Segment 2 - Treating missing values

```
import numpy as np
import pandas as pd

from pandas import Series, DataFrame
```

Figuring out what data is missing

```
missing = np.nan

series_obj = Series(['row 1', 'row 2', missing, 'row 4', 'row 5', 'row 6', missing, 'row 8'])
series_obj
```

	0
0	row 1
1	row 2
2	NaN
3	row 4
4	row 5
5	row 6
6	NaN
7	row 8

dtype: object

```
series_obj.isnull()
```

```
0    False
1    False
2     True
3    False
4    False
5    False
6     True
7    False
dtype: bool
```

Filling in for missing values

```
np.random.seed(25)
DF_obj = DataFrame(np.random.rand(36).reshape(6,6))
DF_obj
```

	0	1	2	3	4	5
0	0.870124	0.582277	0.278839	0.185911	0.411100	0.117376
1	0.684969	0.437611	0.556229	0.367080	0.402366	0.113041
2	0.447031	0.585445	0.161985	0.520719	0.326051	0.699186
3	0.366395	0.836375	0.481343	0.516502	0.383048	0.997541
4	0.514244	0.559053	0.034450	0.719930	0.421004	0.436935
5	0.281701	0.900274	0.669612	0.456069	0.289804	0.525819

```
DF_obj.loc[3:5, 0] = missing
DF_obj.loc[1:4, 5] = missing
DF_obj
```

```
          0         1         2         3         4         5
0  0.870124  0.582277  0.278839  0.185911  0.411100  0.117376
1  0.684969  0.437611  0.556229  0.367080  0.402366       NaN
2  0.447031  0.585445  0.161985  0.520719  0.326051       NaN
3       NaN  0.836375  0.481343  0.516502  0.383048       NaN
4       NaN  0.559053  0.034450  0.719930  0.421004       NaN
5       NaN  0.900274  0.669612  0.456069  0.289804  0.525819
```

```
filled_DF = DF_obj.fillna(0)
filled_DF
```

```
          0         1         2         3         4         5
0  0.870124  0.582277  0.278839  0.185911  0.411100  0.117376
1  0.684969  0.437611  0.556229  0.367080  0.402366  0.000000
2  0.447031  0.585445  0.161985  0.520719  0.326051  0.000000
3  0.000000  0.836375  0.481343  0.516502  0.383048  0.000000
4  0.000000  0.559053  0.034450  0.719930  0.421004  0.000000
5  0.000000  0.900274  0.669612  0.456069  0.289804  0.525819
```

```
filled_DF = DF_obj.fillna({0: 0.1, 5:1.25})
filled_DF
```

```
          0         1         2         3         4         5
0  0.870124  0.582277  0.278839  0.185911  0.411100  0.117376
1  0.684969  0.437611  0.556229  0.367080  0.402366  1.250000
2  0.447031  0.585445  0.161985  0.520719  0.326051  1.250000
3  0.100000  0.836375  0.481343  0.516502  0.383048  1.250000
4  0.100000  0.559053  0.034450  0.719930  0.421004  1.250000
5  0.100000  0.900274  0.669612  0.456069  0.289804  0.525819
```

```
fill_DF = DF_obj.fillna(method='ffill')
fill_DF
```

```
          0         1         2         3         4         5
0  0.870124  0.582277  0.278839  0.185911  0.411100  0.117376
1  0.684969  0.437611  0.556229  0.367080  0.402366  0.117376
2  0.447031  0.585445  0.161985  0.520719  0.326051  0.117376
3  0.447031  0.836375  0.481343  0.516502  0.383048  0.117376
4  0.447031  0.559053  0.034450  0.719930  0.421004  0.117376
5  0.447031  0.900274  0.669612  0.456069  0.289804  0.525819
```

Counting missing values

```
np.random.seed(25)
DF_obj = DataFrame(np.random.rand(36).reshape(6,6))
DF_obj.loc[3:5, 0] = missing
DF_obj.loc[1:4, 5] = missing
DF_obj
```

	0	1	2	3	4	5
0	0.870124	0.582277	0.278839	0.185911	0.411100	0.117376
1	0.684969	0.437611	0.556229	0.367080	0.402366	NaN
2	0.447031	0.585445	0.161985	0.520719	0.326051	NaN
3	NaN	0.836375	0.481343	0.516502	0.383048	NaN
4	NaN	0.559053	0.034450	0.719930	0.421004	NaN
5	NaN	0.900274	0.669612	0.456069	0.289804	0.525819

```
DF_obj.isnull().sum()
```

```
0    3
1    0
2    0
3    0
4    0
5    4
dtype: int64
```

Filtering out missing values

```
DF_no_NaN = DF_obj.dropna()
DF_no_NaN
```

	0	1	2	3	4	5
0	0.870124	0.582277	0.278839	0.185911	0.4111	0.117376

```
DF_no_NaN = DF_obj.dropna(axis=1)
DF_no_NaN
df=pd.read_csv('.csv ')
```

	1	2	3	4
0	0.582277	0.278839	0.185911	0.411100
1	0.437611	0.556229	0.367080	0.402366
2	0.585445	0.161985	0.520719	0.326051
3	0.836375	0.481343	0.516502	0.383048
4	0.559053	0.034450	0.719930	0.421004
5	0.900274	0.669612	0.456069	0.289804

Class exercise 1:

Here's a sample dataset you can use. You can copy this directly into a DataFrame:

data = { 'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve', None], 'Age': [24, None, 22, 23, None, 29], 'Score': [85, 70, None, 88, 95, 90], 'City': ['New York', 'Los Angeles', None, 'Chicago', 'Houston', None] }

df = pd.DataFrame(data)

print(df)

Tasks:

1. Detecting Missing Values Check for missing values in the dataset
2. Dropping Missing Values

- Drop rows with any missing values.
- Drop rows only if all values in the row are missing.
- Drop columns with missing values

3. Filling Missing Values

- Fill with a fixed value (e.g., 0 or 'Unknown').
- Fill using forward-fill (propagate last valid value).
- Fill using backward-fill.

4. Filling with Mean/Median/Mode

- Fill numerical columns with the mean value.
- Fill categorical columns with the mode.

5. Replacing Values with replace()

   Replace specific values like None or NaN.

Class exercise 2:

use https://www.kaggle.com/datasets/gunjanpathak/melb-data dataset and find the missing values and replace it with mean values.

```
Start coding or generate with AI.
```