DATE : 21/03/2023

ASSIGNMENT NO – 11

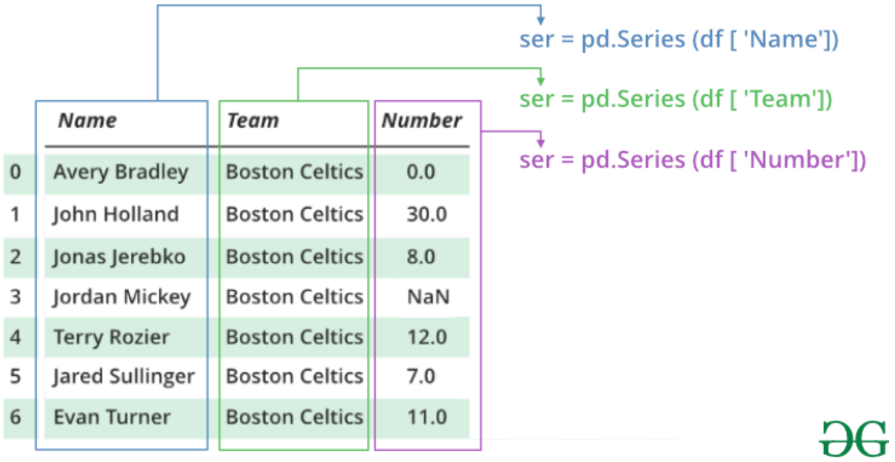
PROBLEM STATEMENT : Program on Pandas in Python

THEORY :

Pandas is a popular open-source library in Python for data manipulation and analysis. It provides data structures and functions for efficiently handling structured data, such as tables or spreadsheets, in a convenient and flexible manner.

Some of the key features of Pandas include:

1. **Data structures** : Pandas has two main data structures - Series and DataFrame. A Series is a one-dimensional array-like object that can hold any data type, while a DataFrame is a two-dimensional table-like data structure that consists of rows and columns, similar to a spreadsheet.
   A. **Series** :
   Pandas Series is a one-dimensional labeled array capable of holding data of any type (integer, string, float, python objects, etc.). The axis labels are collectively called indexes. Pandas Series is nothing but a column in an excel sheet. Labels need not be unique but must be a hashable type. The object supports both integer and label-based indexing and provides a host of methods for performing operations involving the index.



ser = pd.Series (df ['Name'])

ser = pd.Series (df ['Team'])

ser = pd.Series (df ['Number'])

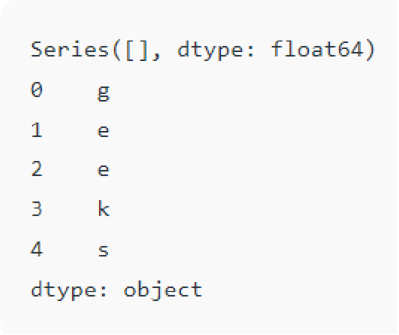
	Name	*Team*	*Number*
0	Avery Bradley	Boston Celtics	0.0
1	John Holland	Boston Celtics	30.0
2	Jonas Jerebko	Boston Celtics	8.0
3	Jordan Mickey	Boston Celtics	NaN
4	Terry Rozier	Boston Celtics	12.0
5	Jared Sullinger	Boston Celtics	7.0
6	Evan Turner	Boston Celtics	11.0

A Pandas Series will be created by loading the datasets from existing storage, storage can be SQL Database, CSV file, an Excel file. Pandas Series can be created from the lists, dictionary, and from a scalar value etc.

Example:

```
import pandas as pd
import numpy as np
ser = pd.Series()
print(ser)
data = np.array(['g', 'e', 'e', 'k', 's'])
ser = pd.Series(data)
print(ser)
```

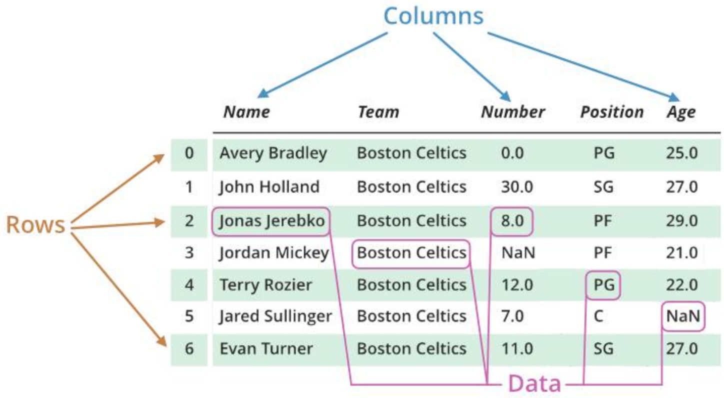
OUTPUT :



```
Series([], dtype: float64)
0    g
1    e
2    e
3    k
4    s
dtype: object
```

B. **DATAFRAME** :

Pandas DataFrame is a 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.



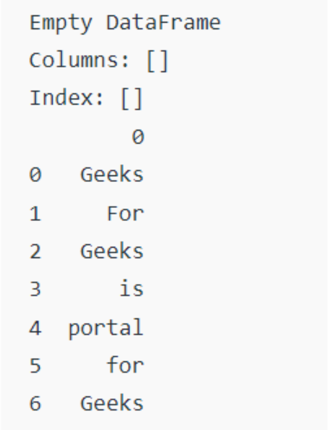
	Name	*Team*	*Number*	*Position*	*Age*
0	Avery Bradley	Boston Celtics	0.0	PG	25.0
1	John Holland	Boston Celtics	30.0	SG	27.0
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN
6	Evan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

A Pandas DataFrame will be created by loading the datasets from existing storage, storage can be SQL Database, CSV file, an Excel file. Pandas DataFrame can be created from the lists, dictionary, and from a list of dictionaries, etc.

<u>EXAMPLE :</u>

```
import pandas as pd
df = pd.DataFrame()
print(df)
lst = ['Geeks', 'For', 'Geeks', 'is', 'portal', 'for', 'Geeks']
df = pd.DataFrame(lst)
print(d
```

OUTPUT :



```
Empty DataFrame
Columns: []
Index: []
        0
0   Geeks
1     For
2   Geeks
3      is
4  portal
5     for
6   Geeks
```

2. **Data manipulation** : Pandas provides various functions for manipulating and transforming data, such as filtering, sorting, grouping, merging, reshaping, and aggregating. These operations can be performed on entire datasets or subsets of the data.
3. **Missing data handling** : Pandas has built-in functions for handling missing data, such as filling missing values with a default value or interpolating missing values based on neighboring values.
4. **Time-series data analysis** : Pandas provides support for time-series data analysis, including date and time manipulation, time-zone handling, and frequency conversion.
5. **Input/output** : Pandas supports reading and writing data in various formats, such as CSV, Excel, SQL databases, and JSON.
6. **Visualization** : Pandas has built-in support for data visualization, allowing users to create various types of plots and charts using the Matplotlib library.

To use Pandas in Python, you can install it using the pip package manager by running pip install pandas in your terminal or command prompt. Once installed, you can import it in your Python code using import pandas as pd.

Advantages of Pandas in Python :

- Fast and efficient for manipulating and analyzing data.
- Data from different file objects can be loaded.
- Easy handling of missing data (represented as NaN) in floating point as well as non-floating point data
- Size mutability: columns can be inserted and deleted from DataFrame and higher dimensional objects
- Data set merging and joining.
- Flexible reshaping and pivoting of data sets
- Provides time-series functionality.
- Powerful group by functionality for performing split-apply-combine operations on data sets

CODE:

```
 import pandas as pd
import numpy as np
series1 = pd.Series([1, 2, 3, 4, 5])
series2 = pd.Series([10, 20, 30, 40, 50])
add_result = series1 + series2
print("Addition Result:\n", add_result)

sub_result = series1 - series2
print("Subtraction Result:\n", sub_result)

mult_result = series1 * series2
print("Multiplication Result:\n", mult_result)

div_result = series1 / series2
print("Division Result:\n", div_result)

series1 = pd.Series([1, 2, 3, 4, 5])
series2 = pd.Series([1, 3, 2, 5, 4])
comp_result = series1 == series2
print("Comparison Result:\n", comp_result)

my_dict = {'a': 100, 'b': 200, 'c': 300, 'd': 400, 'e': 500}
dict_series = pd.Series(my_dict)
print("Dictionary to Series Conversion:\n", dict_series)

my_array = np.array([10, 20, 30, 40, 50])
array_series = pd.Series(my_array)
print("NumPy Array to Series Conversion:\n", array_series)
```

Output:

```
Addition Result:
 0    11
1    22
2    33
3    44
4    55
dtype: int64
Subtraction Result:
 0    -9
1   -18
2   -27
3   -36
4   -45
dtype: int64
Multiplication Result:
 0     10
1     40
2     90
3    160
4    250
dtype: int64
Division Result:
 0    0.1
1    0.1
2    0.1
3    0.1
4    0.1
dtype: float64
Comparison Result:
 0     True
1    False
```

```
2    False
3     True
4    False
dtype: bool
Dictionary to Series Conversion:
 a    100
b    200
c    300
d    400
e    500
dtype: int64
NumPy Array to Series Conversion:
 0    10
1    20
2    30
3    40
4    50
dtype: int64
```

CODE:

```
import pandas as pd
df = pd.read_csv("filename.csv")
print(“before csv:\n”)
print(df)
df.fillna(-999, inplace=True)
df.dropna(how='all', inplace=True)
print(“After csv:\n”)
print(df)
```

Output:

```
Before csv:
```

```
A,B,C,D
1,2,,4
5,,,7
8,9,10,
```

After csv:

```
   A  B      C  D
0  1  2 -999.0  4
1  5 -999.0  -999  7
2  8  9  10.0 -999
```

CODE:

```
import pandas as pd
df1 = pd.DataFrame({
    'id': [1, 2, 3, 4, 5],
    'name': ['Shruti', 'Niyati', 'Ujjwal', 'Kaushik', 'Jagjeet'],
    'age': [22, 33, 25, 27, 29]
})
df2 = pd.DataFrame({
    'id': [1, 2, 4, 5, 6],
    'salary': [50000, 60000, 70000, 80000, 90000],
    'department': ['Sales', 'Marketing', 'IT', 'Finance', 'HR']
})
merged_df = pd.merge(df1, df2, on='id')
print('Merged DataFrame on the basis of id:')
print(merged_df)
merged_df_outer = pd.merge(df1, df2, on='id', how='outer')
print('Merged DataFrame using outer method:')
print(merged_df_outer)
merged_df_inner = pd.merge(df1, df2, on='id', how='inner')
print('Merged DataFrame using inner method:')
print(merged_df_inner)
```

Output:

```
Merged DataFrame on the basis of id:
  id   name      age   salary     department
0  1   Shruti    22    50000      Sales
1  2   Niyati    33    60000      Marketing
2  4   Kaushik   27    70000      Finance
3  5   Jagjeet   29    80000      HR

Merged DataFrame using outer method:
  id   name      age        salary     department
0  1   Shruti    22.0       50000.0    Sales
1  2   Niyati    33.0       60000.0    Marketing
2  3   Ujjwal    25.0       NaN        NaN
3  4   Kaushik   27.0       70000.0    Finance
4  5   Jagjeet   29.0       80000.0    HR
5  6   NaN       NaN        90000.0    IT

Merged DataFrame using inner method:
  id   name      age        salary     department
0  1   Shruti    22         50000      Sales
1  2   Niyati    33         60000      Marketing
2  4   Kaushik   27         70000      Finance
3  5   Jagjeet   29         80000      HR
```