LAB – 2
PYTHON BASIC PRACTICE – II

210905189
Kushala Sarada A V
Roll No. 33

15/01/2024

```
import pandas as pd
from IPython.display import display
data = [['Dinesh',10],['Nithya',12],['Raji',13]]
df = pd.DataFrame(data,columns=['Name','Age'])
display(df)
```

```
210905189_Kushala@networklab:~/Desktop/210905189/Lab2$ python3 sample.py
     Name  Age
0  Dinesh   10
1  Nithya   12
2    Raji   13
210905189_Kushala@networklab:~/Desktop/210905189/Lab2$
```

```
import pandas as pd
from IPython.display import display
data = {'Name':['Kavitha', 'Sudha', 'Raju','Vignesh'],'Age':[28,34,29,42]}
df = pd.DataFrame(data, index=['rank1','rank2','rank3','rank4'])
display(df)
```

```
210905189_Kushala@networklab:~/Desktop/210905189/Lab2$ python3 sample1.py
          Name  Age
rank1  Kavitha   28
rank2    Sudha   34
rank3     Raju   29
rank4  Vignesh   42
210905189_Kushala@networklab:~/Desktop/210905189/Lab2$
```

```
import pandas as pd
import numpy as np
from IPython.display import display

# Create DataFrame df1
df1 = pd.DataFrame({
    'A': pd.Timestamp('20130102'),
    'B': np.array([3] * 4, dtype='int32'),
    'C': pd.Categorical(['Male', 'Female', 'Male', 'Female'])
})

# Display df1
display(df1)

# Display dtypes of df1
print("Data Types:")
print(df1.dtypes)

# Display the first few rows of df1
print("\nHead:")
print(df1.head())
```

```
# Display the last few rows of df1
print("\nTail:")
print(df1.tail())

# Display summary statistics of df1
print("\nSummary:")
print(df1.describe())

# Transpose df1 and display
print("\nTranspose:")
print(df1.T)
```

```
Head:
           A  B       C
0 2013-01-02  3    Male
1 2013-01-02  3  Female
2 2013-01-02  3    Male
3 2013-01-02  3  Female

Tail:
           A  B       C
0 2013-01-02  3    Male
1 2013-01-02  3  Female
2 2013-01-02  3    Male
3 2013-01-02  3  Female

Summary:
                        A    B
count                   4  4.0
mean   2013-01-02 00:00:00  3.0
min    2013-01-02 00:00:00  3.0
25%    2013-01-02 00:00:00  3.0
50%    2013-01-02 00:00:00  3.0
75%    2013-01-02 00:00:00  3.0
max    2013-01-02 00:00:00  3.0
std                    NaN  0.0

Transpose:
                     0                   1                   2                   3
A  2013-01-02 00:00:00  2013-01-02 00:00:00  2013-01-02 00:00:00  2013-01-02 00:00:00
B                    3                   3                   3                   3
C                 Male              Female                Male              Female
210905189_Kushala@networklab:~/Desktop/210905189/Lab2$
```

```
import pandas as pd
import numpy as np

# Creating a DataFrame with random data
dates = pd.date_range('20130101', periods=100)
df = pd.DataFrame(np.random.randn(100, 4), index=dates, columns=list('ABCD'))

# Viewing the first 5 records
print("First 5 records:")
print(df.head())

# Viewing the last 5 records
print("\nLast 5 records:")
print(df.tail())

# Viewing the index
```

```
print("\nIndex:")
print(df.index)

# Viewing the column names
print("\nColumn names:")
print(df.columns)

# Transposing the DataFrame
print("\nTransposed DataFrame:")
print(df.T)

# Sorting by axis (columns in this case) in descending order
print("\nSorting by Axis:")
print(df.sort_index(axis=1, ascending=False))

# Sorting by values in column 'B'
print("\nSorting by Values in Column 'B':")
print(df.sort_values(by='B'))

# Slicing the first 3 records (rows)
print("\nSlicing the first 3 records:")
print(df[0:3])

# Slicing with index name for a specific date range
print("\nSlicing with index name for a date range:")
print(df['20130105':'20130110'])

# Slicing with row and column index
print("\nSlicing with row and column index:")
# Fetching entire 1st row
print(df.iloc[0])
# Fetching 1st row, first 2 columns
print(df.iloc[0, :2])
# Fetching a single element (1st row, 1st column)
print(df.iloc[0, 0])

# Selecting a single column ('A'), which yields a Series
print("\nSelecting a single column ('A'):")
print(df['A'])

# Selecting more than one column ('A' and 'B'), entire 2 columns
print("\nSelecting more than one column ('A' and 'B'):")
print(df[['A', 'B']])

# Selecting more than one column ('A' and 'B') with the first 5 records
print("\nSelecting more than one column with the first 5 records:")
print(df[['A', 'B']][:5])
# Alternatively using loc for a specific date range and selected columns
print(df.loc['20130101':'20130105', ['A', 'B']][:5])
```

```
210905189_Kushala@networklab:~/Desktop/210905189/Lab2$ python3 sample3.py
First 5 records:
                   A         B         C         D
2013-01-01 -0.464549 -0.528514 -0.651055  1.619336
2013-01-02  2.284597 -0.064330  0.224792 -1.914929
2013-01-03 -2.572284  1.158340 -0.667139 -0.620028
2013-01-04  0.785513 -0.180376 -1.034600 -0.953221
2013-01-05 -1.170296 -0.379970  0.972397  1.275571

Last 5 records:
                   A         B         C         D
2013-04-06 -0.045555 -0.303683 -0.031196 -1.557019
2013-04-07 -1.894843 -1.624150 -0.209579  1.487102
2013-04-08 -0.960703 -0.730921  0.286278  0.017152
2013-04-09 -1.985095  1.029690 -1.235847  0.949298
2013-04-10  1.099472 -0.309944  0.244693 -0.023092
```

```
Selecting a single column ('A'):
2013-01-01   -0.464549
2013-01-02    2.284597
2013-01-03   -2.572284
2013-01-04    0.785513
2013-01-05   -1.170296
                ...
2013-04-06   -0.045555
2013-04-07   -1.894843
2013-04-08   -0.960703
2013-04-09   -1.985095
2013-04-10    1.099472
Freq: D, Name: A, Length: 100, dtype: float64

Selecting more than one column ('A' and 'B'):
                   A         B
2013-01-01 -0.464549 -0.528514
2013-01-02  2.284597 -0.064330
2013-01-03 -2.572284  1.158340
2013-01-04  0.785513 -0.180376
2013-01-05 -1.170296 -0.379970
...              ...       ...
2013-04-06 -0.045555 -0.303683
2013-04-07 -1.894843 -1.624150
2013-04-08 -0.960703 -0.730921
2013-04-09 -1.985095  1.029690
2013-04-10  1.099472 -0.309944

[100 rows x 2 columns]

Selecting more than one column with the first 5 records:
                   A         B
2013-01-01 -0.464549 -0.528514
2013-01-02  2.284597 -0.064330
2013-01-03 -2.572284  1.158340
2013-01-04  0.785513 -0.180376
2013-01-05 -1.170296 -0.379970
                   A         B
2013-01-01 -0.464549 -0.528514
2013-01-02  2.284597 -0.064330
2013-01-03 -2.572284  1.158340
2013-01-04  0.785513 -0.180376
2013-01-05 -1.170296 -0.379970
210905189_Kushala@networklab:~/Desktop/210905189/Lab2$
```

```
import pandas as pd
import numpy as np

# Creating a
DataFrame with random data
dates = pd.date_range('20130101', periods=10)
df = pd.DataFrame(np.random.randn(10, 4), index=dates, columns=list('ABCD'))

# Boolean Indexing: Fetching all rows where column 'A' has positive values
print("Boolean Indexing:")
print(df[df['A'] > 0])

# Adding a new column 'F' with categorical character data
df['F'] = ['Male', 'Female', 'Female', 'Male', 'Female', 'Female', 'Male', 'Male', 'Female', 'Male']

# Setting by assigning with a numpy array: Replacing the 'D' column with all 5
df.loc[:, 'D'] = np.array([5] * len(df))
print("\nSetting 'D' column with all 5:")
```

```
print(df)

# Deleting a column: Dropping the 'C' column
df.drop('C', axis=1, inplace=True)
print("\nAfter dropping 'C' column:")
print(df)

# Deleting a row: Dropping the row with index 20130104
df.drop(pd.Timestamp('20130104'), axis=0, inplace=True)
print("\nAfter dropping row with index 20130104:")
print(df)

# Concatenating two DataFrames horizontally
df1 = pd.DataFrame(np.random.randn(10, 5), columns=list('ABCDE'))
df2 = pd.DataFrame(np.random.randn(10, 3), columns=list('FGH'))

df_new = pd.concat([df1, df2], axis=1)
print("\nConcatenation horizontally:")
print(df_new)
print("Shape after concatenation:", df_new.shape)

# Concatenating two DataFrames vertically
A = pd.DataFrame(np.random.randn(10, 5), columns=list('ABCDE'))
B = pd.DataFrame(np.random.randn(15, 5), columns=list('ABCDE'))

D = pd.concat([A, B], axis=0)
print("\nConcatenation vertically:")
print(D)
print("Shape after concatenation:", D.shape)
```

```
210905189_Kushala@networklab:~/Desktop/210905189/Lab2$ python3 sample4.py
Boolean Indexing:
                   A         B         C         D
2013-01-01  0.593535  0.270717  0.104731 -0.158319
2013-01-03  0.106162 -0.707835 -0.402848  0.705766
2013-01-04  0.062085 -0.308419  0.725158  0.210713
2013-01-05  1.451998 -1.020385  0.613440 -1.112928
2013-01-10  0.721252  1.010148  0.618198 -0.358110

Setting 'D' column with all 5:
                   A         B         C    D       F
2013-01-01  0.593535  0.270717  0.104731  5.0    Male
2013-01-02 -0.582429  0.925693 -1.636042  5.0  Female
2013-01-03  0.106162 -0.707835 -0.402848  5.0  Female
2013-01-04  0.062085 -0.308419  0.725158  5.0    Male
2013-01-05  1.451998 -1.020385  0.613440  5.0  Female
2013-01-06 -1.144427 -1.331184  0.739285  5.0  Female
2013-01-07 -0.310689  1.080146 -1.432596  5.0    Male
2013-01-08 -0.498635 -0.628462 -0.207121  5.0    Male
2013-01-09 -0.440630 -0.159880  0.309856  5.0  Female
2013-01-10  0.721252  1.010148  0.618198  5.0    Male

After dropping 'C' column:
                   A         B    D       F
2013-01-01  0.593535  0.270717  5.0    Male
2013-01-02 -0.582429  0.925693  5.0  Female
2013-01-03  0.106162 -0.707835  5.0  Female
2013-01-04  0.062085 -0.308419  5.0    Male
2013-01-05  1.451998 -1.020385  5.0  Female
2013-01-06 -1.144427 -1.331184  5.0  Female
2013-01-07 -0.310689  1.080146  5.0    Male
2013-01-08 -0.498635 -0.628462  5.0    Male
2013-01-09 -0.440630 -0.159880  5.0  Female
2013-01-10  0.721252  1.010148  5.0    Male
```

```
import pandas as pd

# Create the DataFrame
data = {
    'rank1': [28, 34, 29, 42],
    'rank2': ['Kavitha', 'Sudha', 'Raju', 'Vignesh'],
    'rank3': [10, 20, 15, 30],
    'rank4': [5, 8, 12, 7],
    'Age': [28, 34, 29, 42],
    'Name': ['Kavitha', 'Sudha', 'Raju', 'Vignesh']
}

A = pd.DataFrame(data)

# Sort the DataFrame by the 'Age' column
A_sorted = A.sort_values(by='Age')

# Display the sorted DataFrame
print(A_sorted)
```

```
210905189_Kushala@networklab:~/Desktop/210905189/Lab2$ python3 sample5.py
   rank1    rank2  rank3  rank4  Age     Name
0     28  Kavitha     10      5   28  Kavitha
2     29     Raju     15     12   29     Raju
1     34    Sudha     20      8   34    Sudha
3     42  Vignesh     30      7   42  Vignesh
210905189_Kushala@networklab:~/Desktop/210905189/Lab2$
```

```
import pandas as pd

# Reading CSV file with no header
df = pd.read_csv('xyz.csv', header=None)

# Displaying the first 5 records
print("First 5 records:")
print(df.head())

# Displaying the last 5 records
print("\nLast 5 records:")
print(df.tail())

# Assigning custom column names
df.columns = ['preg', 'glu', 'bp', 'sft', 'ins', 'bmi', 'dpf', 'age', 'class']
```

```
210905189_Kushala@networklab:~/Desktop/210905189/Lab2$ python3 sample6.py
First 5 records:
   0    1   2   3    4     5      6   7  8
0  6  148  72  35    0  33.6  0.627  50  1
1  1   85  66  29    0  26.6  0.351  31  0
2  8  183  64   0    0  23.3  0.672  32  1
3  1   89  66  23   94  28.1  0.167  21  0
4  0  137  40  35  168  43.1  2.288  33  1

Last 5 records:
      0    1   2   3    4     5      6   7  8
763  10  101  76  48  180  32.9  0.171  63  0
764   2  122  70  27    0  36.8  0.340  27  0
765   5  121  72  23  112  26.2  0.245  30  0
766   1  126  60   0    0  30.1  0.349  47  1
767   1   93  70  31    0  30.4  0.315  23  0
210905189_Kushala@networklab:~/Desktop/210905189/Lab2$
```

```
import pandas as pd
import matplotlib.pyplot as plt

# Reading CSV file with no header and assigning column names
df = pd.read_csv('xyz.csv', header=None)
df.columns = ['preg', 'glu', 'bp', 'sft', 'ins', 'bmi', 'dpf', 'age', 'class']

# Displaying the first 5 records
print("First 5 records:")
print(df.head())

# Displaying the last 5 records
print("\nLast 5 records:")
print(df.tail())

# Scatter plot of 'bmi' against 'glu'
plt.scatter(df['bmi'], df['glu'])
```

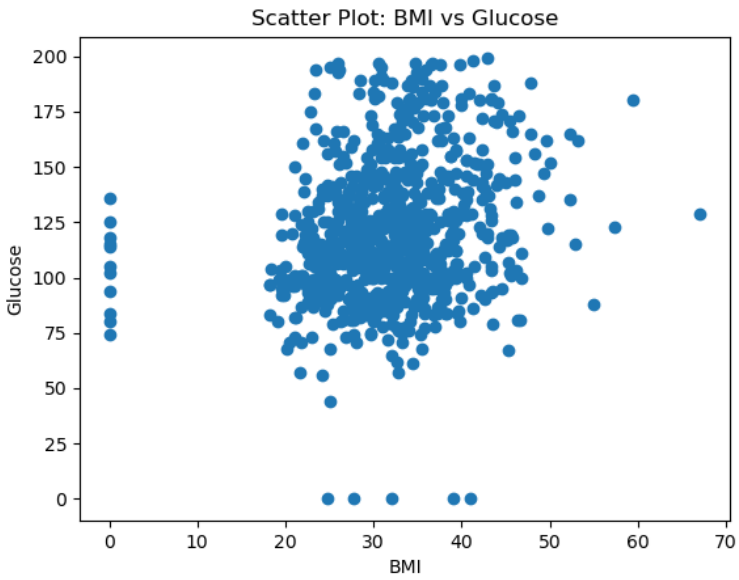
```
plt.xlabel('BMI')
plt.ylabel('Glucose')
plt.title('Scatter Plot: BMI vs Glucose')
plt.show()

# Histogram for the 'age' column
df['age'].hist()
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.title('Histogram: Age Distribution')
plt.show()
```

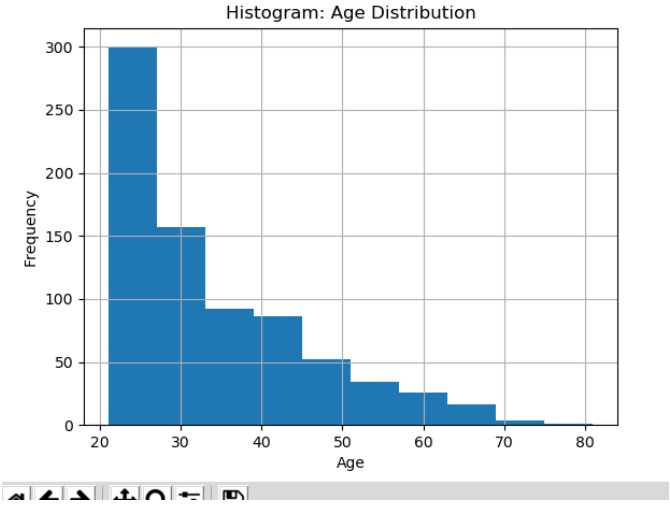
```
210905189_Kushala@networklab:~/Desktop/210905189/Lab2$ python3 sample7.py
First 5 records:
   preg  glu  bp  sft  ins   bmi    dpf  age  class
0     6  148  72   35    0  33.6  0.627   50      1
1     1   85  66   29    0  26.6  0.351   31      0
2     8  183  64    0    0  23.3  0.672   32      1
3     1   89  66   23   94  28.1  0.167   21      0
4     0  137  40   35  168  43.1  2.288   33      1

Last 5 records:
     preg  glu  bp  sft  ins   bmi    dpf  age  class
763    10  101  76   48  180  32.9  0.171   63      0
764     2  122  70   27    0  36.8  0.340   27      0
765     5  121  72   23  112  26.2  0.245   30      0
766     1  126  60    0    0  30.1  0.349   47      1
767     1   93  70   31    0  30.4  0.315   23      0
```

jy



l




```
import pandas as pd

# Reading CSV file with no header
W = pd.read_csv('xyz.xls', header=None)

# Displaying the first 5 records
print("CSV File (xyz.xls):")
print(W.head())

# Reading Excel file with sheet name specified
G = pd.read_excel('xyz.xlsx', sheet_name='Sheet1')

# Displaying the first 5 records
print("\nExcel File (xyz.xlsx):")
print(G.head())
```

```
import pandas as pd
import matplotlib.pyplot as plt

# Reading a text file with tab-separated values
H = pd.read_table('HR.txt')

# Extracting value counts for the 'Department' column
f = H['Department'].value_counts()

# Visualizing the distribution of categorical values using a bar plot
f.plot(kind='bar', color='skyblue', edgecolor='black')
plt.title('Department Distribution')
plt.xlabel('Department')
plt.ylabel('Count')
plt.show()

# Visualizing the above bar plot as a Pie chart
```

```
f.plot(kind='pie', autopct='%1.1f%%', startangle=90, colors=['lightcoral', 'lightgreen', 'lightblue'])
plt.title('Department Distribution (Pie Chart)')
plt.axis('equal')  # Equal aspect ratio ensures that the pie chart is circular.
plt.show()
```