SHARDA UNIVERSITY

GREATER NOIDA

School of Engineering & Technology
Department of Computer Science & Engineering

Pattern Recognition Lab

(CSA 302)

SUBMITTED BY:

NAME: - SHIVAM

CLASS :- CSE-H(G1)

SYSTEM ID:-2020518381

ROLL NO :-200101298

Experiment 1

Aim: Introduction to Python, Operations inPython

Theory:

Python divides the operators in the following groups:

a. Arithmetic operators: They are used with numeric values to perform common mathematical operations. E.g., addition (+), subtraction (-), multiplication (*), etc.

Input:

```
# Python Arithmetic Operators
print (2*3)
print (4/2)
print (4%2)
print (2**2)
print (4//2)
```

Output:

```
6
2.0
0
4
2
4
0
```

b. Assignment operator: They are used to assign values to variables. E.g., =, +=, -=, etc.

Input:

```
# Python Assignment Operation
x = 5
print(x)
x += 3
print(x)
x -= 3
print(x)
x *= 3
print(x)
x%=3
print(x)
x /= 3
print(x)
```

Output:

```
5
8
5
15
0
0.0
```

c. Comparison operators: They are used to compare two values. E.g., equal (=), not equal (!=), greater than (>), etc.

Input:

```
#Comparison Operator
x = 5
y = 3
print(x == y)
print(x != y)
print(x > y)
```

Output:

```
False
True
True
```

d. Logical operators: They are used to combine conditional statements

Input:

```
#logical operator
x = 5
print(x > 3 and x < 10)
print(x > 3 or x < 4)
print(not(x > 3 and x < 10))
```

Output:

```
True
True
False
```

e. Identity operators. E.g., is, is not.

Input:

```
#Identity Operator
x = ["apple", "banana"]
y = ["apple", "banana"]
z = x
print(x is z)
print(x is y)
print(x == y)
print(x is not z)
print(x is not y)
print(x != y)
```

Output:

```
True
False
True
False
True
False
```

f.Membership operators: They are used to test if a sequence is presented in an object. E.g., in, not in.

Input:

```
#Membership Operation
x = ["apple", "banana"]
print("banana" in x)
print("pineapple" not in x)
```

Output:

```
True
True
```

g. Bitwise operators: They are used to compare binary numbers. E.g., AND(&), OR(|), XOR(^), etc. Operator precedence describes the order in which operations are performed.

Input:

```
#Bitwise Operation
print(6 & 3)
print(6 | 3)
print(6 ^ 3)
print(~3)
```

Output:

```
2
7
5
-4
```

Experiment 2

Aim: Write a Python program to do the following operations:
Library: Pandas
a) Loading data from CSV file
b) Compute the basic statistics of given data - shape, no. of columns, mean
c) Splitting a data frame on values of categorical variables
d) Indexing of the data.
e) Display top 10 rows of data

Data Description:

Heart attacks occur when blood flow to a part of the heart muscle is blocked, usually due to a blood clot. Globally,
heart attacks claim millions of lives annually. According to the World Health Organization, in 2022, cardiovascular diseases were the leading cause of death, accounting for approximately 21 million deaths. Timely awareness, lifestyle changes, and medical interventions are crucial in reducing this alarming statistic.
In this model, there are predictions entail training models with historical data to discern patterns, enabling accurate forecasts on new information. Crucial elements include well-defined features, appropriate algorithms, and high-quality data, ensuring the creation of reliable models for diverse applications.

Code:
Importing Libraries

```
import pandas as pd
import numpy as np
```

1. Loading Dataset and creating dataframe

```
data = pd.read_csv("heart.csv")
data.head()
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

2. Computing Shape, No. of columns and Mean

```
print("Shape of Dataset: ",
data.shape)
rows = len(data.axes[0])
cols = len(data.axes[1])
print("Number of Rows: ", rows)
print("Number of Columns", cols)
```

```
Shape of dataset:  (303, 14)
Number of Rows:  303
Number of Columns 14
```

```
data[["cp", "chol", "fbs"]].mean()
```

```
cp          0.966997
chol      246.264026
fbs         0.148515
dtype: float64
```

3. Splitting of dataframe on values of categorial variables
Method : Using Indexing

```
datacategorical_variable =
'output'split_data =
data.groupby(categorical_variable
)first_row = data.loc[0]
Tenth_row = data.loc[9]
fiftieth_row = data.loc[49]
top_10_rows = data.head(10)
pd.set_option('display.max_rows',
None)
pd.set_option('display.max_column
s', None)
pd.set_option('display.width',
None)print(f'c) Data Split on
Categorical Variable
(outout):\n{split_data.head()})
print(f'\nd) Data Indexing :')
print(f'\n first row:\n{first_row}')
print(f'\n Tenth
row:\n{Tenth_row}')print(f'\n
Fiftieth row:\n{fiftieth_row}')
print(f'\ne) Top 10 Rows of
Data:\n{top_10_rows}')
pd.reset_option('display.max_rows')
pd.reset_option('display.max_colum
ns')
pd.reset_option('display.width')
```

c) Data Split on Categorical Variable (outout):

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
165	67	1	0	160	286	0	0	108	1	1.5	1	3	2	0
166	67	1	0	120	229	0	0	129	1	2.6	1	2	3	0
167	62	0	0	140	268	0	0	160	0	3.6	0	2	2	0
168	63	1	0	130	254	0	0	147	0	1.4	1	1	3	0
169	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0

d) Data Indexing :

first row:

```
age         63.0
sex          1.0
cp           3.0
trtbps     145.0
chol       233.0
fbs          1.0
restecg      0.0
thalachh   150.0
exng         0.0
oldpeak      2.3
slp          0.0
caa          0.0
thall        1.0
output       1.0
Name: 0, dtype: float64
```

Tenth row:

```
age         57.0
sex          1.0
cp           2.0
trtbps     150.0
chol       168.0
fbs          0.0
restecg      1.0
thalachh   174.0
exng         0.0
oldpeak      1.6
slp          2.0
caa          0.0
thall        2.0
output       1.0
Name: 9, dtype: float64
```

Fiftieth row:

```
age         53.0
sex          0.0
cp           0.0
trtbps     138.0
chol       234.0
fbs          0.0
restecg      0.0
thalachh   160.0
exng         0.0
oldpeak      0.0
slp          2.0
caa          0.0
thall        2.0
output       1.0
Name: 49, dtype: float64
```

e) Top 10 Rows of Data:

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp
0	63	1	3	145	233	1	0	150	0	2.3	0
1	37	1	2	130	250	0	1	187	0	3.5	0
2	41	0	1	130	204	0	0	172	0	1.4	2
3	56	1	1	120	236	0	1	178	0	0.8	2
4	57	0	0	120	354	0	1	163	1	0.6	2
5	57	1	0	140	192	0	1	148	0	0.4	1
6	56	0	1	140	294	0	0	153	0	1.3	1
7	44	1	1	120	263	0	1	173	0	0.0	2
8	52	1	2	172	199	1	1	162	0	0.5	2
9	57	1	2	150	168	0	1	174	0	1.6	2

	caa	thall	output
0	0	1	1
1	0	2	1
2	0	2	1
3	0	2	1
4	0	2	1
5	0	1	1
6	0	2	1
7	0	3	1
8	0	3	1
9	0	2	1

4. Indexing of the data

#Printing rows 2 to 5
data.loc[2:5]

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
5	57	1	0	140	192	0	1	148	0	0.4	1	0	1	1

#Applying Condition
data.loc[(data.cp< 8)]

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

303 rows × 14 columns

5. Display top 10 rows of data
data.head(10)

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
5	57	1	0	140	192	0	1	148	0	0.4	1	0	1	1
6	56	0	1	140	294	0	0	153	0	1.3	1	0	2	1
7	44	1	1	120	263	0	1	173	0	0.0	2	0	3	1
8	52	1	2	172	199	1	1	162	0	0.5	2	0	3	1
9	57	1	2	150	168	0	1	174	0	1.6	2	0	2	1