In [1]:
```
import numpy as np
import pandas as pd
```

1. Create any Series and print the output

In [3]:
```
a=pd.Series([1,2,3,4,5,6])
print(a)
```

```
0    1
1    2
2    3
3    4
4    5
5    6
dtype: int64
```

2. Create any dataframe of 10x5 with few nan values and print the output

In [8]:
```
df=pd.DataFrame(np.random.rand(10,5))
df[4][4]=np.nan
df[2][6]=np.nan
df[0][3]=np.nan
df
```

Out[8]:

	0	1	2	3	4
0	0.703089	0.349944	0.239923	0.537871	0.715836
1	0.099337	0.632751	0.735349	0.810469	0.599917
2	0.018779	0.078738	0.795391	0.090249	0.140877
3	NaN	0.115318	0.597116	0.049245	0.829471
4	0.358033	0.660131	0.883482	0.062877	NaN
5	0.382389	0.988966	0.834214	0.378425	0.549848
6	0.353578	0.609429	NaN	0.705653	0.533679
7	0.527556	0.307957	0.567367	0.997163	0.901904
8	0.017539	0.059505	0.571829	0.579723	0.923660
9	0.548897	0.391605	0.469035	0.455152	0.400070

3.Display top 7 and last 6 rows and print the output

In [10]:
```
df.head(7)
```

Out[10]:

	0	1	2	3	4
0	0.703089	0.349944	0.239923	0.537871	0.715836
1	0.099337	0.632751	0.735349	0.810469	0.599917
2	0.018779	0.078738	0.795391	0.090249	0.140877
3	NaN	0.115318	0.597116	0.049245	0.829471
4	0.358033	0.660131	0.883482	0.062877	NaN
5	0.382389	0.988966	0.834214	0.378425	0.549848
6	0.353578	0.609429	NaN	0.705653	0.533679

In [11]:
```
df.tail(6)
```

Out[11]:

	0	1	2	3	4
4	0.358033	0.660131	0.883482	0.062877	NaN
5	0.382389	0.988966	0.834214	0.378425	0.549848
6	0.353578	0.609429	NaN	0.705653	0.533679
7	0.527556	0.307957	0.567367	0.997163	0.901904
8	0.017539	0.059505	0.571829	0.579723	0.923660
9	0.548897	0.391605	0.469035	0.455152	0.400070

4. Fill with a constant value and print the output

In [16]:
```
df.fillna("8")
```

Out[16]:

	0	1	2	3	4
0	0.703089	0.349944	0.239923	0.537871	0.715836
1	0.099337	0.632751	0.735349	0.810469	0.599917
2	0.018779	0.078738	0.795391	0.090249	0.140877
3	8	0.115318	0.597116	0.049245	0.829471
4	0.358033	0.660131	0.883482	0.062877	8
5	0.382389	0.988966	0.834214	0.378425	0.549848
6	0.353578	0.609429	8	0.705653	0.533679
7	0.527556	0.307957	0.567367	0.997163	0.901904
8	0.017539	0.059505	0.571829	0.579723	0.92366
9	0.548897	0.391605	0.469035	0.455152	0.40007

5. Drop the column with missing values and print the output

In [14]:
```
df.dropna(axis=1)
```

Out[14]:

	1	3
0	0.349944	0.537871
1	0.632751	0.810469
2	0.078738	0.090249
3	0.115318	0.049245
4	0.660131	0.062877
5	0.988966	0.378425
6	0.609429	0.705653
7	0.307957	0.997163
8	0.059505	0.579723
9	0.391605	0.455152

6. Drop the row with missing values and print the output

In [15]:
```
df.dropna()
```

Out[15]:

	0	1	2	3	4
0	0.703089	0.349944	0.239923	0.537871	0.715836
1	0.099337	0.632751	0.735349	0.810469	0.599917
2	0.018779	0.078738	0.795391	0.090249	0.140877
5	0.382389	0.988966	0.834214	0.378425	0.549848
7	0.527556	0.307957	0.567367	0.997163	0.901904
8	0.017539	0.059505	0.571829	0.579723	0.923660
9	0.548897	0.391605	0.469035	0.455152	0.400070

7. To check the presence of missing values in your dataframe

In [17]:
```
df.isna()
```

Out[17]:

	0	1	2	3	4
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	True	False	False	False	False
4	False	False	False	False	True
5	False	False	False	False	False
6	False	False	True	False	False
7	False	False	False	False	False
8	False	False	False	False	False
9	False	False	False	False	False

8. Use operators and check the condition and print the output

In [18]:
```
df1=df[df>0.5]
df1
```

Out[18]:

	0	1	2	3	4
0	0.703089	NaN	NaN	0.537871	0.715836
1	NaN	0.632751	0.735349	0.810469	0.599917
2	NaN	NaN	0.795391	NaN	NaN
3	NaN	NaN	0.597116	NaN	0.829471
4	NaN	0.660131	0.883482	NaN	NaN
5	NaN	0.988966	0.834214	NaN	0.549848
6	NaN	0.609429	NaN	0.705653	0.533679
7	0.527556	NaN	0.567367	0.997163	0.901904
8	NaN	NaN	0.571829	0.579723	0.923660
9	0.548897	NaN	NaN	NaN	NaN

9. Display your output using loc and iloc, row and column heading

In [19]:
```
df.loc[:2]
```

Out[19]:

	0	1	2	3	4
0	0.703089	0.349944	0.239923	0.537871	0.715836
1	0.099337	0.632751	0.735349	0.810469	0.599917
2	0.018779	0.078738	0.795391	0.090249	0.140877

In [20]:
```
df.iloc[:2]
```

Out[20]:

	0	1	2	3	4
0	0.703089	0.349944	0.239923	0.537871	0.715836
1	0.099337	0.632751	0.735349	0.810469	0.599917

10. Display the statistical summary of data ¶

In [21]:
```
df.describe()
```

Out[21]:

	0	1	2	3	4
count	9.000000	10.000000	9.000000	10.000000	9.000000
mean	0.334355	0.419434	0.632634	0.466683	0.621696
std	0.244375	0.301854	0.203349	0.326852	0.253826
min	0.017539	0.059505	0.239923	0.049245	0.140877
25%	0.099337	0.163478	0.567367	0.162293	0.533679
50%	0.358033	0.370775	0.597116	0.496511	0.599917
75%	0.527556	0.626920	0.795391	0.674170	0.829471
max	0.703089	0.988966	0.883482	0.997163	0.923660

In [ ]:
```

```