Days 3 - Pandas

In [4]:

```
import pandas as pd
import numpy as np
```

1. Create any Series and print the output

In [3]:

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

Out[3]:

```
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 [17]:

```
d=pd.DataFrame(
    {
        "A":1.0,
        "B":pd.Timestamp("20230721"),
        "C":56,
        "D":14,
        "E":pd.Series(index=list(range(10)))
    }
)
d
```

```
<ipython-input-17-c42489e14f55>:7: DeprecationWarning: The default dtype f
or empty Series will be 'object' instead of 'float64' in a future version.
Specify a dtype explicitly to silence this warning.
  "E":pd.Series(index=list(range(10)))
```

Out[17]:

	A	B	C	D	E
0	1.0	2023-07-21	56	14	NaN
1	1.0	2023-07-21	56	14	NaN
2	1.0	2023-07-21	56	14	NaN
3	1.0	2023-07-21	56	14	NaN
4	1.0	2023-07-21	56	14	NaN
5	1.0	2023-07-21	56	14	NaN
6	1.0	2023-07-21	56	14	NaN
7	1.0	2023-07-21	56	14	NaN
8	1.0	2023-07-21	56	14	NaN
9	1.0	2023-07-21	56	14	NaN

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

In [18]:

```
d.head(7)
```

Out[18]:

	A	B	C	D	E
0	1.0	2023-07-21	56	14	NaN
1	1.0	2023-07-21	56	14	NaN
2	1.0	2023-07-21	56	14	NaN
3	1.0	2023-07-21	56	14	NaN
4	1.0	2023-07-21	56	14	NaN
5	1.0	2023-07-21	56	14	NaN
6	1.0	2023-07-21	56	14	NaN

In [19]:

```
d.tail(6)
```

Out[19]:

	A	B	C	D	E
4	1.0	2023-07-21	56	14	NaN
5	1.0	2023-07-21	56	14	NaN
6	1.0	2023-07-21	56	14	NaN
7	1.0	2023-07-21	56	14	NaN
8	1.0	2023-07-21	56	14	NaN
9	1.0	2023-07-21	56	14	NaN

4. Fill with a constant value and print the output

In [13]:

```
df=pd.DataFrame(
    {
        "A":1.0,
        "B":pd.Timestamp("20230721"),
        "C":pd.Series(index=list(range(4)))
    }
)
df
```

```
<ipython-input-13-3d9d544b9ac1>:5: DeprecationWarning: The default dtype f
or empty Series will be 'object' instead of 'float64' in a future version.
Specify a dtype explicitly to silence this warning.
  "C":pd.Series(index=list(range(4)))
```

Out[13]:

	A	B	C
0	1.0	2023-07-21	NaN
1	1.0	2023-07-21	NaN
2	1.0	2023-07-21	NaN
3	1.0	2023-07-21	NaN

In [14]:

```
df.fillna(1)
```

Out[14]:

	A	B	C
0	1.0	2023-07-21	1.0
1	1.0	2023-07-21	1.0
2	1.0	2023-07-21	1.0
3	1.0	2023-07-21	1.0

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

In [20]:

```
df.dropna(axis=1,how='any')
```

Out[20]:

	A	B
0	1.0	2023-07-21
1	1.0	2023-07-21
2	1.0	2023-07-21
3	1.0	2023-07-21

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

In [22]:

```
x=pd.DataFrame(
    {
        "A":1.0,
        "B":2,
        "C":pd.Series(index=list(range(4)))
    }
)
x
```

```
<ipython-input-22-2678f0c96b7e>:5: DeprecationWarning: The default dtype f
or empty Series will be 'object' instead of 'float64' in a future version.
Specify a dtype explicitly to silence this warning.
  "C":pd.Series(index=list(range(4)))
```

Out[22]:

	A	B	C
0	1.0	2	NaN
1	1.0	2	NaN
2	1.0	2	NaN
3	1.0	2	NaN

In [23]:

```
x.dropna()
```

Out[23]:

A	B	C

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

In [24]:

```
pd.isna(x)
```

Out[24]:

	A	B	C
0	False	False	True
1	False	False	True
2	False	False	True
3	False	False	True

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

In [25]:

```
x[x["B"]<=2]
```

Out[25]:

	A	B	C
0	1.0	2	NaN
1	1.0	2	NaN
2	1.0	2	NaN
3	1.0	2	NaN

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

In [28]:

```
x.loc["A":"C"]
```

Out[28]:

A	B	C

In [29]:

```
x.iloc[0:2]
```

Out[29]:

	A	B	C
0	1.0	2	NaN
1	1.0	2	NaN

In [30]:

```
x.columns
```

Out[30]:

```
Index(['A', 'B', 'C'], dtype='object')
```

In [31]:

```
x.index
```

Out[31]:

```
Int64Index([0, 1, 2, 3], dtype='int64')
```

10. Display the statistical summary of data

In [34]:

```
x.describe()
```

Out[34]:

	A	B	C
count	4.0	4.0	0.0
mean	1.0	2.0	NaN
std	0.0	0.0	NaN
min	1.0	2.0	NaN
25%	1.0	2.0	NaN
50%	1.0	2.0	NaN
75%	1.0	2.0	NaN
max	1.0	2.0	NaN

In [ ]:

```

```