PANDAS IN DATA SCIENCE

Presented By

Pranab Bora
A Student of Rajiv Gandhi University, Itanagar

Series

```
import pandas as pd
data = [1,2,3,4]
srs = pd.Series(data)
print(srs)
print(type(srs))
```

Change index

```
import pandas as pd
data = [1,2,3,4]
srs = pd.Series(data, index=['a','b','c','d'])
print(srs)
print(type(srs))
```

Print from to that indexing

```
import pandas as pd
data = [1,2,3,4]
srs = pd.Series(data, index=['a','b','c','d'])
print(srs)
print(srs['a':'c'])
```

Data Frame

```
import pandas as pd
data=[[1,2,3,4,5],[3,4,5]]
df=pd.DataFrame(data)
print(df)
print(type(df))

#dictionary
import pandas as pd
data={"a":[1,2,3,4,5],"b":[3,4,5,6,7]}
df=pd.DataFrame(data)
print(df)

data = {'Name':['Tom', 'nick', 'krish', 'jack'],
        'Age':[20, 21, 19, 18]}
df=pd.DataFrame(data)
print(df)
```

Change column name
df.rename(columns={'Name':'Username'},inplace=True)

Change row name
df.rename(index={'0':'a'},inplace=True)

Change row and column name
df.rename(columns={'Age':'Ages'},index={'1':'b'},inplace=True)

print(df.rename(columns=str.lower, index=str.title))

print(df.add_prefix('X_'))
print(df.add_suffix('_X'))

Concat or outer joining

```
df1=pd.DataFrame({"A":[1,2,3], "B":[4,5,6]})
df2=pd.DataFrame({"A":[2,3,4], "C":[5,6,7]})
print(df1)
print(df2)
df=pd.concat([df1,df2])
print("After concate")
print(df)
```

Inner Join row

```
df1=pd.DataFrame({"A":[1,2,3], "B":[4,5,6]})
df2=pd.DataFrame({"A":[2,3,4], "C":[5,6,7]})
print(df1)
print(df2)
df=pd.concat([df1,df2], axis=0, join='inner')
print("After concate")
print(df)
```

Inner Join col

```
df1=pd.DataFrame({"A":[1,2,3], "B":[4,5,6]})
df2=pd.DataFrame({"A":[2,3,4], "C":[5,6,7]})
print(df1)
print(df2)
df=pd.concat([df1,df2], axis=1, join='inner')
print("After concate")
print(df)
```

#basically we don't much use concat

It depends on how to represent your data is

Merge

```
import pandas as pd
left = pd.DataFrame({
   'id':[1,2,3,4,5],
   'Name': ['Alex', 'Amy', 'Allen', 'Alice', 'Ayoung'],
   'subject_id':['sub1','sub2','sub4','sub6','sub5']})
right = pd.DataFrame(
   {'id':[1,2,3,4,5],
   'Name': ['Billy', 'Brian', 'Bran', 'Bryce', 'Betty'],
   'subject_id':['sub2','sub4','sub3','sub6','sub5']})
print (left)
print (right)

pd.merge(left,right,on='id')
#common

pd.merge(left,right,on='id')
#common

pd.merge(left,right,on=['id','subject_id'])
#two common
```

```
pd.merge(left, right, on='subject_id', how='left')
#Belonging all left table value

pd.merge(left, right, on='subject_id', how='right')
#Belonging all right table value

pd.merge(left, right, on='subject_id', how='inner')
#Belonging all common value of two tables

pd.merge(left, right, on='subject_id', how='outer')
#include all the values of the table
```

Read CSV or Excel file

```
import pandas as pd
cars=pd.read_csv(r"mtcars.csv")
#pd.read_excel(filename="", sheetname="")
Print(cars)
```

Returns first 5 row

```
cars.head()
```

Returns first 10 row

```
cars.head(10)
```

Returns last 5 row

```
cars.tail()
```

Returns last 10 row

```
cars.tail(10)
```

See not null values
cars.info()

cars.mean()
cars.median()
cars.std()
cars.max()
cars.min()

Also for not null
cars.count()

For all mean median std min max etc
cars.describe()

Rename column
cars=cars.rename(columns={'model':'Username'})
cars

Filling empty place mean()

```
cars.qsec=cars.qsec.fillna(cars.qsec.mean())
Cars
```

Delete unnecessary column

```
cars=cars.drop(columns=['S_no'])
Cars
```

Correlation

```
import pandas as pd
df=cars[['mpg','cyl','disp','hp','drat','wt','qsec','vs','am','gear','carb']].corr()
Df
```

```
import pandas as pd
df=cars[['mpg','cyl','disp','hp']].corr()
df
```

Find values in the location(integer)
cars.iloc[5,4]
#iloc[row, col]

cars.iloc[0:5,4]
#iloc[row, col]

cars.iloc[5,0:4]
#iloc[row, col]

cars.iloc[5:,4:]
#iloc[row, col]

Indexing with labeling(str)
cars.loc[0:,'mpg':'disp']
#iloc[row, col]

cars.loc[0:,'mpg':]
#iloc[row, col]

cars['am']

Changing column value

```
cars['am']=1
Cars
```

Filtering

```
cars[cars['vs']==0]
cars[cars['vs']==1]
```

Adding new column with some operation(lambda)

```
f=lambda x:x*2
cars['2am']=cars['am'].apply(f)
Cars
```

```
f=lambda x:x/2
cars['mpg/2']=cars['mpg'].apply(f)
Cars
```

Remove column

```
cars=cars.drop(columns=['mpg/2'])
cars
```

Sorting
ascending

```
cars=cars.sort_values(by='mpg')
Cars
```

descending

```
cars=cars.sort_values(by='mpg', ascending=False)
Cars
```

Filter

```
f=cars['cyl']>6
f
cars[f]
```

```
f=cars['cyl']>6
f
cars[f]
```

```
f=(cars['cyl']>6) & (cars['hp']>300)
f
cars[f]
```

Thank You !!!!