Pandas DataFrame

DataFrame is a widely used data structure which works with a two-dimensional array with labeled axes (rows and columns). DataFrame is defined as a standard way to store data that has two different indexes, i.e., **row index** and **column index**. It consists of the following properties:

- The columns types like int, bool, and so on.
- It can be seen as a dictionary of Series structure where both the rows and columns are indexed. It is denoted as "columns" in case of columns and "index" in case of rows.
- You can think of it like a spreadsheet or SQL table, or a dict of Series objects

data: It consists of different forms like ndarray, series, map, constants, lists, array.

index: The Default np.arrange(n) index is used for the row labels if no index is passed.

columns: The default syntax is np.arrange(n) for the column labels. It shows only true if no index is passed.

dtype: It refers to the data type of each column.

Create a DataFrame

We can create a DataFrame using following ways:

- **dict**
- **Lists**
- **Numpy ndarrrays**
- **Series**

CREATIVE INSTITUTE DATA SCIENCE

Pandas DataFrame

Create an empty DataFrame

```
import pandas as pd
df = pd.DataFrame()
print (df)
```

Create a DataFrame using List:

```
import pandas as pd
l=[10,11,12,13,14,55]

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

Create a DataFrame from Dict of ndarrays/ Lists

```
import pandas as pd
l={'id':[1,2,3],'name':['haresh','shailesh','mohit']}

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

Create a DataFrame from Dict of Series:

```
import pandas as pd

ans={'design':pd.Series([1,2,3],index=['a','b','c']),'android':pd.Series([10,11,12],index=['a','b','c'])}

rs=pd.DataFrame(ans)
print(rs)
```

CREATIVE INSTITUTE DATA SCIENCE

Pandas DataFrame

We can select any column from the DataFrame

```
import pandas as pd

ans={'design':pd.Series([1,2,3],index=['a','b','c']),'android':pd.Series([10,11,12],index=['a','b','c'])}

rs=pd.DataFrame(ans)
rs['design']
```

Column Addition

```
import pandas as pd

ans={'design':pd.Series([1,2,3],index=['a','b','c']),'android':pd.Series([10,11,12],index=['a','b','c'])}

rs=pd.DataFrame(ans)
rs['flutter']=pd.Series([20,30,40],index=['a','b','c'])
print(rs)
```

sum of column

```
import pandas as pd

ans={'design':pd.Series([1,2,3],index=['a','b','c']),'android':pd.Series([10,11,12],index=['a','b','c'])}

rs=pd.DataFrame(ans)
rs['flutter']=pd.Series([20,30,40],index=['a','b','c'])

rs['python']=rs['design']+rs['android']
print(rs)
```

CREATIVE INSTITUTE DATA SCIENCE

Pandas DataFrame

Column Deletion:

delete any column from the existing DataFrame

using del or pop funtion to delete column

```
import pandas as pd

ans={'design':pd.Series([1,2,3],index=['a','b','c']),'android':pd.Series([10,11,12],index=['a','b','c'])}

rs=pd.DataFrame(ans)
rs['flutter']=pd.Series([20,30,40],index=['a','b','c'])

rs['python']=rs['design']+rs['android']
print(rs)

del rs['design']
print(rs)

rs.pop('design')
```

Row Selection, Addition, and Deletion

We can easily select, add, or delete any row at anytime

Selection by Label: using LOC

select any row by passing the row label to a loc function.

```
import pandas as pd

ans={'design':pd.Series([1,2,3],index=['a','b','c']),'android':pd.Series([10,11,12],index=['a','b','c'])}

rs=pd.DataFrame(ans)
rs['flutter']=pd.Series([20,30,40],index=['a','b','c'])

rs['python']=rs['design']+rs['android']
print(rs)
print(rs.loc['a'])
```

CREATIVE INSTITUTE DATA SCIENCE

Pandas DataFrame

Select single row

```
df.loc['r2']
```

Select single column

```
df.loc[:,"Courses"]
```

Select Multiple rows

```
df.loc[["r1",'r2']]
```

Select Multiple columns

```
df.loc[:,["Courses",'Fee']]
```

Select row range

```
df.loc["r1":'r3']
```

select column range

```
df.loc[:,"Fee":"Discount"]
```

select alternative row

```
df.loc["r1":'r5':2]
```

select alternative column

```
df.loc[:,'Courses':"Discount":2]
```

Using Condition

```
df.loc[df['Fee']>=24000]
```

CREATIVE INSTITUTE DATA SCIENCE

Pandas DataFrame

Selection by integer location: Pandas iloc[]

pandas.DataFrame.iloc[] is a property that is used to select rows and columns by position/index. If the position/index does not exist, it gives an index error

- START is the integer index of the row/column.
- STOP is the integer index of the last row/column where you wanted to stop the selection, and
- STEP as the number of indices to advance after each extraction.

Some point to note about iloc[].

- By not providing a start index, iloc[] selects from the first row/column.
- By not providing stop, iloc[] selects all rows/columns from the start index.
- Providing both start and stop, selects all rows/columns in between.

Select Single Row & Column By Index

The rows can also be selected by passing the integer location to an iloc function.

```
print(rs.iloc[1])
```

select column by Index

```
df.iloc[:,0]
```

Select Multiple Rows & Columns by Index

```
df.iloc[[0,4]]
```

Pandas DataFrame

column

```
df.iloc[:,[0,1,2]]
```

Select Rows or Columns by Index Range

```
df.iloc[0:4]
```

Select Columns between two Indexes

```
df.iloc[:,1:4]
```

Select Alternate Rows

```
df.iloc[0:5:2]
```

Select Alternate Columns

```
df.iloc[:,0:4:2]
```

Using Conditions with iloc[]

```
df.iloc[list(df['Fee']>=20000)]
```

Addition of rows:

easily add new rows to the DataFrame using append function. It adds the new rows at the end.

```
import pandas as pd

a1=pd.DataFrame([[10,12],[21,22]],columns=['a','b'])
b1=pd.DataFrame([[33,44],[55,66]],columns=['a','b'])

print(a1)
print(b1)
c1=a1.append(b1)

print(c1)
```

Pandas DataFrame

Deletion of rows:

We can delete or drop any rows from a DataFrame using the **index** label. If in case, the label is duplicate then multiple rows will be deleted

```
import pandas as pd

a1=pd.DataFrame([[10,12],[21,22]],columns=['a','b'])
b1=pd.DataFrame([[33,44],[55,66]],columns=['a','b'])

print(a1)
print(b1)

c1=a1.append(b1)

d1=c1.drop(0)
print(d1)
```