Today Topics

- Pandas

Pandas

- PanelData
- used for data analysis,manipulation and cleaning
  - anlysing the data attributes/columns and statistics of the dataset
  - transforming the data
    - we can concatenate,merge and join multiple dataframes
      - cleaning of data
      - removal of null data points
- 2Data Structures
  1. Series
     - sequential data arranged in either side by side or line by line
     - *module()*
       - pandas.Series(data items)
  2. DataFrame
     - data is arranged in the form of a frame/table
       - it contains rows (index) and columns(attributes/features)
     - *pd.DataFrame()*

series

- sequential data items/points

In [1]:

```python
# str data
import pandas as pd
```

In [2]:

```python
pd.__version__
```

```
...
```

In [4]:

```python
# read a str from user and apply series function on it
st=input("enter the string:")# dynamic str
s=pd.Series(st)
s
```

```
...
```

In [6]:

```
# read a str from user and apply series function on it
st=input("enter the string:").split()# dynamic str
li=pd.Series(st)
li
```

...

In [7]:

```
tp=tuple(st)
pd.Series(tp)
```

...

In [4]:

```
import pandas as pd
tps=(90,34,56,'hi','hello',34,'python','LBRCE')
ss=pd.Series(tps)
ss
```

...

In [ ]:

```
rn=
```

In [5]:

```
ss.dtype # object
```

Out[5]:

```
dtype('O')
```

In [9]:

```
# create the series using dict,range and numpy array
dic={'k1':"hi",'k2':"hello",3:"lbrce",4:"good"}
d=pd.Series(dic)
d
```

...

In [10]:

```
# using range()
pd.Series(range(11))
```

...

In [11]:

```
pd.Series(range(10,25))
```

...

In [14]:

```
pd.Series(range(100,400,7))
```

...

In [15]:

```
import numpy as np
```

In [17]:

```
ar=np.array([num for num in range(10,21)])
pd.Series(ar)
```

...

In [49]:

```
ar=np.array([[num for num in range(10,15)],[num for num in range(12,17)]])
ar
```

Out[49]:

```
array([[10, 11, 12, 13, 14],
       [12, 13, 14, 15, 16]])
```

In [30]:

```
na=np.arange(10,50,5) # nd array:step_count=5
ss=pd.Series(na,index=[num for num in range(20,28)])
ss
```

...

In [34]:

```
rn=np.random.randint(100,1000,300) # 300 values
srn=pd.Series(rn)
srn.index=[num for num in range(600,900)]
srn
```

...

In [36]:

```
# iteration of series of data items
for item in srn:
    print(item) # dict/str/tuple/list
```

...

DataFrame

- rows and columns
- 2d data structure

In [37]:

```python
# dict and 2d array
dic={'name':'Ruthu','Org':'APSSDC','work':'Trainer',
    12:90}
dic # value is assigned to a key
```

Out[37]:

```
{'name': 'Ruthu', 'Org': 'APSSDC', 'work': 'Trainer', 12: 90}
```

In [38]:

```python
'''
key arranged as colum
val arranged as row data '''
```

Out[38]:

```
'\nkey arranged as colum\nval arranged as row data '
```

In [43]:

```python
pd.DataFrame(dic,index=[1])
```

...

In [44]:

```python
pd.DataFrame(dic,index=[num for num in range(5,10)])
```

...

In [48]:

```python
dc={'Int':[1,2,3,4],'float':(56.9,90.45,12.4,89.45),
    'str':['clg','dept','teaching','nn-tchng']}
df=pd.DataFrame(dc)
df.index=['R1','R2','R3','R4']
df
```

...

In [55]:

```python
# try with 2d array
ar=np.array([[num for num in range(10,15)],[num for num in range(12,17)]])
adf=pd.DataFrame(ar,columns=['C1','C2','C3','C4','C5'],
                 index=['first','second'])
adf
```

...

In [56]:

```python
# prepare2 equal size dataframes
# try with 2d array
df
```

...

In [59]:

```
df2=pd.DataFrame({34:'thiryt four',23:90,
                  'tuple':(9,8,4,7),'list':[9,6,4,8]})
df2
```

...

In [61]:

```
pd.concat([df,df2]) # combining the 2equal size
```

...

In [62]:

```
df.join(df2) # joining the multiple dataframes
```

...

In [63]:

```
df3=pd.DataFrame({'int':'thiryt four',23:90,
                  'str':(9,8,4,7),'list':[9,6,4,8]})
df3
```

...

In [64]:

```
df.join(df3)
```

...

In [66]:

```
#merging the
df3=pd.DataFrame({'int_rate':[2,1,2,3],
                 'int_gdp':[50,45,45,67]},
                index=[2001,2002,2003,2204])
df3
```

...

In [67]:

```
df4=pd.DataFrame({'low_tier_hpi':(80,90,70,60),
                 'unemployment':[1,3,5,6]},
                index=[2001,2002,2003,2005])
df4
```

...

In [68]:

```
df3.join(df4) # you adding extra columns
```

...

In [69]:

```
first=pd.DataFrame({'a':[4,5,6,7],'b':[30,50,60,70],
                    'c':('a','b','c','d')},
                  index=['f','s','th','ft'])
first
```

...

In [70]:

```
second=pd.DataFrame({'a':[4,9,6,7],'b':[30,90,60,70],
                    'c':('a','f','c','d')},
                  index=['f','s','th','ft'])
second
```

...

In [73]:

```
first.merge(second)
# non-similar col elements are disappeared
```

...

In [75]:

```
pd.concat([first,second])
```

...

In [77]:

```
third=pd.DataFrame({'x':[4,9,16,7],'y':[30,90,60,70],
                    'z':('a','f','cx','d')},
                  index=['fs','s','th','ft'])
third
```

...

In [78]:

```
first.join(third) # adding extra colums in a frame
```

...

In [80]:

```
first.merge(second,on="c") # merging the multiple dfs
```

...

In [81]:

```
first.merge(second,on='a')
```

...

In [82]:

```
first.columns # column names
```

...

In [83]:

```
second.index # index values
```

...

In [86]:

```
# date_range()
dates=pd.date_range('2022-07-20','2022-07-30')
dt=pd.Series(dates)
dt
```

...

In [87]:

```
dates
```

...

In [89]:

```
# periods
date=pd.date_range("2022-07-22",periods=10)
date
```

...

In [92]:

```
temp=np.random.randint(30,40,10)
temp
```

...

In [94]:

```
spd=pd.Series(temp,index=date)
spd
```

...

In [105]:

```
tm=pd.DataFrame(temp,index=date,
                columns=['Temperature'])
tm
```

Out[105]:

	Temperature
2022-07-22	38
2022-07-23	35
2022-07-24	39
2022-07-25	33
2022-07-26	31
2022-07-27	30
2022-07-28	33
2022-07-29	30
2022-07-30	39
2022-07-31	30

In [107]:

```
tm.index[0]
```

...

In [109]:

```
tm.shape # dataframe
```

...

In [111]:

```
ss.shape # Series
```

...

In [112]:

```
import pandas as pd
```

In [117]:

```
marks=pd.read_csv('marks.csv')
marks # dataset
```

...

In [134]:

```
marks.sample() # generates a random sample
```

```
...
```

In [139]:

```
# 3 random samples
marks.sample(3)
```

```
...
```

In [144]:

```
marks.head(2) # first 2 rows
```

```
...
```

In [145]:

```
marks.head() # first 5 rows by default
```

```
...
```

In [127]:

```
marks.tail() # last 5 rows
```

```
...
```

In [129]:

```
marks.tail(2) # last 2 rows
```

```
...
```

In [152]:

```
marks.describe() # describes the dataset
# statistics of the ds
```

```
...
```

In [150]:

```
marks.info()
```

```
...
```

In [153]:

```
marks.info
```

```
...
```

In [159]:

```
marks[['CS']] # df
```

```
...
```

In [160]:

```
marks['CS'] # series
```

```
...
```

In [158]:

```
marks[['CS','EM']] # 2D format
```

```
...
```

In [161]:

```
# column name as attribute
marks.EM
```

```
...
```

In [162]:

```
marks.EM.value_counts() # counts the unq values
```

```
...
```

In [163]:

```
marks.EM.sum() # summation of entire colums
```

```
...
```

In [165]:

```
marks.CS.all # retrival of CS attribute
```

```
...
```

In [168]:

```
marks.isnull() # checking the null chars
```

```
...
```

In [170]:

```
new=pd.read_csv("marks.csv")
new.isnull()
```

```
...
```

In [171]:

```
df=pd.read_csv("marks.csv",usecols=['PS','LDIC'])
df # you can get the mentioned cols data
```

```
...
```

In [172]:

```
df['PS'].value_counts()
```

```
...
```

In [174]:

```
marks
```

```
...
```

In [175]:

```
marks.sort_values('LDIC') # dataset is rearranged
```

```
...
```

In [176]:

```
marks
```

```
...
```

In [177]:

```
marks.sort_values('Names')
```

```
...
```

In [184]:

```
marks[:10] # slicing
```

```
...
```

In [188]:

```
marks['Total']=100
```

In [189]:

```
marks
```

```
...
```

In [190]:

```
help(marks.drop)
```

```
...
```

In [ ]:

```

```

In [197]:

```
marks.drop('Total',axis=1)
```

...

In [198]:

```
marks_df=pd.read_csv('marks.csv')
marks_df
```

...

In [200]:

```
new=marks_df.sort_values('Names')
new
```

...

In [204]:

```
new.index=[num for num in range(1,21)]
new
```

...

In [214]:

```
new['Total']=new['CS']+new['PS']+new['LDIC']+new['EM']
new
```

...

In [221]:

```
new[:]
```

...

In [226]:

```
# using slicing
new.iloc[:,1:5]
```

...

In [228]:

```
new.iloc[2] # getting row wise data
```

...

In [229]:

```
new
```

...

In [231]:

```
new.iloc[4,2] # 3rd value in 4th row
```

Out[231]:

```
86
```

In [235]:

```
res=pd.read_csv("marks.csv")
res
```

...

In [236]:

```
# check for null chars
res.isnull().sum()
```

...

In [237]:

```
res.isna().sum()
```

...

In [238]:

```
res.isnull()
```

...

In [240]:

```
final=res.dropna()
final # cleaning of data
```

...

In [241]:

```
from sklearn.datasets import load_iris
```

In [242]:

```
data=load_iris()
data
```

Out[242]:

```
{'data': array([[5.1, 3.5, 1.4, 0.2],
        [4.9, 3. , 1.4, 0.2],
        [4.7, 3.2, 1.3, 0.2],
        [4.6, 3.1, 1.5, 0.2],
        [5. , 3.6, 1.4, 0.2],
        [5.4, 3.9, 1.7, 0.4],
        [4.6, 3.4, 1.4, 0.3],
        [5. , 3.4, 1.5, 0.2],
        [4.4, 2.9, 1.4, 0.2],
        [4.9, 3.1, 1.5, 0.1],
        [5.4, 3.7, 1.5, 0.2],
        [4.8, 3.4, 1.6, 0.2],
        [4.8, 3. , 1.4, 0.1],
        [4.3, 3. , 1.1, 0.1],
        [5.8, 4. , 1.2, 0.2],
        [5.7, 4.4, 1.5, 0.4],
        [5.4, 3.9, 1.3, 0.4],
        [5.1, 3.5, 1.4, 0.3],
```

In [245]:

```
data['Target']=data.target
```

In [249]:

```
dtf=data['Target']
dtf
```

Out[249]:

```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

In [250]:

```
data.at
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\__init__.py:11
7, in Bunch.__getattr__(self, key)
    116 try:
--> 117     return self[key]
    118 except KeyError:

KeyError: 'columns'

During handling of the above exception, another exception occurred:

AttributeError                            Traceback (most recent call last)
Input In [250], in <cell line: 1>()
----> 1 data.columns

File C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\__init__.py:11
9, in Bunch.__getattr__(self, key)
    117     return self[key]
    118 except KeyError:
--> 119     raise AttributeError(key)

AttributeError: columns
```

In [ ]:

```

```