Pandas Assignment

Import pandas and numpy with their aliases

```
In [1]: import pandas as pd
```

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

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

Print a, and data type

```
In [3]: a.head()
```

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

```
In [4]: type(a)
```

```
Out[4]: pandas.core.series.Series
```

Using indexing access the element 300 from the series a.

```
In [5]: a.iloc[2]
```

```
Out[5]: 300
```

What are the values of index for series a?

```
In [6]: 
        a.index
```

```
Out[6]: RangeIndex(start=0, stop=4, step=1)
```

Change the index to ['c', 'a', 'b', 'd']

In [7]:

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


a.head()
```

Out[7]:

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

Access the value in the series with index 'd'

In [8]:

```
a.loc["d"]
```

Out[8]:

```
400
```

Sort the values wrt to the index and print it

In [9]:

```
a.sort_index()
```

Out[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

In [10]:

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

Append b series at the end of a series

In [11]:

```
a = a.append(b)
```

```
C:\Users\91883\AppData\Local\Temp\ipykernel_15616\536917386.py:1: FutureWarni
ng: The series.append method is deprecated and will be removed from pandas in
a future version. Use pandas.concat instead.
  a = a.append(b)
```

In [12]:

```
#print a again after appending b into it

a
```

Out[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

In [13]:

```
a.sort_values(ascending=False)
```

Out[13]:

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

In [14]:

```

a.index
```

Out[14]:

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

Pandas DataFrame

Part 1

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

In [15]:

```
df = pd.DataFrame(a)


df
```

Out[15]:

	0
c	100
a	200
b	300
d	400
e	800
f	450
g	100

In [16]:

```
df.shape
```

Out[16]:

```
(7, 1)
```

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

In [17]:

```

df.shape
```

Out[17]:

```
(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'

In [18]:
```
# yep its same as the series.
df.index
```

Out[18]:
```
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?)

In [19]:
```
df.head()
```

Out[19]:

	0
c	100
a	200
b	300
d	400
e	800

In [20]:
```

df.tail()
```

Out[20]:

	0
b	300
d	400
e	800
f	450
g	100

Rename the column of the dataframe as 'points'

In [21]:
```
df.columns = [ 'points']
df
```

Out[21]:

	points
c	100
a	200
b	300
d	400
e	800
f	450
g	100

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']

In [22]:
```
#Create fruits array
import numpy as np
fruits = np.array( ['orange','mango','apple'])
```

In [23]:
```
#Create series fruits out of fruits array

fruits = pd.Series([np.random.choice(fruits)])
fruits
```

Out[23]:
```
0    orange
dtype: object
```

Change the index of fruits to the index of dataframe df

In [ ]:
```
```

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

In [ ]:

```
```

In [ ]:

```
```

Pandas Concatenation

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

In [24]:

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

Print(d1)

In [25]:

```

d1
```

Out[25]:

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

What is the shape of d1.

In [26]:
```
d1.shape
```

Out[26]:
```
(5, 2)
```

Set city = d1['city']

In [27]:
```

d1.city = d1['city']
```

print city
What is the type of city.

In [28]:
```
print(d1.city)
type(d1.city)
```

```
0    Chandigarh
1         Delhi
2        Kanpur
3       Chennai
4        Manali
Name: city, dtype: object
```

Out[28]:
```
pandas.core.series.Series
```

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

In [29]:
```
d2 = pd.DataFrame({
     'city' : ['Bengalaru','Coimbatore','Srirangam','Pondicherry'],
     'Temperature' :  [24,35,36,39]
    })
```

print the shape of this dataframe

In [30]:
```
d2.shape
```

Out[30]:
```
(4, 2)
```

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

In [31]:

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


d3
```

Out[31]:

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

In [32]:

```
d3[d3['Temperature'] <= 20][['city']]
```

Out[32]:

	city
0	Chandigarh
2	Kanpur
4	Manali

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

In [33]:

```
d3[d3['Temperature'] >= 35][['city']]
```

Out[33]:

	city
1	Coimbatore
2	Srirangam
3	Pondicherry

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.

In [34]:

```
# write function here

# def is_ut(x):

#     # write code below

# d3['is_ut'] =

#d4 = pd.DataFrame({'is_ut':['yes','yes','no','no','no','no','no','no','yes',]
d3['is_ut'] =['yes','yes','no','no','no','no','no','no','yes',]
```

In [35]:

```
# print d3
d3
```

Out[35]:

	city	Temperature	is_ut
0	Chandigarh	15	yes
1	Delhi	22	yes
2	Kanpur	20	no
3	Chennai	26	no
4	Manali	-2	no
0	Bengalaru	24	no
1	Coimbatore	35	no
2	Srirangam	36	no
3	Pondicherry	39	yes

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.

In [36]:

```
# write function here


d3[' Fahrenheit'] =[49,71.6,32,78.8,28.4,75.2,95,96.8,48.2,]

d3
```

Out[36]:

	city	Temperature	is_ut	Fahrenheit
0	Chandigarh	15	yes	49.0
1	Delhi	22	yes	71.6
2	Kanpur	20	no	32.0
3	Chennai	26	no	78.8
4	Manali	-2	no	28.4
0	Bengalaru	24	no	75.2
1	Coimbatore	35	no	95.0
2	Srirangam	36	no	96.8
3	Pondicherry	39	yes	48.2

In [37]:

```
# check function c_to_f(10)
```

In [38]:

```
# apply function c_to_f to d3 to create a column 'temp_farenhiet'

d3
```

Out[38]:

	city	Temperature	is_ut	Fahrenheit
0	Chandigarh	15	yes	49.0
1	Delhi	22	yes	71.6
2	Kanpur	20	no	32.0
3	Chennai	26	no	78.8
4	Manali	-2	no	28.4
0	Bengalaru	24	no	75.2
1	Coimbatore	35	no	95.0
2	Srirangam	36	no	96.8
3	Pondicherry	39	yes	48.2

Indexing and selecting rows in DataFrame

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

In [39]:

```
d3[d3['is_ut'] == 'yes'][['city']]
```

Out[39]:

	city
0	Chandigarh
1	Delhi
3	Pondicherry

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

In [43]:

```
d3[d3['Fahrenheit'] > 90.0][['city']]
data[data['Age'] <= 18][['Name']]
```

Select only the first three rows of the dataframe d1.

In [44]:

```
d3.head(3)
```

Out[44]:

	city	Temperature	is_ut	Fahrenheit
0	Chandigarh	15	yes	49.0
1	Delhi	22	yes	71.6
2	Kanpur	20	no	32.0

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

In [45]:

```
d4 = d3[['is_ut', 'Temperature']]
d4
```

Out[45]:

	is_ut	Temperature
0	yes	15
1	yes	22
2	no	20
3	no	26
4	no	-2
0	no	24
1	no	35
2	no	36
3	yes	39

Groupby

In [46]:

```
# Create a dataframe using dictionary of your choice

data = {
    'Name' : ['Ankit', 'Aishwarya', 'Shaurya', 'Shivangi'],
    'Age' : [23, 21, 22, 21],
    'University' : ['BHU', 'JNU', 'DU', 'BHU'],
}

# creating a Dataframe object
df = pd.DataFrame(data)

df
```

Out[46]:

	Name	Age	University
0	Ankit	23	BHU
1	Aishwarya	21	JNU
2	Shaurya	22	DU
3	Shivangi	21	BHU

In [47]:

```
# Use Groupby of single column with aggregate sum()


print(df.groupby(['Name','Age'])[['University']])
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x0000025EC56406D0>
```

In [48]:

```
# Use Groupby of single column with aggregate count()

df.groupby(['Name','Age'])[['University']].count()
```

Out[48]:

		University
Name	**Age**	
Aishwarya	**21**	1
Ankit	**23**	1
Shaurya	**22**	1
Shivangi	**21**	1

In [49]:

```
# Use Groupby of single column with aggregate min() and max()

df.groupby(by='Name').min('Age')
```

Out[49]:

	Age
Name	
Aishwarya	21
Ankit	23
Shaurya	22
Shivangi	21

In [50]:

```
df.groupby(by='Name').max('Age')
```

Out[50]:

	Age
Name	
Aishwarya	21
Ankit	23
Shaurya	22
Shivangi	21

In [51]:
```
# Use Groupby of any 2 columns with aggregate mean()

df.groupby(['Name','Age']).mean()
```

```
C:\Users\91883\AppData\Local\Temp\ipykernel_15616\2329545415.py:3: FutureWarn
ing: Dropping invalid columns in DataFrameGroupBy.mean is deprecated. In a fu
ture version, a TypeError will be raised. Before calling .mean, select only c
olumns which should be valid for the function.
  df.groupby(['Name','Age']).mean()
```

Out[51]:

Name	Age
Aishwarya	21
Ankit	23
Shaurya	22
Shivangi	21

In [52]:
```
# Use Groupby of any 2 columns with aggregate min() and max()

df.groupby(['Name','Age']).min()
df.groupby(['Name','Age']).max()
```

Out[52]:

		University
Name	Age	
Aishwarya	21	JNU
Ankit	23	BHU
Shaurya	22	DU
Shivangi	21	BHU

In [ ]:

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'

In [53]:
```
a = pd.date_range(start ='1-1-2020',
         end ='1-04-2021', freq ='5H')
```

print a

```
In [54]: 

         a
```

```
Out[54]: DatetimeIndex(['2020-01-01 00:00:00', '2020-01-01 05:00:00',
                        '2020-01-01 10:00:00', '2020-01-01 15:00:00',
                        '2020-01-01 20:00:00', '2020-01-02 01:00:00',
                        '2020-01-02 06:00:00', '2020-01-02 11:00:00',
                        '2020-01-02 16:00:00', '2020-01-02 21:00:00',
                        ...
                        '2021-01-02 02:00:00', '2021-01-02 07:00:00',
                        '2021-01-02 12:00:00', '2021-01-02 17:00:00',
                        '2021-01-02 22:00:00', '2021-01-03 03:00:00',
                        '2021-01-03 08:00:00', '2021-01-03 13:00:00',
                        '2021-01-03 18:00:00', '2021-01-03 23:00:00'],
                       dtype='datetime64[ns]', length=1772, freq='5H')
```

What is the len of a?

```
In [55]: 

         len(a)
```

```
Out[55]: 1772
```

What is the type of a?

```
In [56]: type(a)
```

```
Out[56]: pandas.core.indexes.datetimes.DatetimeIndex
```

```
In [ ]: 
```