PANDAS DATAFRAMES

	ID	NAME	DEPT	SEX	EXPERIENCE
0	101	JOHN	ENT	M	12
1	104	SMITH	ORTHPEDIC	M	5
2	107	GEORGE	CARDIOLOGY	M	10
3	109	LARA	SKIN	F	3
4	113	K GEORGE	MEDICINE	F	9
5	115	JOHNSON	ORTHPEDIC	M	10

Made by: Vineeta Garg

DATAFRAME CREATION

SYNTAX

pandas.DataFrame(data, index, column)

One can easily change the default order of row labels to user defined row labels using the index parameter. It can be used to select only desired rows instead of all rows. Also, the columns parameter in the DataFrame() method can be used to change the sequence of DataFrame columns or to display only selected columns.

CUSTOMIZING LABELS

EXAMPLE

```
import pandas as pd
import numpy as np
a=[[1,"Amit"],[2,"Chetan"],[3,"Rajat"],[4,"Vimal"]]
b=pd.DataFrame(a, index=[1,2,3,4], columns=["Roll No.", "Name"])
print(b)
```

Using List of Dictionaries

EXAMPLE

```
import pandas as pd
l1=[{101:"Amit",102:"Binay",103:"Chahal"}, {102:"Arjun",103:"Fazal"}]
df=pd.DataFrame(l1)
print(df)
```

Using List of Lists

EXAMPLE

```
import pandas as pd
a=[[1,"Amit"],[2,"Chetan"],[3,"Rajat"],[4,"Vimal"]]
b=pd.DataFrame(a)
print(b)
```

Dictionary of Lists

EXAMPLE

```
import pandas as pd
d1={"Rollno":[1,2,3], "Total":[350,5,400,420], "Percentage":[70,80,84]}
df1=pd.DataFrame(d1)
print(df1)
```

Dictionary of Series

EXAMPLE

```
import pandas as pd
d2={"Rollno":pd.Series([1,2,3,4]), "Total":pd.Series([350,5,400,420]),
"Percentage":pd.Series([70,80,84,80])}
df2=pd.DataFrame(d2)
print(df2)
```

ROW/COLUMN OPERATIONS

ADDING ROW

Adding new row using loc() method.

```
BILL.loc[4]=["Scale", 5, 5, 15]
```

ADDING COLUMN

To add a new column Amount to DataFrame BILL by multiplying Quantity and Price.

```
BILL['Amount']=BILL['Quantity']*BILL['Price']
```

Adding new column using loc() method.

```
BILL.loc[:,"Amount"]=[20,80,25]
```

Adding neww column as the last column

```
BILL['Amount']=[230,340,140]
```

Insert the column Amount in the dataframe BILL at the second position

```
BILL.insert(1,"Amount",[20,80,25])
```

RENAMING ROW

The DataFrame.rename() method is used to rename the labels of rows.

```
BILL.rename({'0':'R1', '1':'R2', '2':'R3', '3':'R4'}, axis=' index ')
```

RENAMING COLUMN

The DataFrame.rename() method is used to rename the labels of columns.

```
BILL.rename({'Item Name':'Iname', 'Amount':'Amt',
'Quantity':'Qty', 'Price':'Pr'}, axis='columns')
```

DELETING COLUMN

Deleting the column Amount from the dataframe BILL

```
BILL.drop("Amount", axis=1)
```

DELETING ROW

Deleting the row with label 2 from dataframe BILL

```
BILL.drop(2, axis=0)
```

ACCESSING DATAFRAMES

INDEXING

Positional indexes are used to extract a data element present at a particular index location from a dataframe using iloc() method.

Label indexes are used to extract a data element present at a particular index label from a dataframe using loc() method.

Boolean indexing is used to filter data by applying certain condition on data using relational operators like ==, >, <, <=, >= and logical operators like ~(not), &(and) and |(or)

SLICING

Slicing is used to extract a subset of a DataFrame. You can specify beginning parameter (beg) and end parameter (end) to indicate the size of the slice to be extracted from the dataframe.

head() & tail() functions

The head function is used to return a specified number of rows from the beginning of a DataFrame. If no parameter is passed it returns top 5 rows.

The tail function is used to return a specified number of rows from the end of a DataFrame. If no parameter is passed it returns top 5 rows.

ITERATING OVER DATAFRAMES

iteritems(): Helps to iterate over each element of the set, column-wise.

itertuple(): Helps to iterate over each row and form a tuple out of them.

iterrows(): Helps to iterate over each element of the set, row-wise.