Pandas Tutorial for Data Science

pandas :

1. Series : One-Dimentinal data structure
2. Data-Frame : Multi-Dimentional data structure

```
import pandas as pd
```

```
n1 = pd.Series([10,20,30,40,50])
n1
```

```
0    10
1    20
2    30
3    40
4    50
dtype: int64
```

```
type(n1)
```

```
pandas.core.series.Series
```

```
n1 = pd.Series([11,21,31,41,51], index = ['a','b','c','d','e'])
```

```
n1
```

```
a    11
b    21
c    31
d    41
e    51
dtype: int64
```

```
pd.Series({'west Bengal' : 'kolkata', 'Maharastra' : 'Mumbai', 'Tamilr
```

```
west Bengal    kolkata
Maharastra      Mumbai
Tamilnadu      Chennai
dtype: object
```

```
n2 = pd.Series({'c1' : 'kolkata', 'c2' : 'Mumbai', 'c3' : 'Chennai'},
n2
```

```
c2     Mumbai
c4        NaN
c1    kolkata
```

```
c3    Chennai
dtype: object
```

```
n3 = pd.Series([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
```

```
n3[4]
```

```
5
```

```
n3[:7]
```

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

```
n3[-4:]
```

```
6     7
7     8
8     9
9    10
dtype: int64
```

```
n3 + 4
```

```
0     5
1     6
2     7
3     8
4     9
5    10
6    11
7    12
8    13
9    14
dtype: int64
```

```
n4 = pd.Series([10,20,30,40,50,60,70,80,90,100])
```

```
n3 + n4
```

```
0     11
1     22
2     33
3     44
4     55
5     66
6     77
```

```
7     88
8     99
9    110
dtype: int64
```

```
n4 // 5
```

```
0     2
1     4
2     6
3     8
4    10
5    12
6    14
7    16
8    18
9    20
dtype: int64
```

DataFrame in pandas

```
d1 = pd.DataFrame({'Name' : ['Sasanka', 'Raktim', 'Puronjit'], 'Marks'
```

```
d1
```

	Name	Marks
0	Sasanka	90
1	Raktim	92
2	Puronjit	91

```
dataset = pd.load_csv('california_housing_test.csv')
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-32-69e24274f530> in <module>
----> 1 dataset = pd.load_csv('california_housing_test.csv')

/usr/local/lib/python3.8/dist-packages/pandas/__init__.py in __getattr__(name)
    242         return _SparseArray
    243 
--> 244     raise AttributeError(f"module 'pandas' has no attribute '{name}'")
    245 
    246 

AttributeError: module 'pandas' has no attribute 'load_csv'
```

SEARCH STACK OVERFLOW

```
from sklearn import datasets
```

```python
iris = datasets.load_iris()
```

```python
iris.target[:10]
```

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

```python
iris
```

```
        [6.7, 3. , 5.2, 2.3],
        [6.3, 2.5, 5. , 1.9],
        [6.5, 3. , 5.2, 2. ],
        [6.2, 3.4, 5.4, 2.3],
        [5.9, 3. , 5.1, 1.8]]),
 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
        2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
        2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]),
 'frame': None,
 'target_names': array(['setosa', 'versicolor', 'virginica'], dtype='<U10'),
 'DESCR': '.. _iris_dataset:\n\nIris plants dataset\n--------------------
\n\n**Data Set Characteristics:**\n\n    :Number of Instances: 150 (50 in each of
three classes)\n    :Number of Attributes: 4 numeric, predictive attributes and
the class\n    :Attribute Information:\n        - sepal length in cm\n        -
sepal width in cm\n        - petal length in cm\n        - petal width in cm\n
- class:\n              - Iris-Setosa\n                - Iris-Versicolour\n
- Iris-Virginica\n                \n    :Summary Statistics:\n\n
============== ==== ==== ======= ===== ====================\n
Min  Max   Mean    SD   Class Correlation\n    ============== ==== ==== =======
===== ====================\n    sepal length:   4.3  7.9   5.84   0.83
0.7826\n    sepal width:    2.0  4.4   3.05   0.43   -0.4194\n    petal length:
1.0  6.9   3.76   1.76    0.9490  (high!)\n    petal width:    0.1  2.5   1.20
0.76    0.9565  (high!)\n    ============== ==== ==== ======= =====
====================\n\n    :Missing Attribute Values: None\n    :Class
Distribution: 33.3% for each of 3 classes.\n    :Creator: R.A. Fisher\n
:Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)\n    :Date: July,
1988\n\nThe famous Iris database, first used by Sir R.A. Fisher. The dataset is
taken\nfrom Fisher\'s paper. Note that it\'s the same as in R, but not as in the
UCI\nMachine Learning Repository, which has two wrong data points.\n\nThis is
perhaps the best known database to be found in the\npattern recognition
literature.  Fisher\'s paper is a classic in the field and\nis referenced
frequently to this day.  (See Duda & Hart, for example.)  The\ndata set contains
3 classes of 50 instances each, where each class refers to a\ntype of iris plant.
One class is linearly separable from the other 2; the\nlatter are NOT linearly
separable from each other.\n\n.. topic:: References\n\n   - Fisher, R.A. "The use
of multiple measurements in taxonomic problems"\n     Annual Eugenics, 7, Part
II, 179-188 (1936); also in "Contributions to\n     Mathematical Statistics"
(John Wiley, NY, 1950).\n   - Duda, R.O., & Hart, P.E. (1973) Pattern
Classification and Scene Analysis.\n     (Q327.D83) John Wiley & Sons.  ISBN 0-
471-22361-1.  See page 218.\n   - Dasarathy, B.V. (1980) "Nosing Around the
Neighborhood: A New System\n     Structure and Classification Rule for
Recognition in Partially Exposed\n     Environments".  IEEE Transactions on
Pattern Analysis and Machine\n     Intelligence, Vol. PAMI-2, No. 1, 67-71.\n   -
```

```
Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule". IEEE Transactions\n
on Information Theory, May 1972, 431-433.\n   - See also: 1988 MLC Proceedings,
54-64.  Cheeseman et al"s AUTOCLASS II\n     conceptual clustering system finds 3
classes in the data.\n   - Many, many more ...',
 'feature_names': ['sepal length (cm)',
  'sepal width (cm)',
  'petal length (cm)',
  'petal width (cm)'],
 'filename': 'iris.csv',
 'data_module': 'sklearn.datasets.data'}
```

```
import pandas as pd
```

```
cht = pd.read_csv("/content/sample_data/california_housing_test.csv")
```

```
cht
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	populati
0	-122.05	37.37	27.0	3885.0	661.0	153
1	-118.30	34.26	43.0	1510.0	310.0	80
2	-117.81	33.78	27.0	3589.0	507.0	148
3	-118.36	33.82	28.0	67.0	15.0	4
4	-119.67	36.33	19.0	1241.0	244.0	85
...	...	...	...	...	...	
2995	-119.86	34.42	23.0	1450.0	642.0	125
2996	-118.14	34.06	27.0	5257.0	1082.0	349
2997	-119.70	36.30	10.0	956.0	201.0	69
2998	-117.12	34.10	40.0	96.0	14.0	4
2999	-119.63	34.42	42.0	1765.0	263.0	75

3000 rows × 9 columns

```
cht.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population
0	-122.05	37.37	27.0	3885.0	661.0	1537.0
1	-118.30	34.26	43.0	1510.0	310.0	809.0
2	-117.81	33.78	27.0	3589.0	507.0	1484.0
3	-118.36	33.82	28.0	67.0	15.0	49.0
4	-119.67	36.33	19.0	1241.0	244.0	850.0

```
cht.tail()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	populati
2995	-119.86	34.42	23.0	1450.0	642.0	125
2996	-118.14	34.06	27.0	5257.0	1082.0	349
2997	-119.70	36.30	10.0	956.0	201.0	69
2998	-117.12	34.10	40.0	96.0	14.0	4
2999	-119.63	34.42	42.0	1765.0	263.0	75

```
cht.describe()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	po
count	3000.000000	3000.00000	3000.000000	3000.000000	3000.000000	30
mean	-119.589200	35.63539	28.845333	2599.578667	529.950667	14
std	1.994936	2.12967	12.555396	2155.593332	415.654368	10
min	-124.180000	32.56000	1.000000	6.000000	2.000000	
25%	-121.810000	33.93000	18.000000	1401.000000	291.000000	7
50%	-118.485000	34.27000	29.000000	2106.000000	437.000000	11
75%	-118.020000	37.69000	37.000000	3129.000000	636.000000	17
max	-114.490000	41.92000	52.000000	30450.000000	5419.000000	119

```
cht.head().iloc[1:3, 0:2]
```

	longitude	latitude
1	-118.30	34.26
2	-117.81	33.78

```
cht.loc[:3,('latitude', 'total_bedrooms', 'population')]
```

latitude total_bedrooms population

```
cht.head(7)
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population
0	-122.05	37.37	27.0	3885.0	661.0	1537.0
1	-118.30	34.26	43.0	1510.0	310.0	809.0
2	-117.81	33.78	27.0	3589.0	507.0	1484.0
3	-118.36	33.82	28.0	67.0	15.0	49.0
4	-119.67	36.33	19.0	1241.0	244.0	850.0
5	-119.56	36.51	37.0	1018.0	213.0	663.0
6	-121.43	38.63	43.0	1009.0	225.0	604.0

```
new_set = cht.drop([2,4,7], axis = 0)
```

```
new_set.head(7)
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population
0	-122.05	37.37	27.0	3885.0	661.0	1537.0
1	-118.30	34.26	43.0	1510.0	310.0	809.0
3	-118.36	33.82	28.0	67.0	15.0	49.0
5	-119.56	36.51	37.0	1018.0	213.0	663.0
6	-121.43	38.63	43.0	1009.0	225.0	604.0
8	-122.84	38.40	15.0	3080.0	617.0	1446.0
9	-118.02	34.08	31.0	2402.0	632.0	2830.0

```
new_set = cht.drop('total_rooms', axis = 1)
```

```
new_set.iloc[:4]
```

	longitude	latitude	housing_median_age	total_bedrooms	population	households
0	-122.05	37.37	27.0	661.0	1537.0	606.0
1	-118.30	34.26	43.0	310.0	809.0	277.0
2	-117.81	33.78	27.0	507.0	1484.0	495.0
3	-118.36	33.82	28.0	15.0	49.0	11.0

cht.min()

```
longitude              -124.1800
latitude                 32.5600
housing_median_age        1.0000
total_rooms               6.0000
total_bedrooms            2.0000
population                5.0000
households                2.0000
median_income             0.4999
median_house_value    22500.0000
dtype: float64
```

cht.max()

```
longitude              -114.4900
latitude                 41.9200
housing_median_age       52.0000
total_rooms           30450.0000
total_bedrooms         5419.0000
population            11935.0000
households             4930.0000
median_income            15.0001
median_house_value   500001.0000
dtype: float64
```

cht.median()

```
longitude              -118.48500
latitude                 34.27000
housing_median_age       29.00000
total_rooms            2106.00000
total_bedrooms          437.00000
population             1155.00000
households              409.50000
median_income             3.48715
median_house_value   177650.00000
dtype: float64
```

cht.mean()

```
longitude              -119.589200
latitude                 35.635390
housing_median_age       28.845333
total_rooms            2599.578667
total_bedrooms          529.950667
population             1402.798667
households              489.912000
median_income             3.807272
median_house_value   205846.275000
dtype: float64
```

cht.sum()

```
longitude            -3.587676e+05
```

```
latitude              1.069062e+05
housing_median_age    8.653600e+04
total_rooms           7.798736e+06
total_bedrooms        1.589852e+06
population            4.208396e+06
households            1.469736e+06
median_income         1.142182e+04
median_house_value    6.175388e+08
dtype: float64
```

```
cht.keys()
```

```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value'],
      dtype='object')
```

```
def oneFourth(val):
  return val * 0.25
```

```
cht[['latitude','households']].apply(oneFourth).head()
```

	latitude	households
0	9.3425	151.50
1	8.5650	69.25
2	8.4450	123.75
3	8.4550	2.75
4	9.0825	59.25

```
cht['households'].value_counts()
```

```
273.0     12
375.0     12
614.0     12
363.0     11
287.0     11
          ..
685.0      1
89.0       1
973.0      1
802.0      1
1036.0     1
Name: households, Length: 1026, dtype: int64
```

```
cht.sort_values(by = 'population').tail(10)
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	populati
1283	-117.18	32.92	4.0	15025.0	2616.0	756
947	-117.23	33.91	9.0	11654.0	2100.0	759
2014	-117.22	32.86	4.0	16289.0	4585.0	760
33	-118.08	34.55	5.0	16181.0	2971.0	815
321	-121.73	37.68	17.0	20354.0	3493.0	876
1597	-117.12	33.49	4.0	21988.0	4055.0	882
2429	-117.20	33.58	2.0	30450.0	5033.0	941
1146	-117.27	33.15	4.0	23915.0	4135.0	1087
2186	-116.14	34.45	12.0	8796.0	1721.0	1113