Introduction to Pandas

It's documentation: http://pandas.pydata.org/pandas-docs/stable/

- Created by Wes Mckinney in 2008
- An open source project under permissive licence
- Strong community with 100 different contributors

The pandas data structures: `Series` and `DataFrame`

A `Series` is a **one dimensional labeled array** (single column in a excel sheet) cable of holding data of any type (integer, string, float, python objects etc.). The axis labels are collectively called as index.

A `DataFrame` is a **tablular data structure** (multi-dimensional object to hold labeled data) comprised of rows and columns,You can think of it as multiple Series object which share the same index.

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

```
pd.Series?
```

Creating a series from an array

```
data=np.array([10,20,30,40,50])
ser=pd.Series(data)
print(ser)
```

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

Creating a series from a list

```
l=['a','e','i','o','u']
ser=pd.Series(l)
```

```
print(ser)
```

```
0    a
1    e
2    i
3    o
4    u
dtype: object
```

```
#To set index for the series
subj=["Maths","Science","Social science","Language"]
marks=[100,98,87,89]
pd.Series(marks, index=subj)
```

```
Maths             100
Science            98
Social science     87
Language           89
dtype: int64
```

```
#Create a series from dictionary
sub_mark={"Maths":100,"Science":98,"Social Science":87,"Language":89}
pd.Series(sub_mark)
```

```
Maths             100
Science            98
Social Science     87
Language           89
dtype: int64
```

```
#Series with missing values
subj=["Maths","Science","Social Science","Computer science"]
mark_series=pd.Series(sub_mark,index=subj)
mark_series
```

```
Maths               100.0
Science              98.0
Social Science       87.0
Computer science      NaN
dtype: float64
```

```
#Manipulating Series
# to check null values
mark_series.isnull()
```

```
Maths              False
Science            False
Social Science     False
Computer science    True
dtype: bool
```

```
mark_series.notnull()
```

```
Maths                True
Science              True
Social Science       True
Computer science    False
dtype: bool
```

```
mark_series>90
```

```
Maths                True
Science              True
Social Science      False
Computer science    False
dtype: bool
```

```
#extracting subjects above 90
mark_series[mark_series>90]
```

```
Maths      100.0
Science     98.0
dtype: float64
```

```
# to sort values
#mark_series.sort_values()
mark_series.sort_values(ascending=False)
```

```
Maths               100.0
Science              98.0
Social Science       87.0
Computer science      NaN
dtype: float64
```

```
#ranking values
#mark_series.rank()
mark_series.rank(ascending=False)
```

```
Maths               1.0
Science             2.0
Social Science      3.0
Computer science    NaN
dtype: float64
```

```
#basic statistics
mark_series.sum()
mark_series.mean()
mark_series.median()
mark_series.std()
mark_series.max()
mark_series.idxmax()
mark_series.min()
mark_series.idxmin()
mark_series.count()
```

```
3
```

```
#summary statistics
mark_series.describe()
```

```
count      3.0
mean      95.0
std        7.0
min       87.0
25%       92.5
50%       98.0
75%       99.0
max      100.0
dtype: float64
```

```
#to get unique values in a series
mark_series.unique()
```

```
array([100.,  98.,  87.,  nan])
```

```
#to get count of unique values
mark_series.nunique()
```

```
3
```

```
#to get the count of each unique value
mark_series.value_counts()
```

```
100.0    1
98.0     1
87.0     1
dtype: int64
```

```
mark_series.dropna()
#mark_series.dropna(inplace=True)
```

```
Maths             100.0
Science            98.0
Social Science     87.0
dtype: float64
```

```
mark_series
```

```
Maths               100.0
Science              98.0
Social Science       87.0
Computer science      NaN
dtype: float64
```

Accessing elements of a series

There are two ways through which we can access element of series, they are :

- Accessing Element from Series with Position
- Accessing Element Using Label (index)

```
#accessing element from series with position
mark_series[:2]
```

```
Maths      100.0
Science     98.0
dtype: float64
```

```
#accessing element using Label
data=np.array(['a','e','i','o','u'])
ser=pd.Series(data,index=[5,10,5,20,25])
ser
```

```
5     a
10    e
5     i
20    o
25    u
dtype: object
```

```
#ser[2]
ser[5]
```

```
5    a
5    i
dtype: object
```

```
#using iloc
ser.iloc[2]
```

```
'i'
```

```
#using loc
ser.loc[15]
```

```
mark_series
```

```
mark_series.iloc[1]
```

```
mark_series.loc["Science"]
```

```
98.0
```

```
mark_series
```

```
mark_series.iloc[:2]
```

```
mark_series.loc[:"Social Science"]
```

```
r = np.array([1, 2])
ser = pd.Series(r, copy=False)
ser.iloc[0] = 999
print(r)
print(ser)
```

Loading data into data frame

```
#pandas dataframe
#creating from dictionary
data={'subj':["Maths","Science","Social Science","Computer science"],'marks':[100,98,87,89],'grade
data
df=pd.DataFrame(data)
df
```

	subj	marks	grade
0	Maths	100	A
1	Science	98	A
2	Social Science	87	B
3	Computer science	89	B

```
import pandas as pd
# create an Empty DataFrame object
df = pd.DataFrame()
print(df)
# append columns to an empty DataFrame
df['Name'] = ['Ankit', 'Ankita', 'Yashvardhan']
df['Age'] = [19,18,20]
df['Gender'] = ['M','F','M']
df
del(df['Age'])
df
```

```
Empty DataFrame
```

```
df1 = pd.DataFrame([[1, 2], [3, 4]], columns = ['a','b'])
df1.loc[len(df1.index)] =[7,7]
df1
```

	a	b
0	1	2
1	3	4
2	7	7

```
# create an Empty DataFrame object
df = pd.DataFrame()
print(df)

# append columns to an empty DataFrame
df['Name'] = ['Ankit', 'Ankita', 'Yashvardhan']
df['Age'] = [19,18,20]
df['Gender'] = ['M','F','M']
df
```

```
Empty DataFrame
Columns: []
Index: []
```

	Name	Age	Gender
0	Ankit	19	M
1	Ankita	18	F
2	Yashvardhan	20	M

```
#creating from series
subj=pd.Series(["Maths","Science","Social Science","Computer science"])
marks=pd.Series([100,98,87,89])
grade=pd.Series(('A','A','B','B'))
#df=pd.DataFrame([subj,marks,grade])
#print(df)

df=pd.DataFrame([subj,marks,grade],index=['subj','marks','grade']).T
print(df)
```

```
                subj marks grade
0              Maths   100     A
1            Science    98     A
2    Social Science    87     B
3  Computer science    89     B
```

```
import pandas as pd
import numpy as np


df1 = pd.DataFrame([[1, 2], [3, 4]], columns = ['a','b'])
df2 = pd.DataFrame([[5, 6], [7, 8]], columns = ['a','b'])
df1 = df1.append(df2)
print(df1)
```

```
   a  b
0  1  2
1  3  4
0  5  6
1  7  8
<ipython-input-46-744627740865>:3: FutureWarning: The frame.append method is deprecated and w
  df1 = df1.append(df2)
```

```
#create an Empty DataFrame object
df = pd.DataFrame()

# append columns to an empty DataFrame
df['Name'] = ['Ankit', 'Ankita', 'Yashv','Vikal']
df['CAT1_Mark'] = [15,22,24,25]
df['CAT2_Mark'] = [18,24,21,23]
df['FAT_Mark']=[45,50,37,49]
print(df)
df.loc[len(df.index)] = ['Pavan',23,23,41]
print(df)
```

```
     Name  CAT1_Mark  CAT2_Mark  FAT_Mark
0   Ankit         15         18        45
1  Ankita         22         24        50
2   Yashv         24         21        37
3   Vikal         25         23        49
     Name  CAT1_Mark  CAT2_Mark  FAT_Mark
0   Ankit         15         18        45
1  Ankita         22         24        50
2   Yashv         24         21        37
3   Vikal         25         23        49
4   Pavan         23         23        41
```

```
df['Total']=df['CAT1_Mark']+df['CAT2_Mark']+df['FAT_Mark']
df
df.sort_values('Total')
```

	Name	CAT1_Mark	CAT2_Mark	FAT_Mark	Total
0	Ankit	15	18	45	78
2	Yashv	24	21	37	82

```
df.sort_values('Total')
```

	Name	CAT1_Mark	CAT2_Mark	FAT_Mark	Total
0	Ankit	15	18	45	78
2	Yashv	24	21	37	82
4	Pavan	23	23	41	87
1	Ankita	22	24	50	96
3	Vikal	25	23	49	97

```
df.iloc[1][2]
```

```
24
```

```
24
```

```
df['Name'][3]
```

```
'Vikal'
```

```
#pandas dataframe
#creating from dictionary
data={'subj':["Maths","Science","Social Science","Computer science"],'marks':[100,98,87,89],'grade
data
df=pd.DataFrame(data)
df
```

	subj	marks	grade
0	Maths	100	A
1	Science	98	A
2	Social Science	87	B
3	Computer science	89	B

```
# iterating over rows using iterrows() function
for i, j in df.iterrows():
    print(i,j)
 # iterating over columns
```

```
for k in df.columns:
  print(df[k])
```

```
0 subj     Maths
marks      100
grade        A
Name: 0, dtype: object
1 subj     Science
marks         98
grade          A
Name: 1, dtype: object
2 subj     Social Science
marks                87
grade                 B
Name: 2, dtype: object
3 subj     Computer science
marks                  89
grade                   B
Name: 3, dtype: object
0               Maths
1             Science
2      Social Science
3    Computer science
Name: subj, dtype: object
0    100
1     98
2     87
3     89
Name: marks, dtype: int64
0    A
1    A
2    B
3    B
Name: grade, dtype: object
```

```
#creating from series
subj=pd.Series(["Maths","Science","Social Science","Computer science"])
marks=pd.Series([100,98,87,89])
grade=pd.Series(('A','A','B','B'))
df=pd.DataFrame([subj,marks,grade])
print(df)
df=pd.DataFrame([subj,marks,grade],index=['subj','marks','grade'])
print(df)
df=pd.DataFrame([subj,marks,grade],index=['subj','marks','grade']).T
print(df)
```

```
       0        1               2                 3
0  Maths  Science  Social Science  Computer science
1    100       98              87                89
2      A        A               B                 B
           0        1               2                 3
subj   Maths  Science  Social Science  Computer science
marks    100       98              87                89
grade      A        A               B                 B
               subj marks grade
0             Maths   100     A
1           Science    98     A
```

```
      2    Social Science    87     B
      3  Computer science    89     B
```

```
#loading hard coded data
df_h=pd.DataFrame([['Jan', 58, 42, 74, 22, 2.95],
                ['Feb', 61, 45, 78, 26, 3.02],
               ['Mar', 65, 48, 84, 25, 2.34],
               ['Apr', 67, 50, 92, 28, 1.02],
               ['May', 71, 53, 98, 35, 0.48],
               ['Jun', 75, 56, 107, 41, 0.11],
               ['Jul', 77, 58, 105, 44, 0.0],
               ['Aug', 77, 59, 102, 43, 0.03],
               ['Sep', 77, 57, 103, 40, 0.17],
               ['Oct', 73, 54, 96, 34, 0.81],
               ['Nov', 64, 48, 84, 30, 1.7],
               ['Dec', 58, 42, 73, 21, 2.56]],
                index=[0,1,2,3,4,5,6,7,8,9,10,11],
               columns=['month','avg_low','avg_high','record_high','record_low','avg_preci'])
print(df_h)
```

```
      month  avg_low  avg_high  record_high  record_low  avg_preci
0     Jan       58        42           74          22       2.95
1     Feb       61        45           78          26       3.02
2     Mar       65        48           84          25       2.34
3     Apr       67        50           92          28       1.02
4     May       71        53           98          35       0.48
5     Jun       75        56          107          41       0.11
6     Jul       77        58          105          44       0.00
7     Aug       77        59          102          43       0.03
8     Sep       77        57          103          40       0.17
9     Oct       73        54           96          34       0.81
10    Nov       64        48           84          30       1.70
11    Dec       58        42           73          21       2.56
```

```
dates=pd.date_range('20220101',periods=5)
dates
```

```
DatetimeIndex(['2022-01-01', '2022-01-02', '2022-01-03', '2022-01-04',
               '2022-01-05'],
              dtype='datetime64[ns]', freq='D')
```

```
a=np.random.random((5,4))
a
```

```
array([[0.98777457, 0.12513649, 0.46661298, 0.7014642 ],
       [0.69795484, 0.28946887, 0.29363442, 0.51014844],
       [0.87456276, 0.98095062, 0.12407944, 0.14830578],
       [0.14994703, 0.01924441, 0.30536278, 0.55096644],
       [0.94572629, 0.56105703, 0.35127255, 0.95207901]])
```

```
df=pd.DataFrame(np.random.random((5,4)),index=dates,columns=list('ABCD'))
df
```

	A	B	C	D
2022-01-01	0.048114	0.491222	0.280125	0.840043
2022-01-02	0.097718	0.009601	0.262051	0.173596
2022-01-03	0.179057	0.303074	0.887830	0.678528
2022-01-04	0.523071	0.597963	0.861085	0.210535
2022-01-05	0.709964	0.232737	0.769772	0.631873

```
df1=pd.DataFrame({'kind':['cat','dog','cat','dog'],
                  'height':[9.1, 6.0, 9.5, 34.0],
                  'weight':[7.9, 7.5, 9.9, 198.0]})
print(df1)
```

```
   kind  height  weight
0  cat     9.1     7.9
1  dog     6.0     7.5
2  cat     9.5     9.9
3  dog    34.0   198.0
```

```
df1.loc[len(df1.index)] = ['rabbit',10,8]
df1
```

	kind	height	weight
0	cat	9.1	7.9
1	dog	6.0	7.5
2	cat	9.5	9.9
3	dog	34.0	198.0
4	rabbit	10.0	8.0

```
#creating data frame from series
stud1=pd.Series({"regno":"20MCB1001","name":"Ravi","CGPA":9.5})
stud2=pd.Series({"regno":"20MCB1002","name":"Raja","CGPA":8.5})
stud3=pd.Series({"regno":"20MCB1003","name":"Rama","CGPA":7.5})

st=pd.DataFrame([stud1,stud2,stud3],index=["chennai","Mumbai","chennai"])
st
```

	regno	name	CGPA
chennai	20MCB1001	Ravi	9.5
Mumbai	20MCB1002	Raja	8.5
chennai	20MCB1003	Rama	7.5

```
st["name"]
```

```
chennai    Ravi
Mumbai     Raja
chennai    Rama
Name: name, dtype: object
```

```
st.name
```

```
chennai    Ravi
Mumbai     Raja
chennai    Rama
Name: name, dtype: object
```

```
#st["name"]
st.name
```

```
chennai    Ravi
Mumbai     Raja
chennai    Rama
Name: name, dtype: object
```

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

```
#reading data from csv files
filename='/content/weather.csv'
df=pd.read_csv(filename)
print(df)
```

```
   month  avg_low  avg_high  record_high  record_low  avg_preci
0    Jan       58        42           74          22       2.95
1    Feb       61        45           78          26       3.02
2    Mar       65        48           84          25       2.34
3    Apr       67        50           92          28       1.02
4    May       71        53           98          35       0.48
5    Jun       75        56          107          41       0.11
6    Jul       77        58          105          44       0.00
7    Aug       77        59          102          43       0.03
8    Sep       77        57          103          40       0.17
9    Oct       73        54           96          34       0.81
10   Nov       64        48           84          30       1.70
11   Dec       58        42           73          21       2.56
```

Displaying data

```
#reading data from csv files
filename='/content/weather.csv'
df=pd.read_csv(filename)
print(df)
```

```
    month  avg_low  avg_high  record_high  record_low  avg_preci
0     Jan       58        42           74          22       2.95
1     Feb       61        45           78          26       3.02
2     Mar       65        48           84          25       2.34
3     Apr       67        50           92          28       1.02
4     May       71        53           98          35       0.48
5     Jun       75        56          107          41       0.11
6     Jul       77        58          105          44       0.00
7     Aug       77        59          102          43       0.03
8     Sep       77        57          103          40       0.17
9     Oct       73        54           96          34       0.81
10    Nov       64        48           84          30       1.70
11    Dec       58        42           73          21       2.56
```

```
#display first few rows
#print(df.head())
df.head(2)
```

	month	avg_low	avg_high	record_high	record_low	avg_preci
0	Jan	58	42	74	22	2.95
1	Feb	61	45	78	26	3.02

```
#display last few rows
#print(df.tail())
df.tail()
```

	month	avg_low	avg_high	record_high	record_low	avg_preci
7	Aug	77	59	102	43	0.03
8	Sep	77	57	103	40	0.17
9	Oct	73	54	96	34	0.81
10	Nov	64	48	84	30	1.70
11	Dec	58	42	73	21	2.56

```
df.sample(5)
```

	month	avg_low	avg_high	record_high	record_low	avg_preci
9	Oct	73	54	96	34	0.81
10	Nov	64	48	84	30	1.70
2	Mar	65	48	84	25	2.34
8	Sep	77	57	103	40	0.17
7	Aug	77	59	102	43	0.03

Description of data

```
#To get the shape of the data
df.shape
```

```
(12, 6)
```

```
#To get data types
df.dtypes
```

```
month           object
avg_low          int64
avg_high         int64
record_high      int64
record_low       int64
avg_preci      float64
dtype: object
```

```
#To get index
df.index
```

```
RangeIndex(start=0, stop=12, step=1)
```

```
#To get columns
df.columns
```

```
Index(['month', 'avg_low', 'avg_high', 'record_high', 'record_low',
       'avg_preci'],
      dtype='object')
```

```
df.head(2)
```

	month	avg_low	avg_high	record_high	record_low	avg_preci
0	Jan	58	42	74	22	2.95
1	Feb	61	45	78	26	3.02

```
df['avg_high'].head(2)
```

```
df.columns=df.columns.str.lstrip()
```

```
df.columns
```

```
Index(['month', 'avg_low', 'avg_high', 'record_high', 'record_low',
       'avg_preci'],
      dtype='object')
```

```
#To get values
df.values
```

```
array([['Jan', 58, 42, 74, 22, 2.95],
       ['Feb', 61, 45, 78, 26, 3.02],
       ['Mar', 65, 48, 84, 25, 2.34],
       ['Apr', 67, 50, 92, 28, 1.02],
       ['May', 71, 53, 98, 35, 0.48],
       ['Jun', 75, 56, 107, 41, 0.11],
       ['Jul', 77, 58, 105, 44, 0.0],
       ['Aug', 77, 59, 102, 43, 0.03],
       ['Sep', 77, 57, 103, 40, 0.17],
       ['Oct', 73, 54, 96, 34, 0.81],
       ['Nov', 64, 48, 84, 30, 1.7],
       ['Dec', 58, 42, 73, 21, 2.56]], dtype=object)
```

```
#statistical summary of each column
df.describe()
```

	avg_low	avg_high	record_high	record_low	avg_preci
count	12.000000	12.000000	12.000000	12.000000	12.000000
mean	68.583333	51.000000	91.333333	32.416667	1.265833
std	7.366488	6.060303	12.323911	8.240238	1.186396
min	58.000000	42.000000	73.000000	21.000000	0.000000
25%	63.250000	47.250000	82.500000	25.750000	0.155000
50%	69.000000	51.500000	94.000000	32.000000	0.915000
75%	75.500000	56.250000	102.250000	40.250000	2.395000
max	77.000000	59.000000	107.000000	44.000000	3.020000

```
df.describe(include="all")
```

	month	avg_low	avg_high	record_high	record_low	avg_preci
count	12	12.000000	12.000000	12.000000	12.000000	12.000000

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12 entries, 0 to 11
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   month        12 non-null     object 
 1   avg_low      12 non-null     int64  
 2   avg_high     12 non-null     int64  
 3   record_high  12 non-null     int64  
 4   record_low   12 non-null     int64  
 5   avg_preci    12 non-null     float64
dtypes: float64(1), int64(4), object(1)
memory usage: 704.0+ bytes
```

Sorting data

```
#sort the records by any column
df.sort_values('record_high',ascending=False)
```

	month	avg_low	avg_high	record_high	record_low	avg_preci
5	Jun	75	56	107	41	0.11
6	Jul	77	58	105	44	0.00
8	Sep	77	57	103	40	0.17
7	Aug	77	59	102	43	0.03
4	May	71	53	98	35	0.48
9	Oct	73	54	96	34	0.81
3	Apr	67	50	92	28	1.02
2	Mar	65	48	84	25	2.34
10	Nov	64	48	84	30	1.70
1	Feb	61	45	78	26	3.02
0	Jan	58	42	74	22	2.95
11	Dec	58	42	73	21	2.56

```
#sorting by an axis
#df.sort_index(axis=1)
df.sort_index(axis=1,ascending=False)
```

	record_low	record_high	month	avg_preci	avg_low	avg_high
0	22	74	Jan	2.95	58	42
1	26	78	Feb	3.02	61	45
2	25	84	Mar	2.34	65	48
3	28	92	Apr	1.02	67	50
4	35	98	May	0.48	71	53
5	41	107	Jun	0.11	75	56
6	44	105	Jul	0.00	77	58
7	43	102	Aug	0.03	77	59
8	40	103	Sep	0.17	77	57
9	34	96	Oct	0.81	73	54
10	30	84	Nov	1.70	64	48
11	21	73	Dec	2.56	58	42

```
sample_df=df.sample(5)
sample_df
```

	month	avg_low	avg_high	record_high	record_low	avg_preci
2	Mar	65	48	84	25	2.34
7	Aug	77	59	102	43	0.03
0	Jan	58	42	74	22	2.95
8	Sep	77	57	103	40	0.17
10	Nov	64	48	84	30	1.70

```
#sample_df.sort_index(axis=0)
sample_df.sort_index(axis=0)
```

	month	avg_low	avg_high	record_high	record_low	avg_preci
0	Jan	58	42	74	22	2.95
2	Mar	65	48	84	25	2.34
7	Aug	77	59	102	43	0.03
8	Sep	77	57	103	40	0.17
10	Nov	64	48	84	30	1.70

Slicing data

```
df.columns
```

```
Index(['month', 'avg_low', 'avg_high', 'record_high', 'record_low',
       'avg_preci'],
      dtype='object')
```

```
#slicing a column
df.avg_low
```

```
0     58
1     61
2     65
3     67
4     71
5     75
6     77
7     77
8     77
9     73
10    64
11    58
Name: avg_low, dtype: int64
```

```
#slicing a column
df['avg_low']
```

```
0     58
1     61
2     65
3     67
4     71
5     75
6     77
7     77
8     77
9     73
10    64
11    58
Name: avg_low, dtype: int64
```

```
#slicing rows
df[2:4]
```

	month	avg_low	avg_high	record_high	record_low	avg_preci
2	Mar	65	48	84	25	2.34
3	Apr	67	50	92	28	1.02

```
df[['month','record_high']]
```

	month	record_high
0	Jan	74
1	Feb	78
2	Mar	84
3	Apr	92
4	May	98
5	Jun	107
6	Jul	105
7	Aug	102
8	Sep	103
9	Oct	96
10	Nov	84

```
#slicing all rows and few columns
df.loc[:,['avg_low','record_high']]
```

	avg_low	record_high
0	58	74
1	61	78
2	65	84
3	67	92
4	71	98
5	75	107
6	77	105
7	77	102
8	77	103
9	73	96
10	64	84
11	58	73

```
#retrieving a particular value along a row and a column(s)
df.loc[9,['avg_low','record_high']]
```

```
avg_low        73
record_high    96
Name: 9, dtype: object
```

```
df.at[9,'avg_low']
df.loc[9,'avg_low']
```

```
73
```

```
df.loc[9,'avg_low']
```

```
73
```

```
df.head()
```

	month	avg_low	avg_high	record_high	record_low	avg_preci
0	Jan	58	42	74	22	2.95
1	Feb	61	45	78	26	3.02
2	Mar	65	48	84	25	2.34
3	Apr	67	50	92	28	1.02
4	May	71	53	98	35	0.48

```
#slicing by position
df.iloc[3]
```

```
month           Apr
avg_low          67
avg_high         50
record_high      92
record_low       28
avg_preci      1.02
Name: 3, dtype: object
```

```
df.columns
```

```
Index(['month', 'avg_low', 'avg_high', 'record_high', 'record_low',
       'avg_preci'],
      dtype='object')
```

```
#slicing by position
df.iloc[:,[2,5]]
```

	avg_high	avg_preci
0	42	2.95
1	45	3.02
2	48	2.34
3	50	1.02
4	53	0.48
5	56	0.11
6	58	0.00

```
df.iloc[3:5,[2,3]]
#df.iloc[[3,5],[2,3]]
```

	avg_high	record_high
3	50	92
4	53	98

```
#to get a scalar value explicitly
df.iloc[1,1]
```

```
61
```

```
#faster access to the scalar value
df.iat[1,1]
```

```
61
```

```
df
```

	month	avg_low	avg_high	record_high	record_low	avg_preci
0	Jan	58	42	74	22	2.95
1	Feb	61	45	78	26	3.02

Filtering

```
df[df["month"]=="Feb"]
```

	month	avg_low	avg_high	record_high	record_low	avg_preci
1	Feb	61	45	78	26	3.02
8	Sep	77	57	103	40	0.17

```
df[df['month'].isin(['Feb','Jun','Dec'])]
```

	month	avg_low	avg_high	record_high	record_low	avg_preci
1	Feb	61	45	78	26	3.02
5	Jun	75	56	107	41	0.11
11	Dec	58	42	73	21	2.56

```
df.month
```

```
0     Jan
1     Feb
2     Mar
3     Apr
4     May
5     Jun
6     Jul
7     Aug
8     Sep
9     Oct
10    Nov
11    Dec
Name: month, dtype: object
```

```
df[df["month"].isin(['May','Sep','Dec'])]
```

	month	avg_low	avg_high	record_high	record_low	avg_preci
4	May	71	53	98	35	0.48
8	Sep	77	57	103	40	0.17
11	Dec	58	42	73	21	2.56

```
#Assignment
df.loc[9,['avg_percipitation']]=0.89


df.loc[9,['avg_percipitation']]
```

```
avg_percipitation    0.89
Name: 9, dtype: object
```

```
df.avg_low
```

```
0     58
1     61
2     65
3     67
4     71
5     75
6     77
7     77
8     77
9     73
10    64
11    58
Name: avg_low, dtype: int64
```

```
l=[5]
l*5
```

```
[5, 5, 5, 5, 5]
```

```
#Assignment
df.loc[:,['avg_low']]=np.array([5]*len(df))
df.avg_low
```

```
0     5
1     5
2     5
3     5
4     5
5     5
6     5
7     5
8     5
9     5
10    5
11    5
Name: avg_low, dtype: int64
```

```
df.columns
```

```
Index(['month', 'avg_low', 'avg_high', 'record_high', 'record_low',
       'avg_preci', 'avg_percipitation'],
      dtype='object')
```

```
#Creating a new column
df['avg_day']=(df.avg_low+df.avg_high)/2
df.head()
```

	month	avg_low	avg_high	record_high	record_low	avg_preci	avg_percipitation	avg_day
0	Jan	58	42	74	22	2.95	NaN	50.0
1	Feb	61	45	78	26	3.02	NaN	53.0
2	Mar	65	48	84	25	2.34	NaN	56.5
3	Apr	67	50	92	28	1.02	NaN	58.5
4	May	71	53	98	35	0.48	NaN	62.0

```
df.columns
```

```
Index(['month', 'avg_low', 'avg_high', 'record_high', 'record_low',
       'avg_preci', 'avg_day'],
      dtype='object')
```

Renaming columns

```
df.rename(columns={'avg_low':'avg_l'})
```

	month	avg_l	avg_high	record_high	record_low	avg_preci	avg_percipitation	avg_day
0	Jan	58	42	74	22	2.95	NaN	50.0
1	Feb	61	45	78	26	3.02	NaN	53.0
2	Mar	65	48	84	25	2.34	NaN	56.5
3	Apr	67	50	92	28	1.02	NaN	58.5
4	May	71	53	98	35	0.48	NaN	62.0
5	Jun	75	56	107	41	0.11	NaN	65.5
6	Jul	77	58	105	44	0.00	NaN	67.5
7	Aug	77	59	102	43	0.03	NaN	68.0
8	Sep	77	57	103	40	0.17	NaN	67.0
9	Oct	73	54	96	34	0.81	0.89	63.5
10	Nov	64	48	84	30	1.70	NaN	56.0
11	Dec	58	42	73	21	2.56	NaN	50.0

```
df.columns
```

```
Index(['month', 'avg_low', 'avg_high', 'record_high', 'record_low',
       'avg_preci', 'avg_percipitation', 'avg_day'],
      dtype='object')
```

```
df.rename(columns={'avg_percipitation':'avg_preci'},inplace=True)
df.head()
```

	month	av_hi	av_lo	rec_hi	rec_lo	av_rain	av_dy
0	Jan	58	42	74	22	2.95	NaN
1	Feb	61	45	78	26	3.02	NaN
2	Mar	65	48	84	25	2.34	NaN
3	Apr	67	50	92	28	1.02	NaN
4	May	71	53	98	35	0.48	NaN

```
df.columns
```

```
Index(['month', 'av_hi', 'av_lo', 'rec_hi', 'rec_lo', 'av_rain', 'av_dy'], dtype='object')
```

```
#reading data from csv files
filename='/content/weather.csv'
df=pd.read_csv(filename)
print(df)
```

```
    month  avg_low  avg_high  record_high  record_low  avg_preci
0     Jan       58        42           74          22       2.95
1     Feb       61        45           78          26       3.02
2     Mar       65        48           84          25       2.34
3     Apr       67        50           92          28       1.02
4     May       71        53           98          35       0.48
5     Jun       75        56          107          41       0.11
6     Jul       77        58          105          44       0.00
7     Aug       77        59          102          43       0.03
8     Sep       77        57          103          40       0.17
9     Oct       73        54           96          34       0.81
10    Nov       64        48           84          30       1.70
11    Dec       58        42           73          21       2.56
```

```
df.columns=['month','av_hi','av_lo','rec_hi','rec_lo','av_rain','av_dy']
df.head()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-68-e17b9c70d650> in <cell line: 1>()
----> 1 df.columns=['month','av_hi','av_lo','rec_hi','rec_lo','av_rain','av_dy']
      2 df.head()

4 frames
/usr/local/lib/python3.10/dist-packages/pandas/core/internals/base.py in
_validate_set_axis(self, axis, new_labels)
     68
     69         elif new_len != old_len:
---> 70             raise ValueError(
     71                 f"Length mismatch: Expected axis has {old_len} elements, new "
     72                 f"values have {new_len} elements"
```

```
#iterating over data
for index,row in df.iterrows():
    print(index,row['month'],row['avg_high'])
```

```
0 Jan 42
1 Feb 45
2 Mar 48
3 Apr 50
4 May 53
5 Jun 56
6 Jul 58
7 Aug 59
8 Sep 57
9 Oct 54
10 Nov 48
11 Dec 42
```

```
df.head(3)
```

	month	avg_low	avg_high	record_high	record_low	avg_preci
0	Jan	58	42	74	22	2.95
1	Feb	61	45	78	26	3.02
2	Mar	65	48	84	25	2.34

```
#breaking data frame into pieces
pieces=[df[:2],df[2:4],df[4:]]
pieces
```

```
[  month  avg_low  avg_high  record_high  record_low  avg_preci
 0   Jan       58        42           74          22       2.95
 1   Feb       61        45           78          26       3.02,
   month  avg_low  avg_high  record_high  record_low  avg_preci
 2   Mar       65        48           84          25       2.34
 3   Apr       67        50           92          28       1.02,
    month  avg_low  avg_high  record_high  record_low  avg_preci
 4    May       71        53           98          35       0.48
 5    Jun       75        56          107          41       0.11
```

```
6    Jul        77         58          105          44        0.00
7    Aug        77         59          102          43        0.03
8    Sep        77         57          103          40        0.17
9    Oct        73         54           96          34        0.81
10   Nov        64         48           84          30        1.70
11   Dec        58         42           73          21        2.56]
```

```
pd.concat(pieces)
```

	month	avg_low	avg_high	record_high	record_low	avg_preci
0	Jan	58	42	74	22	2.95
1	Feb	61	45	78	26	3.02
2	Mar	65	48	84	25	2.34
3	Apr	67	50	92	28	1.02
4	May	71	53	98	35	0.48
5	Jun	75	56	107	41	0.11
6	Jul	77	58	105	44	0.00
7	Aug	77	59	102	43	0.03
8	Sep	77	57	103	40	0.17
9	Oct	73	54	96	34	0.81
10	Nov	64	48	84	30	1.70
11	Dec	58	42	73	21	2.56

```
piece1=df[:2]
piece2=df[2:4]
print(pd.merge(piece1,piece2))
```

```
Empty DataFrame
Columns: [month, avg_low, avg_high, record_high, record_low, avg_preci]
Index: []
```

```
#appending data
s=df.iloc[8]
df.append(s)
```

```
<ipython-input-29-6d90692f3193>:3: FutureWarning: The frame.append method is deprecated and w
  df.append(s)
```

	month	avg_low	avg_high	record_high	record_low	avg_preci
0	Jan	58	42	74	22	2.95
1	Feb	61	45	78	26	3.02
2	Mar	65	48	84	25	2.34
3	Apr	67	50	92	28	1.02
4	May	71	53	98	35	0.48
5	Jun	75	56	107	41	0.11
6	Jul	77	58	105	44	0.00
7	Aug	77	59	102	43	0.03
8	Sep	77	57	103	40	0.17

```
df.head()
```

	month	avg_low	avg_high	record_high	record_low	avg_preci
0	Jan	58	42	74	22	2.95
1	Feb	61	45	78	26	3.02
2	Mar	65	48	84	25	2.34
3	Apr	67	50	92	28	1.02
4	May	71	53	98	35	0.48

```
df['newcol']=[0,0,0,0,1,1,1,1,2,2,2,2]
df
```

	month	avg_low	avg_high	record_high	record_low	avg_preci	newcol
0	Jan	58	42	74	22	2.95	0

```
#groupby operation allows you to partition an array into groups based on the value in one or more c
#and then perform operations on each group separately
df.groupby('newcol').sum()
```

```
<ipython-input-9-43a83063c71e>:3: FutureWarning: The default value of numeric_only in DataFra
  df.groupby('newcol').sum()
```

newcol	avg_low	avg_high	record_high	record_low	avg_preci
0	251	185	328	101	9.33
1	300	226	412	163	0.62
2	272	201	356	125	5.24

```
print(df.groupby('newcol'))
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x79d42792e260>
```

```
#df.groupby('newcol')['avg_low'].sum()
df.groupby('newcol')['avg_low','record_low'].sum()
```

```
<ipython-input-57-adcc81b97855>:2: FutureWarning: Indexing with multiple keys (implicitly con
  df.groupby('newcol')['avg_low','record_low'].sum()
```

newcol	avg_low	record_low
0	251	101
1	300	163
2	272	125

```
df.groupby('newcol')['avg_high'].mean()
```

```
newcol
0    46.25
1    56.50
2    50.25
Name: avg_high, dtype: float64
```

```
#df.groupby('newcol')['avg_high'].aggregate(['min',np.median,max])
df['avg_high'].aggregate([min,np.median,max])
```

```
min       42.0
median    51.5
max       59.0
Name: avg_high, dtype: float64
```

```
df.groupby('newcol').aggregate({'avg_low':[min,max],'avg_high':[np.mean,np.median]})
```

	avg_low		avg_high	
	min	max	mean	median
newcol				
0	58	67	46.25	46.5
1	71	77	56.50	57.0
2	58	77	50.25	51.0

```
df.groupby('newcol')['avg_low'].std()
```

```
newcol
0    4.031129
1    2.828427
2    8.602325
Name: avg_low, dtype: float64
```

```
df['Catcol']=['A','A','A','B','B','B','C','C','C','D','D','D']
df.head()
df.groupby(['newcol','catcol']).sum()
```

	month	avg_low	avg_high	record_high	record_low	avg_preci	newcol	Catcol
0	Jan	58	42	74	22	2.95	0	A
1	Feb	61	45	78	26	3.02	0	A
2	Mar	65	48	84	25	2.34	0	A
3	Apr	67	50	92	28	1.02	0	B
4	May	71	53	98	35	0.48	1	B

```
df.groupby(['newcol','Catcol']).sum()
```

```
<ipython-input-18-0698e2d92f5e>:1: FutureWarning: The default value of numeric_only in D
  df.groupby(['newcol','Catcol']).sum()
```

		avg_low	avg_high	record_high	record_low	avg_preci
newcol	**Catcol**					
0	**A**	184	135	236	73	8.31
	B	87	59	98	32	1.93
2	**C**	77	57	103	40	6.17

```
#writing to csv files

filename1='/content/weathernew.csv'
df.to_csv(filename1)
```