Pandas Tutorial

Import Libraries

In [ ]:

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

Object Series

In [ ]:

```python
series1 = pd.Series([1,2,3,4,5,6,7,8,9,10])
series1
```

Out[ ]:

```
0     1
1     2
2     3
3     4
4     5
5     6
6     7
7     8
8     9
9    10
dtype: int64
```

In [ ]:

```python
dates = pd.date_range("20220101", periods=20)
dates
```

Out[ ]:

```
DatetimeIndex(['2022-01-01', '2022-01-02', '2022-01-03', '2022-01-04',
               '2022-01-05', '2022-01-06', '2022-01-07', '2022-01-08',
               '2022-01-09', '2022-01-10', '2022-01-11', '2022-01-12',
               '2022-01-13', '2022-01-14', '2022-01-15', '2022-01-16',
               '2022-01-17', '2022-01-18', '2022-01-19', '2022-01-20'],
              dtype='datetime64[ns]', freq='D')
```

Creating Data-Frame

In [ ]:

```python
df=pd.DataFrame(np.random.randn(20,4), index=dates,columns=list("ABCD"))
df
```

Out[ ]:

	A	B	C	D
2022-01-01	-0.243597	-0.807238	0.546472	0.407585
2022-01-02	-0.453285	0.501168	0.098791	0.060577
2022-01-03	-0.542659	-1.559297	-1.319488	-0.104209
2022-01-04	0.603553	-1.062545	-1.369933	-0.771599
2022-01-05	1.078564	1.351742	0.813419	-0.246128
2022-01-06	0.031488	-0.132891	-0.602976	-0.110989
2022-01-07	0.836313	1.767735	-0.452541	-0.938514
2022-01-08	-0.964943	-0.665613	-0.248710	0.096500
2022-01-09	-0.347374	-0.275315	-0.564968	0.817769
2022-01-10	2.146198	0.227445	0.366495	1.291886
2022-01-11	0.045809	0.226590	-0.445373	-0.786563
2022-01-12	1.017386	-1.550262	0.308897	-0.369246
2022-01-13	0.358586	0.417658	0.886548	0.847326
2022-01-14	-0.621551	0.848705	-0.348097	-0.671176
2022-01-15	1.061288	-0.765717	1.715390	0.468456
2022-01-16	1.119669	1.251402	0.203521	1.244519
2022-01-17	-0.363791	-0.470904	0.171403	-0.472825
2022-01-18	0.340325	-1.526882	0.285516	0.144267
2022-01-19	-0.410259	0.400678	1.087062	-1.059641
2022-01-20	-0.530977	2.066283	0.250032	0.172869

In [ ]:

```python
df2 = pd.DataFrame(
    {
        "A": 1.0,
        "B": pd.Timestamp("20220101"),
        "C": pd.Series(1, index=list(range(4)), dtype="float32"),
        "D": np.array([3]*4, dtype="int32"),
        "E": pd.Categorical(["girl","women","girl","women"]),
        "F": "female",
    }
)
df2
```

Out[ ]:

	A	B	C	D	E	F
0	1.0	2022-01-01	1.0	3	girl	female
1	1.0	2022-01-01	1.0	3	women	female
2	1.0	2022-01-01	1.0	3	girl	female
3	1.0	2022-01-01	1.0	3	women	female

In [ ]:

```python
df2.dtypes
```

Out[ ]:

```
A           float64
B    datetime64[ns]
C           float32
D             int32
E          category
F            object
dtype: object
```

head() & tail() Fuctions

In [ ]:

```python
df.head(4)
```

Out[ ]:

	A	B	C	D
2022-01-01	-0.243597	-0.807238	0.546472	0.407585
2022-01-02	-0.453285	0.501168	0.098791	0.060577
2022-01-03	-0.542659	-1.559297	-1.319488	-0.104209
2022-01-04	0.603553	-1.062545	-1.369933	-0.771599

In [ ]:

```python
df.tail(4)
```

Out[ ]:

	A	B	C	D
2022-01-17	-0.363791	-0.470904	0.171403	-0.472825
2022-01-18	0.340325	-1.526882	0.285516	0.144267
2022-01-19	-0.410259	0.400678	1.087062	-1.059641
2022-01-20	-0.530977	2.066283	0.250032	0.172869

Index()

In [ ]:

```python
df.index
```

Out[ ]:

```
DatetimeIndex(['2022-01-01', '2022-01-02', '2022-01-03', '2022-01-04',
               '2022-01-05', '2022-01-06', '2022-01-07', '2022-01-08',
               '2022-01-09', '2022-01-10', '2022-01-11', '2022-01-12',
               '2022-01-13', '2022-01-14', '2022-01-15', '2022-01-16',
               '2022-01-17', '2022-01-18', '2022-01-19', '2022-01-20'],
              dtype='datetime64[ns]', freq='D')
```

Converting Data Frame into Numpy Array using numpy()

In [ ]:

```python
df.to_numpy()
```

Out[ ]:

```
array([[-0.24359652, -0.80723815,  0.54647183,  0.40758465],
       [-0.4532853 ,  0.50116843,  0.09879093,  0.06057657],
       [-0.54265886, -1.55929731, -1.31948775, -0.1042089 ],
       [ 0.60355305, -1.06254535, -1.36993306, -0.77159945],
       [ 1.07856413,  1.35174152,  0.81341914, -0.24612845],
       [ 0.03148845, -0.13289095, -0.60297642, -0.11098892],
       [ 0.83631343,  1.767735  , -0.45254131, -0.93851375],
       [-0.96494305, -0.66561329, -0.24871002,  0.09650032],
       [-0.34737427, -0.27531453, -0.56496802,  0.81776875],
       [ 2.14619839,  0.22744517,  0.36649505,  1.29188602],
       [ 0.04580937,  0.22658978, -0.44537331, -0.78656264],
       [ 1.01738646, -1.5502623 ,  0.30889725, -0.36924591],
       [ 0.35858593,  0.41765806,  0.88654817,  0.84732649],
       [-0.62155061,  0.8487051 , -0.34809732, -0.67117605],
       [ 1.0612881 , -0.76571719,  1.71538965,  0.46845607],
       [ 1.11966891,  1.25140199,  0.203521  ,  1.24451943],
       [-0.36379149, -0.47090382,  0.17140343, -0.47282478],
       [ 0.34032525, -1.52688247,  0.2855161 ,  0.14426691],
       [-0.41025906,  0.40067775,  1.08706189, -1.05964131],
       [-0.53097709,  2.06628323,  0.25003195,  0.17286856]])
```

describe() this fuction describe about data Mean,Median Mode

In [ ]:

```python
df.describe()
```

Out[ ]:

	A	B	C	D
count	20.000000	20.000000	20.000000	20.000000
mean	0.208037	0.012137	0.069073	0.001043
std	0.801647	1.087864	0.764892	0.694549
min	-0.964943	-1.559297	-1.369933	-1.059641
25%	-0.421016	-0.776097	-0.447165	-0.522413
50%	0.038649	0.046849	0.187462	-0.021816
75%	0.881582	0.588053	0.411489	0.422803
max	2.146198	2.066283	1.715390	1.291886

How to Transpose Data (Row to Column and Column to Row)

In [ ]:

```python
df2.T
```

Out[ ]:

	0	1	2	3
A	1.0	1.0	1.0	1.0
B	2022-01-01 00:00:00	2022-01-01 00:00:00	2022-01-01 00:00:00	2022-01-01 00:00:00
C	1.0	1.0	1.0	1.0
D	3	3	3	3
E	girl	women	girl	women
F	female	female	female	female

In [ ]:

```python
# using Axis=0 data rows values will be in assending form
df.sort_index(axis=0,ascending=True)
```

Out[ ]:

	A	B	C	D
2022-01-01	-0.243597	-0.807238	0.546472	0.407585
2022-01-02	-0.453285	0.501168	0.098791	0.060577
2022-01-03	-0.542659	-1.559297	-1.319488	-0.104209
2022-01-04	0.603553	-1.062545	-1.369933	-0.771599
2022-01-05	1.078564	1.351742	0.813419	-0.246128
2022-01-06	0.031488	-0.132891	-0.602976	-0.110989
2022-01-07	0.836313	1.767735	-0.452541	-0.938514
2022-01-08	-0.964943	-0.665613	-0.248710	0.096500
2022-01-09	-0.347374	-0.275315	-0.564968	0.817769
2022-01-10	2.146198	0.227445	0.366495	1.291886
2022-01-11	0.045809	0.226590	-0.445373	-0.786563
2022-01-12	1.017386	-1.550262	0.308897	-0.369246
2022-01-13	0.358586	0.417658	0.886548	0.847326
2022-01-14	-0.621551	0.848705	-0.348097	-0.671176
2022-01-15	1.061288	-0.765717	1.715390	0.468456
2022-01-16	1.119669	1.251402	0.203521	1.244519
2022-01-17	-0.363791	-0.470904	0.171403	-0.472825
2022-01-18	0.340325	-1.526882	0.285516	0.144267
2022-01-19	-0.410259	0.400678	1.087062	-1.059641
2022-01-20	-0.530977	2.066283	0.250032	0.172869

In [ ]:

```python
### using Axis=1 data Columns values will be in assending form
df.sort_index(axis=1,ascending=True)
```

Out[ ]:

	A	B	C	D
2022-01-01	-0.243597	-0.807238	0.546472	0.407585
2022-01-02	-0.453285	0.501168	0.098791	0.060577
2022-01-03	-0.542659	-1.559297	-1.319488	-0.104209
2022-01-04	0.603553	-1.062545	-1.369933	-0.771599
2022-01-05	1.078564	1.351742	0.813419	-0.246128
2022-01-06	0.031488	-0.132891	-0.602976	-0.110989
2022-01-07	0.836313	1.767735	-0.452541	-0.938514
2022-01-08	-0.964943	-0.665613	-0.248710	0.096500
2022-01-09	-0.347374	-0.275315	-0.564968	0.817769
2022-01-10	2.146198	0.227445	0.366495	1.291886
2022-01-11	0.045809	0.226590	-0.445373	-0.786563
2022-01-12	1.017386	-1.550262	0.308897	-0.369246
2022-01-13	0.358586	0.417658	0.886548	0.847326
2022-01-14	-0.621551	0.848705	-0.348097	-0.671176
2022-01-15	1.061288	-0.765717	1.715390	0.468456
2022-01-16	1.119669	1.251402	0.203521	1.244519
2022-01-17	-0.363791	-0.470904	0.171403	-0.472825
2022-01-18	0.340325	-1.526882	0.285516	0.144267
2022-01-19	-0.410259	0.400678	1.087062	-1.059641
2022-01-20	-0.530977	2.066283	0.250032	0.172869

Sort by Specific Columns

In [ ]:

```python
df.sort_values(by="B",)
# df.sort_values(by="B",ascending=False)
```

Out[ ]:

	A	B	C	D
2022-01-03	-0.542659	-1.559297	-1.319488	-0.104209
2022-01-12	1.017386	-1.550262	0.308897	-0.369246
2022-01-18	0.340325	-1.526882	0.285516	0.144267
2022-01-04	0.603553	-1.062545	-1.369933	-0.771599
2022-01-01	-0.243597	-0.807238	0.546472	0.407585
2022-01-15	1.061288	-0.765717	1.715390	0.468456
2022-01-08	-0.964943	-0.665613	-0.248710	0.096500
2022-01-17	-0.363791	-0.470904	0.171403	-0.472825
2022-01-09	-0.347374	-0.275315	-0.564968	0.817769
2022-01-06	0.031488	-0.132891	-0.602976	-0.110989
2022-01-11	0.045809	0.226590	-0.445373	-0.786563
2022-01-10	2.146198	0.227445	0.366495	1.291886
2022-01-19	-0.410259	0.400678	1.087062	-1.059641
2022-01-13	0.358586	0.417658	0.886548	0.847326
2022-01-02	-0.453285	0.501168	0.098791	0.060577
2022-01-14	-0.621551	0.848705	-0.348097	-0.671176
2022-01-16	1.119669	1.251402	0.203521	1.244519
2022-01-05	1.078564	1.351742	0.813419	-0.246128
2022-01-07	0.836313	1.767735	-0.452541	-0.938514
2022-01-20	-0.530977	2.066283	0.250032	0.172869

(Filtering)

- Selecting a Specific Column

In [ ]:

```python
df["B"]
```

Out[ ]:

```
2022-01-01   -0.807238
2022-01-02    0.501168
2022-01-03   -1.559297
2022-01-04   -1.062545
2022-01-05    1.351742
2022-01-06   -0.132891
2022-01-07    1.767735
2022-01-08   -0.665613
2022-01-09   -0.275315
2022-01-10    0.227445
2022-01-11    0.226590
2022-01-12   -1.550262
2022-01-13    0.417658
2022-01-14    0.848705
2022-01-15   -0.765717
2022-01-16    1.251402
2022-01-17   -0.470904
2022-01-18   -1.526882
2022-01-19    0.400678
2022-01-20    2.066283
Freq: D, Name: B, dtype: float64
```

Row Wise Selection

In [ ]:

```python
df[10:]
```

Out[ ]:

	A	B	C	D
2022-01-11	0.045809	0.226590	-0.445373	-0.786563
2022-01-12	1.017386	-1.550262	0.308897	-0.369246
2022-01-13	0.358586	0.417658	0.886548	0.847326
2022-01-14	-0.621551	0.848705	-0.348097	-0.671176
2022-01-15	1.061288	-0.765717	1.715390	0.468456
2022-01-16	1.119669	1.251402	0.203521	1.244519
2022-01-17	-0.363791	-0.470904	0.171403	-0.472825
2022-01-18	0.340325	-1.526882	0.285516	0.144267
2022-01-19	-0.410259	0.400678	1.087062	-1.059641
2022-01-20	-0.530977	2.066283	0.250032	0.172869

LOC [] & iloc[] Function

- The main distinction between loc and iloc is: loc is label-based, which means that you have to specify rows and columns based on their row and column labels. iloc is integer position-based, so you have to specify rows and columns by their integer position values (0-based integer position)

loc()

In [ ]:

```python
# Getteing Index Wise Data/Row wise data
df.loc[dates[0]]
```

Out[ ]:

```
A   -0.243597
B   -0.807238
C    0.546472
D    0.407585
Name: 2022-01-01 00:00:00, dtype: float64
```

In [ ]:

```python
# Selecting Columns using Labels.
df.loc[:,["A","B"]]
```

Out[ ]:

	A	B
2022-01-01	-0.243597	-0.807238
2022-01-02	-0.453285	0.501168
2022-01-03	-0.542659	-1.559297
2022-01-04	0.603553	-1.062545
2022-01-05	1.078564	1.351742
2022-01-06	0.031488	-0.132891
2022-01-07	0.836313	1.767735
2022-01-08	-0.964943	-0.665613
2022-01-09	-0.347374	-0.275315
2022-01-10	2.146198	0.227445
2022-01-11	0.045809	0.226590
2022-01-12	1.017386	-1.550262
2022-01-13	0.358586	0.417658
2022-01-14	-0.621551	0.848705
2022-01-15	1.061288	-0.765717
2022-01-16	1.119669	1.251402
2022-01-17	-0.363791	-0.470904
2022-01-18	0.340325	-1.526882
2022-01-19	-0.410259	0.400678
2022-01-20	-0.530977	2.066283

Getting a specific data using index & lebels Name

In [ ]:

```python
df.loc["20220102":"20220106",["A","B"]]
```

Out[ ]:

	A	B
2022-01-02	-0.453285	0.501168
2022-01-03	-0.542659	-1.559297
2022-01-04	0.603553	-1.062545
2022-01-05	1.078564	1.351742
2022-01-06	0.031488	-0.132891

df.at use for

- Access a single value for a row/column label pair. Similar to loc, in that both provide label-based lookups. Use at if you only need to get or set a single value in a DataFrame or Series.

In [ ]:

```python
# What is inside Dates index 3 and column B
df.at[dates[3],"B"]
```

Out[ ]:

```
-1.0625453469570858
```

iLoc[ ]

In [ ]:

```python
# implicity
df.iloc[0:5, :]
```

Out[ ]:

	A	B	C	D
2022-01-01	-0.243597	-0.807238	0.546472	0.407585
2022-01-02	-0.453285	0.501168	0.098791	0.060577
2022-01-03	-0.542659	-1.559297	-1.319488	-0.104209
2022-01-04	0.603553	-1.062545	-1.369933	-0.771599
2022-01-05	1.078564	1.351742	0.813419	-0.246128

In [ ]:

```python
df.iloc[:, 0:2]
```

Out[ ]:

	A	B
2022-01-01	-0.243597	-0.807238
2022-01-02	-0.453285	0.501168
2022-01-03	-0.542659	-1.559297
2022-01-04	0.603553	-1.062545
2022-01-05	1.078564	1.351742
2022-01-06	0.031488	-0.132891
2022-01-07	0.836313	1.767735
2022-01-08	-0.964943	-0.665613
2022-01-09	-0.347374	-0.275315
2022-01-10	2.146198	0.227445
2022-01-11	0.045809	0.226590
2022-01-12	1.017386	-1.550262
2022-01-13	0.358586	0.417658
2022-01-14	-0.621551	0.848705
2022-01-15	1.061288	-0.765717
2022-01-16	1.119669	1.251402
2022-01-17	-0.363791	-0.470904
2022-01-18	0.340325	-1.526882
2022-01-19	-0.410259	0.400678
2022-01-20	-0.530977	2.066283

In [ ]:

```python
#Selection or Fillteration
df[df["A","B"]>1.5]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, tolerance)
   3360             try:
-> 3361                 return self._engine.get_loc(casted_key)
   3362             except KeyError as err:

~\anaconda3\lib\site-packages\pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

~\anaconda3\lib\site-packages\pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: ('A', 'B')

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
C:\Users\M6205~1.TAN\AppData\Local\Temp/ipykernel_3688/1066112665.py in <module>
      1 #Selection or Fillteration
----> 2 df[df["A","B"]>1.5]

~\anaconda3\lib\site-packages\pandas\core\frame.py in __getitem__(self, key)
   3456             if self.columns.nlevels > 1:
   3457                 return self._getitem_multilevel(key)
-> 3458             indexer = self.columns.get_loc(key)
   3459             if is_integer(indexer):
   3460                 indexer = [indexer]

~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, tolerance)
   3361                 return self._engine.get_loc(casted_key)
   3362             except KeyError as err:
-> 3363                 raise KeyError(key) from err
   3364 
   3365         if is_scalar(key) and isna(key) and not self.hasnans:

KeyError: ('A', 'B')
```

In [ ]:

```python
#Select those values which are greater then 0 in entire data frame
df[df>0]
```

Out[ ]:

	A	B	C	D
2022-01-01	NaN	NaN	0.546472	0.407585
2022-01-02	NaN	0.501168	0.098791	0.060577
2022-01-03	NaN	NaN	NaN	NaN
2022-01-04	0.603553	NaN	NaN	NaN
2022-01-05	1.078564	1.351742	0.813419	NaN
2022-01-06	0.031488	NaN	NaN	NaN
2022-01-07	0.836313	1.767735	NaN	NaN
2022-01-08	NaN	NaN	NaN	0.096500
2022-01-09	NaN	NaN	NaN	0.817769
2022-01-10	2.146198	0.227445	0.366495	1.291886
2022-01-11	0.045809	0.226590	NaN	NaN
2022-01-12	1.017386	NaN	0.308897	NaN
2022-01-13	0.358586	0.417658	0.886548	0.847326
2022-01-14	NaN	0.848705	NaN	NaN
2022-01-15	1.061288	NaN	1.715390	0.468456
2022-01-16	1.119669	1.251402	0.203521	1.244519
2022-01-17	NaN	NaN	0.171403	NaN
2022-01-18	0.340325	NaN	0.285516	0.144267
2022-01-19	NaN	0.400678	1.087062	NaN
2022-01-20	NaN	2.066283	0.250032	0.172869

is in Methods

In [ ]:

```python
df2 = df.copy()
df2["E"] = ["one","Two","Three","Four","five",
"one","Two","Three","Four","five",
            "one","Two","Three","Four","five",
            "one","Two","Three","Four","five"]
df2
```

Out[ ]:

	A	B	C	D	E
2022-01-01	-0.243597	-0.807238	0.546472	0.407585	one
2022-01-02	-0.453285	0.501168	0.098791	0.060577	Two
2022-01-03	-0.542659	-1.559297	-1.319488	-0.104209	Three
2022-01-04	0.603553	-1.062545	-1.369933	-0.771599	Four
2022-01-05	1.078564	1.351742	0.813419	-0.246128	five
2022-01-06	0.031488	-0.132891	-0.602976	-0.110989	one
2022-01-07	0.836313	1.767735	-0.452541	-0.938514	Two
2022-01-08	-0.964943	-0.665613	-0.248710	0.096500	Three
2022-01-09	-0.347374	-0.275315	-0.564968	0.817769	Four
2022-01-10	2.146198	0.227445	0.366495	1.291886	five
2022-01-11	0.045809	0.226590	-0.445373	-0.786563	one
2022-01-12	1.017386	-1.550262	0.308897	-0.369246	Two
2022-01-13	0.358586	0.417658	0.886548	0.847326	Three
2022-01-14	-0.621551	0.848705	-0.348097	-0.671176	Four
2022-01-15	1.061288	-0.765717	1.715390	0.468456	five
2022-01-16	1.119669	1.251402	0.203521	1.244519	one
2022-01-17	-0.363791	-0.470904	0.171403	-0.472825	Two
2022-01-18	0.340325	-1.526882	0.285516	0.144267	Three
2022-01-19	-0.410259	0.400678	1.087062	-1.059641	Four
2022-01-20	-0.530977	2.066283	0.250032	0.172869	five

In [ ]:

```python
df2["new"] = [1.2,2.2,3.2,4.4,5.5,1.2,2.2,3.2,4.4,5.5,
1.2,2.2,3.2,4.4,5.5,1.2,2.2,3.2,4.4,5.5]
df2
```

In [ ]:

```python
df2["mean"] = [1.2,2.2,3.2,4.4,5.5,1.2,2.2,3.2,4.4,5.5,
1.2,2.2,3.2,4.4,5.5,1.2,2.2,3.2,4.4,5.5]
df2
```

Out[ ]:

	A	B	C	D	E	new	mean
2022-01-01	-0.243597	-0.807238	0.546472	0.407585	one	1.2	1.2
2022-01-02	-0.453285	0.501168	0.098791	0.060577	Two	2.2	2.2
2022-01-03	-0.542659	-1.559297	-1.319488	-0.104209	Three	3.2	3.2
2022-01-04	0.603553	-1.062545	-1.369933	-0.771599	Four	4.4	4.4
2022-01-05	1.078564	1.351742	0.813419	-0.246128	five	5.5	5.5
2022-01-06	0.031488	-0.132891	-0.602976	-0.110989	one	1.2	1.2
2022-01-07	0.836313	1.767735	-0.452541	-0.938514	Two	2.2	2.2
2022-01-08	-0.964943	-0.665613	-0.248710	0.096500	Three	3.2	3.2
2022-01-09	-0.347374	-0.275315	-0.564968	0.817769	Four	4.4	4.4
2022-01-10	2.146198	0.227445	0.366495	1.291886	five	5.5	5.5
2022-01-11	0.045809	0.226590	-0.445373	-0.786563	one	1.2	1.2
2022-01-12	1.017386	-1.550262	0.308897	-0.369246	Two	2.2	2.2
2022-01-13	0.358586	0.417658	0.886548	0.847326	Three	3.2	3.2
2022-01-14	-0.621551	0.848705	-0.348097	-0.671176	Four	4.4	4.4
2022-01-15	1.061288	-0.765717	1.715390	0.468456	five	5.5	5.5
2022-01-16	1.119669	1.251402	0.203521	1.244519	one	1.2	1.2
2022-01-17	-0.363791	-0.470904	0.171403	-0.472825	Two	2.2	2.2
2022-01-18	0.340325	-1.526882	0.285516	0.144267	Three	3.2	3.2
2022-01-19	-0.410259	0.400678	1.087062	-1.059641	Four	4.4	4.4
2022-01-20	-0.530977	2.066283	0.250032	0.172869	five	5.5	5.5

Taking Row Wise Mean using diferent collumns then putting it into specific column

In [ ]:

```python
df2["mean"]=df2[["A","B","C","D","new"]].mean(axis=1)
df2
```

Out[ ]:

	A	B	C	D	E	new	mean
2022-01-01	-0.243597	-0.807238	0.546472	0.407585	one	1.2	0.220644
2022-01-02	-0.453285	0.501168	0.098791	0.060577	Two	2.2	0.481450
2022-01-03	-0.542659	-1.559297	-1.319488	-0.104209	Three	3.2	-0.065131
2022-01-04	0.603553	-1.062545	-1.369933	-0.771599	Four	4.4	0.359895
2022-01-05	1.078564	1.351742	0.813419	-0.246128	five	5.5	1.699519
2022-01-06	0.031488	-0.132891	-0.602976	-0.110989	one	1.2	0.076926
2022-01-07	0.836313	1.767735	-0.452541	-0.938514	Two	2.2	0.682599
2022-01-08	-0.964943	-0.665613	-0.248710	0.096500	Three	3.2	0.283447
2022-01-09	-0.347374	-0.275315	-0.564968	0.817769	Four	4.4	0.806022
2022-01-10	2.146198	0.227445	0.366495	1.291886	five	5.5	1.906405
2022-01-11	0.045809	0.226590	-0.445373	-0.786563	one	1.2	0.048093
2022-01-12	1.017386	-1.550262	0.308897	-0.369246	Two	2.2	0.321355
2022-01-13	0.358586	0.417658	0.886548	0.847326	Three	3.2	1.142024
2022-01-14	-0.621551	0.848705	-0.348097	-0.671176	Four	4.4	0.721576
2022-01-15	1.061288	-0.765717	1.715390	0.468456	five	5.5	1.595883
2022-01-16	1.119669	1.251402	0.203521	1.244519	one	1.2	1.003822
2022-01-17	-0.363791	-0.470904	0.171403	-0.472825	Two	2.2	0.212777
2022-01-18	0.340325	-1.526882	0.285516	0.144267	Three	3.2	0.488645
2022-01-19	-0.410259	0.400678	1.087062	-1.059641	Four	4.4	0.883568
2022-01-20	-0.530977	2.066283	0.250032	0.172869	five	5.5	1.491641

In [ ]: