**Aim:** Practical based on Pandas Data Structures

**IDE:**

What is Python Pandas?

Pandas is a powerful, open-source data analysis and manipulation package for Python. It provides data structures and functions needed to work on structured data seamlessly and efficiently.

What Is Pandas Used For?

Pandas is extensively used for:

* Data Cleaning: Handling missing values, duplications, and incorrect data formats.
* Data Manipulation: Filtering, transforming, and merging datasets.
* Data Analysis: Performing statistical analysis and aggregations.
* Data Visualization: Creating plots and charts to visualize data trends and patterns.
* Time Series Analysis: Handling and manipulating time series data.

Run the following command to install Pandas:

pip install pandas

import pandas as pd

print(pd.\_\_version\_\_)

Pandas Series

A Pandas Series is a one-dimensional labeled array capable of holding any data type. It is similar to a column in a spreadsheet or a SQL table.

Example:

import pandas as pd

# Creating a Series

data = [1, 2, 3, 4, 5]

series = pd.Series(data)

print(series)

Output:

A screenshot of a computer program

AI-generated content may be incorrect.

Basic Operations on Series

Perform various operations on Series, such as arithmetic operations, filtering, and statistical calculations.

Example:

# Arithmetic Operations

series2 = series + 10

print(series2)

# Filtering

filtered\_series = series[series > 2]

print(filtered\_series)

# Statistical Calculations

mean\_value = series.mean()

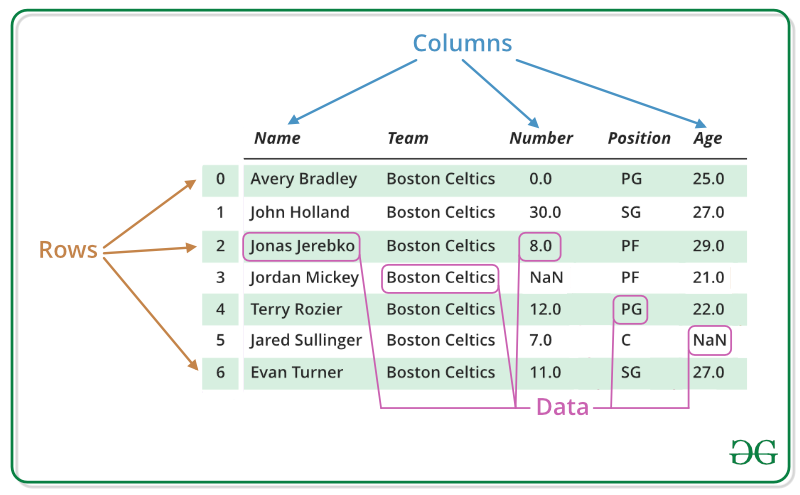
print(mean\_value)

Output

A screenshot of a computer program

AI-generated content may be incorrect.

Pandas Dataframe

Pandas DataFrame is two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns. Pandas DataFrame consists of three principal components, the data, rows, and columns.

# Creating a DataFrame

data = {

    'Name': ['Alice', 'Bob', 'Charlie'],

    'Age': [25, 30, 35],

    'City': ['New York', 'Los Angeles', 'Chicago']

}

df = pd.DataFrame(data)

print(df)

Output

A computer screen shot of a program

AI-generated content may be incorrect.

**Basic Operations on Dataframes**

DataFrames support a wide range of operations for data manipulation and analysis.

# Accessing Columns (# select one column)

print(df[['Name']])

Output

A screenshot of a computer

AI-generated content may be incorrect.

# Adding a New Column

df['Salary'] = [70000, 80000, 90000]

print(df)

Output

A screenshot of a computer program

AI-generated content may be incorrect.

# Dropping a Column

df = df.drop('City', axis=1)

print(df)

Output

A screenshot of a computer program

AI-generated content may be incorrect.

The DataFrame is like a table with rows and columns.

Pandas use the loc attribute to return one or more specified row(s)

# Return row 0:

print(df.loc[[0]])

Output

A screenshot of a computer program

AI-generated content may be incorrect.

#Return row 0 and 1:

#use a list of indexes:

print(df.loc[[0, 1]])

Output

A screenshot of a computer program

AI-generated content may be incorrect.

**Named Indexes**

With the index argument, you can name your own indexes.

Example:

Add a list of names to give each row a name:

import pandas as pd

data = {

"calories": [420, 380, 390],

"duration": [50, 40, 45]

}

df = pd.DataFrame(data, index = ["day1", "day2", "day3"])

print(df)

Output

A screen shot of a computer

AI-generated content may be incorrect.

Explanation of Key Pandas Functions

Reading and Writing Data:

Reading Data: Read a CSV file into a DataFrame.

Example:

dat = pd.read\_csv("data.csv")

print(dat)

Output

A screenshot of a computer

AI-generated content may be incorrect.

Writing Data: Write a DataFrame to a CSV file.

Note: Other Ways to Save Pandas DataFrames (to\_excel(), to\_json(), to\_hdf(), to\_sql(), to\_pickle())

Example:

Biodata = {'Name': ['John', 'Emily', 'Mike', 'Lisa'],

'Age': [28, 23, 35, 31],

'Gender': ['M', 'F', 'M', 'F']

}

df = pd.DataFrame(Biodata)

# Save the dataframe to a CSV file

df.to\_csv('Biodata.csv', index=False)

Output

A screen shot of a computer

AI-generated content may be incorrect.

**Data Inspection:**

df.head(): Display the first few rows of the DataFrame.

df.tail(): Display the last few rows of the DataFrame.

df.info(): Display a summary of the DataFrame.

df.describe(): Provide descriptive statistics for numerical columns. (count: the number of non-null entries, mean: the mean value, std: the standard deviation, min: the minimum value, 25%, 50%, 75%: the lower, median, and upper quartiles, max: the maximum value)

Example:

dat = pd.read\_csv("data.csv")

print(dat.info())

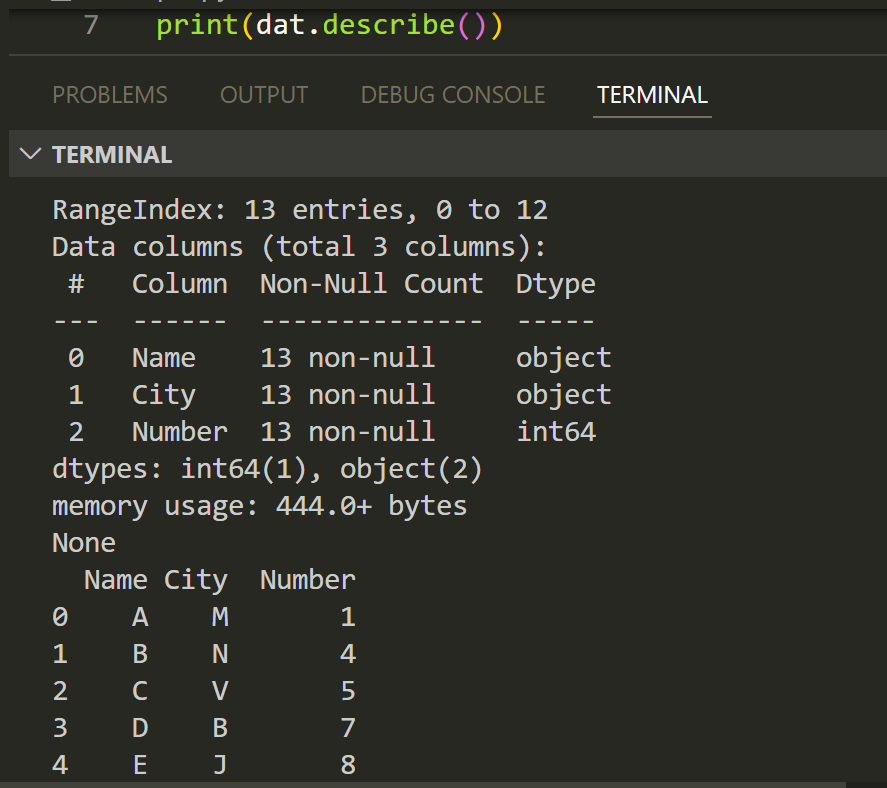
# shows first and last five rows

print(dat.head())

print(dat.tail())

print(dat.describe())

Output

 A screenshot of a computer

AI-generated content may be incorrect.

**Data Selection and Indexing:**

dat[['A']]: Select a column.

dat[['A', 'B']]: Select multiple columns.

dat.loc[[0]]: Select a row by label.

Example:

print(dat[['Name']])

print(dat[['Name','Number']])

print(dat.loc[[1]])

Output

A screenshot of a computer

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect.

**Data Manipulation:**

dat['A'] = dat['A'] \* 2: Modify a column.

dat['F'] = dat['A'] + dat['B']: Create a new column based on existing columns.

dat.drop(columns=['A']): Drop a column.

dat.drop(index=[0]): Drop a row.

Task

Create a DataFrame with 5 numeric columns

data = {

'A': [np.nan, 2, 3, 4, 5, 6, 7, 8, 9, 10],

'B': np.random.normal(50, 15, 10),

'C': np.random.rand(10) \* 100,

'D': np.linspace(1, 10, 10),

'E': np.logspace(1, 2, 10)

}

df = pd.DataFrame(data)

Output

A screen shot of a computer program

AI-generated content may be incorrect.

**Post Lab Exercise:**

1. Write a Pandas program to add, subtract, multiple and divide two Pandas Series.

import pandas as pd

s1 = pd.Series([10, 20, 30, 40])

s2 = pd.Series([1, 2, 3, 4])

print("Addition:\n", s1 + s2)

print("Subtraction:\n", s1 - s2)

print("Multiplication:\n", s1 \* s2)

print("Division:\n", s1 / s2)

1. Write a Pandas program to convert a dictionary to a Pandas series.

import pandas as pd

data = {'a': 100, 'b': 200, 'c': 300}

series = pd.Series(data)

print(series)

1. Write a Pandas program to create a series from a list, numpy array and dict

import pandas as pd

import numpy as np

series\_from\_list = pd.Series([10, 20, 30])

series\_from\_array = pd.Series(np.array([1, 2, 3, 4]))

series\_from\_dict = pd.Series({'x': 5, 'y': 10, 'z': 15})

print("Series from List:\n", series\_from\_list)

print("Series from Array:\n", series\_from\_array)

print("Series from Dict:\n", series\_from\_dict)

1. Write a Pandas program to stack two series vertically and horizontally

import pandas as pd

series\_a = pd.Series([1, 2, 3])

series\_b = pd.Series([4, 5, 6])

vertical\_stack = pd.concat([series\_a, series\_b], *axis*=0)

print("Vertical Stack:\n", vertical\_stack)

horizontal\_stack = pd.concat([series\_a, series\_b], *axis*=1)

print("Horizontal Stack:\n", horizontal\_stack)