AMRUTVAHINI COLLEGE OF ENGINEERING, SANGAMNER

BE(E&C)	Data Science & Visualization Lab	
Experiment No.:01	**Data Manipulation in Python using Pandas.**	**Page:** /08

Aim: Data Manipulation in Python using Pandas.

Software Used: Python 3.12, Jupyter Notebook

Learning Objective

1. Understand the importance of effective data manipulation in data analysis tasks.
2. Learn how to use Pandas, a powerful Python library, for handling and manipulating structured data efficiently.

Learning Outcomes:

After performing the experiment students will be able to-

1. Gain practical skills in data manipulation, includingvarious techniques and functions in Pandas for tasks such as adding columns/rows, dropping columns/rows andrenaming columns/rows.

Theory:

In today's data-driven world, effective data manipulation is essential for extracting valuable insights andmaking informed decisions. Pandas, a powerful Python library, provides a versatile toolkit for handling andmanipulating structured data.

Panda Library

Pandas, a Python library for data analysis and manipulation, is open-source and built on top of NumPy. Itoffers powerful data structures like the Pandas DataFrame and Series for working with structured dataefficiently. Named after "Panel Data," it excels in handling time series and structured datasets. Pandas providesseamless integration with Python, SQL, and variousalgorithms. With its support for data visualization, Pandas is a go-to tool for exploring and analyzing data.

Installation

Install via pip using the following command,

```
pip install pandas
```

Install via anaconda using the following command,

```
conda install pandas
```

PandasDataframe- A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns. A pandas DataFrame can be created using various inputs like – Lists, dictionary, series, Numpyndarrays, another DataFrame.

Example

```
import pandas aspd
data =[['Ram',18],['Shyam',16],['Shiv',20]]
df=pd.DataFrame(data,columns=['Name','Age'])
print (df)
```

Output-

```
       Name      Age
0      Ram       18
1      Shyam     16
2      Shiv      20
```

Data Manipulation using Pandas

DataFrame manipulation in Pandas involves editing andmodifying existing DataFrames. Some commonDataFrame manipulation operations are:

1. Adding rows/columns
2. Removing rows/columns
3. Renaming rows/columns

1. Add a New Column to a PandasDataFrame:

```
data = {'Name': ['Ram','Shyam','Shiv'],
        'Age': [18, 16, 20]}
df = pd.DataFrame(data)
# declare a new list
address = ['New Delhi', 'Mumbai', 'Kolkata']
# assign the list as a column
df['Address'] = address
print(df)
```

Output-

```
       Name      Age    Address
0      Ram       18     New Delhi
```

```
1      Shyam    16    Mumbai
2      Shiv     20   Kolkata
```

2. **Add a New Row to a Pandas DataFrame:**

   Adding rows to a DataFrame is not quite asstraightforward as adding columns in Pandas. We use the.loc property to add a new row to a Pandas DataFrame.

```
data = {'Name': ['Ram','Shyam','Shiv'],
        'Age': [18, 16, 20],
        'Address':['New Delhi', 'Mumbai', 'Kolkata']}
df = pd.DataFrame(data)
print("Original DataFrame:")
print(df)
print()
# add a new row
df.loc[len(df.index)] = ['Ganesh', 12, 'Hyderabad']
print("Modified DataFrame:")
print(df)
```

Output-

```
Original DataFrame:
      Name     Age     Address
0     Ram      18      New Delhi
1     Shyam    16      Mumbai
2     Shiv     20     Kolkata
Modified DataFrame:

      Name     Age     Address
0     Ram      18      New Delhi
1     Shyam    16      Mumbai
2     Shiv     20     Kolkata
3     Ganesh   12     Hyderabad
```

3. **Remove Rows/Columns from aPandas DataFrame**

   We can use drop() to delete rows and columns from a DataFrame.

Example: Delete Rows

```
import pandas aspd
```

```
data = {'Name':['Ram', 'Shyam', 'Shiv', 'Radha', 'Gauri',
'Girija'],
        'Age': [25, 20, 30, 18, 26, 25],
        'City': ['Delhi', 'Mumbai', 'Kailash', 'Ujjain',
'Chennai',
'Shimla']}
df=pd.DataFrame(data)
print("Original DataFrame:")
print(df)
print()
# delete row with index 4
df.drop(4, axis=0, inplace=True)
# delete row with index 5
df.drop(index=5, inplace=True)
# delete rows with index 1 and 3
df.drop([1, 3], axis=0, inplace=True)
# display the modified DataFrame after deleting rows
print("Modified DataFrame:")
print(df)
```

Output-

```
Original DataFrame:
     Name  Age     City
0     Ram   25    Delhi
1   Shyam   20   Mumbai
2    Shiv   30  Kailash
3   Radha   18   Ujjain
4   Gauri   26  Chennai
5  Girija   25   Shimla

Modified DataFrame:
   Name  Age     City
0   Ram   25    Delhi
2  Shiv   30  Kailash
```

Here,

axis=0 : indicates that rows are to be deleted

inplace=True : indicates that the changes are to bemade in the original DataFrame

Example: Delete columns

```
import pandas as pd
# create a sample DataFrame
data = {'Name': ['Ravi', 'Anil', 'Mukseh', 'Rahul'],
        'Age': [25, 30, 35, 40],
        'City': ['New Delhi', 'Bhopal', 'Varanasi', 'Pune'],
        'Height': ['165', '178', '185', '171'],
        'Profession': ['Engineer', 'Entrepreneur', 'Unemployed', 
'Actor'],
        'Marital       Status':       ['Single',       'Married', 
'Divorced','Engaged']}
df = pd.DataFrame(data)

# display the original DataFrame
print("Original DataFrame:")
print(df)
print()


# delete age column
df.drop('Age', axis=1, inplace=True)
# delete marital status column
df.drop(columns='Marital Status', inplace=True)
# delete height and profession columns
df.drop(['Height', 'Profession'], axis=1, inplace=True)
# display the modified DataFrame after deleting rows
print("Modified DataFrame:")
print(df)
```

Output-

```
Original DataFrame:
      Name  Age      City  Height    Profession  Marital Status
0     Ravi   25  New Delhi   165      Engineer         Single
1     Anil   30     Bhopal   178  Entrepreneur        Married
2   Mukseh   35   Varanasi   185    Unemployed       Divorced
3    Rahul   40       Pune   171         Actor        Engaged

Modified DataFrame:
      Name     City
0     Ravi    New Delhi
1     Anil    Bhopal
2   Mukseh    Varanasi
3    Rahul    Pune
```

Here,

axis=1: indicates that columns are to be deleted

inplace=True: indicates that the changes are to be made in the original DataFrame

4. Rename Labels in a DataFrame

We can rename columns in a Pandas DataFrame using the rename() function.

Example: Rename Columns

```
data = {'Name': ['Alice', 'Bob', 'Charlie', 'David'],
        'Age': [25, 30, 35, 40],
        'City': ['New York', 'London', 'Paris', 'Tokyo']}
df = pd.DataFrame(data)
# display the original DataFrame
print("Original DataFrame:")
print(df)
print()
# rename column 'Name' to 'First_Name'
df.rename(columns= {'Name': 'First_Name'}, inplace=True)
# rename columns 'Age' and 'City'
df.rename(mapper= {'Age': 'Number', 'City':'Address'}, axis=1,
inplace=True)
# display the DataFrame after renaming column
print("Modified DataFrame:")
print(df)
```

Output-

```
Original DataFrame:
      Name    Age     City
0     Alice   25     New York
1       Bob   30     London
2   Charlie   35     Paris
3     David   40     Tokyo

Modified DataFrame:
   First_Name  Number   Address
0      Alice      25     New York
1        Bob      30     London
```

```
2    Charlie     35     Paris
3      David     40     Tokyo
```

Here,

axis=1: indicates that columns are to be renamed

inplace=True: indicates that the changes are to be made in the original DataFrame

Example: Rename Row Labels

```
import pandas as pd
# create a sample DataFrame
data = {'Name': ['Alice', 'Bob', 'Charlie', 'David'],
        'Age': [25, 30, 35, 40],
        'City': ['New York', 'London', 'Paris', 'Tokyo']}
df = pd.DataFrame(data)

# display the original DataFrame
print("Original DataFrame:")
print(df)
print()

# rename column one index label
df.rename(index={0: 7}, inplace=True)
# rename columns multiple index labels
df.rename(mapper={1: 10, 2: 100}, axis=0, inplace=True)
# display the DataFrame after renaming column
print("Modified DataFrame:")
print(df)
```

Output-

```
Original DataFrame:
      Name  Age    City
0    Alice   25   New York
1      Bob   30    London
2   Charlie  35      Paris
3    David   40    Tokyo

Modified DataFrame:
        Name   Age   City
```

```
7      Alice   25    New York
10       Bob   30    London
100  Charlie   35    Paris
3      David   40    Tokyo
```

Here,

axis=0: indicates that rows are to be renamed

inplace=True: indicates that the changes are to be made in the original DataFrame.

Conclusion:

____

____

____

____

Questions:

1. What is pandas in Python?
2. Differentiate between Series and DataFrame?
3. Show two different ways to create a pandas DataFrame.
4. How do you add multiple rows to an existing DataFrame?
5. How do we select specific columns from a DataFrame?