	Marwadi University **Faculty of Engineering & Technology** **Department of Information and Communication Technology**	
Subject: Programming With Python (01CT1309)	**Aim:** Practical based on Pandas Data Structures	
Experiment No: 09	**Date:**	**Enrollment No: 92510133028**

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__)
```

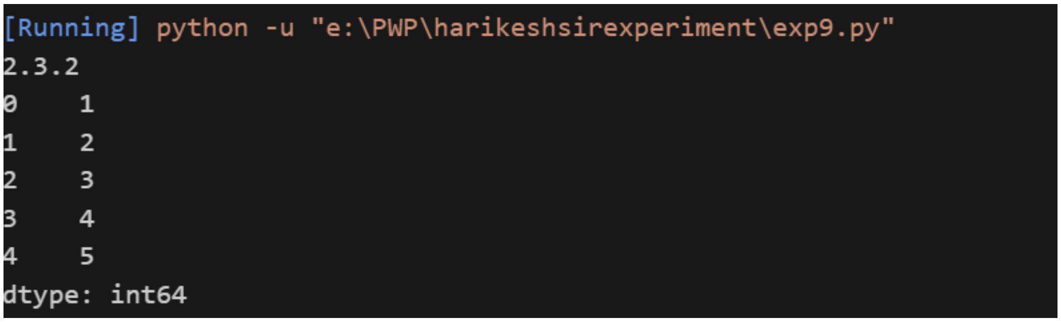
Pandas Series

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:



```
[Running] python -u "e:\PWP\harikeshsirexperiment\exp9.py"
2.3.2
0    1
1    2
2    3
3    4
4    5
dtype: int64
```

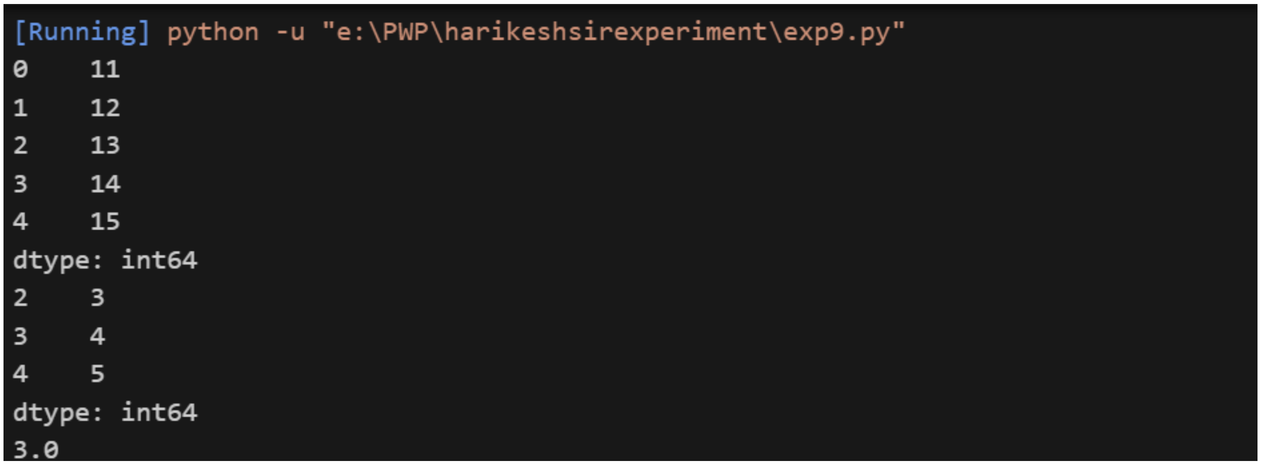
Basic Operations on Series

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)
```

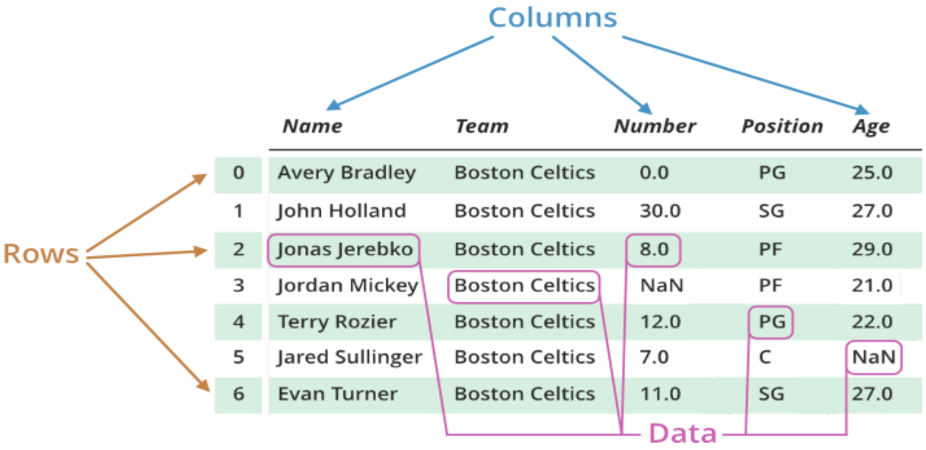
Output



```
[Running] python -u "e:\PWP\harikeshsirexperiment\exp9.py"
0    11
1    12
2    13
3    14
4    15
dtype: int64
2    3
3    4
4    5
dtype: int64
3.0
```

Pandas Dataframe

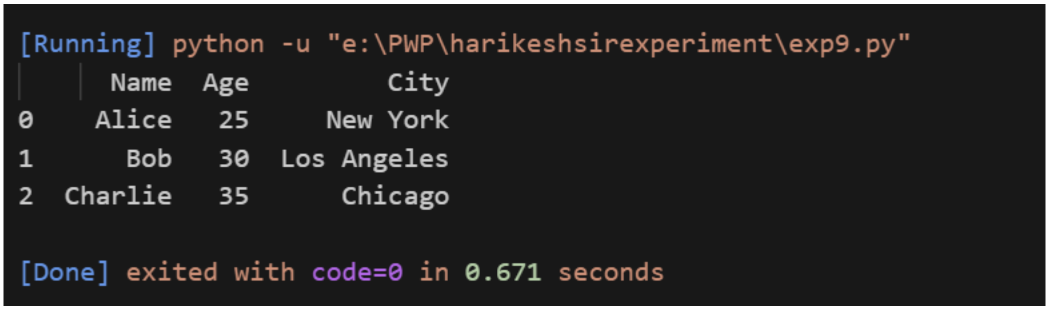
Pandas DataFrame is two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns. Pandas DataFrame consists of three principal components,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



	Name	Team	Number	Position	Age
0	Avery Bradley	Boston Celtics	0.0	PG	25.0
1	John Holland	Boston Celtics	30.0	SG	27.0
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN
6	Evan Turner	Boston Celtics	11.0	SG	27.0

```
# 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

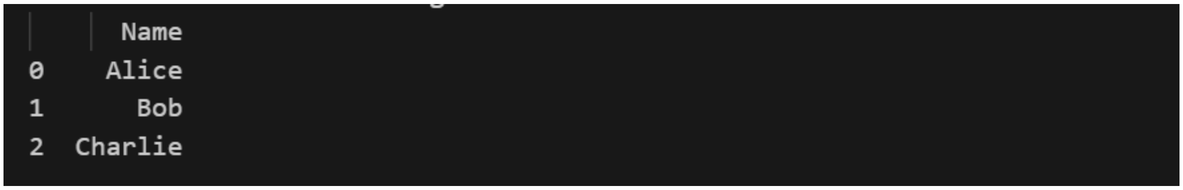


```
[Running] python -u "e:\PWP\harikeshsirexperiment\exp9.py"
      Name  Age         City
0    Alice   25     New York
1      Bob   30  Los Angeles
2  Charlie   35      Chicago

[Done] exited with code=0 in 0.671 seconds
```

Basic Operations on Dataframes

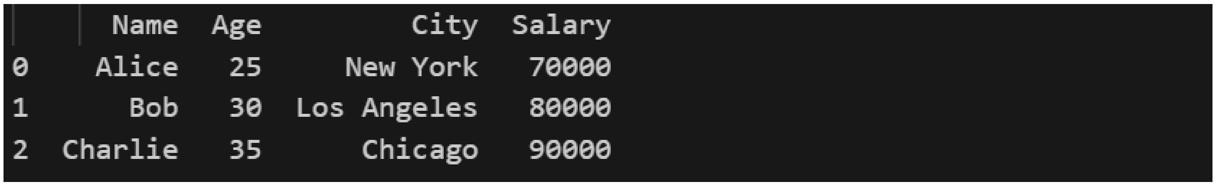
DataFrames support a wide range of operations for data manipulation and analysis.

# Accessing Columns (# select one column)
print(df[['Name']])
Output



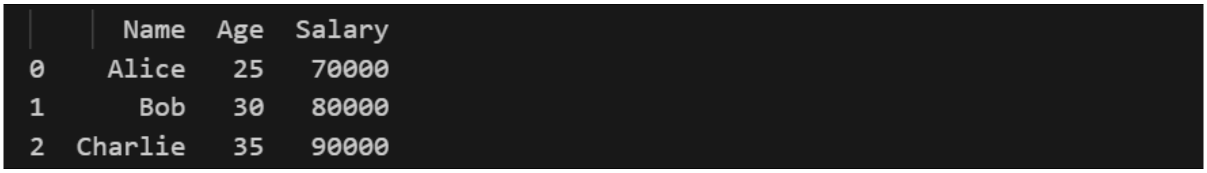
```
      Name
0    Alice
1      Bob
2  Charlie
```

# Adding a New Column
df['Salary'] = [70000, 80000, 90000]
print(df)
Output



```
      Name  Age         City  Salary
0    Alice   25     New York   70000
1      Bob   30  Los Angeles   80000
2  Charlie   35      Chicago   90000
```

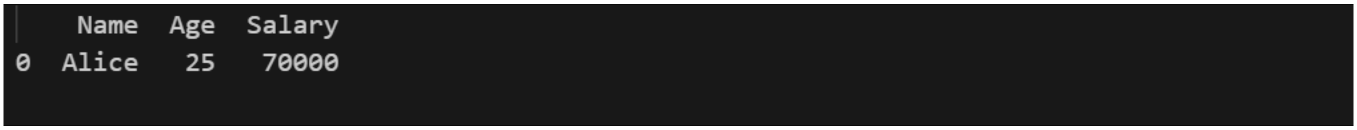
# Dropping a Column
df = df.drop('City', axis=1)
print(df)
Output



```
      Name  Age  Salary
0    Alice   25   70000
1      Bob   30   80000
2  Charlie   35   90000
```

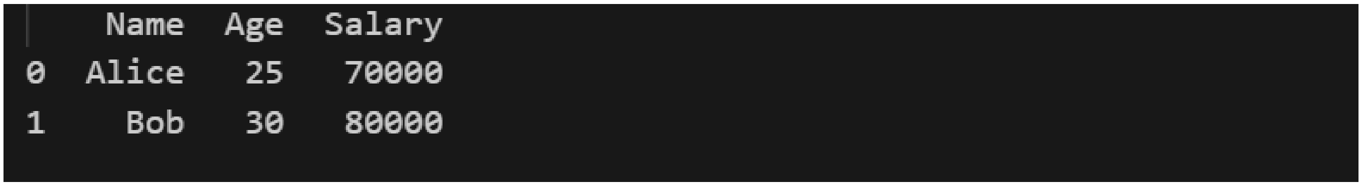
The DataFrame is like a table with rows and columns.
Pandas use the loc attribute to return one or more specified row(s)
# Return row 0:
print(df.loc[[0]])

Output



```
    Name  Age  Salary
0  Alice   25   70000
```

#Return row 0 and 1:
#use a list of indexes:
print(df.loc[[0, 1]])
Output



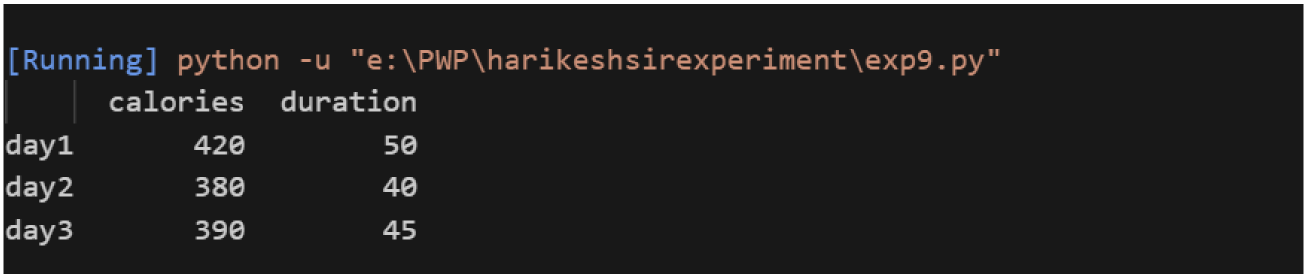
```
    Name  Age  Salary
0  Alice   25   70000
1    Bob   30   80000
```

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)
```

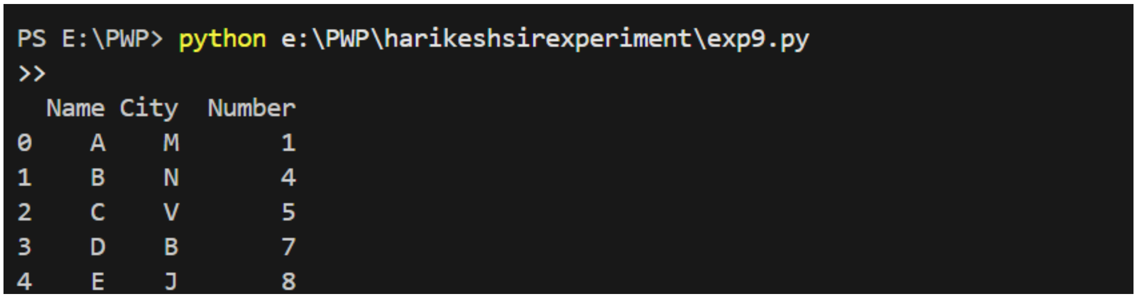
Output



```
[Running] python -u "e:\PWP\harikeshsirexperiment\exp9.py"
      calories  duration
day1       420        50
day2       380        40
day3       390        45
```

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



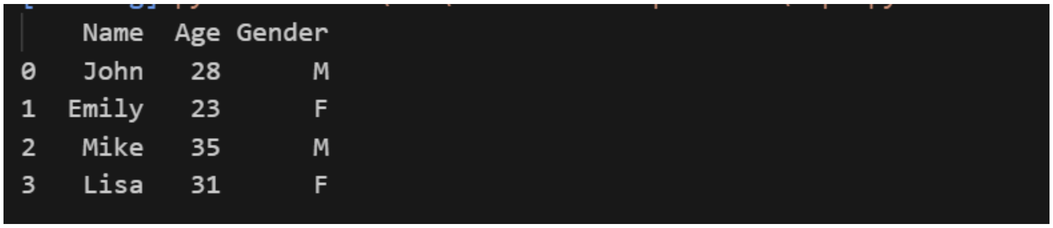
```
PS E:\PWP> python e:\PWP\harikeshsirexperiment\exp9.py
>>
  Name City  Number
0    A    M       1
1    B    N       4
2    C    V       5
3    D    B       7
4    E    J       8
```

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



```
    Name  Age Gender
0   John   28      M
1  Emily   23      F
2   Mike   35      M
3   Lisa   31      F
```

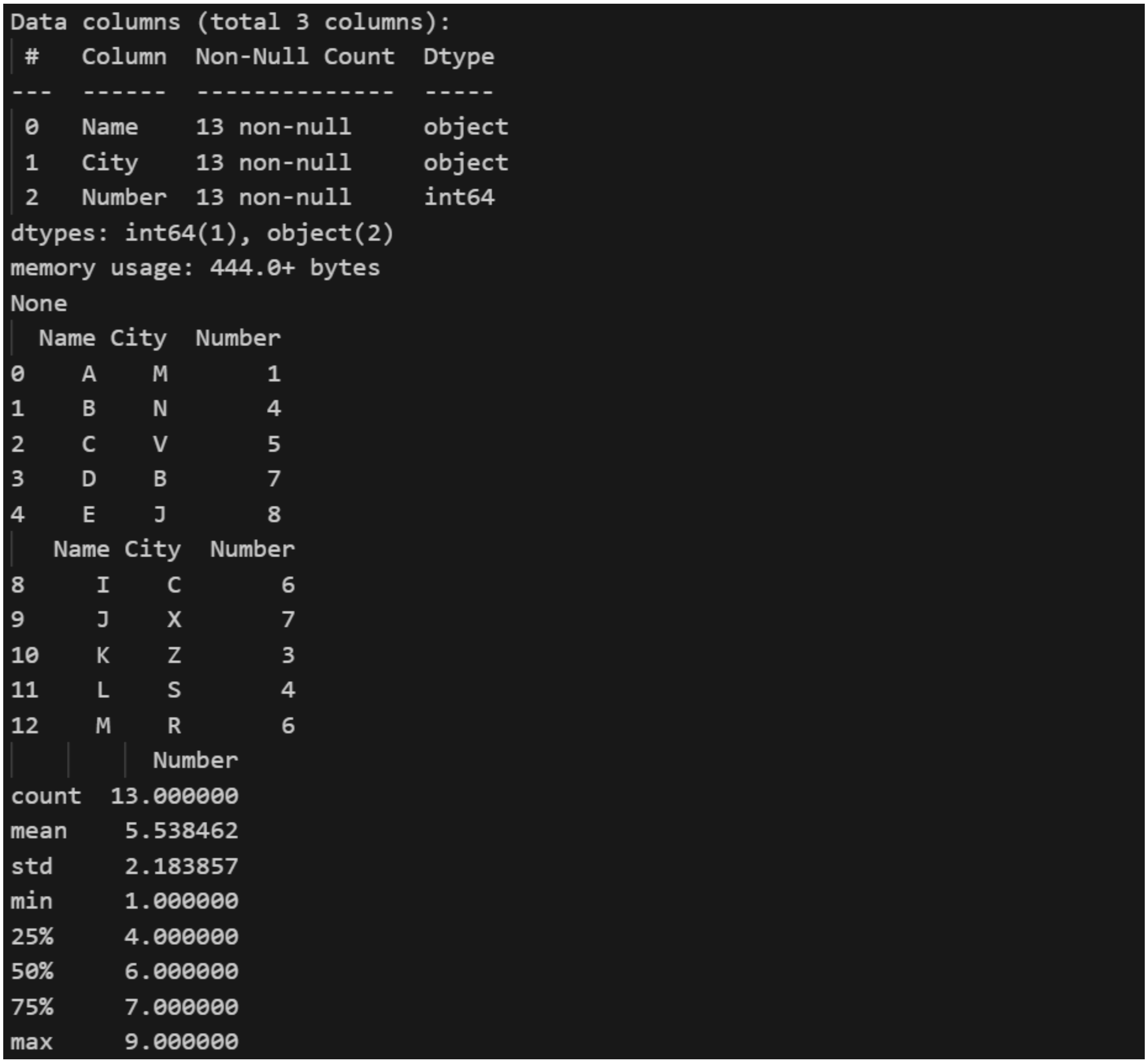
Data Inspection:

`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



```
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Name    13 non-null     object
 1   City    13 non-null     object
 2   Number  13 non-null     int64
dtypes: int64(1), object(2)
memory usage: 444.0+ bytes
None
  Name City  Number
0    A    M       1
1    B    N       4
2    C    V       5
3    D    B       7
4    E    J       8
   Name City  Number
8     I    C       6
9     J    X       7
10    K    Z       3
11    L    S       4
12    M    R       6
          Number
count  13.000000
mean    5.538462
std     2.183857
min     1.000000
25%     4.000000
50%     6.000000
75%     7.000000
max     9.000000
```

Data Selection and Indexing:

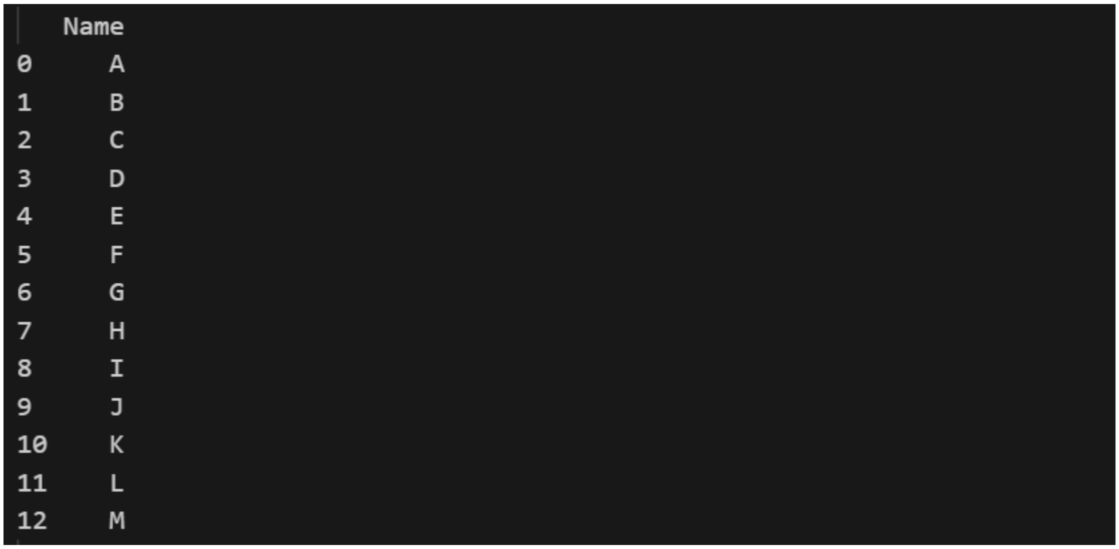
`dat[['A']]`: Select a column.

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

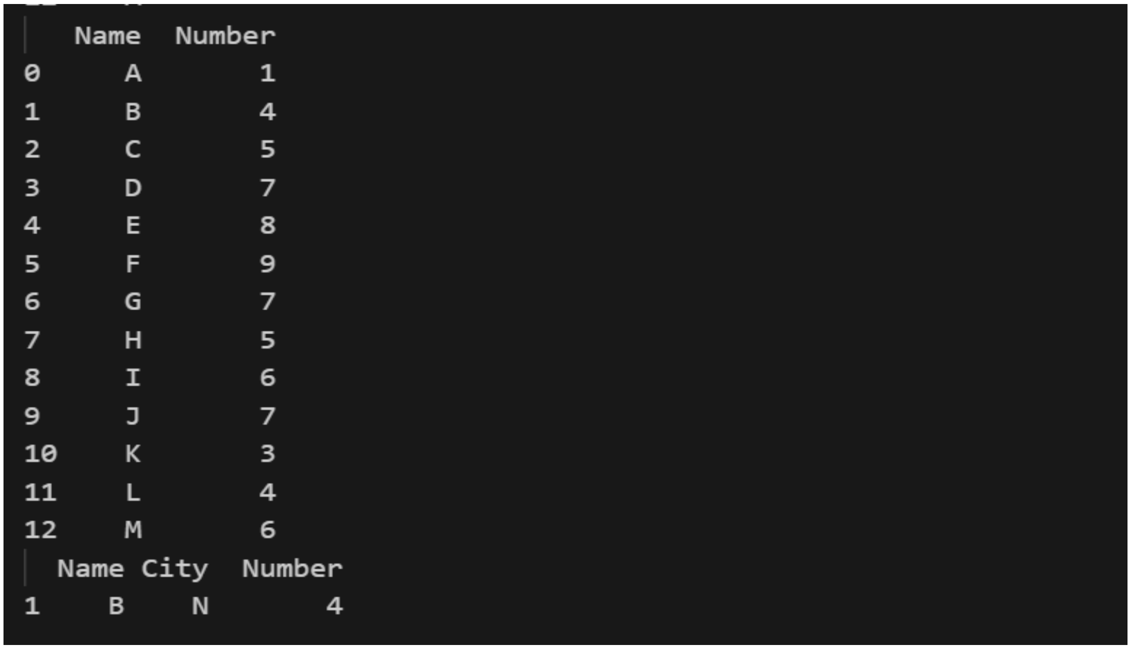
`dat.loc[[0]]`: Select a row by label.

Example:

print(dat[['Name']])
print(dat[['Name','Number']])
print(dat.loc[[1]])
Output



```
   Name
0     A
1     B
2     C
3     D
4     E
5     F
6     G
7     H
8     I
9     J
10    K
11    L
12    M
```



```
    Name  Number
0      A       1
1      B       4
2      C       5
3      D       7
4      E       8
5      F       9
6      G       7
7      H       5
8      I       6
9      J       7
10     K       3
11     L       4
12     M       6
  Name City  Number
1    B    N       4
```

Data Manipulation:

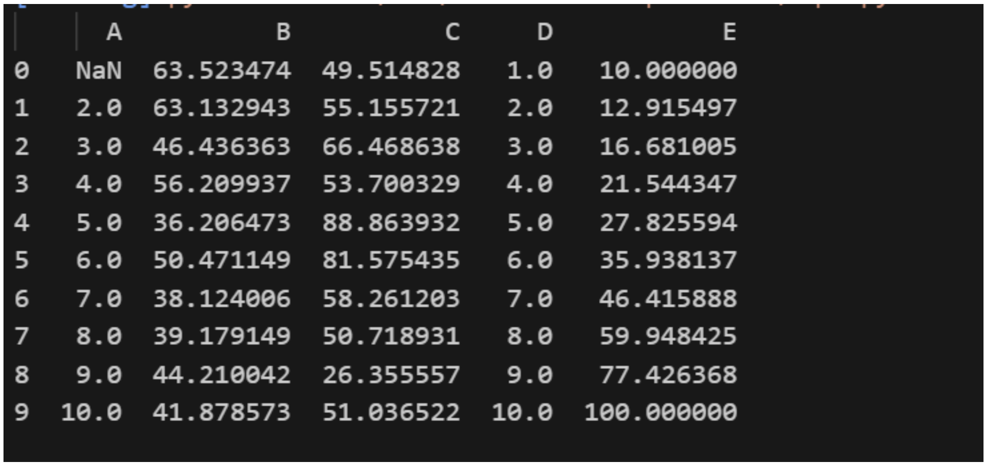
dat['A'] = dat['A'] * 2: Modify a column.
dat['F'] = dat['A'] + dat['B']: Create a new column based on existing columns.
dat.drop(columns=['A']): Drop a column.
dat.drop(index=[0]): Drop a row.

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

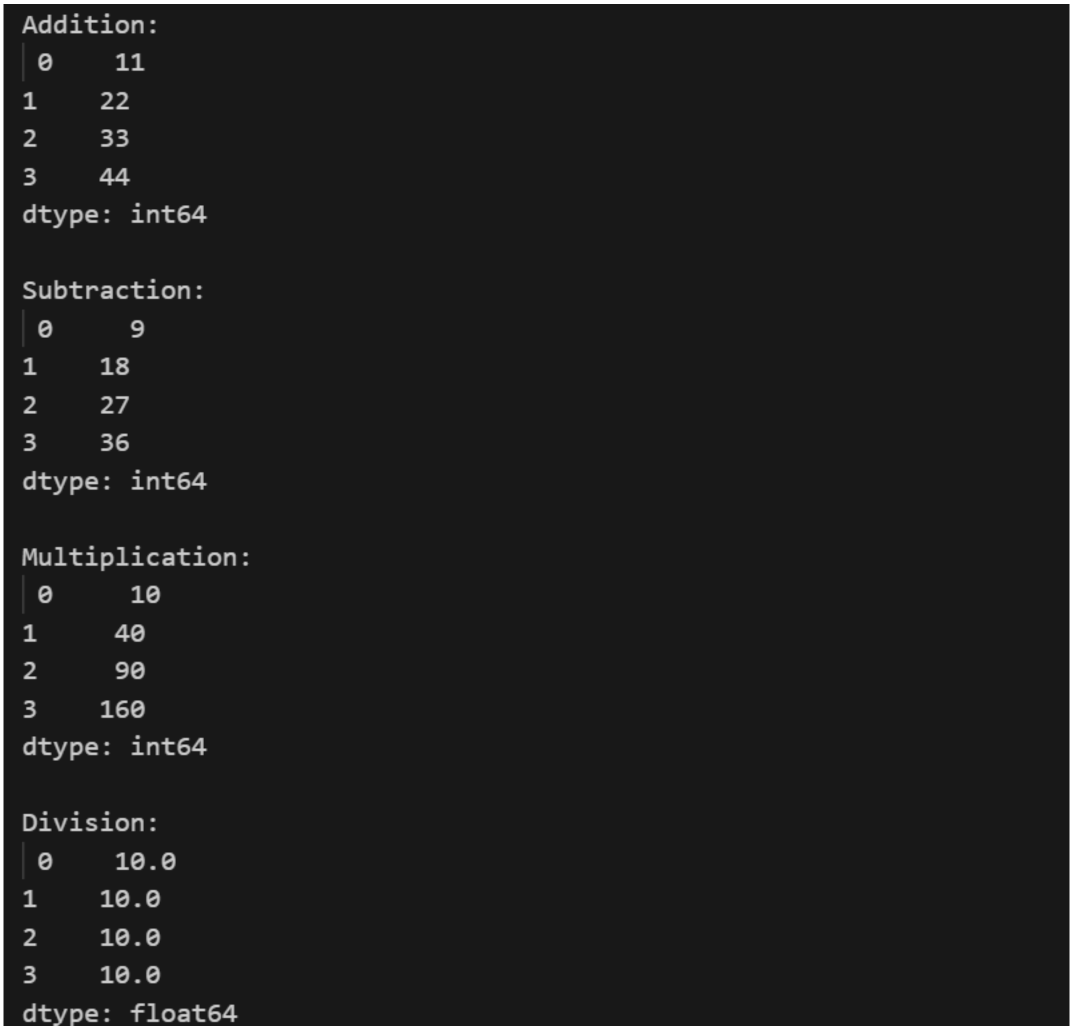


```
      A          B          C     D           E
0   NaN  63.523474  49.514828   1.0   10.000000
1   2.0  63.132943  55.155721   2.0   12.915497
2   3.0  46.436363  66.468638   3.0   16.681005
3   4.0  56.209937  53.700329   4.0   21.544347
4   5.0  36.206473  88.863932   5.0   27.825594
5   6.0  50.471149  81.575435   6.0   35.938137
6   7.0  38.124006  58.261203   7.0   46.415888
7   8.0  39.179149  50.718931   8.0   59.948425
8   9.0  44.210042  26.355557   9.0   77.426368
9  10.0  41.878573  51.036522  10.0  100.000000
```

Post Lab Exercise:

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

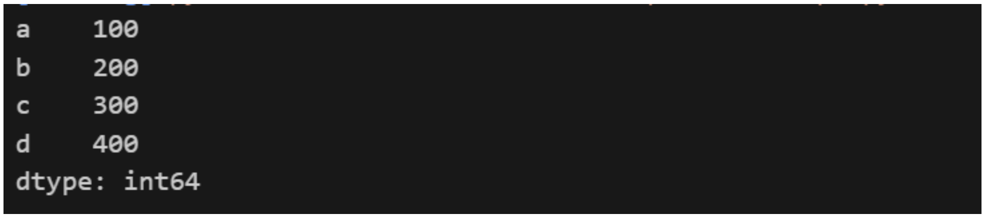
```
import pandas as pd

s1 = pd.Series([10, 20, 30, 40])
s2 = pd.Series([1, 2, 3, 4])

print("Addition:\n", s1 + s2)
print("\nSubtraction:\n", s1 - s2)
print("\nMultiplication:\n", s1 * s2)
print("\nDivision:\n", s1 / s2)
```



```
Addition:
 0    11
1    22
2    33
3    44
dtype: int64

Subtraction:
 0     9
1    18
2    27
3    36
dtype: int64

Multiplication:
 0     10
1     40
2     90
3    160
dtype: int64

Division:
 0    10.0
1    10.0
2    10.0
3    10.0
dtype: float64
```

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

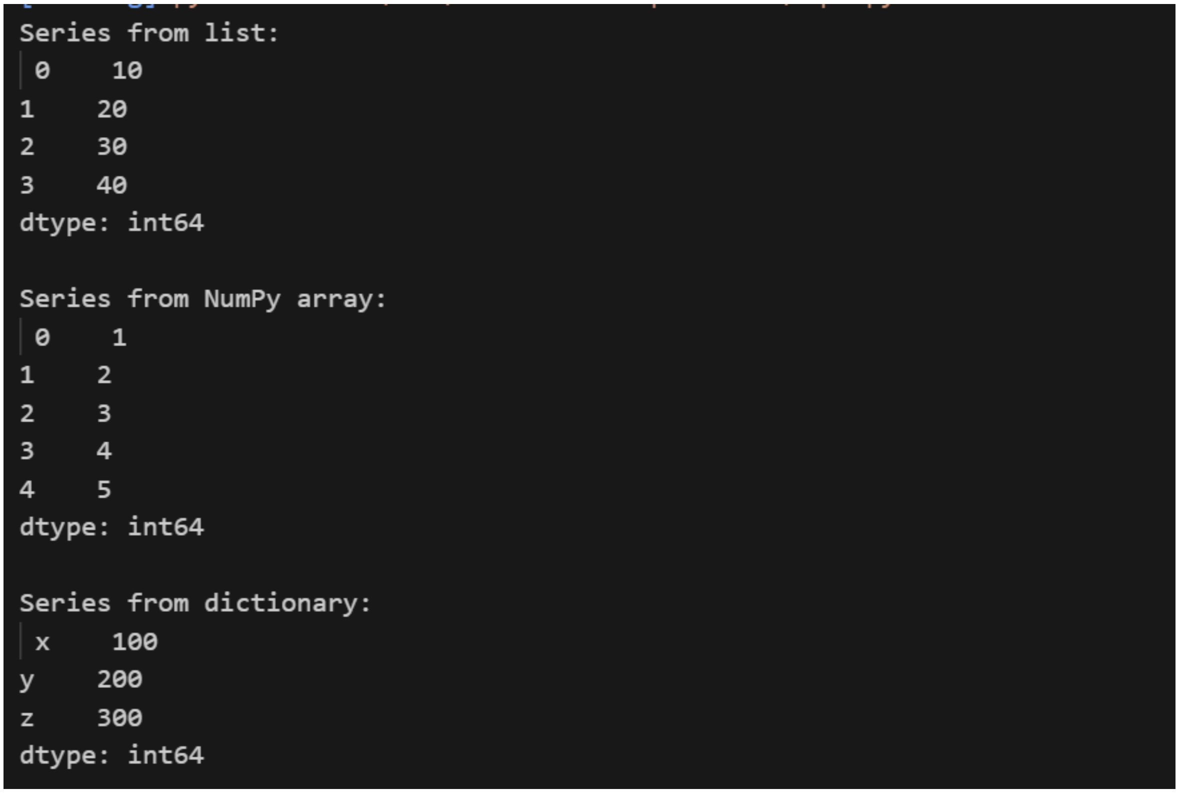
```
data = {'a': 100, 'b': 200, 'c': 300, 'd': 400}
s = pd.Series(data)
print(s)
```



```
a    100
b    200
c    300
d    400
dtype: int64
```

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

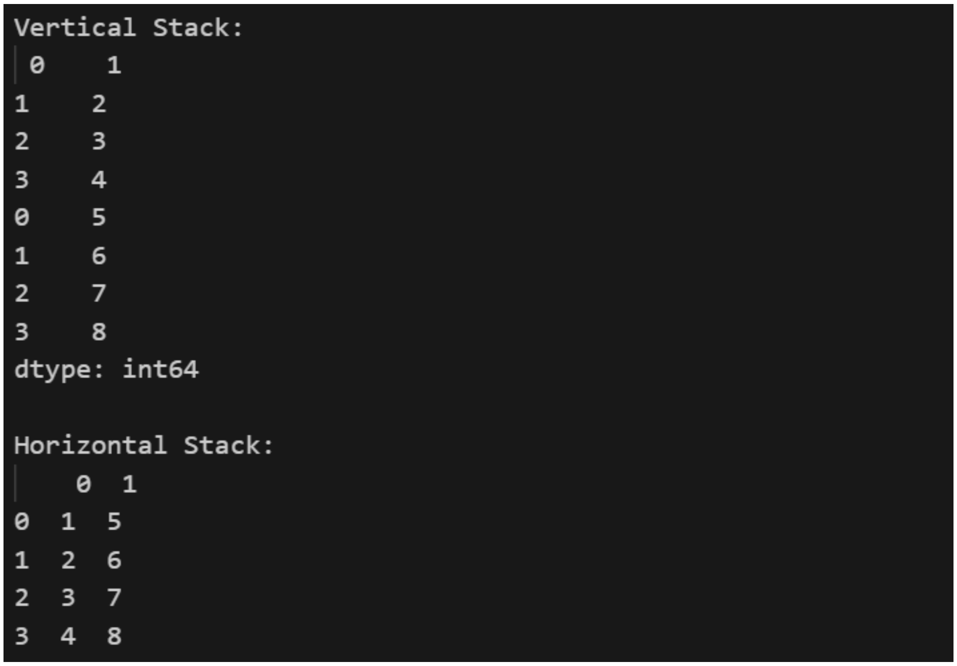
```
import numpy as np

# From list
s1 = pd.Series([10, 20, 30, 40])
print("Series from list:\n", s1)

# From NumPy array
arr = np.array([1, 2, 3, 4, 5])
s2 = pd.Series(arr)
print("\nSeries from NumPy array:\n", s2)

# From dict
d = {'x': 100, 'y': 200, 'z': 300}
s3 = pd.Series(d)
print("\nSeries from dictionary:\n", s3)
```



```
Series from list:
 0    10
1    20
2    30
3    40
dtype: int64

Series from NumPy array:
 0    1
1    2
2    3
3    4
4    5
dtype: int64

Series from dictionary:
 x    100
y    200
z    300
dtype: int64
```

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

```
s1 = pd.Series([1, 2, 3, 4])
s2 = pd.Series([5, 6, 7, 8])

# Vertical stacking → concatenation
vertical = pd.concat([s1, s2])
print("Vertical Stack:\n", vertical)

# Horizontal stacking → DataFrame
horizontal = pd.concat([s1, s2], axis=1)
print("\nHorizontal Stack:\n", horizontal)
```



```
Vertical Stack:
 0    1
1    2
2    3
3    4
0    5
1    6
2    7
3    8
dtype: int64

Horizontal Stack:
    0  1
0  1  5
1  2  6
2  3  7
3  4  8
```