_Colonne1","Nom_Colonne2"]]	
Description	**Exemple**	**Résultat**
Retourne les données de la colonne1 et de colonne2	import pandas as ps df=ps.read_csv("pays.csv",sep=";") print(df[["Pays","Habitant"]])	Pays Habitant 0 Russ 146781095 1 Canada 37560207 2 USA 331883986 3 Chine 1394112547 4 Brésil 210301591 5 Australie 25105503
Syntaxe :	dataframe.iloc[Position_Initiale : Position_Finale]	
Description	**Exemple**	**Résultat**
Retourne les données des lignes de la position initiale et la position finale - 1	import pandas as ps df=ps.read_csv("pays.csv",sep=";") print (df.iloc[0:3])	Pays Superficie Habitant 0 Russ 17098242 146781095 1 Canada 998467 37560207 2 USA 9826675 331883986
Syntaxe :	dataframe.nom colonne [Position_Initiale : Position_Finale]	
Description	**Exemple**	**Résultat**
Retourne les données d'une colonne entre une position initiale et une position finale - 1	import pandas as ps df=ps.read_csv("pays.csv",sep=";") print (df.Pays[2:5])	2 USA 3 Chine 4 Brésil Name: Pays, dtype: object

A.4.g) Affichage des données d'un dataframe selon une ou plusieurs conditions

- Pour afficher des données d'un dataframe selon une condition, on utilise l'instruction :

dataframe[dataframe ["nom colonne"] opérateur_comparaison valeur]

LES OPERATEURS DE COMPARAISON				
Opérateur	**Nomination**	**Rôle**	**Exemple**	**Résultat**
==	opérateur d'égalité	Vérifie qu'une variable est égalité à une valeur	print (df[df["Superficie"] == 9826675])	Pays Superficie Habitant 2 USA 9826675 331883986
<	opérateur d'infériorité stricte	Vérifie qu'une variable est strictement inférieure à une valeur	print (df[df["Superficie"] <9826675])	Pays Superficie Habitant 1 Canada 998467 37560207 3 Chine 9596961 1394112547 4 Brésil 8514877 210301591 5 Australie 774122 25105503
<=	opérateur d'infériorité	Vérifie qu'une variable est inférieure ou égale à une valeur	print (df[df["Superficie"] <=9826675])	Pays Superficie Habitant 1 Canada 998467 37560207 2 USA 9826675 331883986 3 Chine 9596961 1394112547 4 Brésil 8514877 210301591 5 Australie 774122 25105503
>	opérateur de supériorité stricte	Vérifie qu'une variable est strictement supérieure à une valeur	print (df[df["Superficie"] > 9826675])	Pays Superficie Habitant 0 Russ 17098242 146781095

>=	opérateur de supériorité	Vérifie qu'une variable est supérieure ou égale à une valeur	print (df[df["Superficie"] >= 9826675])	Pays Superficie Habitant 0 Russ 17098242 146781095 2 USA 9826675 331883986
!=	opérateur de différence	Vérifie qu'une variable est différente d'une valeur	print (df[df["Superficie"]!= 9826675])	Pays Superficie Habitant 0 Russ 17098242 146781095 1 Canada 998467 37560207 3 Chine 9596961 1394112547 4 Brésil 8514877 210301591 5 Australie 774122 25105503
isin	dans	Vérifier que la valeur d'une variable dans une liste de valeurs	print (df[df["Superficie"].isin ([9826675,17098242])])	Pays Superficie Habitant 0 Russ 17098242 146781095 2 USA 9826675 331883986

- Pour afficher des données d'un dataframe selon plusieurs conditions, on utilise l'instruction :

```
dataframe[ (dataframe ["Nom_Colonne"] Opérateur_Comparaison  Valeur) Opérateur_Logique
(dataframe ["Nom_Colonne"] Opérateur_Comparaison  Valeur)]
```

LES OPERATEURS LOGIQUES				
Opérateur	**Nomination**	**Rôle**	**Exemple**	**Résultat**
|	**OU** logique	Vérifie qu'une des conditions est réalisée	print (df[(df["Superficie"] >= 9826675) | (df["Superficie"] >= 1000000)])	Pays Superficie Habitant 0 Russ 17098242 146781095 2 USA 9826675 331883986 3 Chine 9596961 1394112547 4 Brésil 8514877 210301591
&	**ET** logique	Vérifie que toutes les conditions sont réalisées	print (df[(df["Superficie"]>= 9826675) & (df["Superficie"]>= 1000000)])	Pays Superficie Habitant 0 Russ 17098242 146781095 2 USA 9826675 331883986

A.4.h) Tri d'un dataframe

Pour trier les données d'un dataframe, on utilise l'instruction suivante :

```
dataframe = dataframe.sort_values(by = [Liste des colonnes], ascending = [critère pour chaque
colonne (True/False])
```

Exemple :

Le code

```
import pandas as ps
df=ps.read_csv("pays.csv",sep=";")
df=df.sort_values(by = ["Superficie","Habitant"],ascending = [False,True])
print (df)
```

dataframe avant exécution du code

	Pays	Superficie	Habitant
0	Russ	17098242	146781095
1	Canada	998467	37560207
2	USA	9826675	331883986
3	Chine	9596961	1394112547
4	Brésil	8514877	210301591
5	Australie	774122	25105503

Affichage après exécution du code

	Pays	Superficie	Habitant
0	Russ	17098242	146781095
1	USA	9826675	331883986
2	Chine	9596961	1394112547
3	Brésil	8514877	210301591
4	Canada	998467	37560207
5	Australie	774122	25105503

A.4.i) Modification d'une cellule d'un dataframe

Pour modifier le contenu d'une cellule dans un dataframe, on utilise l'instruction suivante :

```
Dataframe.loc[N° ligne, "Nom_colonne"]=Valeur
```

Exemple :

Le code

```
import pandas as ps
df=ps.read_csv("pays.csv",sep=";")
print (df)
df.loc[0, "Pays"] = "Russie"
print (df)
```

dataframe avant exécution du code

	Pays	Superficie	Habitant
0	Russ	17098242	146781095
1	Canada	998467	37560207
2	USA	9826675	331883986
3	Chine	9596961	1394112547
4	Brésil	8514877	210301591
5	Australie	774122	25105503

Affichage après exécution du code

	Pays	Superficie	Habitant
0	**Russie**	17098242	146781095
1	Canada	998467	37560207
2	USA	9826675	331883986
3	Chine	9596961	1394112547
4	Brésil	8514877	210301591
5	Australie	774122	25105503

A.4.j) Modification des données d'une ligne d'un dataframe

Pour modifier le contenu d'une ligne d'un dataframe, on utilise l'instruction suivante :

```
Dataframe.loc[N° ligne]=[La liste des valeurs]
```

Exemple :

Le code

```
import pandas as ps
df=ps.read_csv("pays.csv",sep=";")
print (df)
df.loc[0] =["Russie" , 17098555,146781999]
print (df)
```

dataframe avant exécution du code

	Pays	Superficie	Habitant
0	Russ	17098242	146781095
1	Canada	998467	37560207
2	USA	9826675	331883986
3	Chine	9596961	1394112547
4	Brésil	8514877	210301591
5	Australie	774122	25105503

Affichage après exécution du code

	Pays	Superficie	Habitant
0	**Russie**	**17098555**	**146781999**
1	Canada	998467	37560207
2	USA	9826675	331883986
3	Chine	9596961	1394112547
4	Brésil	8514877	210301591
5	Australie	774122	25105503

A.4.k) Création et remplissage des colonnes suite à une condition simple

Pour ajouter une colonne remplie par des données dans un dataframe, on utilise la commande :

```
dataframe["Nom_Colonne_Resultat"]= numpy.where(dataframe ["Nom_Colonne]
opérateur_comparaison  valeur, Valeur1_Si_vrai, Valeur2_si_Faux)
```

Noter Bien : Pour utiliser « **numpy.where** », il faut importer la bibliothèque « **numpy** » par l'instruction « **import numpy** »

Exemple :
Ajouter la colonne « Observation » sachant que :

- Observation = Très dense dans le cas où la densité est > 60,
- Observation = Moyenne dans le cas contraire.

Le code

```
import pandas as ps
import numpy
df=ps.read_csv("pays.csv",sep=";")
df["Densité"]=df.Habitant/df.Superficie
df["Observation"]=numpy.where(df["Densité"]>60,"Très dense", "Moyenne")
print(df)
```

L'affichage :

	Pays	Superficie	Habitant	Densité	**Observation**
0	Russ	17098242	146781095	8.584572	**Moyenne**
1	Canada	998467	37560207	37.617875	**Moyenne**
2	USA	9826675	331883986	33.773783	**Moyenne**
3	Chine	9596961	1394112547	145.266043	**Très dense**
4	Brésil	8514877	210301591	24.698136	**Moyenne**
5	Australie	774122	25105503	32.430939	**Moyenne**

A.4.L) Création et remplissage des colonnes suite à plusieurs conditions

Pour ajouter une colonne remplie par des données dans un dataframe on peut utiliser l'un des deux méthodes suivantes :

- **1ère méthode : Utilisation de la structure de contrôle itérative et de la structure conditionnelle**

Exemple :

Ajouter et remplir la colonne « observation », sachant que :

- Observation = Très dense dans le cas où densité ≥ 100
- Observation = Normale dans le cas où 50 ≤ densité < 100
- Observation = Moyenne dans le cas où 20 ≤ densité < 50
- Observation = Faible dans le cas où densité < 20

Le code

```
import pandas as ps
df=ps.read_csv("pays.csv",sep=";")
df["Densité"]=df.Habitant/df.Superficie
obser=[]
for x in df["Densité"] : # parcourir les valeurs de la colonne densité valeur par valeur
    if x >=100 : # vérifier si la valeur de la densité >=100
        obser.append("Très dense")
    else :
        if x >=50 : # vérifier si la valeur de la densité >=50
            obser.append("Normale")
        else :
            if x >=20 : # vérifier si la valeur de la densité >=20
                obser.append("Moyenne")
            else :
                 obser.append("Faible")
df["Observation"]=obser
print(df)
```

L'affichage :

	Pays	Superficie	Habitant	Densité	**Observation**
0	Russ	17098242	146781095	8.584572	**Faible**
1	Canada	998467	37560207	37.617875	**Moyenne**
2	USA	9826675	331883986	33.773783	**Moyenne**
3	Chine	9596961	1394112547	145.266043	**Très dense**
4	Brésil	8514877	210301591	24.698136	**Moyenne**
5	Australie	774122	25105503	32.430939	**Moyenne**

- **2ème méthode : Utilisation d'une fonction utilisateur (def) et la fonction prédéfinie apply()**

La fonction **apply()** permet d'appliquer une fonction sur la table de donnée dataframe

```
dataframe ["Nom colonne].apply(fonction utilisateur)
```

Exemple :

Ajouter et remplir la colonne « Observation », sachant que :

- Observation = Très dense dans le cas où densité ≥ 100
- Observation = Normale dans le cas où 50 ≤ densité < 100
- Observation = Moyenne dans le cas où 20 ≤ densité < 50
- Observation = Faible dans le cas où densité < 20

Le code

```
import pandas as ps
def calcul(x):
    if x >=100 : # vérifier si la valeur de la densité >=100
        return "Très dense"
    else :
        if x >=50 : # vérifier si la valeur de la densité >=50
            return "Normale"
        else :
            if x >=20 : # vérifier si la valeur de la densité >=20
                return "Moyenne"
            else :
                 return "Faible"
df=ps.read_csv("pays.csv",sep=";")
df["Densité"]=df.Habitant/df.Superficie
df["Observation"]=df["Densité"].apply(calcul)
print(df)
```

L'affichage :

	Pays	Superficie	Habitant	Densité	**Observation**
0	Russ	17098242	146781095	8.584572	**Faible**
1	Canada	998467	37560207	37.617875	**Moyenne**
2	USA	9826675	331883986	33.773783	**Moyenne**
3	Chine	9596961	1394112547	145.266043	**Très dense**
4	Brésil	8514877	210301591	24.698136	**Moyenne**
5	Australie	774122	25105503	32.430939	**Moyenne**

A.4.m) Concaténation de plusieurs dataframes en un seul

Pour concaténer plusieurs dataframes en un seul, on utilise l'instruction suivante :

```
dataframe_resultat = pandas.concat([df1, df2, fdn], ignore_index = True)
```

Exemple :

Le code

```
import pandas as ps
df0=ps.read_csv("pays.csv",sep=";")
print(" La liste N° 1 \n")
print (df0)
df1=ps.read_csv("pays1.csv",sep=";")
print(" La liste N° 2 \n")
print (df1)
df_res=ps.concat([df0,df1],ignore_index = True)
print(" La liste Finale \n")
print (df_res)
```

dataframe avant exécution de l'instruction

La liste N° 1

	Pays	Superficie	Habitant
0	Russ	17098242	146781095
1	Canada	998467	37560207
2	USA	9826675	331883986
3	Chine	9596961	1394112547
4	Brésil	8514877	210301591
5	Australie	774122	25105503

La liste N° 2

	Pays	Superficie	Habitant
0	Nigéria	923768	212871345
1	Algérie	2381741	43003767

Affichage après exécution de l'instruction

La liste Finale

	Pays	Superficie	Habitant
0	Russ	17098242	146781095
1	Canada	998467	37560207
2	USA	9826675	331883986
3	Chine	9596961	1394112547
4	Brésil	8514877	210301591
5	Australie	774122	25105503
6	**Nigéria**	**923768**	**212871345**
7	**Algérie**	**2381741**	**43003767**

A.4.n) Ajout des données dans d'un dataframe

Pour ajouter des données dans un dataframe, on utilise les instructions suivantes :

```
Nouveau_dataframe= pandas.DataFrame([[La liste des valeurs]],columns=[La liste des colonnes])
dataframe= dataframe.append(Nouveau_dataframe ,sort=False, ignore_index=True/False )
```

Exemple N° 1 :
Ajouter les informations concernant la Tunisie avec une superficie de 162155 et un nombre d'habitants =11582075 à la fin du dataframe.

Le code

```
import pandas as ps
df=ps.read_csv("pays.csv",sep=";")
df1=ps.DataFrame([["Tunisie",162155,11582075]],columns=["Pays","Superficie","Habitant"])
df=df.append(df1,sort=False,ignore_index=True)
print (df)
```

L'affichage :

	Pays	Superficie	Habitant
0	Russ	17098242	146781095
1	Canada	998467	37560207
2	USA	9826675	331883986
3	Chine	9596961	1394112547
4	Brésil	8514877	210301591
5	Australie	774122	25105503
6	**Tunisie**	**162155**	**11582075**

Exemple N° 2 :

Ajouter les informations concernant la Tunisie avec une superficie de 162155 et un nombre d'habitants =11582075 à la 4ème position.

Le code

```
import pandas as ps
df=ps.read_csv("pays.csv",sep=";")
df0=df.iloc[0:3]
df1=df.iloc[3:]
df2=ps.DataFrame([["Tunisie",162155,11582075]],columns=["Pays","Superficie","Habitant"])
df0=df0.append(df2,sort=False,ignore_index=True)
df=df0.append(df1,sort=False,ignore_index=True)
print (df)
```

L'affichage :

	Pays	Superficie	Habitant
0	Russ	17098242	146781095
1	Canada	998467	37560207
2	USA	9826675	331883986
3	**Tunisie**	**162155**	**11582075**
4	Chine	9596961	1394112547
5	Brésil	8514877	210301591

6	Australie	774122	25105503

A.4.o) Suppression des données dans d'un dataframe

Pour supprimer une ligne d'un dataframe, on utilise l'instruction suivante :

```
dataframe = dataframe.drop([indice_ligne])
```

Exemple :
Supprimer la 1ère ligne.

Le code

```
import pandas as ps
df=ps.read_csv("pays.csv",sep=";")
df = df.drop([0])
print (df)
```

L'affichage :

	Pays	Superficie	Habitant
1	Canada	998467	37560207
2	USA	9826675	331883986
3	Chine	9596961	1394112547
4	Brésil	8514877	210301591
5	Australie	774122	25105503

A.4.p) Suppression des doublons dans d'un dataframe

Pour supprimer les doublons d'un dataframe, on utilise l'instruction suivante :

```
dataframe = dataframe. drop_duplicates()
```

Exemple :
Supprimer les doublons.

Le code :

```
import pandas as ps
df=ps.read_csv("pays03.csv",sep=";")
print (df)
df=df.drop_duplicates()
print (df)
```

dataframe avant exécution du code

	Pays	Superficie	Habitant
0	Russ	17098242	146781095
1	Canada	998467	37560207
2	USA	9826675	331883986
3	Tunisie	162155	11582075
4	Chine	9596961	1394112547
5	Brésil	8514877	210301591
6	Australie	774122	25105503
7	Tunisie	162155	11582075

Affichage après exécution du code

	Pays	Superficie	Habitant
0	Russ	17098242	146781095
1	Canada	998467	37560207
2	USA	9826675	331883986
3	Tunisie	162155	11582075
4	Chine	9596961	1394112547
5	Brésil	8514877	210301591
6	Australie	774122	25105503

A.5 / Les fonctions statistiques sur les dataframes

Fonction	Description	Exemple	Résultat
sum()	La somme des éléments	#la somme des surfaces print (df.Superficie.sum())	46809344
prod()	Le produit des éléments	#le produit des surfaces print (df.Superficie.prod())	230507958054795876
len(dataframe)	Le nombre de lignes dans dataframe	print (len(df))	6
count()	Le nombre d'éléments d'un objets (ou series, dataframe, liste, etc)	#le nombre d'élément dans chaque colonne print (df.count())	Pays 6 Superficie 6 Habitant 6 dtype: int64
mean()	La moyenne des éléments	#La moyenne des habitants print (df.habitant.mean())	357624154.8333333
median()	Médiane des éléments	#La médiane des habitants print (df.habitant.median())	178541343.0
min()	Le minimum des éléments	#La surface la plus petite print (df.Superficie.min())	774122
max()	Le maximum des éléments	#La surface la plus grande print (df.Superficie.max())	17098242
describe()	Réaliser des statistiques descriptives sur un dataframe ou une colonne	print(df.describe())	Superficie Habitant count 6.000000e+00 6.000000e+00 mean 7.801557e+06 3.576242e+08 std 6.163256e+06 5.204112e+08 min 7.741220e+05 2.510550e+07 25% 2.877570e+06 6.486543e+07 50% 9.055919e+06 1.785413e+08 75% 9.769246e+06 3.014884e+08 max 1.709824e+07 1.394113e+09

B / Présentation des données à travers des graphiques :

B.1 / Installation de la bibliothèque matplotlib

```
python -m pip install matplotlib
```

B.2 / Importation de la bibliothèque matplotlib

```
import matplotlib.pyplot as alias
ou bien
import matplotlib.pyplot
```

B.3 / Création d'un graphique à Barres

- **Syntaxe :**

Pour créer un graphique à barres, on utilise l'instruction suivante :

```
pyplot.bar ([Position1 , Position2 , PositionN] , [Valeur1 , Valeur2 , ValeurN] , align="Position" , width =Taille ,  color="Couleur/Code_Couleur")
```

- **Explication détaillée :**

Commande	Description
pyplot.bar	Un graphique de type Barre
[Position1 , Position2 , PositionN]	Un tableau qui contient les indices des positions dans l'axe des X
[Valeur1 , Valeur2 , ValeurN]	Un tableau qui contient les indices des valeurs dans l'axe des Y
align = "Position"	La position des barres Position = center / left / right
width = Taille	La taille de la barre
color = "Couleur/Code couleur"	La couleur de la barre

- **Autres options**

Commande	Description
pyplot.xticks([Position1, Position2, positionN], [Valeur1, Valeur2, ValeurN])	Permet d'ajouter les valeurs (Valeur1..ValeurN) sur l'axe des X
pyplot.xlabel("Titre X")	Permet d'ajouter un titre pour l'axe des X
pyplot.ylabel("Titre Y")	Permet d'ajouter un titre pour l'axe des Y
pyplot.title("Titre graphique")	Permet d'ajouter un titre pour le graphique
pyplot.legend(["Légende1","Légende2", "Légende N"], loc="Position")	Permet d'ajouter des légendes pour chaque barre du graphique dans une position donnée Position="upper left" / "upper right "/ "best"/"lower left" /"lower right" / "right" / "center left" / "center right" / "lower center"/ "upper center"/ "center"
pyplot.show()	Afficher le graphique

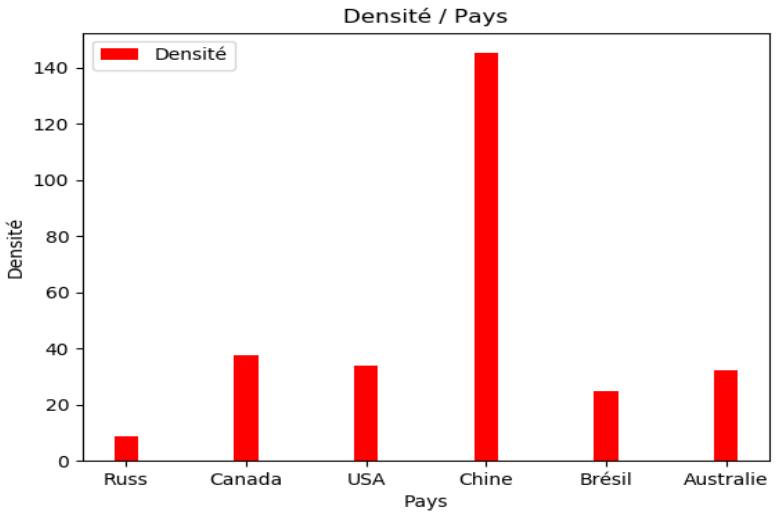
Exemple :

Créer un graphique à barres qui permet de représenter pour chaque pays sa densité.

Le code :

```
import pandas as ps
import matplotlib.pyplot as plt
df=ps.read_csv("pays.csv",sep=";")
#------------------------------------------------ Calcul de la densité pour chaque pays
df["Densité"]=df.Habitant/df.Superficie
#------------------------------------------------ Récupération des indices du dataframe
ind=df.index.values
#------------------------------------------------ Création du graphique de type barre
plt.bar(ind,df["Densité"],align='center',width =0.2, color='red')
#------------------------------------------------ Les valeurs de l'axe des X
plt.xticks(ind, df.Pays)
#------------------------------------------------ Titre de l'axe des X
plt.xlabel('Pays')
#------------------------------------------------ Titre de l'axe des Y
plt.ylabel('Densité')
#------------------------------------------------ Titre du graphique
plt.title('Densité / Pays')
#------------------------------------------------ Les légendes
plt.legend(['Densité'], loc='upper left')
#------------------------------------------------ L'affichage
plt.show()
```

L'affichage :



B.4 / Création d'un graphique en histogramme

- **Syntaxe :**

Pour créer un graphique en histogramme, on utilise l'instruction suivante :

```
pyplot.hist([Valeur1 , Valeur2 , ValeurN] , bins = Nombre , color = "Couleur/Code_Couleur", label = "Libellé")
```

- **Explication détaillée :**

Commande	Description
pyplot.hist	Un graphique en histogramme
[Valeur1, Valeur2, ValeurN]	Un tableau qui contient les indices des positions dans l'axe des X
color="Couleur/Code_Couleur"	La couleur de la barre
label="Libellé"	Représente le libellé des histogrammes

- **Autres options**

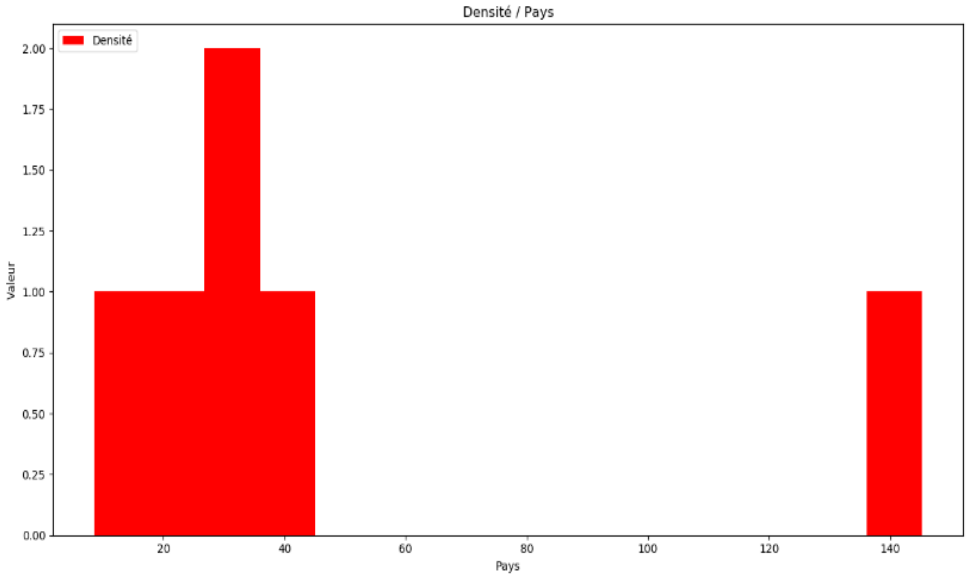
Commande	Description
pyplot.xlabel("Titre X")	Permet d'ajouter un titre à l'axe des X
pyplot.ylabel("Titre Y")	Permet d'ajouter un titre à l'axe des Y
pyplot.title("Titre graphique")	Permet d'ajouter un titre au graphique
pyplot.legend(["Légende1","Légende2", "Lengende N"], loc="position")	Permet d'ajouter des légendes pour chaque barre du graphique dans une position donnée Position="upper left" / "upper right "/ "best"/"lower left" /"lower right" / "right" / "center left" / "center right" / "lower center"/ "upper center"/ "center"
pyplot.show()	Afficher le graphique

Exemple :
Créer un graphique en histogramme qui permet de représenter les densités.

Le code :

```
import pandas as ps
import matplotlib.pyplot as plt
import numpy as np
df=ps.read_csv("pays.csv",sep=";")
#---------------------------------------------- Calcul de la densité pour chaque pays
df["Densité"]=df.Habitant/df.Superficie
#---------------------------------------------- Création d'un graphique de type histogramme
plt.hist(df["Densité"],bins=15, color='red',label="Densité")
#---------------------------------------------- Titre de l'axe des X
plt.xlabel('Pays')
# ----------------------------------------------Titre de l'axe des Y
plt.ylabel('Valeur')
# ----------------------------------------------Titre du graphique
plt.title('Densité / Pays')
#---------------------------------------------- Les légendes
plt.legend(['Densité'], loc='upper left')
#---------------------------------------------- l'affichage
plt.show()
```

L'affichage :



B.5 / Création d'un graphique en camembert

- **Syntaxe :**

Pour créer un graphique de type camembert, on utilise l'instruction suivante :

```
Nouvelle_liste=Tableau des données

pyplot.pie(Nouvelle_liste , labels=[Libellé1, Libellé2 , LibelléN] ,  autopct = lambda
Nouvelle_liste: str(round(Nouvelle_liste)) + '%', shadow = True/False, pctdistance =Nombre,
labeldistance = Nombre)
```

- **Explication détaillée :**

Commande	Description
pyplot.pie	Un graphique en camembert.
Nouvelle_liste	Un tableau contenant les valeurs des portions du camembert.
labels=[Libellé1, Libellé1, LibelléN]	Représente le libellé de chaque secteur du Camembert.
autopct = lambda Nouvelle_liste: str(round(Nouvelle_liste)) + '%'	Une fonction qui prend le pourcentage du secteur et renvoie ce qui doit être affiché en pourcentage.
shadow = True/False	Indique qu'il faut afficher une ombre.
pctdistance =Nombre	La position de l'affichage de la valeur du pourcentage par rapport au centre (Rayon du cercle = 1).
labeldistance = Nombre	La position de l'affichage des libellés de chaque secteur par rapport au centre (Rayon du cercle = 1).

- **Autres options**

Commande	Description
pyplot.title("Titre graphique")	Permet d'ajouter un titre pour le graphique
pyplot.legend(loc="position")	Permet d'ajouter des légendes pour chaque barre du graphique dans une position donnée Position="upper left" / "upper right "/ "best"/"lower left" /"lower right" / "right" / "center left" / "center right" / "lower center"/ "upper center"/ "center"
pyplot.show()	Afficher le graphique

Exemple :
Créer un graphique en Camembert qui permet de représenter le taux de la densité de chaque pays.

Le code :

```
import pandas as ps
import matplotlib.pyplot as plt
df=ps.read_csv("pays.csv",sep=";")
#---------------------------------------------------- Calcul de la densité pour chaque pays
df["Densité"]=df.Habitant/df.Superficie
#---------------------------------------------------- Calcul du taux de densité
df["Taux"]=df["Densité"]/ df.Densité.sum()
#---------------------------------------------------- Création d'un graphique de type histogramme
list0=df["Taux"]
plt.pie(list0, labels=df.Pays,  autopct = lambda list0: str(round(list0)) + '%',shadow = True,
pctdistance = 0.7, labeldistance = 1.2)
#---------------------------------------------------- Titre du graphique
plt.title('Taux / Pays')
# ---------------------------------------------------- Les légendes
plt.legend(loc='upper left')
plt.show()      #---------------------------------------------------- L'affichage
```

L'affichage :