PANDAS

- analyze big data and make conclusions based on statistical theories.
- created by Wes McKinney in 2008.
- Size mutability:
- Pandas gives you answers about the data:Is there a correlation between two or more columns? What is average value?Max value?Min value?
- Fast and efficient for manipulating and analyzing data.
- Three Types: Series(1-D), DataFrame(2-D), Panel(3-D) Table

```
1 import pandas as pd
2 print(pd.__version__)
```

```
1.3.5
```

pd.Series(data, index, dtype, copy)

data

data takes various forms like ndarray, list, constants

index

Index values must be unique and hashable, same length as data. Default np.arrange(n) if no index is passed.

dtype

dtype is for data type. If None, data type will be inferred

copy

Copy data. Default False

```
1 #Create an Empty Series
2 import pandas as pd
3 s = pd.Series()
4 print (s)
```

```
Series([], dtype: float64)
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: DeprecationWa
  This is separate from the ipykernel package so we can avoid doing imports u
```

Create a Series from ndarray

```
1 import pandas as pd
2 import numpy as np
3 data = np.array(['a','b','c','d'])
4 s = pd.Series(data)
5 print (s)
```

```
0    a
1    b
2    c
3    d
dtype: object
```

```
1 import pandas as pd
2 import numpy as np
3 data = np.array(['a','b','c','d'])
4 s = pd.Series(data,index=[100,101,102,103])
5 print (s)
6 print(type(s))
```

```
100    a
101    b
102    c
103    d
dtype: object
<class 'pandas.core.series.Series'>
```

```
1 import pandas as pd
2 import numpy as np
3 data = {'a' : 0., 'b' : 1., 'c' : 2.}
4 s = pd.Series(data)
5 print (s)
```

```
a    0.0
b    1.0
c    2.0
dtype: float64
```

```
1 import pandas as pd
2 import numpy as np
3 data = {'a' : 0., 'b' : 1., 'c' : 2.}
4 s = pd.Series(data,index=['b','c','d','a'])
5 print (s)
```

```
b    1.0
c    2.0
d    NaN
a    0.0
dtype: float64
```

Create a Series from Scalar

If data is a scalar value, an index must be provided. The value will be repeated to match the length of index

```
import pandas as pd
import numpy as np
s = pd.Series(5, index=[0, 1, 2, 3])
print (s)
```

```
0    5
1    5
2    5
3    5
dtype: int64
```

DATA access

- position
- label

```
import pandas as pd
s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])

#retrieve the first element
print(s)
print("\n")
print (s[0])
print("\n")
print (s[0:4])
print("\n")
print (len(s))
```

```
a    1
b    2
c    3
d    4
e    5
dtype: int64


1


a    1
b    2
c    3
d    4
dtype: int64


5
```

```
1 print (s[:4])
```

```
a    1
b    2
c    3
d    4
dtype: int64
```

```
1 print (s[-4:])
```

```
b    2
c    3
d    4
e    5
dtype: int64
```

```
1 print(s['a'])
2 print(s['c'])
3 #print(s['g'])-------------give error
```

```
1
3
```

```
1 print (s[['a','c','d']])
```

```
a    1
c    3
d    4
dtype: int64
```

```
1 print (s[['a','c','d']])
```

```
a    1
c    3
d    4
dtype: int64
```

Data updata is possible(**mutable**)

Resize is not possible(**immutable**)

```
1 s['a']=40
2 print(s)
```

```
a    40
b     2
c     3
d     4
e     5
dtype: int64
```

```
1 s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])
```

```
arr=s.to_numpy()
print(type(s))
print(type(arr))
print(np.max(s))
print(np.max(s[:3]))
print(np.min(s[['a','e']]))
```

```
<class 'pandas.core.series.Series'>
<class 'numpy.ndarray'>
5
3
1
```

DATA FRAME

- a two-dimensional data structure
- data is aligned in a tabular fashion in rows and columns

pd.DataFrame(data, index, columns, dtype, copy)

data

data takes various forms like ndarray, series, map, lists, dict, constants and also another DataFrame.

index

For the row labels, the Index to be used for the resulting frame is Optional Default np.arange(n) if no index is passed.

columns

For column labels, the optional default syntax is - np.arange(n). This is only true if no index is passed.

dtype

Data type of each column.

copy

This command (or whatever it is) is used for copying of data, if the default is False.

Create DataFrame

```
df = pd.DataFrame()
print (df)
```

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

```
data = [1,2,3,4,5]
df = pd.DataFrame(data)
```

```
print (df)
print (df.shape)
```

```
   0
0  1
1  2
2  3
3  4
4  5
(5, 1)
```

```
df=pd.DataFrame([1,2,3,4,5],columns=['number'])
print(df)
print (df.shape)
print (df.size)
print (df.ndim)

```

```
   number
0       1
1       2
2       3
3       4
4       5
(5, 1)
5
2
```

```
df=pd.DataFrame([1,2,3,4,5],columns=['number'],index=['student1','student2','stu
print(df)
```

```
           number
student1       1
student2       2
student3       3
student4       4
student5       5
```

```
data = [['Alex',10],['Bob',12],['Clarke',13]]
df = pd.DataFrame(data,columns=['Name','Age'],index=['student1','student2','stu
print (df)
```

```
             Name  Age
student1    Alex   10
student2     Bob   12
student3  Clarke   13
```

shape,size,dimension, conversion to numpy

```
print (df.shape)
```

```
print (df.size)
print (df.ndim)
```

```
(3, 2)
6
2
```

```
print(df.values)
print(type(df.values))
```

```
[['Alex' 10]
 ['Bob' 12]
 ['Clarke' 13]]
<class 'numpy.ndarray'>
```

```
a=df.to_numpy()
print(a)
print(type(a))
```

```
[['Alex' 10]
 ['Bob' 12]
 ['Clarke' 13]]
<class 'numpy.ndarray'>
```

```
data = [['Alex',10],['Bob',12],['Clarke',13]]
df = pd.DataFrame(data,columns=['Name','Age'],dtype=float)
print (df)
```

```
     Name   Age
0    Alex  10.0
1     Bob  12.0
2  Clarke  13.0
/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:3326:
  exec(code_obj, self.user_global_ns, self.user_ns)
```

** DataFrame from Dict of ndarrays / Lists**

```
data = {'Name':['Tom', 'Jack', 'Steve', 'Ricky'],'Age':[28,34,29,42]}
df = pd.DataFrame(data)
print (df)
```

```
    Name  Age
0    Tom   28
1   Jack   34
2  Steve   29
3  Ricky   42
```

```
print(data)
```

```
{'Name': ['Tom', 'Jack', 'Steve', 'Ricky'], 'Age': [28, 34, 29, 42]}
```

```
data = {'Name':['Tom', 'Jack', 'Steve', 'Ricky'],'Age':[28,34,29,42]}
```

```
df = pd.DataFrame(data, index=['rank1','rank2','rank3','rank4'])
print (df)
```

```
          Name  Age
rank1    Tom   28
rank2   Jack   34
rank3  Steve   29
rank4  Ricky   42
```

DataFrame from List of Dicts

```
data = [{'a': 1, 'b': 2},{'a': 5, 'b': 10, 'c': 20}]
df = pd.DataFrame(data)
print (df)
```

```
   a   b     c
0  1   2   NaN
1  5  10  20.0
```

```
df = pd.DataFrame(data,index=[3,4])
print (df)
```

```
   a   b     c
3  1   2   NaN
4  5  10  20.0
```

list of dictionary

```
data = [{'a': 1, 'b': 2},{'a': 5, 'b': 10, 'c': 20}]
df = pd.DataFrame(data)
print (df)
```

```
   a   b     c
0  1   2   NaN
1  5  10  20.0
```

```
data = [{'a': 1, 'b': 2},{'a': 5, 'b': 10, 'c': 20}]
df = pd.DataFrame(data, index=['first', 'second'])
print (df)
```

```
          a   b     c
first     1   2   NaN
second    5  10  20.0
```

```
data = [{'a': 1, 'b': 2},{'a': 5, 'b': 10, 'c': 20}]

#With two column indices, values same as dictionary keys
df1 = pd.DataFrame(data, index=['first', 'second'], columns=['a', 'b'])

#With two column indices with one index with other name
df2 = pd.DataFrame(data, index=['first', 'second'], columns=['a', 'b1'])
```

```
print (df1)
print (df2)
```

```
          a   b
first     1   2
second    5  10
          a  b1
first     1 NaN
second    5 NaN
```

DataFrame from Dict of Series

```
d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),
   'two' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}

df = pd.DataFrame(d)
print (df)
```

```
     one  two
a    1.0    1
b    2.0    2
c    3.0    3
d    NaN    4
```

```
df=pd.DataFrame([1,2,3,4,5])
print(df)
```

```
     0
0    1
1    2
2    3
3    4
4    5
```

```
df=pd.DataFrame([[1,2,3,4,5]])
print(df)
```

```
     0  1  2  3  4
0    1  2  3  4  5
```

```
df=pd.DataFrame([[1,2,3,4,5],[6,7,8,9,10]])
print(df)
```

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

```
df=pd.DataFrame([[1,2,3,4,5],[6,7,8,9,10]],columns=['col1','col2','col3','col4'
print(df)
```

```
      col1  col2  col3  col4  col5
row1     1     2     3     4     5
row2     6     7     8     9    10
```

```
print(len(df))
```

```
2
```

Rows and Column

number of rows and column

```
print(len(df.axes[0]))
print(len(df.axes[1]))
```

```
2
5
```

display and access column name

```
print(df.columns)
print(df.columns[:2])
print(df.columns[3])
print(df.columns[-3:])
```

```
Index(['col1', 'col2', 'col3', 'col4', 'col5'], dtype='object')
Index(['col1', 'col2'], dtype='object')
col4
Index(['col3', 'col4', 'col5'], dtype='object')
```

display and acces row name

```
print(df.index)
print(df.index[0])
print(df.index[1])
print(df.index[0:])
```

```
Index(['row1', 'row2'], dtype='object')
row1
row2
Index(['row1', 'row2'], dtype='object')
```

```
# write column name and row name
df=pd.DataFrame([[1,2,3,4,5],[6,7,8,9,10]])
print(df)
```

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

```
column=[]
for i in range(len(df.axes[1])):
  column.append("col"+str(i+1))
```

```
df=pd.DataFrame([[1,2,3,4,5],[6,7,8,9,10]],columns=column)
print(df)
```

```
   col1  col2  col3  col4  col5
0     1     2     3     4     5
1     6     7     8     9    10
```

```
row=[]
for i in range(len(df.axes[0])):
  row.append("row"+str(i+1))
```

```
df=pd.DataFrame([[1,2,3,4,5],[6,7,8,9,10]],columns=column,index=row)
print(df)
```

```
      col1  col2  col3  col4  col5
row1     1     2     3     4     5
row2     6     7     8     9    10
```

column selection and access column value

```
print(df['col1'])
print(df['col1'][0])
print(df['col1'][1])
print(df['col1'][0:2])
print(df['col1'][:2])
```

```
row1    1
row2    6
Name: col1, dtype: int64
1
6
row1    1
row2    6
Name: col1, dtype: int64
row1    1
row2    6
Name: col1, dtype: int64
```

Adding new column

```
#declaring a new list as a column
df['col6']=[11,12]
print(df)
```

```
      col1  col2  col3  col4  col5  col6
row1     1     2     3     4     5    11
row2     6     7     8     9    10    12
```

DataFrameName.insert(loc, column, value, allow_duplicates = False)

```
1 df.insert(2, "col7", [21, 23], True)
2 print(df)
```

```
      col1  col2  col7  col3  col4  col5  col6
row1     1     2    21     3     4     5    11
row2     6     7    23     8     9    10    12
```

```
1 df.insert(3, "col7", [21, 23], True)
2 print(df)
```

```
      col1  col2  col7  col7  col3  col4  col5  col6
row1     1     2    21    21     3     4     5    11
row2     6     7    23    23     8     9    10    12
```

```
1 # df.insert(4, "col7", [21, 23], False)
2 #print(df)---------- cannot insert col7, already exists
```

Delete column

- del
- pop

```
1 del (df['col7'])
2 print (df)
```

```
      col1  col2  col3  col4  col5  col6
row1     1     2     3     4     5    11
row2     6     7     8     9    10    12
```

```
1 df.pop('col6')
2 print (df)
```

```
      col1  col2  col3  col4  col5
row1     1     2     3     4     5
row2     6     7     8     9    10
```

Adding new column from existing column by doing arithmetic operation

```
1 df['col7']=df['col1']+df['col2']#addition
2 print(df)
```

```
      col1  col2  col3  col4  col5  col7
row1     1     2     3     4     5     3
```

```
row2     6     7     8     9    10    13
```

```
df['col1']=df['col1']-df['col2']#difference
print(df)
```

```
      col1  col2  col3  col4  col5  col7
row1    -1     2     3     4     5     3
row2    -1     7     8     9    10    13
```

```
df['col1']=df['col1']*df['col2']#multiplication
print(df)
```

```
      col1  col2  col3  col4  col5  col7
row1    -2     2     3     4     5     3
row2    -7     7     8     9    10    13
```

```
df['col1']=df['col1']/df['col2']# division
print(df)
```

```
      col1  col2  col3  col4  col5  col7
row1  -1.0     2     3     4     5     3
row2  -1.0     7     8     9    10    13
```

Selecting rows

- using label
- using integer location

```
df=pd.DataFrame([[1,2,3,4,5],[6,7,8,9,10]],columns=column,index=row)# dataframe
print(df)
```

```
      col1  col2  col3  col4  col5
row1     1     2     3     4     5
row2     6     7     8     9    10
```

```
print(df.loc['row1'])
```

```
col1    1
col2    2
col3    3
col4    4
col5    5
Name: row1, dtype: int64
```

```
print(df.iloc[0])
```

```
col1    1
col2    2
col3    3
col4    4
col5    5
Name: row1, dtype: int64
```

```
print(df.iloc[0:2])
```

```
      col1  col2  col3  col4  col5
row1     1     2     3     4     5
row2     6     7     8     9    10
```

```
print(df.iloc[0:1,2:4])
```

```
      col3  col4
row1     3     4
```

```
df = pd.DataFrame(np.random.randn(8, 4), columns = ['A', 'B', 'C', 'D'])
print(df)
# Slicing through list of values
print (df.iloc[[1, 3, 5], [1, 3]])
print('\n')
print (df.iloc[1:3, :])
print('\n')
print (df.iloc[:,1:3])
```

```
          A         B         C         D
0 -0.482213  0.000815 -1.407117 -1.052961
1 -0.606946 -0.227393  0.749983 -1.416531
2 -0.027671 -0.363526  1.083394 -0.019393
3 -1.081470  0.456000 -0.376977  0.791940
4  0.305042  1.819971 -0.397577 -0.207182
5  0.142531 -0.299000 -1.578550  1.827448
6  0.472126 -0.466925  0.913910 -0.290647
7  0.412752 -1.344350  1.042152  0.995806
          B         D
1 -0.227393 -1.416531
3  0.456000  0.791940
5 -0.299000  1.827448


          A         B         C         D
1 -0.606946 -0.227393  0.749983 -1.416531
2 -0.027671 -0.363526  1.083394 -0.019393


          B         C
0  0.000815 -1.407117
1 -0.227393  0.749983
2 -0.363526  1.083394
3  0.456000 -0.376977
4  1.819971 -0.397577
5 -0.299000 -1.578550
6 -0.466925  0.913910
7 -1.344350  1.042152
```

row slicing

```
print(df[0:1])
```

```
      col1  col2  col3  col4  col5
row1     1     2     3     4     5
```

deletion of rows

- Use index label to delete or drop rows from a DataFrame. If label is duplicated, then multiple rows will be dropped.

```
df = df.drop('row1')
print(df)
```

```
      col1  col2  col3  col4  col5
row2     6     7     8     9    10
```

Addition of Rows

```
df2=pd.DataFrame([[6,7,8,9,10]],columns=['col1','col2','col3','col4','col5'])
df=df.append(df2)
print(df)
```

```
      col1  col2  col3  col4  col5
row2     6     7     8     9    10
0        6     7     8     9    10
```

```
df2=pd.DataFrame([[6,7,8,9,10]],columns=['col1','col2','col3','col4','col5'],in
df=df.append(df2)
print(df)
```

```
      col1  col2  col3  col4  col5
row2     6     7     8     9    10
0        6     7     8     9    10
row1     6     7     8     9    10
```

```
df2=pd.DataFrame([6,7,8,9,10])
df=df.append(df2)
print(df)
```

```
      col1  col2  col3  col4  col5     0
row2   6.0   7.0   8.0   9.0  10.0   NaN
0      6.0   7.0   8.0   9.0  10.0   NaN
row1   6.0   7.0   8.0   9.0  10.0   NaN
0      NaN   NaN   NaN   NaN   NaN   6.0
1      NaN   NaN   NaN   NaN   NaN   7.0
2      NaN   NaN   NaN   NaN   NaN   8.0
3      NaN   NaN   NaN   NaN   NaN   9.0
4      NaN   NaN   NaN   NaN   NaN  10.0
```

```
dff=pd.DataFrame([[1,2,3,4,5],[5,6,7,8,9],[40,50,60,70,80]])
print(dff)
print('\n')
```

```
df1=pd.DataFrame([[10,11,12,13,14]])
print(df1)
```

```
    0   1   2   3   4
0   1   2   3   4   5
1   5   6   7   8   9
2  40  50  60  70  80
```

```
    0   1   2   3   4
0  10  11  12  13  14
```

```
dff.append(df1,ignore_index = True)
print(dff)
```

```
    0   1   2   3   4
0   1   2   3   4   5
1   5   6   7   8   9
2  40  50  60  70  80
```

Insertion of row at particular place

```
s1 = pd.Series([1, 2, 3])
s2 = pd.Series([4, 5, 6])
cols =  ["A", "B", "C"]
df = pd.DataFrame([list(s1), list(s2)],  columns =  cols)
print(df)
```

```
   A  B  C
0  1  2  3
1  4  5  6
```

```
new_row = [7, 8, 9]# append rows
#df.loc[-1] = new_row
#print(df)
```

```
import numpy as np
df = pd.DataFrame(np.insert(df.values, 1, new_row, axis=0))
print(df)
```

```
   0  1  2
0  1  2  3
1  7  8  9
2  4  5  6
```

```
df4=df.copy()
print(df4)
```

```
   0  1  2
0  1  2  3
1  7  8  9
2  4  5  6
```

```
df4.loc[1.5]=new_row
print(df4)
```

```
     0  1  2
0.0  1  2  3
1.0  7  8  9
2.0  4  5  6
1.5  7  8  9
```

```
sortindex=df4.sort_index()
print(sortindex)
```

```
     0  1  2
0.0  1  2  3
1.0  7  8  9
1.5  7  8  9
2.0  4  5  6
```

```
resetindex=sortindex.reset_index(drop=True)
print(resetindex)
print("\n")
print(df4)
```

```
   0  1  2
0  1  2  3
1  7  8  9
2  7  8  9
3  4  5  6


     0  1  2
0.0  1  2  3
1.0  7  8  9
2.0  4  5  6
1.5  7  8  9
```

```
df4=df4.sort_index().reset_index(drop=True)
print(df4)
```

```
   0  1  2
0  1  2  3
1  7  8  9
2  7  8  9
3  4  5  6
```

DATA visualization

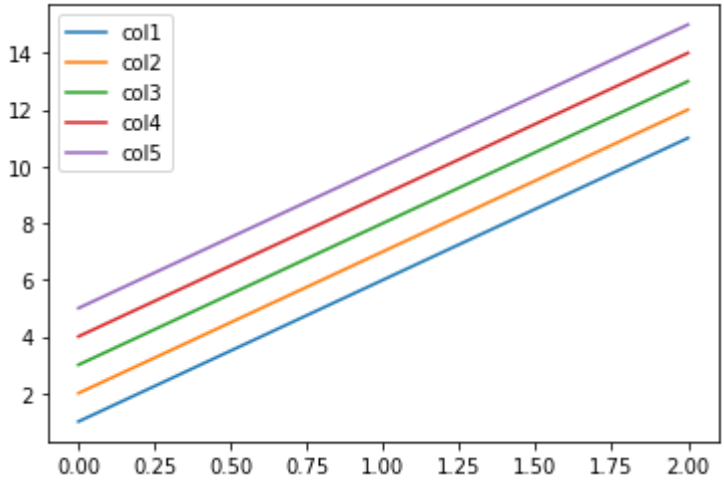
```
df=pd.DataFrame([[1,2,3,4,5],[6,7,8,9,10],[11,12,13,14,15]],columns=['col1','co
print(df)
```

```
    col1  col2  col3  col4  col5
```

0	1	2	3	4	5
1	6	7	8	9	10
2	11	12	13	14	15

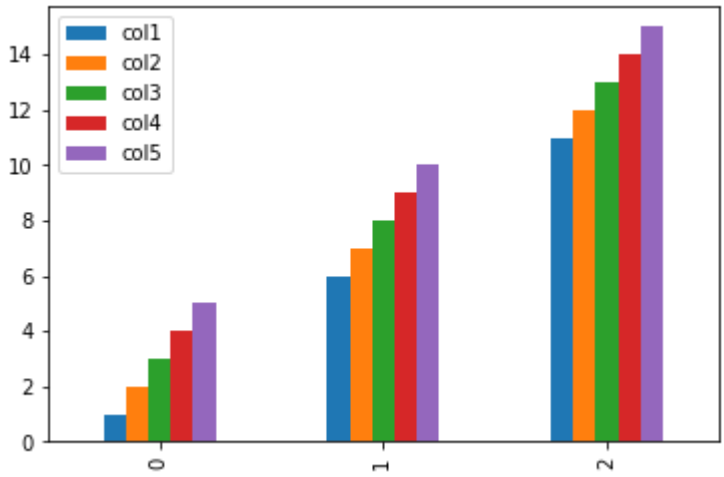
```
1 df.plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd3d7351e10>
```




```
1 df.plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd3d613d910>
```

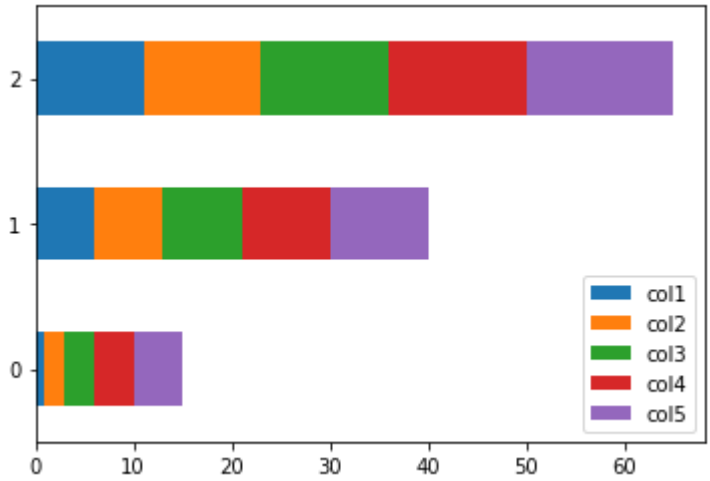



```
1 df.plot.bar(stacked=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd3d6059310>
```

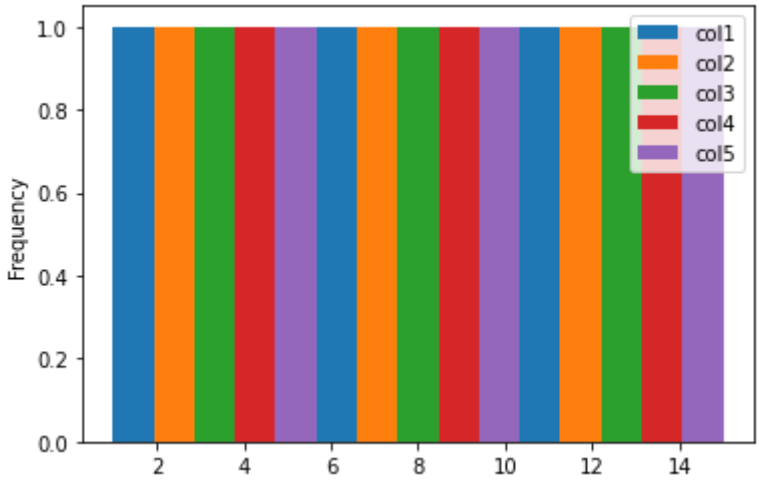
```
1 df.plot.barh(stacked=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd3d60baed0>
```




```
1 df.plot.hist(bins=15)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd3d5f5af50>
```

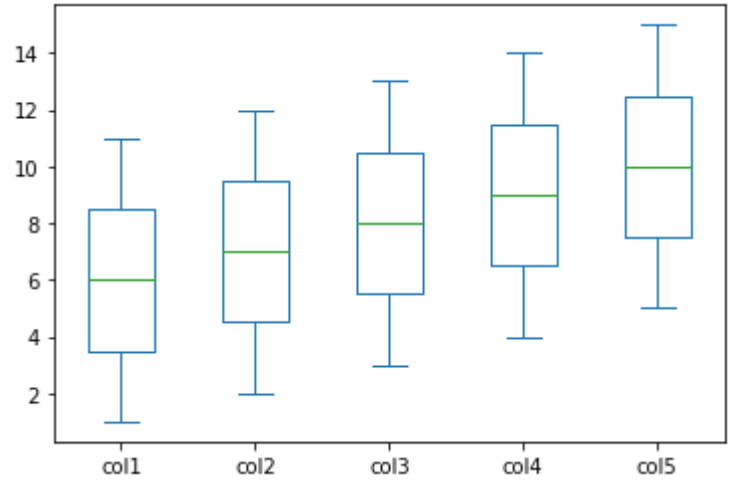



```
1 df.hist()
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fd3d5df4350>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd3d5dbcb50>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fd3d5d72f10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd3d5d36410>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fd3d5cec950>,
```

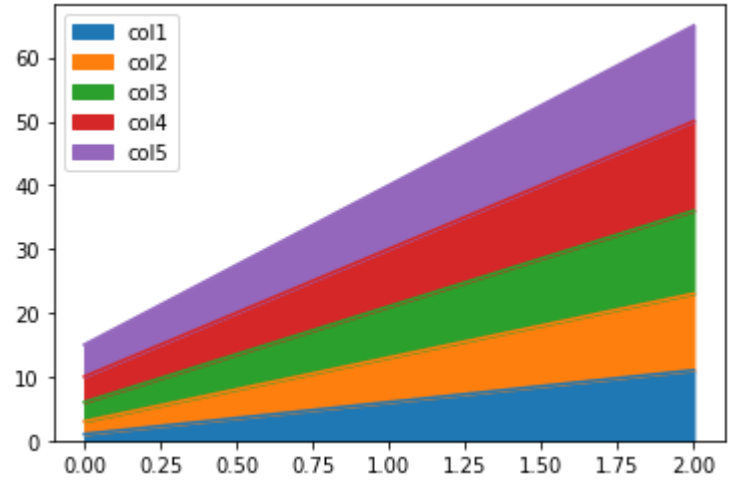
```
1 df.plot.box()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd3d5c71450>
```




```
1 df.plot.area()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd3d5b1fa90>
```

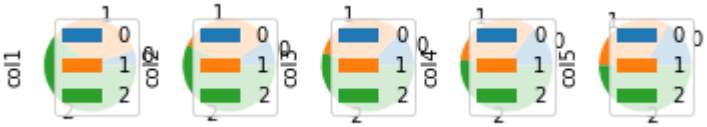



```
1 df.plot.scatter(x='col1', y='col3',s=100,c='red')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd3d59eb790>
```


```
1 df.plot.pie(subplots=True)
```

```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x7fd3d594af10>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7fd3d59789d0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7fd3d592fc90>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7fd3d58e9fd0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7fd3d58ac3d0>],
      dtype=object)
```



Save and load Table

```
1 df=pd.DataFrame([[1,2,3,4,5],[6,7,8,9,10]],columns=['col1','col2','col3','col4'
2 print(df)
```

```
      col1  col2  col3  col4  col5
row1     1     2     3     4     5
row2     6     7     8     9    10
```

```
1 df.to_csv('file1.csv')#give the file path where we need to save the table
```

```
1 dataframe = pd.read_csv('file1.csv')# give the file path
2 print(dataframe)
```

```
  Unnamed: 0  col1  col2  col3  col4  col5
0       row1     1     2     3     4     5
1       row2     6     7     8     9    10
```

if the csv file is present in google drive

```
1 from google.colab import drive
2 drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
```

```
1 path='/content/drive/MyDrive/Numpy_tutorial/'
2 df.to_csv(path+'file2.csv')# sving file in drive
```

```
1 dataframe1 = pd.read_csv(path+'file2.csv')# give the file path
2 print(dataframe1)
```

```
  Unnamed: 0  col1  col2  col3  col4  col5
0       row1     1     2     3     4     5
1       row2     6     7     8     9    10
```

saving and loading excel

```
1 df.to_excel(path+'file2.xlsx')
```

```
1 dataframe2 = pd.read_excel(path+'file2.xlsx')# give the file path
2 print(dataframe2)
```

```
  Unnamed: 0  col1  col2  col3  col4  col5
0       row1     1     2     3     4     5
1       row2     6     7     8     9    10
```

*DATE *

```
1 a=pd.date_range('1/1/2022', periods=5)
2 print (a)
3 print (a[0])
4 print (a[:3])
```

```
DatetimeIndex(['2022-01-01', '2022-01-02', '2022-01-03', '2022-01-04',
               '2022-01-05'],
              dtype='datetime64[ns]', freq='D')
2022-01-01 00:00:00
DatetimeIndex(['2022-01-01', '2022-01-02', '2022-01-03'], dtype='datetime64[n
```

```
1 print (pd.date_range('1/1/2011', periods=5,freq='M'))
```

```
DatetimeIndex(['2011-01-31', '2011-02-28', '2011-03-31', '2011-04-30',
               '2011-05-31'],
              dtype='datetime64[ns]', freq='M')
```

```
1 print (pd.date_range('1/1/2011', periods=5,freq='A'))
```

```
DatetimeIndex(['2011-12-31', '2012-12-31', '2013-12-31', '2014-12-31',
               '2015-12-31'],
              dtype='datetime64[ns]', freq='A-DEC')
```

```
1 start = pd.datetime(2011, 1, 1)
2 end = pd.datetime(2011, 1, 5)
3 print (pd.date_range(start, end))
```

```
DatetimeIndex(['2011-01-01', '2011-01-02', '2011-01-03', '2011-01-04',
               '2011-01-05'],
              dtype='datetime64[ns]', freq='D')
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning
  """Entry point for launching an IPython kernel.
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: FutureWarning
```

Iteration on Data Frame

```
# Define a dictionary containing students data
data = {'Name': ['Ankit', 'Amit',
                 'Aishwarya', 'Priyanka'],
        'Age': [21, 19, 20, 18],
        'Stream': ['Math', 'Commerce',
                   'Arts', 'Biology'],
        'Percentage': [88, 92, 95, 70]}

# Convert the dictionary into DataFrame
df = pd.DataFrame(data, columns=['Name', 'Age',
                                 'Stream', 'Percentage'])
```

1. **Index method**

```
print("Given Dataframe :\n", df)

print("\nIterating over rows using index attribute :\n")

# iterate through each row and select
# 'Name' and 'Stream' column respectively.
for ind in df.index:
    print(df['Name'][ind], df['Stream'][ind])
```

```
Given Dataframe :
         Name  Age    Stream  Percentage
0      Ankit   21      Math          88
1       Amit   19  Commerce          92
2  Aishwarya   20      Arts          95
3   Priyanka   18   Biology          70

Iterating over rows using index attribute :

Ankit Math
Amit Commerce
Aishwarya Arts
Priyanka Biology
```

2. **Location method**

```
print("Given Dataframe :\n", df)

print("\nIterating over rows using loc function :\n")

```

```
# iterate through each row and select
# 'Name' and 'Age' column respectively.
for i in range(len(df)):
    print(df.loc[i, "Name"], df.loc[i, "Age"])
```

```
Given Dataframe :
         Name  Age    Stream  Percentage
0      Ankit   21      Math          88
1       Amit   19  Commerce          92
2  Aishwarya   20      Arts          95
3   Priyanka   18   Biology          70

Iterating over rows using loc function :

Ankit 21
Amit 19
Aishwarya 20
Priyanka 18
```

```
print("Given Dataframe :\n", df)

print("\nIterating over rows using iloc function :\n")

# iterate through each row and select
# 0th and 2nd index column respectively.
for i in range(len(df)):
    print(df.iloc[i, 0], df.iloc[i, 2])
```

```
Given Dataframe :
         Name  Age    Stream  Percentage
0      Ankit   21      Math          88
1       Amit   19  Commerce          92
2  Aishwarya   20      Arts          95
3   Priyanka   18   Biology          70

Iterating over rows using iloc function :

Ankit Math
Amit Commerce
Aishwarya Arts
Priyanka Biology
```

3. **iterrows()**

```
print("\nIterating over rows using iterrows() method :\n")

# iterate through each row and select
# 'Name' and 'Age' column respectively.
for index, row in df.iterrows():
    print(row["Name"], row["Age"])
```

```
Iterating over rows using iterrows() method :
```

```
Ankit 21
Amit 19
Aishwarya 20
Priyanka 18
```

4. **itertuples()**

```
print("\nIterating over rows using itertuples() method :\n")

# iterate through each row and select
# 'Name' and 'Percentage' column respectively.
for row in df.itertuples(index=True, name='Pandas'):
    print(getattr(row, "Name"), getattr(row, "Percentage"))
```

```
Iterating over rows using itertuples() method :

Ankit 88
Amit 92
Aishwarya 95
Priyanka 70
```

5. **apply()**

```
print("\nIterating over rows using apply function :\n")

# iterate through each row and concatenate
# 'Name' and 'Percentage' column respectively.
print(df.apply(lambda row: row["Name"] + " " +
               str(row["Percentage"]), axis=1))
```

```
Iterating over rows using apply function :

0        Ankit 88
1         Amit 92
2    Aishwarya 95
3     Priyanka 70
dtype: object
```

Colab paid products - Cancel contracts here

0s completed at 9:48 AM