Université Sultan Moulay Slimane Faculté Polydisciplinaire Béni Mellal
Département INFORMATIQUE (MIP)

Filière : Science de données et sécurité des systèmes d'information
A.U : 2023-2024

Sujet

Compte Rendu de TP05 – Python pour la science des données

Présenté Par :
MAFTOUH Omar

Encadré Par :
Z. BOUSALEM
Y. MADANI

Introduction :

Le traitement de données avec la bibliothèque Pandas en Python est une compétence cruciale pour tout analyste de données ou scientifique des données. Dans cet exercice, nous allons explorer diverses fonctionnalités de Pandas en utilisant un ensemble de données contenu dans le fichier CSV "Personne.csv". Nous allons effectuer des opérations telles que la création d'un DataFrame, l'exploration des données, la sélection et la manipulation des données, le calcul de statistiques descriptives et la création de nouvelles colonnes dérivées. Tout au long de cet exercice, nous utiliserons des fonctions et des méthodes de Pandas pour obtenir des informations précieuses sur notre ensemble de données.

Application :

- **Question 01 & 02 :**

```
import pandas as pd
import numpy as np

# 1 Creation d'un DataFrame à partir du fichier Personne.csv
df = pd.read_csv("Personne - Personne.csv" , sep=',' , index_col="id")

# 2 les 10 premières entrées après la création du DataFrame
print(f"- Les 10 premières entrées : \n{df.loc[0 : 10]}")
```

- **Résultat :**

```
        Nom     Prénom  Age  Taille  Poids   Salaire
id
1     Nom_1   Prenom_1   39  173.27  73.21  10084.34
2     Nom_2   Prenom_2   37  166.89  54.76  11533.22
3     Nom_3   Prenom_3   22  191.06  50.02  10808.93
4     Nom_4   Prenom_4   29  193.09  71.94  11697.27
5     Nom_5   Prenom_5   36  159.19  52.21   9210.91
6     Nom_6   Prenom_6   39  172.40  78.27  11190.55
7     Nom_7   Prenom_7   42  171.66  79.95   8663.09
8     Nom_8   Prenom_8   39  168.05  58.51  11220.31
9     Nom_9   Prenom_9   42  182.98  66.85   8793.78
10   Nom_10  Prenom_10   38  195.00  56.57  11440.60
```

- **Question 03 , 04 , 05 , 06 , 07 et 08 :**

```
# 3 nombre dobservation
print(f"- Nombre d\'observation : {df.shape[0]}")

# 4 Nombre de colone
print(f"- Nombre de colones : {df.shape[1]}")

# 5 le nom de toutes les colonnes
print(f"- Le nom de tous les colones : \n{df.columns}")

# 6 indexation
print(f"- L\'ensemble de données est indexé : \n{df.index}")

# 7 Type de chaque colone
print(f"- Type de chaque colone : \n{df.dtypes}")

# 8 Afficher la colonne « Salaire »
print(f"- Le contenu du colone salaire : \n{df.loc[: , ['Salaire']]}")
```

- **Résultat :**

```
- Nombre d'observation : 5000
- Nombre de colones : 6
- Le nom de tous les colones :
Index(['Nom', 'Prénom', 'Age', 'Taille', 'Poids', 'Salaire'], dtype='object')
- L'ensemble de données est indexé :
Index([   1,    2,    3,    4,    5,    6,    7,    8,    9,   10,
       ...
       4991, 4992, 4993, 4994, 4995, 4996, 4997, 4998, 4999, 5000],
      dtype='int64', name='id', length=5000)
- Type de chaque colone :
Nom         object
Prénom      object
Age          int64
Taille     float64
Poids      float64
Salaire    float64
dtype: object
- Le contenu du colone salaire :
       Salaire
id
1     10084.34
2     11533.22
3     10808.93
4     11697.27
5      9210.91
...
4999  13289.69
5000  12371.97
```

- **Question 09, 10 et 11 :**

```
# 9 Le nombre d'âges différents dans l'ensemble de données
print(f"Le nombre d\'âges différents dans l\'ensemble de données :
{df["Age"].nunique()}")

# 10 l'âge le plus fréquent
print(f"l\'âge le plus fréquent : {df["Age"].value_counts().idxmax()}")

# 11 le salaire le moins fréquent
print(f"le salaire le moins fréquent :
{df["Salaire"].value_counts().idxmin()}")
```

- **Résultat :**

```
Le nombre d'âges différents dans l'ensemble de données : 22
l'âge le plus fréquent : 36
le salaire le moins fréquent : 6780.66
```

- **Question 12 :**

```
# 12 Résumez le DataFrame
print(f"Résumez le DataFrame : \n{df.describe()}")
```

- **Résultat :**

```
Résumez le DataFrame :
               Age       Taille        Poids       Salaire
count  5000.000000  5000.000000  5000.000000   5000.000000
mean     31.569400   174.944166    72.346476  10049.931716
std       6.399295    14.359552    12.991692   2305.314949
min      21.000000   150.010000    50.000000   6001.460000
25%      26.000000   162.535000    61.065000   8036.570000
50%      32.000000   174.790000    72.210000  10065.770000
75%      37.000000   187.482500    83.642500  12064.845000
max      42.000000   200.000000    94.990000  13999.970000
```

- **Interprétation :**

* Pour la colonne "Age", le nombre d'observations est de 5000. La moyenne d'âge est d'environ 31,6 ans, avec un écart-type d'environ 6,4 ans. L'âge minimum est de 21 ans, tandis que l'âge maximum est de 42 ans.
* Pour la colonne "Taille", le nombre d'observations est également de 5000. La moyenne de la taille est d'environ 174,9 cm, avec un écart-type d'environ 14,4 cm. La taille minimum est d'environ 150 cm, tandis que la taille maximum est de 200 cm.
* Pour la colonne "Poids", les statistiques sont similaires. Le poids moyen est d'environ 72,3 kg, avec un écart-type d'environ 13 kg. Le poids varie entre 50 kg (minimum) et 95 kg (maximum).
* Pour la colonne "Salaire", le salaire moyen est d'environ 10049,93 unités, avec un écart-type d'environ 2305,31 unités. Le salaire minimum est d'environ 6001,46 unités et le salaire maximum est d'environ 14000 unités

Question 13 :

```
# 13 Résumer toutes les colonnes du Dataframe
print(f"Résumer toutes les colonnes du Dataframe :
\n{df.describe(include='all')}")
```

Résultat :

```
Résumer toutes les colonnes du Dataframe :
          Nom    Prénom          Age       Taille        Poids       Salaire
count    5000      5000  5000.000000  5000.000000  5000.000000   5000.000000
unique   5000      5000          NaN          NaN          NaN           NaN
top     Nom_1  Prenom_1          NaN          NaN          NaN           NaN
freq        1         1          NaN          NaN          NaN           NaN
mean      NaN       NaN    31.569400   174.944166    72.346476  10049.931716
std       NaN       NaN     6.399295    14.359552    12.991692   2305.314949
min       NaN       NaN    21.000000   150.010000    50.000000   6001.460000
25%       NaN       NaN    26.000000   162.535000    61.065000   8036.570000
50%       NaN       NaN    32.000000   174.790000    72.210000  10065.770000
75%       NaN       NaN    37.000000   187.482500    83.642500  12064.845000
max       NaN       NaN    42.000000   200.000000    94.990000  13999.970000
```

Interprétation :

* Pour les colonnes de texte comme "Nom" et "Prénom", les statistiques comprennent le nombre d'observations uniques, la valeur la plus fréquente (top), et la fréquence de cette valeur la plus fréquente (freq). Dans ce cas, chaque nom et prénom est unique, donc le top et la fréquence sont NaN.
* Les statistiques descriptives pour les colonnes numériques restent les mêmes que pour la question 12, fournissant des informations sur l'âge, la taille, le poids et le salaire.

Question 14 :

```
# 14 "Age" est supérieure à 25
print(f"Age sup que 25 : \n{df["Age"].loc[df["Age"]>25]}")
```

Résultat :

```
Age sup que 25 :
id
1       39
2       37
4       29
5       36
6       39
        ..
4996    34
4997    28
4998    30
4999    33
5000    42
Name: Age, Length: 3854, dtype: int64
```

- **Question 15 & 16 :**

```
# 15 la valeur à la deuxième ligne et à la troisième colonne
print(f"La valeur deuxième ligne et troisième colonne : {df.iloc[1 , 2]}")

# 16 Toutes les lignes pour lesquelles la colonne "Taille" est supérieure à
170
print(f"Les Valeurs sup a 170 : \n{df["Taille"].loc[df["Taille"]>170]}")
```

- **Résultat :**

```
La valeur deuxième ligne et troisième colonne : 21
Les Valeurs sup a 170 :
           Nom       Prénom  Age  Taille  Poids   Salaire  Prime Annuelle  \
id
1607  NOM_1607  PRENOM_1607   21  188.34  93.89   8729.97         872.997
1717  NOM_1717  PRENOM_1717   21  192.97  83.90  12401.14        1240.114
3729  NOM_3729  PRENOM_3729   21  199.22  50.83  12771.95        1277.195
563    NOM_563   PRENOM_563   21  191.11  73.95   9000.85         900.085
4628  NOM_4628  PRENOM_4628   21  191.23  75.48   8743.19         874.319
...        ...          ...  ...     ...    ...       ...             ...
3669  NOM_3669  PRENOM_3669   42  180.96  61.51   8182.64         818.264
1381  NOM_1381  PRENOM_1381   42  190.67  85.54  12289.93        1228.993
742    NOM_742   PRENOM_742   42  178.47  86.14   8335.32         833.532
895    NOM_895   PRENOM_895   42  195.86  87.18  13589.59        1358.959
5000  NOM_5000  PRENOM_5000   42  191.06  52.72  12371.97        1237.197

     catégorie de poids
id
1607              Lourd
1717              Moyen
3729              Lèger
563               Moyen
4628              Moyen
...                 ...
3669              Lèger
...
895               Moyen
5000              Lèger

[2996 rows x 8 columns]
```

Question 17 & 18 :

```
# 17 La première ligne et les deux premières colonnes
print(f"La première ligne et les deux premières colonnes : \n{df.iloc[ 0 , : 2
]}")

# 18 Toutes les lignes où la colonne "Salaire" est supérieure à 7000 et la
colonne "Age" est égale à 30
print(f"Salaire supérieur à 7000 & Age supérieur à 30
:\n{df.loc[(df['Salaire'] > 7000) & (df['Age'] > 30)]}")
```

Résultat :

```
- La première ligne et les deux premières colonnes :
Nom          Nom_1
Prénom    Prenom_1
Name: 1, dtype: object
- Salaire supérieur à 7000 & Age supérieur à 30 :
           Nom       Prénom  Age  Taille  Poids   Salaire
id
1        Nom_1     Prenom_1   39  173.27  73.21  10084.34
2        Nom_2     Prenom_2   37  166.89  54.76  11533.22
5        Nom_5     Prenom_5   36  159.19  52.21   9210.91
6        Nom_6     Prenom_6   39  172.40  78.27  11190.55
7        Nom_7     Prenom_7   42  171.66  79.95   8663.09
...        ...          ...  ...     ...    ...       ...
4994  Nom_4994  Prenom_4994   32  157.29  70.26   7816.15
4995  Nom_4995  Prenom_4995   42  164.68  90.31   7047.74
4996  Nom_4996  Prenom_4996   34  171.76  51.14   9244.03
4999  Nom_4999  Prenom_4999   33  178.79  76.82  13289.69
5000  Nom_5000  Prenom_5000   42  191.06  52.72  12371.97
```

Question 19 :

```
# 19 Trier le Dataframe par âge des personnes
print(df.sort_values("Age"))
```

Résultat :

```
           Nom       Prénom  Age  Taille  Poids   Salaire
id
1607  Nom_1607  Prenom_1607   21  188.34  93.89   8729.97
1693  Nom_1693  Prenom_1693   21  166.47  80.79   8702.91
1717  Nom_1717  Prenom_1717   21  192.97  83.90  12401.14
1736  Nom_1736  Prenom_1736   21  169.47  63.24   6361.12
3729  Nom_3729  Prenom_3729   21  199.22  50.83  12771.95
...        ...          ...  ...     ...    ...       ...
2042  Nom_2042  Prenom_2042   42  159.41  61.87  12717.57
1697  Nom_1697  Prenom_1697   42  164.59  86.05  11898.82
742    Nom_742   Prenom_742   42  178.47  86.14   8335.32
895    Nom_895   Prenom_895   42  195.86  87.18  13589.59
5000  Nom_5000  Prenom_5000   42  191.06  52.72  12371.97
```

- **Question 20 :**

```
# 20 le salaire de la personne la plus jeune
print(f"le salaire de la personne la plus jeune :
\n{df.sort_values(by="Age").iloc[0]}")
```

- **Résultat :**

```
le salaire de la personne la plus jeune :
Nom           Nom_1607
Prénom     Prenom_1607
Age                21
Taille         188.34
Poids           93.89
Salaire        8729.97
Name: 1607, dtype: object
```

- **Question 21 & 22 :**

```
# 21 fonction lambda qui mettra en majuscule les chaînes de caractères.
upperCase = lambda chaine : chaine.upper()

# 22 mettre nom & prenom en majuscule
df["Nom"] = df["Nom"].apply(upperCase)
df["Prénom"] = df["Prénom"].apply(upperCase)
print(f"Nom & prenom en majuscule : \n{df[["Nom" , "Prénom"]]}")
```

- **Résultat :**

```
Nom & prenom en majuscule :
           Nom       Prénom
id
1607  NOM_1607  PRENOM_1607
1693  NOM_1693  PRENOM_1693
1717  NOM_1717  PRENOM_1717
1736  NOM_1736  PRENOM_1736
3729  NOM_3729  PRENOM_3729
...        ...          ...
2042  NOM_2042  PRENOM_2042
1697  NOM_1697  PRENOM_1697
742    NOM_742   PRENOM_742
895    NOM_895   PRENOM_895
5000  NOM_5000  PRENOM_5000
```

Question 23 :

```
# 23 la prime annuelle de chaque employé
def primeAnnuel ( salaire ) :
    return salaire * 0.10

df["Prime Annuelle"] = df["Salaire"].apply(primeAnnuel)
```

Résultat :

```
           Nom       Prénom  Age  Taille  Poids   Salaire  Prime Annuelle
id
1        Nom_1     Prenom_1   39  173.27  73.21  10084.34        1008.434
2        Nom_2     Prenom_2   37  166.89  54.76  11533.22        1153.322
3        Nom_3     Prenom_3   22  191.06  50.02  10808.93        1080.893
4        Nom_4     Prenom_4   29  193.09  71.94  11697.27        1169.727
5        Nom_5     Prenom_5   36  159.19  52.21   9210.91         921.091
...        ...          ...  ...     ...    ...       ...             ...
4996  Nom_4996  Prenom_4996   34  171.76  51.14   9244.03         924.403
4997  Nom_4997  Prenom_4997   28  177.82  50.51   8413.23         841.323
4998  Nom_4998  Prenom_4998   30  153.11  91.33   7099.65         709.965
4999  Nom_4999  Prenom_4999   33  178.79  76.82  13289.69        1328.969
5000  Nom_5000  Prenom_5000   42  191.06  52.72  12371.97        1237.197
```

Question 24 :

```
# 24 Grouper selon Age & Calculer la moyenne des salaires pour chaque groupe
d'âge
groupAge = df.groupby("Age")["Salaire"].mean()
```

Résultat :

```
21    10119.726420
22     9981.018515
23    10048.084820
24     9880.982837
25     9898.889747
26    10230.765455
27    10444.025256
28    10078.641106
29    10138.074234
30    10210.449773
31    10069.967048
32    10126.509957
33     9892.771587
34     9764.361472
35    10055.175684
36     9977.249494
37     9876.418834
38    10139.296776
39    10055.887439
40     9984.453964
41     9995.321759
42    10096.159431
```

- **Question 25 :**

```
# 25 colonne 'catégorie de poids’
def categoriePoids ( poid ) :
    if poid <= 50 :
        return "Très léger"
    elif poid > 50  and poid <= 70 :
        return "Lèger"
    elif poid > 70 and poid <= 90 :
        return "Moyen"
    elif poid > 90 :
        return "Lourd"

df["catégorie de poids"] = df["Poids"].apply(categoriePoids)
```

- **Résultat :**

```
           Nom       Prénom  Age  Taille  Poids   Salaire  Prime Annuelle  \
id
1        Nom_1     Prenom_1   39  173.27  73.21  10084.34        1008.434
2        Nom_2     Prenom_2   37  166.89  54.76  11533.22        1153.322
3        Nom_3     Prenom_3   22  191.06  50.02  10808.93        1080.893
4        Nom_4     Prenom_4   29  193.09  71.94  11697.27        1169.727
5        Nom_5     Prenom_5   36  159.19  52.21   9210.91         921.091
...        ...          ...  ...     ...    ...       ...             ...
4996  Nom_4996  Prenom_4996   34  171.76  51.14   9244.03         924.403
4997  Nom_4997  Prenom_4997   28  177.82  50.51   8413.23         841.323
4998  Nom_4998  Prenom_4998   30  153.11  91.33   7099.65         709.965
4999  Nom_4999  Prenom_4999   33  178.79  76.82  13289.69        1328.969
5000  Nom_5000  Prenom_5000   42  191.06  52.72  12371.97        1237.197

     catégorie de poids
id
1                 Moyen
2                 Lèger
3                 Lèger
4                 Moyen
5                 Lèger
...                 ...
4996              Lèger
4997              Lèger
4998              Lourd
4999              Moyen
5000              Lèger
```

- **Question 26 & 27 :**

```
# 26 La catégorie de poids la plus courante
print(f"Categorie du poid plus courante : {df["catégorie de
poids"].value_counts().idxmax()}")
# 27 groupe d'âge et catégorie de poids, puis calculer la moyenne des salaires
pour chaque combinaison.
moySalaire = df.groupby(["Age" , "catégorie de poids"])["Salaire"].mean()
print(f"Moyenne de salaire par Age & categorie de poids : \n{moySalaire}")
```

- **Résultat :**

```
- Categorie du poid plus courante : Moyen
- Moyenne de salaire par Age & categorie de poids :
Age  catégorie de poids
21   Lourd                 9998.593704
     Lèger                10068.749817
     Moyen                10202.222056
22   Lourd                 9586.951176
     Lèger                10003.921727
                             ...
41   Lèger                10048.863370
     Moyen                10074.402360
42   Lourd                 9644.189200
     Lèger                10288.649292
     Moyen                 9999.380741
```

- **Question 28 :**

```
df.groupby("catégorie de poids")["Taille"].describe()
```

- **Résultat :**

```
                     count        mean        std     min       25%     50%  \
catégorie de poids
Lourd                512.0  175.775488  14.911754  150.07  162.2000  176.06
Lèger               2241.0  174.965199  14.318334  150.02  162.5900  174.92
Moyen               2246.0  174.744439  14.265837  150.01  162.6175  174.01
Très léger             1.0  150.760000        NaN  150.76  150.7600  150.76

                         75%     max
catégorie de poids
Lourd                189.5725  199.85
Lèger                187.2300  199.99
Moyen                187.3700  200.00
Très léger           150.7600  150.76
```

- **Question 29 :**

```
df.groupby("catégorie de poids")["Salaire"].agg(["mean" , "max" , "min" , "median"])
```

- **Résultat :**

```
                           mean       max       min     median
catégorie de poids
Lourd                 9936.055449  13949.11   6005.97   9866.335
Lèger                10041.194516  13999.55   6002.86  10080.400
Moyen                10083.478246  13999.97   6001.46  10093.470
Très léger           12589.140000  12589.14  12589.14  12589.140
```

- **Question 30 :**

```
      df.to_csv("DataFrame.csv")
```

Conclusion :

Dans cet exercice, nous avons parcouru diverses fonctionnalités de la bibliothèque Pandas en manipulant l'ensemble de données contenu dans le fichier CSV "Personne.csv". Nous avons créé un DataFrame à partir du fichier CSV, exploré les données, effectué des opérations de sélection et de manipulation, calculé des statistiques descriptives et créé de nouvelles colonnes dérivées. Ces opérations nous ont permis d'obtenir des informations précieuses sur les caractéristiques de nos données, telles que les distributions d'âge, de salaire et de poids, ainsi que les relations entre ces variables. En fin de compte, Pandas s'est avéré être un outil puissant et polyvalent pour l'analyse et la manipulation de données en Python.