untitled3-1

May 30, 2023

[ ]:
```
pip install numpy
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages
(1.22.4)
```

[ ]:
```
import numpy as np
```

1 D ARRAY

[ ]:
```
a=np.array([1,2,3])
a
```

[ ]:
```
array([1, 2, 3])
```

[ ]:
```
a.shape
```

[ ]:
```
(3,)
```

[ ]:
```
len(a)
```

[ ]:
```
3
```

[ ]:
```
a.ndim
```

[ ]:
```
1
```

[ ]:
```
a.size
```

[ ]:
```
3
```

[ ]:
```
a.dtype
```

[ ]:
```
dtype('int64')
```

[ ]:
```
np.zeros(5)
```

[ ]:
```
array([0., 0., 0., 0., 0.])
```

[ ]:
```
b=np.ones(9)
b
```

[ ]:
```
array([1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

[ ]:
```
c=np.arange(3,6,9)
c
```

[ ]:
```
array([3])
```

[ ]:
```
d=np.linspace(4,5,3)
d
```

[ ]:
```
array([4. , 4.5, 5. ])
```

[ ]:
```
ARITHMETIC OPERATION
```

ADDITION

[ ]:
```
a=np.array([3,4,5])
b=np.array([6,7,8])
a+b
```

[ ]:
```
array([ 9, 11, 13])
```

SUBTRACTION

[ ]:
```
a=np.array([3,4,5])
b=np.array([6,7,8])
a-b
```

[ ]:
```
array([-3, -3, -3])
```

MULTIPLICATION

[ ]:
```
a=np.array([3,4,5])
b=np.array([6,7,8])
a*b
```

[ ]:
```
array([18, 28, 40])
```

DIVISION

[ ]:
```
a=np.array([3,4,5])
b=np.array([6,7,8])
a/b
```

[ ]:
```
array([0.5       , 0.57142857, 0.625     ])
```

EXPONENTIAL

[8]:
```
b=np.array([1,2,3])
np.sqrt(b)
```

[8]:
```
array([1.        , 1.41421356, 1.73205081])
```

AGGREGATE FUNCTION

[10]:
```
a=np.array([1,2,3])
a.sum()
```

[10]:
```
6
```

[11]:
```
a.min()
```

[11]:
```
1
```

[12]:
```
a.max()
```

[12]:
```
3
```

[13]:
```
a.mean()
```

[13]:
```
2.0
```

[14]:
```
np.std(a)
```

[14]:
```
0.816496580927726
```

####2 ARRAY

[15]:
```
import numpy as np
```

[20]:
```
a=np.array([[1,2,3],[3,4,5]])
a.shape
```

[20]:
```
(2, 3)
```

[21]:
```
a
```

[21]:
```
array([[1, 2, 3],
       [3, 4, 5]])
```

[22]:
```
len(a)
```

[22]:
```
2
```

[24]:
```
a.ndim
```

[24]:
```
2
```

[25]:
```
a.size
```

[25]:
```
6
```

[30]:
```
a.dtype
```

[30]:
```
dtype('int64')
```

CREATE AN ARRAY OF ONE

[33]:
```
a1=np.ones(5)
a1
```

[33]:
```
array([1., 1., 1., 1., 1.])
```

[34]:
```
b=np.ones(5)
b
```

[34]:
```
array([1., 1., 1., 1., 1.])
```

[37]:
```
c=np.arange(3,4,5)
c
```

[37]:
```
array([3])
```

[38]:
```
d=np.linspace(3,4,5)
d
```

[38]:
```
array([3.  , 3.25, 3.5 , 3.75, 4.  ])
```

ARITHMETIC OPERATION

[39]:
```
a=np.array([[1,2,3],[4,5,6]])
b=np.array([[5,6,7],[7,8,9]])
a+b
```

[39]:
```
array([[ 6,  8, 10],
       [11, 13, 15]])
```

SUBTRACTION

[41]:
```
a-b
```

[41]:
```
array([[-4, -4, -4],
       [-3, -3, -3]])
```

MULTIPLICATION

[42]:
```
a*b
```

[42]:
```
array([[ 5, 12, 21],
       [28, 40, 54]])
```

DIVISION

[43]:
```
a/b
```

[43]:
```
array([[0.2       , 0.33333333, 0.42857143],
       [0.57142857, 0.625     , 0.66666667]])
```

EXPONENTIAL

[44]:
```
np.exp(b)
```

[44]:
```
array([[ 148.4131591 ,  403.42879349, 1096.63315843],
       [1096.63315843, 2980.95798704, 8103.08392758]])
```

SQUARE ROOT

[45]:
```
np.sqrt(b)
```

[45]:
```
array([[2.23606798, 2.44948974, 2.64575131],
       [2.64575131, 2.82842712, 3.        ]])
```

AGGREGATE FUNCTION

[46]:
```
a.sum()
```

[46]:
```
21
```

[47]:
```
a.min()
```

[47]:
```
1
```

[48]:
```
a.min()
```

[48]:
```
1
```

[49]:
```
a.max()
```

[49]:
```
6
```

[50]:
```
a.mean()
```

[50]:
```
3.5
```

[51]:
```
np.std(a)
```

[51]:
```
1.707825127659933
```

3 D ARRAY

[52]:
```
import numpy as np
```

[53]:
```
a=np.array([[[1,2,3],[4,5,6,],[7,8,9]]])
a
```

[53]:
```
array([[[1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]]])
```

[54]:
```
a.shape
```

[54]:
```
(1, 3, 3)
```

[55]:
```
len(a)
```

[55]:
```
1
```

[56]:
```
a.ndim
```

[56]:
```
3
```

[57]:
```
a.size
```

[57]:
```
9
```

[58]:
```
a.dtype
```

[58]:
```
dtype('int64')
```

[59]:
```
b=np.zeros(5)
b
```

[59]:
```
array([0., 0., 0., 0., 0.])
```

CREATE AN ARRAY OF ONE

[61]:
```
c=np.ones(5)
c
```

[61]:
```
array([1., 1., 1., 1., 1.])
```

[63]:
```
d=np.arange(4,5,6)
d
```

[63]:
```
array([4])
```

[64]:
```
e=np.linspace(4,5,6)
e
```

[64]:
```
array([4. , 4.2, 4.4, 4.6, 4.8, 5. ])
```

ARITHMETIC OPERATION

ADDITION

[65]:
```
a=np.array([[[1,2,3],[4,5,6],[7,8,9]]])
b=np.array([[[9,8,7],[6,5,4],[3,2,1]]])
a+b
```

[65]:
```
array([[[10, 10, 10],
        [10, 10, 10],
        [10, 10, 10]]])
```

SUBTRACTION

[66]:
```
a-b
```

[66]:
```
array([[[-8, -6, -4],
        [-2,  0,  2],
        [ 4,  6,  8]]])
```

MULTIPLICATION

[67]:
```
a*b
```

[67]:
```
array([[[ 9, 16, 21],
        [24, 25, 24],
        [21, 16,  9]]])
```

DIVISION

[68]:
```
a/b
```

[68]:
```
array([[[0.11111111, 0.25      , 0.42857143],
        [0.66666667, 1.        , 1.5       ],
        [2.33333333, 4.        , 9.        ]]])
```

EXPONENTIAL

[69]:
```
np.exp(b)
```

[69]:
```
array([[[8.10308393e+03, 2.98095799e+03, 1.09663316e+03],
        [4.03428793e+02, 1.48413159e+02, 5.45981500e+01],
        [2.00855369e+01, 7.38905610e+00, 2.71828183e+00]]])
```

SUARE ROOT

[70]:
```
np.sqrt(b)
```

[70]:
```
array([[[3.        , 2.82842712, 2.64575131],
        [2.44948974, 2.23606798, 2.        ],
        [1.73205081, 1.41421356, 1.        ]]])
```

AGGREGATE FUNCTION

[71]:
```
a.sum()
```

[71]:
```
45
```

[72]:
```
a.min()
```

[72]:
```
1
```

[74]:
```
a.max()
```

[74]:
```
9
```

[75]:
```
a.mean()
```

[75]:
```
5.0
```

[76]:
```
np.std(a)
```

[76]:
```
2.581988897471611
```

PANDAS:-

[77]:
```
import pandas as pd
```

[83]:
```
a = pd.Series([1,2,3,4,5,6,7,8],index=list("KJSFHSJD"))
a
```

[83]:
```
K    1
J    2
S    3
F    4
H    5
S    6
J    7
D    8
```

```
dtype: int64
```

[84]:
```
b=pd.DataFrame({"sher ali":18,"nabeel":8,"asad":9}, index=[1,2,3])
b
```

[84]:
```
   sher ali  nabeel  asad
1        18       8     9
2        18       8     9
3        18       8     9
```

WORKING ON TITANIC DATASET FROM SEABORN LIBRARY

[85]:
```
import seaborn as sns
df = sns.load_dataset("titanic")
df
```

[85]:
```
     survived  pclass     sex   age  sibsp  parch     fare embarked   class  \
0           0       3    male  22.0      1      0   7.2500        S   Third
1           1       1  female  38.0      1      0  71.2833        C   First
2           1       3  female  26.0      0      0   7.9250        S   Third
3           1       1  female  35.0      1      0  53.1000        S   First
4           0       3    male  35.0      0      0   8.0500        S   Third
..        ...     ...     ...   ...    ...    ...      ...      ...     ...
886         0       2    male  27.0      0      0  13.0000        S  Second
887         1       1  female  19.0      0      0  30.0000        S   First
888         0       3  female   NaN      1      2  23.4500        S   Third
889         1       1    male  26.0      0      0  30.0000        C   First
890         0       3    male  32.0      0      0   7.7500        Q   Third

       who  adult_male deck  embark_town alive  alone
0      man        True  NaN  Southampton    no  False
1    woman       False    C    Cherbourg   yes  False
2    woman       False  NaN  Southampton   yes   True
3    woman       False    C  Southampton   yes  False
4      man        True  NaN  Southampton    no   True
..     ...         ...  ...          ...   ...    ...
886    man        True  NaN  Southampton    no   True
887  woman       False    B  Southampton   yes   True
888  woman       False  NaN  Southampton    no  False
889    man        True    C    Cherbourg   yes   True
890    man        True  NaN   Queenstown    no   True

[891 rows x 15 columns]
```

CHECKING INFORMATION ABOUT DATA

[86]:
```
df.info
```

[86]:
```
<bound method DataFrame.info of      survived  pclass     sex   age  sibsp
parch     fare embarked   class  \
0           0       3    male  22.0      1      0   7.2500        S   Third
1           1       1  female  38.0      1      0  71.2833        C   First
2           1       3  female  26.0      0      0   7.9250        S   Third
3           1       1  female  35.0      1      0  53.1000        S   First
4           0       3    male  35.0      0      0   8.0500        S   Third
..        …     …      …   …    …     …       …      …     …
886         0       2    male  27.0      0      0  13.0000        S  Second
887         1       1  female  19.0      0      0  30.0000        S   First
888         0       3  female   NaN      1      2  23.4500        S   Third
889         1       1    male  26.0      0      0  30.0000        C   First
890         0       3    male  32.0      0      0   7.7500        Q   Third

       who  adult_male deck  embark_town alive  alone
0      man        True  NaN  Southampton    no  False
1    woman       False    C    Cherbourg   yes  False
2    woman       False  NaN  Southampton   yes   True
3    woman       False    C  Southampton   yes  False
4      man        True  NaN  Southampton    no   True
..     …          …  …          …    …    …
886    man        True  NaN  Southampton    no   True
887  woman       False    B  Southampton   yes   True
888  woman       False  NaN  Southampton    no  False
889    man        True    C    Cherbourg   yes   True
890    man        True  NaN   Queenstown    no   True

[891 rows x 15 columns]>
```

CHECK NO.OF ROWS AND COLUMNS

[87]:
```
df.shape
```

[87]:
```
(891, 15)
```

[88]:
```
df.columns
```

[88]:
```
Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare',
       'embarked', 'class', 'who', 'adult_male', 'deck', 'embark_town',
       'alive', 'alone'],
      dtype='object')
```

[89]:
```
df.index
```

[89]:
```
RangeIndex(start=0, stop=891, step=1)
```

[91]:
```
df.head()
```

[91]:
```
   survived  pclass     sex   age  sibsp  parch     fare embarked  class  \
0         0       3    male  22.0      1      0   7.2500        S  Third
1         1       1  female  38.0      1      0  71.2833        C  First
2         1       3  female  26.0      0      0   7.9250        S  Third
3         1       1  female  35.0      1      0  53.1000        S  First
4         0       3    male  35.0      0      0   8.0500        S  Third

     who  adult_male deck  embark_town alive  alone
0    man        True  NaN  Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True  NaN  Southampton    no   True
```

[92]:
```
df.tail()
```

[92]:
```
     survived  pclass     sex   age  sibsp  parch   fare embarked   class  \
886         0       2    male  27.0      0      0  13.00        S  Second
887         1       1  female  19.0      0      0  30.00        S   First
888         0       3  female   NaN      1      2  23.45        S   Third
889         1       1    male  26.0      0      0  30.00        C   First
890         0       3    male  32.0      0      0   7.75        Q   Third

       who  adult_male deck  embark_town alive  alone
886    man        True  NaN  Southampton    no   True
887  woman       False    B  Southampton   yes   True
888  woman       False  NaN  Southampton    no  False
889    man        True    C    Cherbourg   yes   True
890    man        True  NaN   Queenstown    no   True
```

BASIC STATISTIC

[93]:
```
df.describe()
```

[93]:
```
          survived      pclass         age       sibsp       parch        fare
count  891.000000  891.000000  714.000000  891.000000  891.000000  891.000000
mean     0.383838    2.308642   29.699118    0.523008    0.381594   32.204208
std      0.486592    0.836071   14.526497    1.102743    0.806057   49.693429
min      0.000000    1.000000    0.420000    0.000000    0.000000    0.000000
25%      0.000000    2.000000   20.125000    0.000000    0.000000    7.910400
50%      0.000000    3.000000   28.000000    0.000000    0.000000   14.454200
75%      1.000000    3.000000   38.000000    1.000000    0.000000   31.000000
max      1.000000    3.000000   80.000000    8.000000    6.000000  512.329200
```

REMOVNG SPECIFIC COLUMNS

[96]:
```
df9=df.drop(["fare","alone"],axis =1)
df9
```

[96]:
```
     survived  pclass     sex   age  sibsp  parch embarked   class    who  \
0           0       3    male  22.0      1      0        S   Third    man
1           1       1  female  38.0      1      0        C   First  woman
2           1       3  female  26.0      0      0        S   Third  woman
3           1       1  female  35.0      1      0        S   First  woman
4           0       3    male  35.0      0      0        S   Third    man
..        ...     ...     ...   ...    ...    ...      ...     ...    ...
886         0       2    male  27.0      0      0        S  Second    man
887         1       1  female  19.0      0      0        S   First  woman
888         0       3  female   NaN      1      2        S   Third  woman
889         1       1    male  26.0      0      0        C   First    man
890         0       3    male  32.0      0      0        Q   Third    man

     adult_male deck  embark_town alive
0          True  NaN  Southampton    no
1         False    C    Cherbourg   yes
2         False  NaN  Southampton   yes
3         False    C  Southampton   yes
4          True  NaN  Southampton    no
..          ...  ...          ...   ...
886        True  NaN  Southampton    no
887       False    B  Southampton   yes
888       False  NaN  Southampton    no
889        True    C    Cherbourg   yes
890        True  NaN   Queenstown    no

[891 rows x 13 columns]
```

GROUPING

[98]:
```
df9.groupby(["sex"]).mean()
```

```
<ipython-input-98-c1f9ad6e4b0f>:1: FutureWarning: The default value of
numeric_only in DataFrameGroupBy.mean is deprecated. In a future version,
numeric_only will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
  df9.groupby(["sex"]).mean()
```

[98]:
```
        survived    pclass        age     sibsp     parch  adult_male
sex
female  0.742038  2.159236  27.915709  0.694268  0.649682    0.000000
male    0.188908  2.389948  30.726645  0.429809  0.235702    0.930676
```

[99]:
```
df9.groupby(["sex","class"]).mean()
```

```
<ipython-input-99-7c1888990b5e>:1: FutureWarning: The default value of
numeric_only in DataFrameGroupBy.mean is deprecated. In a future version,
numeric_only will default to False. Either specify numeric_only or select only
```

```
columns which should be valid for the function.
  df9.groupby(["sex","class"]).mean()
```

[99]:

sex	class	survived	pclass	age	sibsp	parch	adult_male
female	First	0.968085	1.0	34.611765	0.553191	0.457447	0.000000
	Second	0.921053	2.0	28.722973	0.486842	0.605263	0.000000
	Third	0.500000	3.0	21.750000	0.895833	0.798611	0.000000
male	First	0.368852	1.0	41.281386	0.311475	0.278689	0.975410
	Second	0.157407	2.0	30.740707	0.342593	0.222222	0.916667
	Third	0.135447	3.0	26.507589	0.498559	0.224784	0.919308

CHECKING MISSING VALUES

[100]:
```
df.isnull().sum()
```

[100]:
```
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```