Pandas

What is Pandas?

- Pandas is a tool for data processing which helps in data analysis.
- It provides functions and methods to efficiently manipulate large dataset.
- Data Structure in Pandas: Series(one dimentional array) and DataFrame(two dimentional array).
- In Pandas the data can be of any datatype.

1. Pandas DataFrame
2. Pandas Series
3. Pandas Basic Operation

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
np.arange(0,20).reshape(5,4)
```

Out[2]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15],
       [16, 17, 18, 19]])
```

create a DataFrame

In [10]:
```
df=pd.DataFrame(np.arange(0,20).reshape(5,4),index=['Row1','Row2','Row3','
                columns=['Column1','Column2','Column3','Column4'])
```

In [12]:
```
type(df)
```

Out[12]:
```
pandas.core.frame.DataFrame
```

In [13]:
```
df
```

Out[13]:

	Column1	Column2	Column3	Column4
Row1	0	1	2	3
Row2	4	5	6	7
Row3	8	9	10	11
Row4	12	13	14	15
Row5	16	17	18	19

In [14]:
```
df.head()
```

Out[14]:

	Column1	Column2	Column3	Column4
Row1	0	1	2	3
Row2	4	5	6	7
Row3	8	9	10	11
Row4	12	13	14	15
Row5	16	17	18	19

In [16]:
```
df.tail(3)
```

Out[16]:

	Column1	Column2	Column3	Column4
Row3	8	9	10	11
Row4	12	13	14	15
Row5	16	17	18	19

In [17]:
```
df.describe()
```

Out[17]:

	Column1	Column2	Column3	Column4
count	5.000000	5.000000	5.000000	5.000000
mean	8.000000	9.000000	10.000000	11.000000
std	6.324555	6.324555	6.324555	6.324555
min	0.000000	1.000000	2.000000	3.000000
25%	4.000000	5.000000	6.000000	7.000000
50%	8.000000	9.000000	10.000000	11.000000
75%	12.000000	13.000000	14.000000	15.000000
max	16.000000	17.000000	18.000000	19.000000

In [18]:
```
# i just want to read 3 column col1,col2,col3

df[['Column1','Column2','Column3']]
```

Out[18]:

	Column1	Column2	Column3
Row1	0	1	2
Row2	4	5	6
Row3	8	9	10
Row4	12	13	14
Row5	16	17	18

In [19]:
```
# for 1 column
df[['Column1']]
```

Out[19]:

	Column1
Row1	0
Row2	4
Row3	8
Row4	12
Row5	16

In [20]:
```
type(df[['Column1']])
```

Out[20]:
```
pandas.core.frame.DataFrame
```

In [21]:
```
df['Column1']
```

Out[21]:
```
Row1     0
Row2     4
Row3     8
Row4    12
Row5    16
Name: Column1, dtype: int32
```

In [22]:
```
# This Bascially called as series

type(df['Column1'])
```

Out[22]:
```
pandas.core.series.Series
```

In [23]:
```
df
```

Out[23]:

	Column1	Column2	Column3	Column4
Row1	0	1	2	3
Row2	4	5	6	7
Row3	8	9	10	11
Row4	12	13	14	15
Row5	16	17	18	19

In [24]:
```
# Retrieve data with respect to row index
# wenever u write location(loc) which mean u can actually give your row ind

df.loc[['Row1','Row3']]
```

Out[24]:

	Column1	Column2	Column3	Column4
Row1	0	1	2	3
Row3	8	9	10	11

In [28]:
```
# Another way
# when ever u want to retrieve record with the help of row no at that time
# iloc basically means index location

df.iloc[0:3]
```

Out[28]:

	Column1	Column2	Column3	Column4
Row1	0	1	2	3
Row2	4	5	6	7
Row3	8	9	10	11

In [29]:
```
df.iloc[0:3,0:2]
```

Out[29]:

	Column1	Column2
Row1	0	1
Row2	4	5
Row3	8	9

In [30]:
```
df.iloc[2:4,1:3]
```

Out[30]:

	Column2	Column3
Row3	9	10
Row4	13	14

In [31]:
```
# Convert datafram into array
# put values in the last

df.iloc[2:4,1:3].values
```

Out[31]:
```
array([[ 9, 10],
       [13, 14]])
```

In [33]:
```
# nan basically mean null value

np.nan
```

Out[33]:
```
nan
```

In [39]:
```
df=pd.DataFrame(data=[[1,np.nan,2],[1,2,3],[5,4,3]],index =['Row1',
                                                            'Row2','Row3'],colu



df
```

Out[39]:

	Column1	Column2	Column3
Row1	1	NaN	2
Row2	1	2.0	3
Row3	5	4.0	3

In [40]:
```
# For checking null value

df.isnull()
```

Out[40]:

	Column1	Column2	Column3
Row1	False	True	False
Row2	False	False	False
Row3	False	False	False

In [41]:
```
# it will sum all the null value
df.isnull().sum()
```

Out[41]:
```
Column1    0
Column2    1
Column3    0
dtype: int64
```

In [42]:
```
df.isna().sum()
```

Out[42]:
```
Column1    0
Column2    1
Column3    0
dtype: int64
```

In [43]:
```
df
```

Out[43]:

	Column1	Column2	Column3
Row1	1	NaN	2
Row2	1	2.0	3
Row3	5	4.0	3

In [44]:
```
# i want to see what eachn and every column data is about

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 3 entries, Row1 to Row3
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   Column1  3 non-null      int64  
 1   Column2  2 non-null      float64
 2   Column3  3 non-null      int64  
dtypes: float64(1), int64(2)
memory usage: 96.0+ bytes
```

In [45]:
```
df.head()
```

Out[45]:

	Column1	Column2	Column3
Row1	1	NaN	2
Row2	1	2.0	3
Row3	5	4.0	3

In [49]:
```
# to check unique value(value_counts in series)

df['Column3'].value_counts()
```

Out[49]:
```
Column3
3    2
2    1
Name: count, dtype: int64
```

In [51]:
```
df['Column1'].unique()
```

Out[51]:
```
array([1, 5], dtype=int64)
```

In [52]:
```
df = pd.DataFrame({'A':[1,2,3], 'B':[1,1,1]})
```

In [53]:
```
df
```

Out[53]:

	A	B
0	1	1
1	2	1
2	3	1

In [54]:
```
df['B'].unique()
```

Out[54]:
```
array([1], dtype=int64)
```

In [56]:
```
# df.n unique say no of unique value

df.nunique()
```

Out[56]:
```
A    3
B    1
dtype: int64
```

In [57]:
```
# if we put axis = 0 which mean i going to see column wise

df.nunique(axis=0)
```

Out[57]:
```
A    3
B    1
dtype: int64
```

In [58]:
```
df.nunique(axis=1)
```

Out[58]:
```
0    1
1    2
2    2
dtype: int64
```

In [59]:
```
df=pd.DataFrame(np.arange(0,20).reshape(5,4),index=['Row1','Row2','Row3','
                columns=['Column1','Column2','Column3','Column4'])
```

In [61]:
```
df.head()
```

Out[61]:

	Column1	Column2	Column3	Column4
Row1	0	1	2	3
Row2	4	5	6	7
Row3	8	9	10	11
Row4	12	13	14	15
Row5	16	17	18	19

In [62]:
```
df['Column2']>5
```

Out[62]:
```
Row1    False
Row2    False
Row3     True
Row4     True
Row5     True
Name: Column2, dtype: bool
```

In [64]:
```
# df on top of df means take off all row greater than 5

df[df['Column2']>5]
```

Out[64]:

	Column1	Column2	Column3	Column4
Row3	8	9	10	11
Row4	12	13	14	15
Row5	16	17	18	19

In [65]:
```
df.to_csv('test.csv')
```

In [68]:
```
df1=pd.read_csv('test.csv')
df1
```

Out[68]:

	Unnamed: 0	Column1	Column2	Column3	Column4
0	Row1	0	1	2	3
1	Row2	4	5	6	7
2	Row3	8	9	10	11
3	Row4	12	13	14	15
4	Row5	16	17	18	19

In [70]:
```
# for removing unnamed row write index = false
df.to_csv('test.csv',index=False)
```

In [71]:
```
df1=pd.read_csv('test.csv')
df1
```

Out[71]:

	Column1	Column2	Column3	Column4
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15
4	16	17	18	19

In [78]:
```
df.to_excel('test.xlsx',index=False)
```

In [79]:
```
pd.read_excel('test.xlsx')
```

Out[79]:

	Column1	Column2	Column3	Column4
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15
4	16	17	18	19

In [ ]:
```
pd.read_csv('https://download.bls.gov/pub/time.series/cu/cu.item',sep='\t'
```

In [91]:
```
data = '{"employee_name":"Kamran","email":"kamran@gmail.com","job_profile"
```

In [92]:
```
data
```

Out[92]:
```
'{"employee_name":"Kamran","email":"kamran@gmail.com","job_profile":[{"ti
tle":"ML engineer","role":"Dev"}]}'
```

In [93]:
```
type(data)
```

Out[93]:
```
str
```

In [94]:
```
df2=pd.read_json(data)
```

In [95]:
```
df2
```

Out[95]:

	employee_name	email	job_profile
0	Kamran	kamran@gmail.com	{'title': 'ML engineer', 'role': 'Dev'}

In [98]:
```
df2['salary']=90000
df2['status']='single'
```

In [99]:
```
df2
```

Out[99]:

	employee_name	email	job_profile	salary	status
0	Kamran	kamran@gmail.com	{'title': 'ML engineer', 'role': 'Dev'}	90000	single

In [106]:
```
dict(df2['job_profile'])[0]['role']="Developer"
```

In [107]:
```
df2
```

Out[107]:

	employee_name	email	job_profile	salary	status
0	Kamran	kamran@gmail.com	{'title': 'ML engineer', 'role': 'Developer'}	90000	single

In [108]:
```
data
```

Out[108]:
```
'{"employee_name":"Kamran","email":"kamran@gmail.com","job_profile":[{"ti
tle":"ML engineer","role":"Dev"}]}'
```

In [112]:
```


pd.read_json(data)
```

Out[112]:

	employee_name	email	job_profile
0	Kamran	kamran@gmail.com	{'title': 'ML engineer', 'role': 'Dev'}

In [114]:
```
### Orientation(record,index,columns)
```

In [117]:
```
# record orient(same as data)

pd.read_json(data,orient='records')
```

Out[117]:

	employee_name	email	job_profile
0	Kamran	kamran@gmail.com	{'title': 'ML engineer', 'role': 'Dev'}

In [115]:
```
# index orient

pd.read_json(data,orient='index')
```

Out[115]:

	0
employee_name	Kamran
email	kamran@gmail.com
job_profile	[{'title': 'ML engineer', 'role': 'Dev'}]

In [118]:
```
# Column orient(same as data)

pd.read_json(data,orient='columns')
```

Out[118]:

	employee_name	email	job_profile
0	Kamran	kamran@gmail.com	{'title': 'ML engineer', 'role': 'Dev'}

In [119]:
```
import pandas as pd
df = pd.DataFrame([['a','b'],['c','d']],
                  index=['row 1','row 2'],
                  columns=['col 1','col 2'])
```

In [120]:
```
df
```

Out[120]:

	col 1	col 2
row 1	a	b
row 2	c	d

In [121]:
```
df.to_json()
```

Out[121]:
```
'{"col 1":{"row 1":"a","row 2":"c"},"col 2":{"row 1":"b","row 2":"d"}}'
```

In [123]:
```
df.to_json(orient='records')
```

Out[123]:
```
'[{"col 1":"a","col 2":"b"},{"col 1":"c","col 2":"d"}]'
```

In [124]:
```
df.to_json(orient='columns')
```

Out[124]:
```
'{"col 1":{"row 1":"a","row 2":"c"},"col 2":{"row 1":"b","row 2":"d"}}'
```

In [128]:
```
# header = none put the col number else it will take 1st col as header

pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/wine
```

Out[128]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
173	3	13.71	5.65	2.45	20.5	95	1.68	0.61	0.52	1.06	7.70	0.64	1.74	740
174	3	13.40	3.91	2.48	23.0	102	1.80	0.75	0.43	1.41	7.30	0.70	1.56	750
175	3	13.27	4.28	2.26	20.0	120	1.59	0.69	0.43	1.35	10.20	0.59	1.56	835
176	3	13.17	2.59	2.37	20.0	120	1.65	0.68	0.53	1.46	9.30	0.60	1.62	840
177	3	14.13	4.10	2.74	24.5	96	2.05	0.76	0.56	1.35	9.20	0.61	1.60	560

178 rows × 14 columns

In [ ]:
```

```