pandas-assignment

January 12, 2023

#

Pandas Assignment

Import pandas and numpy with their aliases

[1]:
```
import numpy as np
import pandas as pd
```

Create a variable a = pd.Series([100, 200, 300, 400])

[2]:
```
a =pd.Series([100,200,300,400])
```

Print a, and data type

[3]:
```
a
```

[3]:
```
0    100
1    200
2    300
3    400
dtype: int64
```

Using indexing access the element 300 from the series a.

[4]:
```
print(a[2])
```

```
300
```

What are the values of index for series a?

[5]:
```
#getting index
print(a.index)
```

```
RangeIndex(start=0, stop=4, step=1)
```

[6]:
```
#getting values
print(a.values)
```

```
[100 200 300 400]
```

Change the index to [‘c’, ‘a’, ‘b’, ‘d’]

[7]:
```
a=pd.Series([100,200,300,400],index=['c','a','b','d'])
print(a)
```

```
c    100
a    200
b    300
d    400
dtype: int64
```

Access the value in the series with index ‘d’

[8]:
```
print(a['d'])
```

```
400
```

Sort the values wrt to the index and print it

[9]:
```
a.sort_index()
```

[9]:
```
a    200
b    300
c    100
d    400
dtype: int64
```

Create a new Pandas Series b having index as ‘e’, ‘f’, and ‘g’ and value 800,450,100 and print it

[10]:
```
b =pd.Series([800,450,100],index=['e','f','g'])
print(b)
```

```
e    800
f    450
g    100
dtype: int64
```

Append b series at the end of a series

[11]:
```
a = a.append(b)
```

[12]:
```
#print a again after appending b into it
a
```

[12]:
```
c    100
a    200
b    300
d    400
e    800
f    450
```

```
g    100
dtype: int64
```

Sort the values in descending order of a and print the index of the sorted series

[13]:
```
a.sort_index()
```

[13]:
```
a    200
b    300
c    100
d    400
e    800
f    450
g    100
dtype: int64
```

[14]:
```
a.sort_index(ascending=False)
```

[14]:
```
g    100
f    450
e    800
d    400
c    100
b    300
a    200
dtype: int64
```

0.1 Pandas DataFrame

0.1.1 Part 1

Create a pandas dataframe df from the series 'a' that we used in the last section, print the dataframe

[15]:
```
df = pd.DataFrame(a)
df
```

[15]:
```
     0
c  100
a  200
b  300
d  400
e  800
f  450
g  100
```

[15]:
```

```

What is the shape of the datafarme (also, what does it imply?)

[16]:
```
df.shape
```

[16]:
```
(7, 1)
```

Hey! remember shape (7,1) implies dataframe has 7 rows and 1 column.

What is the index of the dataframe, is it same as the series 'a'

[17]:
```
# yep its same as the series.
df.index
```

[17]:
```
Index(['c', 'a', 'b', 'd', 'e', 'f', 'g'], dtype='object')
```

print the head and tail of the dataframe. Additional - (what does head and tali represent?)

[18]:
```
#head prints 1st 5 rows and tail represents the last 5 rows.
df.head()
```

[18]:

	0
c	100
a	200
b	300
d	400
e	800

[19]:
```
df.tail()
```

[19]:

	0
b	300
d	400
e	800
f	450
g	100

Rename the column of the dataframe as 'points'

[20]:
```
df.rename(columns={0:'points'},inplace=True)
```

Create another Series 'fruits', which contains random names of fruits from ['orange','mango','apple']. The series should contain 7 elements, randomly selected from ['orange','mango','apple']

[21]:
```
#Create fruits array

fruits =['orange','mango','apple','mango','apple','apple','orange']
```

[22]:
```
#Create series fruits out of fruits array

fruits =pd.Series(['orange','mango','apple','mango','apple','apple','orange'])
```

Change the index of fruits to the index of dataframe df

[23]:
```
fruits =pd.
↪Series(['orange','mango','apple','mango','apple','apple','orange'],index=['c',␣
↪'a', 'b', 'd', 'e', 'f', 'g'])


fruits
```

[23]:
```
c    orange
a     mango
b     apple
d     mango
e     apple
f     apple
g    orange
dtype: object
```

Add this fruits series as a new column to the dataframe df with its column name as 'fruits' print the head of the dataframe to verify

[24]:
```
df['fruits']=fruits
```

[25]:
```
df
```

[25]:

	points	fruits
c	100	orange
a	200	mango
b	300	apple
d	400	mango
e	800	apple
f	450	apple
g	100	orange

0.2 Pandas Concatenation

Create a dataframe d1 where the cols are 'city' : ['Chandigarh', 'Delhi', 'Kanpur', 'Chennai', 'Manali'] and 'Temperature' : [15, 22, 20, 26,-2]

[26]:
```
d1 = pd.DataFrame({
    'city' : ['Chandigarh', 'Delhi', 'Kanpur', 'Chennai', 'Manali' ],
    'Temperature' : [15, 22, 20, 26,-2],
})
```

Print(d1)

[27]:
```
print(d1)
```

```
          city  Temperature
0  Chandigarh           15
1       Delhi           22
2      Kanpur           20
3     Chennai           26
4      Manali           -2
```

What is the shape of d1.

[28]:
```
d1.shape
```

[28]:
```
(5, 2)
```

Set city = d1[‘city’]

[29]:
```
city=d1['city']
```

print city What is the type of city.

[30]:
```
print(city,type(city))
```

```
0    Chandigarh
1         Delhi
2        Kanpur
3       Chennai
4        Manali
Name: city, dtype: object <class 'pandas.core.series.Series'>
```

Create another datafeame ‘d2’ where the columns are ‘city’ - [‘Bengalaru’,‘Coimbatore’,‘Srirangam’,‘Pondicherry’] ‘Temperature’ - [24,35,36,39]

[31]:
```
d2 = pd.DataFrame({
    'city':['Bengaluru','Coimbatore','Srirangam','Pondicherry'],
    'Temperature':[24,35,36,39],
})
```

print the shape of this dataframe

[32]:
```
d2.shape
```

[32]:
```
(4, 2)
```

merge the two dataframes together, save it in a new dataframe named ‘d3’

[33]:
```
d3 =pd.concat([d1,d2])
d3
```

[33]:
```
          city  Temperature
0   Chandigarh           15
1        Delhi           22
```

```
2       Kanpur           20
3      Chennai           26
4       Manali           -2
0    Bengaluru           24
1   Coimbatore           35
2    Srirangam           36
3  Pondicherry           39
```

Select the part of the dataframe such that it contains cities wherer temp is less then or equal to 20 How many cities are there?

[34]:
```
# dfreq=d3['Temperature'].where(d3['Temperature']<=20).count()

dfreq=d3.where(d3['Temperature']<=20)
```

[35]:
```
dfreq
```

[35]:
```
          city  Temperature
0  Chandigarh         15.0
1         NaN          NaN
2      Kanpur         20.0
3         NaN          NaN
4      Manali         -2.0
0         NaN          NaN
1         NaN          NaN
2         NaN          NaN
3         NaN          NaN
```

Select the part of the dataframe such that it contains the cities where tempearature greater than or equal to 35

[36]:
```
# dfreq1=d3['Temperature'].where(d3['Temperature']>=35).count()
dfreq1=d3.where(d3['Temperature']>=35)
print(dfreq1)
```

```
          city  Temperature
0          NaN          NaN
1          NaN          NaN
2          NaN          NaN
3          NaN          NaN
4          NaN          NaN
0          NaN          NaN
1   Coimbatore         35.0
2    Srirangam         36.0
3  Pondicherry         39.0
```

0.3 Applying functions to columns and creating new columns

We need to create another column in d3, which contains a boolean value for each city to indicate whether it's a union territory or not. - HINT: Chandigarh, Pondicherry and Delhi are only 3 union territories here.

[38]:
```
import numpy as np
conditions = [d3['city'] == 'Chandigarh',d3['city'] == 'Pondicherry',d3['city']␣
 ↪== 'Delhi']
values =[1,1,1]
d3['territory'] = np.select(conditions, values)
```

[39]:
```
# print d3
d3
```

[39]:

	city	Temperature	territory
0	Chandigarh	15	1
1	Delhi	22	1
2	Kanpur	20	0
3	Chennai	26	0
4	Manali	-2	0
0	Bengaluru	24	0
1	Coimbatore	35	0
2	Srirangam	36	0
3	Pondicherry	39	1

[40]:
```
#we can also use:
union_terrs = ["Chandigarh", "Pondicherry", "Delhi"]
d3["territory"] = d3["city"].isin(union_terrs).astype(int)
```

[41]:
```
d3
```

[41]:

	city	Temperature	territory
0	Chandigarh	15	1
1	Delhi	22	1
2	Kanpur	20	0
3	Chennai	26	0
4	Manali	-2	0
0	Bengaluru	24	0
1	Coimbatore	35	0
2	Srirangam	36	0
3	Pondicherry	39	1

The temperatures mentioned in 'Temperature' column are mentioned in Celsius, we need another column which contains the same in Fahrenheit.

HINT - - Define a function c_to_f which takes input temp in celsius and returns a value with temperature in Fahrenheit. - To check: c_to_f(10) should return 50.

[42]:
```
# write function here
def c_to_f(x):
  f=x*1.8+32
  return f
```

[44]:
```
# check function c_to_f(10)
print(c_to_f(10))
```

```
50.0
```

[51]:
```
d3['temp_fahrenheit']=d3['Temperature'].apply(lambda x: x*1.8+32)
```

[52]:
```
d3
```

[52]:

	city	Temperature	territory	temp_fahrenheit
0	Chandigarh	15	1	59.0
1	Delhi	22	1	71.6
2	Kanpur	20	0	68.0
3	Chennai	26	0	78.8
4	Manali	-2	0	28.4
0	Bengaluru	24	0	75.2
1	Coimbatore	35	0	95.0
2	Srirangam	36	0	96.8
3	Pondicherry	39	1	102.2

0.4 Indexing and selecting rows in DataFrame

Select subset of the dataframe d1 such that it contains the cities which are union territories.

[58]:
```
conditions = [d1['city'] == 'Chandigarh',d1['city'] == 'Pondicherry',d1['city']␣
 ↪== 'Delhi']
values =[1,1,1]
d1['territory'] = np.select(conditions, values)
print(d1.where(d1['territory']==1))
```

```
         city  Temperature  territory
0  Chandigarh         15.0        1.0
1       Delhi         22.0        1.0
2         NaN          NaN        NaN
3         NaN          NaN        NaN
4         NaN          NaN        NaN
```

[55]:
```
d1
```

[55]:

	city	Temperature	territory
0	Chandigarh	15	1
1	Delhi	22	1
2	Kanpur	20	0

```
3     Chennai           26          0
4      Manali           -2          0
```

Select a subset of the dataframe d1 such that it contains the cities which only have temperature above 90 Farenhiet.

[59]:
```
d1['temp_fahrenheit']=d1['Temperature'].apply(lambda x: x*1.8+32)
d1
```

[59]:
```
          city  Temperature  territory  temp_fahrenheit
0   Chandigarh           15          1             59.0
1        Delhi           22          1             71.6
2       Kanpur           20          0             68.0
3      Chennai           26          0             78.8
4       Manali           -2          0             28.4
```

[60]:
```
print(d1.where(d1['temp_fahrenheit']>90))
```

```
  city  Temperature  territory  temp_fahrenheit
0  NaN          NaN        NaN              NaN
1  NaN          NaN        NaN              NaN
2  NaN          NaN        NaN              NaN
3  NaN          NaN        NaN              NaN
4  NaN          NaN        NaN              NaN
```

Select only the first three rows of the dataframe d1.

[61]:
```
d1.head(3)
```

[61]:
```
          city  Temperature  territory  temp_fahrenheit
0   Chandigarh           15          1             59.0
1        Delhi           22          1             71.6
2       Kanpur           20          0             68.0
```

Select all the rows and last two columns in the dataframe.

[66]:
```
#df.iloc[row_start:row_end , col_start, col_end]
d1.iloc[:,-2:]
```

[66]:
```
   territory  temp_fahrenheit
0          1             59.0
1          1             71.6
2          0             68.0
3          0             78.8
4          0             28.4
```

0.5 Groupby

[69]:
```
# Create a dataframe using dictionary of your choice


import pandas as pd
technologies   = ({
    'Courses':
 ↪["Spark","PySpark","Hadoop","Python","Pandas","Hadoop","Spark","Python"],
    'Fee' :[22000,25000,23000,24000,26000,25000,25000,22000],
    'Duration':
 ↪['30days','50days','55days','40days','60days','35days','55days','50days'],
    'Discount':[1000,2300,1000,1200,2500,1300,1400,1600]
                })
df = pd.DataFrame(technologies, columns=['Courses','Fee','Duration','Discount'])
print(df)
```

```
   Courses    Fee Duration  Discount
0    Spark  22000   30days      1000
1  PySpark  25000   50days      2300
2   Hadoop  23000   55days      1000
3   Python  24000   40days      1200
4   Pandas  26000   60days      2500
5   Hadoop  25000   35days      1300
6    Spark  25000   55days      1400
7   Python  22000   50days      1600
```

[71]:
```
# Use Groupby of single column with aggregate sum()

da2=df.groupby('Courses').sum()
da2
```

[71]:
```
           Fee  Discount
Courses
Hadoop   48000      2300
Pandas   26000      2500
PySpark  25000      2300
Python   46000      2800
Spark    47000      2400
```

[72]:
```
# Use Groupby of single column with aggregate count()

da3=df.groupby('Courses').count()
da3
```

[72]:
```
         Fee  Duration  Discount
Courses
```

```
Hadoop      2          2          2
Pandas      1          1          1
PySpark     1          1          1
Python      2          2          2
Spark       2          2          2
```

[73]:
```
# Use Groupby of single column with aggregate min() and max()

da4=df.groupby('Discount').min()
da4
```

[73]:
```
          Courses    Fee Duration
Discount
1000       Hadoop  22000   30days
1200       Python  24000   40days
1300       Hadoop  25000   35days
1400        Spark  25000   55days
1600       Python  22000   50days
2300      PySpark  25000   50days
2500       Pandas  26000   60days
```

[74]:
```
da5=df.groupby('Discount').max()
da5
```

[74]:
```
          Courses    Fee Duration
Discount
1000        Spark  23000   55days
1200       Python  24000   40days
1300       Hadoop  25000   35days
1400        Spark  25000   55days
1600       Python  22000   50days
2300      PySpark  25000   50days
2500       Pandas  26000   60days
```

[76]:
```
# Use Groupby of any 2 columns with aggregate mean()

da6 = df.groupby(['Courses','Duration'])['Fee'].sum()
print(da6)
```

```
Courses  Duration
Hadoop   35days      25000
         55days      23000
Pandas   60days      26000
PySpark  50days      25000
Python   40days      24000
         50days      22000
Spark    30days      22000
```

```
          55days      25000
Name: Fee, dtype: int64
```

[78]:
```
# Use Groupby of any 2 columns with aggregate min() and max()

da7 = df.groupby(['Fee','Duration']).min()
print(da7)
```

```
                Courses  Discount
Fee   Duration
22000 30days      Spark      1000
      50days     Python      1600
23000 55days     Hadoop      1000
24000 40days     Python      1200
25000 35days     Hadoop      1300
      50days    PySpark      2300
      55days      Spark      1400
26000 60days     Pandas      2500
```

[80]:
```
da8 = df.groupby(['Fee','Duration']).max()
print(da8)
```

```
                Courses  Discount
Fee   Duration
22000 30days      Spark      1000
      50days     Python      1600
23000 55days     Hadoop      1000
24000 40days     Python      1200
25000 35days     Hadoop      1300
      50days    PySpark      2300
      55days      Spark      1400
26000 60days     Pandas      2500
```

0.6 Data Range

Create a pandas daterange where starting date is 1st of January,2020 and end date is 1st of April 2021, store it in a new variable named 'a'

[81]:
```
a = pd.date_range(start='2020-01-01',end='2022-04-01')
```

print a

[82]:
```
a
```

[82]:
```
DatetimeIndex(['2020-01-01', '2020-01-02', '2020-01-03', '2020-01-04',
               '2020-01-05', '2020-01-06', '2020-01-07', '2020-01-08',
               '2020-01-09', '2020-01-10',
               …
               '2022-03-23', '2022-03-24', '2022-03-25', '2022-03-26',
```

```
               '2022-03-27', '2022-03-28', '2022-03-29', '2022-03-30',
               '2022-03-31', '2022-04-01'],
              dtype='datetime64[ns]', length=822, freq='D')
```

What is the len of a?

[83]:
```
len(a)
```

[83]:
```
822
```

What is the type of a?

[84]:
```
type(a)
```

[84]:
```
pandas.core.indexes.datetimes.DatetimeIndex
```

[ ]:
```

```