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

1. Create any Series and print the output

In [7]:
```
a=pd.Series([1,2,3,4,5])
a
```

Out[7]:
```
0    1
1    2
2    3
3    4
4    5
dtype: int64
```

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

In [11]:
```
a=pd.DataFrame({
    "a":[1,2,3,4,5,6,7,8,9,10],
    "b":[6,7,8,9,0,np.nan,np.nan,1,2,3],
    "c":[1,2,3,4,5,6,np.nan,np.nan,np.nan,2],
    "d":[1,2,3,np.nan,np.nan,6,7,8,9,3],
    "e":[1,2,3,4,5,6,7,8,np.nan,np.nan]
})
a
```

Out[11]:

	a	b	c	d	e
0	1	6.0	1.0	1.0	1.0
1	2	7.0	2.0	2.0	2.0
2	3	8.0	3.0	3.0	3.0
3	4	9.0	4.0	NaN	4.0
4	5	0.0	5.0	NaN	5.0
5	6	NaN	6.0	6.0	6.0
6	7	NaN	NaN	7.0	7.0
7	8	1.0	NaN	8.0	8.0
8	9	2.0	NaN	9.0	NaN
9	10	3.0	2.0	3.0	NaN

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

In [14]:
```
a.head(7)
```

Out[14]:

	a	b	c	d	e
0	1	6.0	1.0	1.0	1.0
1	2	7.0	2.0	2.0	2.0
2	3	8.0	3.0	3.0	3.0
3	4	9.0	4.0	NaN	4.0
4	5	0.0	5.0	NaN	5.0
5	6	NaN	6.0	6.0	6.0
6	7	NaN	NaN	7.0	7.0

4. Fill with a constant value and print the output

In [6]:
```
d=pd.DataFrame(np.random.randn(3,4))
d
```

Out[6]:

	0	1	2	3
0	0.488602	-1.314332	0.148557	1.023907
1	-1.336487	1.330602	-1.421489	0.659434
2	-0.584856	0.248430	1.720381	-0.742871

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

In [18]:
```
a.dropna(axis=1,how="any")
```

Out[18]:

	a
0	1
1	2
2	3
3	4
4	5
5	6
6	7
7	8
8	9
9	10

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

In [17]:
```
a.dropna()
```

Out[17]:

	a	b	c	d	e
0	1	6.0	1.0	1.0	1.0
1	2	7.0	2.0	2.0	2.0
2	3	8.0	3.0	3.0	3.0

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

In [19]:
```
a.isna()
```

Out[19]:

	a	b	c	d	e
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	True	False
4	False	False	False	True	False
5	False	True	False	False	False
6	False	True	True	False	False
7	False	False	True	False	False
8	False	False	True	False	True
9	False	False	False	False	True

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

In [20]:
```
a[a["b"]>7]
```

Out[20]:

	a	b	c	d	e
2	3	8.0	3.0	3.0	3.0
3	4	9.0	4.0	NaN	4.0

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

In [21]:
```
data.loc[2:5]
```

Out[21]:

	a	b	c	d	e
2	3	8.0	3.0	3.0	3.0
3	4	9.0	4.0	NaN	4.0
4	5	0.0	5.0	NaN	5.0
5	6	NaN	6.0	6.0	6.0

10. Display the statistical summary of data

In [22]:
```
a.describe()
```

Out[22]:

	a	b	c	d	e
count	10.00000	8.000000	7.000000	8.000000	8.00000
mean	5.50000	4.500000	3.285714	4.875000	4.50000
std	3.02765	3.422614	1.799471	2.997022	2.44949
min	1.00000	0.000000	1.000000	1.000000	1.00000
25%	3.25000	1.750000	2.000000	2.750000	2.75000
50%	5.50000	4.500000	3.000000	4.500000	4.50000
75%	7.75000	7.250000	4.500000	7.250000	6.25000
max	10.00000	9.000000	6.000000	9.000000	8.00000

In [ ]:
```

```