Fun & iloc

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

In [78]:
```
df=pd.read_csv('Visha.csv')
df.dtypes
print(df)
```

```
    Roll   FSD  Python   COA   TOC Remarks
0      1   NaN    10.0   8.0  10.0       g
1      2   6.0    10.0   7.0   8.0       a
2      3   NaN     NaN   NaN   NaN       b
3      4   6.0    10.0   7.0   8.0       e
4      5   2.0    10.0   5.0   3.0       b
5      6   7.0    10.0   7.0   5.0       v
6      7   8.0    10.0   8.0   5.0       v
7      8   9.0    10.0   9.0   5.0       v
8      9  10.0    10.0  10.0   5.0       v
9     10  11.0    10.0  11.0   5.0       v
10    11  12.0    10.0  12.0   5.0       v
11    12  13.0    10.0  13.0   5.0       v
12    13  14.0    10.0  14.0   5.0       v
13    14  15.0    10.0  15.0   5.0       v
14    15  16.0    10.0  16.0   5.0       v
15    16  17.0    10.0  17.0   5.0       v
16    17  18.0    10.0  18.0   5.0       v
17    18  19.0    10.0  19.0   5.0       v
```

In [9]:
```
df.dtypes
```

Out[9]:
```
Roll         int64
FSD        float64
Python     float64
COA        float64
TOC        float64
Remarks     object
dtype: object
```

In [10]:
```
1 df.index
```

Out[10]:
```
RangeIndex(start=0, stop=6, step=1)
```

In [20]:
```
1 df.columns
2
```

Out[20]:
```
Index(['Roll', 'FSD', 'Python', 'COA', 'TOC', 'Remarks'], dtype='object')
```

In [21]:
```
1 df.axes
```

Out[21]:
```
[RangeIndex(start=0, stop=6, step=1),
 Index(['Roll', 'FSD', 'Python', 'COA', 'TOC', 'Remarks'], dtype='object')]
```

In [23]:
```
1 df.values
```

Out[23]:
```
array([[1, nan, 10.0, 8.0, 10.0, 'g'],
       [2, 6.0, 10.0, 7.0, 8.0, 'a'],
       [3, nan, nan, nan, nan, 'b'],
       [4, 6.0, 10.0, 7.0, 8.0, 'e'],
       [5, 2.0, 10.0, 5.0, 3.0, 'b'],
       [6, 7.0, 10.0, 7.0, 5.0, 'v']], dtype=object)
```

In [25]:
```
1 df.shape
```

Out[25]:
```
(6, 6)
```

In [30]:
```
1 df.size
```

Out[30]:
```
36
```

In [6]:
```
1 df.iloc[0]
```

Out[6]:
```
Roll         1
FSD        NaN
Python      10
COA          8
TOC         10
Remarks      g
Name: 0, dtype: object
```

In [11]:
```
df.iloc[0,0]
```

Out[11]:
```
1
```

In [19]:
```
print(type(df.iloc[1]))
df.iloc[1]
```

```
<class 'pandas.core.series.Series'>
```

Out[19]:
```
Roll        2
FSD         6
Python     10
COA         7
TOC         8
Remarks     a
Name: 1, dtype: object
```

In [18]:
```
print(type(df.iloc[[1]]))
df.iloc[[1]]

```

```
<class 'pandas.core.frame.DataFrame'>
```

Out[18]:

	Roll	FSD	Python	COA	TOC	Remarks
1	2	6.0	10.0	7.0	8.0	a

In [22]:
```
df.iloc[0:5,0:3]
```

Out[22]:

	Roll	FSD	Python
0	1	NaN	10.0
1	2	6.0	10.0
2	3	NaN	NaN
3	4	6.0	10.0
4	5	2.0	10.0

In [24]:
```
1 df.iloc[:,[0,2]]
```

Out[24]:

	Roll	Python
0	1	10.0
1	2	10.0
2	3	NaN
3	4	10.0
4	5	10.0
5	6	10.0

In [29]:
```
1 df.iloc[1]
```

Out[29]:
```
Roll        2
FSD         6
Python     10
COA         7
TOC         8
Remarks     a
Name: 1, dtype: object
```

In [32]:
```
1 df.iloc[::5,::5]
```

Out[32]:

	Roll	Remarks
0	1	g
5	6	v

In [36]:
```
1 df.iloc[[0,-1],[0,-1]]
```

Out[36]:

	Roll	Remarks
0	1	g
5	6	v

In [37]:
```
df[["Roll","Python"]]
```

Out[37]:

	Roll	Python
0	1	10.0
1	2	10.0
2	3	NaN
3	4	10.0
4	5	10.0
5	6	10.0

Loc

In [38]:
```
df=df.set_index("Roll")
```

In [45]:
```
print(df)
```

```
      FSD  Python  COA   TOC Remarks
Roll
1     NaN    10.0  8.0  10.0       g
2     6.0    10.0  7.0   8.0       a
3     NaN     NaN  NaN   NaN       b
4     6.0    10.0  7.0   8.0       e
5     2.0    10.0  5.0   3.0       b
6     7.0    10.0  7.0   5.0       v
```

In [50]:
```
df.iloc[0,2]
```

Out[50]: 8.0

In [51]:
```
df.iloc[0,0]
```

Out[51]: nan

In [43]:
```
df.loc[2,"FSD"]
```

Out[43]:
```
6.0
```

In [44]:
```
df.loc[1,"FSD"]
```

Out[44]:
```
nan
```

In [54]:
```
df["FSD"]>5

```

Out[54]:
```
Roll
1    False
2     True
3    False
4     True
5    False
6     True
Name: FSD, dtype: bool
```

In [55]:
```
df[df["FSD"]>5]
```

Out[55]:

Roll	FSD	Python	COA	TOC	Remarks
2	6.0	10.0	7.0	8.0	a
4	6.0	10.0	7.0	8.0	e
6	7.0	10.0	7.0	5.0	v

In [68]:
```
df[(df["FSD"]>5) & (df["Remarks"]=="a")]
```

Out[68]:

Roll	FSD	Python	COA	TOC	Remarks
2	6.0	10.0	7.0	8.0	a

In [70]:
```
df.loc[2,"FSD"]=10
```

In [71]:
```
print(df)
```

```
       FSD  Python  COA   TOC Remarks
Roll
1      NaN    10.0  8.0  10.0       g
2     10.0    10.0  7.0   8.0       a
3      NaN     NaN  NaN   NaN       b
4      6.0    10.0  7.0   8.0       e
5      2.0    10.0  5.0   3.0       b
6      7.0    10.0  7.0   5.0       v
```

In [74]:
```
df["FSD"].iloc[1]=8
```

```
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexing.py:670: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  iloc._setitem_with_indexer(indexer, value)
```

In [79]:
```
1 print(df)
```

```
    Roll   FSD  Python   COA   TOC Remarks
0      1   NaN    10.0   8.0  10.0       g
1      2   6.0    10.0   7.0   8.0       a
2      3   NaN     NaN   NaN   NaN       b
3      4   6.0    10.0   7.0   8.0       e
4      5   2.0    10.0   5.0   3.0       b
5      6   7.0    10.0   7.0   5.0       v
6      7   8.0    10.0   8.0   5.0       v
7      8   9.0    10.0   9.0   5.0       v
8      9  10.0    10.0  10.0   5.0       v
9     10  11.0    10.0  11.0   5.0       v
10    11  12.0    10.0  12.0   5.0       v
11    12  13.0    10.0  13.0   5.0       v
12    13  14.0    10.0  14.0   5.0       v
13    14  15.0    10.0  15.0   5.0       v
14    15  16.0    10.0  16.0   5.0       v
15    16  17.0    10.0  17.0   5.0       v
16    17  18.0    10.0  18.0   5.0       v
17    18  19.0    10.0  19.0   5.0       v
```

In [83]:
```
1 df.head(5)
```

Out[83]:

	Roll	FSD	Python	COA	TOC	Remarks
0	1	NaN	10.0	8.0	10.0	g
1	2	6.0	10.0	7.0	8.0	a
2	3	NaN	NaN	NaN	NaN	b
3	4	6.0	10.0	7.0	8.0	e
4	5	2.0	10.0	5.0	3.0	b

In [84]:
```
1 df.tail(5)
```

Out[84]:

	Roll	FSD	Python	COA	TOC	Remarks
13	14	15.0	10.0	15.0	5.0	v
14	15	16.0	10.0	16.0	5.0	v
15	16	17.0	10.0	17.0	5.0	v
16	17	18.0	10.0	18.0	5.0	v
17	18	19.0	10.0	19.0	5.0	v

In [86]:
```
1 df.sample(5)
```

Out[86]:

	Roll	FSD	Python	COA	TOC	Remarks
14	15	16.0	10.0	16.0	5.0	v
13	14	15.0	10.0	15.0	5.0	v
5	6	7.0	10.0	7.0	5.0	v
8	9	10.0	10.0	10.0	5.0	v
9	10	11.0	10.0	11.0	5.0	v

In [88]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18 entries, 0 to 17
Data columns (total 6 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Roll     18 non-null     int64
 1   FSD      16 non-null     float64
 2   Python   17 non-null     float64
 3   COA      17 non-null     float64
 4   TOC      17 non-null     float64
 5   Remarks  18 non-null     object
dtypes: float64(4), int64(1), object(1)
memory usage: 992.0+ bytes
```

In [89]:
```
1 df.describe()
```

Out[89]:

	Roll	FSD	Python	COA	TOC
count	18.000000	16.000000	17.0	17.000000	17.000000
mean	9.500000	11.437500	10.0	11.529412	5.529412
std	5.338539	4.912145	0.0	4.374895	1.624717
min	1.000000	2.000000	10.0	5.000000	3.000000
25%	5.250000	7.750000	10.0	8.000000	5.000000
50%	9.500000	11.500000	10.0	11.000000	5.000000
75%	13.750000	15.250000	10.0	15.000000	5.000000
max	18.000000	19.000000	10.0	19.000000	10.000000

In [96]:
```
1 df.describe(percentiles=[0.2,0.6,0.8,0.9,1,0.67])
```

Out[96]:

	Roll	FSD	Python	COA	TOC
count	18.000000	16.000000	17.0	17.000000	17.000000
mean	9.500000	11.437500	10.0	11.529412	5.529412
std	5.338539	4.912145	0.0	4.374895	1.624717
min	1.000000	2.000000	10.0	5.000000	3.000000
20%	4.400000	7.000000	10.0	7.200000	5.000000
50%	9.500000	11.500000	10.0	11.000000	5.000000
60%	11.200000	13.000000	10.0	12.600000	5.000000
67%	12.390000	14.050000	10.0	13.720000	5.000000
80%	14.600000	16.000000	10.0	15.800000	5.000000
90%	16.300000	17.500000	10.0	17.400000	8.000000
100%	18.000000	19.000000	10.0	19.000000	10.000000
max	18.000000	19.000000	10.0	19.000000	10.000000

In [98]:
```
1 df.describe(include="all")
```

Out[98]:

	Roll	FSD	Python	COA	TOC	Remarks
count	18.000000	16.000000	17.0	17.000000	17.000000	18
unique	NaN	NaN	NaN	NaN	NaN	5
top	NaN	NaN	NaN	NaN	NaN	v
freq	NaN	NaN	NaN	NaN	NaN	13
mean	9.500000	11.437500	10.0	11.529412	5.529412	NaN
std	5.338539	4.912145	0.0	4.374895	1.624717	NaN
min	1.000000	2.000000	10.0	5.000000	3.000000	NaN
25%	5.250000	7.750000	10.0	8.000000	5.000000	NaN
50%	9.500000	11.500000	10.0	11.000000	5.000000	NaN
75%	13.750000	15.250000	10.0	15.000000	5.000000	NaN
max	18.000000	19.000000	10.0	19.000000	10.000000	NaN

In [ ]:
```
1 help(df.describe)
```

In [103]:
```
1 df.describe(exclude=np.number)
```

Out[103]:

	Remarks
count	18
unique	5
top	v
freq	13

In [105]:
```
1 df.isna().sum()
```

Out[105]:
```
Roll       0
FSD        2
Python     1
COA        1
TOC        1
Remarks    0
dtype: int64
```

In [106]:
```
1 df.isnull()
```

Out[106]:

	Roll	FSD	Python	COA	TOC	Remarks
0	False	True	False	False	False	False
1	False	False	False	False	False	False
2	False	True	True	True	True	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False
5	False	False	False	False	False	False
6	False	False	False	False	False	False
7	False	False	False	False	False	False
8	False	False	False	False	False	False
9	False	False	False	False	False	False
10	False	False	False	False	False	False
11	False	False	False	False	False	False
12	False	False	False	False	False	False
13	False	False	False	False	False	False
14	False	False	False	False	False	False
15	False	False	False	False	False	False
16	False	False	False	False	False	False
17	False	False	False	False	False	False

In [107]:
```
1 help(df.fillna)
```

```
Help on method fillna in module pandas.core.frame:

fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None) -> Union[ForwardRef('Data
Frame'), NoneType] method of pandas.core.frame.DataFrame instance
    Fill NA/NaN values using the specified method.

    Parameters
    ----------
    value : scalar, dict, Series, or DataFrame
        Value to use to fill holes (e.g. 0), alternately a
        dict/Series/DataFrame of values specifying which value to use for
        each index (for a Series) or column (for a DataFrame).  Values not
        in the dict/Series/DataFrame will not be filled. This value cannot
        be a list.
    method : {'backfill', 'bfill', 'pad', 'ffill', None}, default None
        Method to use for filling holes in reindexed Series
        pad / ffill: propagate last valid observation forward to next valid
        backfill / bfill: use next valid observation to fill gap.
    axis : {0 or 'index', 1 or 'columns'}
        Axis along which to fill missing values.
    inplace : bool, default False
        If True, fill in-place. Note: this will modify any
        other views on this object (e.g., a no-copy slice for a column in a
        DataFrame).
    limit : int, default None
        If method is specified, this is the maximum number of consecutive
        NaN values to forward/backward fill. In other words, if there is
        a gap with more than this number of consecutive NaNs, it will only
        be partially filled. If method is not specified, this is the
        maximum number of entries along the entire axis where NaNs will be
        filled. Must be greater than 0 if not None.
    downcast : dict, default is None
        A dict of item->dtype of what to downcast if possible,
        or the string 'infer' which will try to downcast to an appropriate
        equal type (e.g. float64 to int64 if possible).

    Returns
    -------
    DataFrame or None
        Object with missing values filled or None if ``inplace=True``.

    See Also
    --------
```

```
interpolate : Fill NaN values using interpolation.
reindex : Conform object to new index.
asfreq : Convert TimeSeries to specified frequency.

Examples
--------
>>> df = pd.DataFrame([[np.nan, 2, np.nan, 0],
...                    [3, 4, np.nan, 1],
...                    [np.nan, np.nan, np.nan, 5],
...                    [np.nan, 3, np.nan, 4]],
...                   columns=list('ABCD'))
>>> df
     A    B   C  D
0  NaN  2.0 NaN  0
1  3.0  4.0 NaN  1
2  NaN  NaN NaN  5
3  NaN  3.0 NaN  4

Replace all NaN elements with 0s.

>>> df.fillna(0)
    A   B   C   D
0   0.0 2.0 0.0 0
1   3.0 4.0 0.0 1
2   0.0 0.0 0.0 5
3   0.0 3.0 0.0 4

We can also propagate non-null values forward or backward.

>>> df.fillna(method='ffill')
    A   B   C   D
0   NaN 2.0 NaN 0
1   3.0 4.0 NaN 1
2   3.0 4.0 NaN 5
3   3.0 3.0 NaN 4

Replace all NaN elements in column 'A', 'B', 'C', and 'D', with 0, 1,
2, and 3 respectively.

>>> values = {'A': 0, 'B': 1, 'C': 2, 'D': 3}
>>> df.fillna(value=values)
    A   B   C   D
0   0.0 2.0 2.0 0
```

```
1   3.0 4.0 2.0 1
2   0.0 1.0 2.0 5
3   0.0 3.0 2.0 4

Only replace the first NaN element.

>>> df.fillna(value=values, limit=1)
    A   B   C   D
0   0.0 2.0 2.0 0
1   3.0 4.0 NaN 1
2   NaN 1.0 NaN 5
3   NaN 3.0 NaN 4
```

In [110]:
```
df.fillna(df.mean())
```

Out[110]:

	Roll	FSD	Python	COA	TOC	Remarks
0	1	11.4375	10.0	8.000000	10.000000	g
1	2	6.0000	10.0	7.000000	8.000000	a
2	3	11.4375	10.0	11.529412	5.529412	b
3	4	6.0000	10.0	7.000000	8.000000	e
4	5	2.0000	10.0	5.000000	3.000000	b
5	6	7.0000	10.0	7.000000	5.000000	v
6	7	8.0000	10.0	8.000000	5.000000	v
7	8	9.0000	10.0	9.000000	5.000000	v
8	9	10.0000	10.0	10.000000	5.000000	v
9	10	11.0000	10.0	11.000000	5.000000	v
10	11	12.0000	10.0	12.000000	5.000000	v
11	12	13.0000	10.0	13.000000	5.000000	v
12	13	14.0000	10.0	14.000000	5.000000	v
13	14	15.0000	10.0	15.000000	5.000000	v
14	15	16.0000	10.0	16.000000	5.000000	v
15	16	17.0000	10.0	17.000000	5.000000	v
16	17	18.0000	10.0	18.000000	5.000000	v
17	18	19.0000	10.0	19.000000	5.000000	v

In [116]:
```
df["FSD"]=df["FSD"].fillna(100)
```

In [117]:
```
print(df)
```

```
    Roll   FSD  Python   COA   TOC Remarks
0      1   0.0    10.0   8.0  10.0       g
1      2   6.0    10.0   7.0   8.0       a
2      3   0.0     NaN   NaN   NaN       b
3      4   6.0    10.0   7.0   8.0       e
4      5   2.0    10.0   5.0   3.0       b
5      6   7.0    10.0   7.0   5.0       v
6      7   8.0    10.0   8.0   5.0       v
7      8   9.0    10.0   9.0   5.0       v
8      9  10.0    10.0  10.0   5.0       v
9     10  11.0    10.0  11.0   5.0       v
10    11  12.0    10.0  12.0   5.0       v
11    12  13.0    10.0  13.0   5.0       v
12    13  14.0    10.0  14.0   5.0       v
13    14  15.0    10.0  15.0   5.0       v
14    15  16.0    10.0  16.0   5.0       v
15    16  17.0    10.0  17.0   5.0       v
16    17  18.0    10.0  18.0   5.0       v
17    18  19.0    10.0  19.0   5.0       v
```

In [118]:
```
1 df.fillna(method="bfill")
```

Out[118]:

	Roll	FSD	Python	COA	TOC	Remarks
0	1	0.0	10.0	8.0	10.0	g
1	2	6.0	10.0	7.0	8.0	a
2	3	0.0	10.0	7.0	8.0	b
3	4	6.0	10.0	7.0	8.0	e
4	5	2.0	10.0	5.0	3.0	b
5	6	7.0	10.0	7.0	5.0	v
6	7	8.0	10.0	8.0	5.0	v
7	8	9.0	10.0	9.0	5.0	v
8	9	10.0	10.0	10.0	5.0	v
9	10	11.0	10.0	11.0	5.0	v
10	11	12.0	10.0	12.0	5.0	v
11	12	13.0	10.0	13.0	5.0	v
12	13	14.0	10.0	14.0	5.0	v
13	14	15.0	10.0	15.0	5.0	v
14	15	16.0	10.0	16.0	5.0	v
15	16	17.0	10.0	17.0	5.0	v
16	17	18.0	10.0	18.0	5.0	v
17	18	19.0	10.0	19.0	5.0	v

In [ ]:
```
1
```