Pandas Assignment

Import pandas and numpy with their aliases

In [6]:
```python
import pandas as pd
import numpy as np
```

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

In [7]:
```python
a = pd.Series([100, 200, 300, 400])
print(a)
```

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

Print a, and data type

In [8]:
```python
print(a)
print("\nData type of Series 'a':", type(a))
```

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

Data type of Series 'a': <class 'pandas.core.series.Series'>
```

Using indexing access the element 300 from the series a.

In [9]:
```python
element_300 = a[2]
print("Element 300 from the Series 'a':", element_300)
```

```
Element 300 from the Series 'a': 300
```

What are the values of index for series a?

In [10]:
```python
a = pd.Series([100, 200, 300, 400])
print("Index values for Series 'a':", a.index)
```

```
Index values for Series 'a': RangeIndex(start=0, stop=4, step=1)
```

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

In [11]:
```python
a.index = ['c', 'a', 'b', 'd']
print("Series 'a' with new index:")
print(a)
```

```
Series 'a' with new index:
c    100
a    200
b    300
d    400
dtype: int64
```

Access the value in the series with index 'd'

In [12]:
```python
value_d = a['d']
print("Value in the Series with index 'd':", value_d)
```

```
Value in the Series with index 'd': 400
```

Sort the values wrt to the index and print it

In [13]:
```python
sorted_a = a.sort_index()
print("Sorted Series 'a' with respect to the index:")
print(sorted_a)
```

```
Sorted Series 'a' with respect to the index:
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 [14]:
```python
b = pd.Series([800, 450, 100], index=['e', 'f', 'g'])

# Printing the Series 'b'
print(b)
```

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

Append b series at the end of a series

In [15]:
```python
a_appended = pd.concat([a, b])

# Printing the appended Series
print("Appended Series:")
print(a_appended)
```

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

In [16]:
```python
#print a again after appending b into it
print(a_appended)
```

```
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 [17]:
```python
sorted_a_appended = a_appended.sort_values(ascending=False)

# Print the index of the sorted series
print(sorted_a_appended.index)
```

```
Index(['e', 'f', 'd', 'b', 'a', 'c', '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 [18]:
```python
df = pd.DataFrame(a_appended, columns=['Values'])

print("DataFrame 'df' from Series 'a':")
print(df)
```

```
DataFrame 'df' from Series 'a':
   Values
c     100
a     200
b     300
d     400
e     800
f     450
g     100
```

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

In [19]:
```python
print("Shape of the DataFrame 'df':", df.shape)
```

```
Shape of the DataFrame 'df': (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 [20]:
```python
# yep its same as the series.
print("Index of the DataFrame 'df':", df.index)
```

```
Index of the DataFrame 'df': 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 [21]:
```python
print("Head of the DataFrame 'df':")
print(df.head())
```

```
Head of the DataFrame 'df':
   Values
c     100
a     200
b     300
d     400
e     800
```

In [22]:
```python
print("\nTail of the DataFrame 'df':")
print(df.tail())
```

```
Tail of the DataFrame 'df':
   Values
b     300
d     400
e     800
f     450
g     100
```

Rename the column of the dataframe as 'points'

In [23]:
```python
df = df.rename(columns={'Values': 'points'})
print("DataFrame 'df' with column 'points':")
print(df)
```

```
DataFrame 'df' with column 'points':
   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 [24]:
```python
#Create fruits array
import random
fruit_list = ['orange', 'mango', 'apple']
fruits = pd.Series(random.choices(fruit_list, k=7))
print("Series 'fruits' with random fruit names:")
print(fruits)
```

```
Series 'fruits' with random fruit names:
0     apple
1     apple
2     apple
3     apple
4    orange
5     apple
6    orange
dtype: object
```

In [25]:
```python
#Create series fruits out of fruits array

fruits_array = ['orange', 'mango', 'apple', 'orange', 'mango', 'apple', 'orange']
fruits = pd.Series(fruits_array)
print("Series 'fruits' from the fruits array:")
print(fruits)
```

```
Series 'fruits' from the fruits array:
0    orange
1     mango
2     apple
3    orange
4     mango
5     apple
6    orange
dtype: object
```

Change the index of fruits to the index of dataframe df

In [26]:
```python
fruits.index = df.index
print("Series 'fruits' with the index matching DataFrame 'df':")
print(fruits)
```

```
Series 'fruits' with the index matching DataFrame 'df':
c    orange
a     mango
b     apple
d    orange
e     mango
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

In [27]:
```python
df['fruits'] = fruits
```

In [28]:
```python
print("Head of the DataFrame 'df' with the new 'fruits' column:")
print(df.head())
```

```
Head of the DataFrame 'df' with the new 'fruits' column:
   points  fruits
c     100  orange
a     200   mango
b     300   apple
d     400  orange
e     800   mango
```

Pandas Concatenation

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

In [29]:
```python
data = {
    'city': ['Chandigarh', 'Delhi', 'Kanpur', 'Chennai', 'Manali'],
    'Temperature': [15, 22, 20, 26, -2]
}

d1 = pd.DataFrame(data)
```

Print(d1)

In [30]:
```python
print("DataFrame 'd1' with city and Temperature:")
print(d1)
```

```
DataFrame 'd1' with city and Temperature:
         city  Temperature
0  Chandigarh           15
1       Delhi           22
2      Kanpur           20
3     Chennai           26
4      Manali           -2
```

What is the shape of d1.

In [31]:
```python
d1.shape
```

Out[31]:
```
(5, 2)
```

Set city = d1['city']

In [32]:
```python
city = d1['city']
```

print city
What is the type of city.

In [33]:
```python
# Print the variable 'city'
print("Variable 'city':")
print(city)
```

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

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

In [34]:
```python
# Creating the data for DataFrame 'd2'
data_d2 = {
    'city': ['Bengaluru', 'Coimbatore', 'Srirangam', 'Pondicherry'],
    'Temperature': [24, 35, 36, 39]
}

# Creating the DataFrame 'd2'
d2 = pd.DataFrame(data_d2)

# Printing the DataFrame 'd2'
print("DataFrame 'd2' with city and Temperature:")
print(d2)
```

```
DataFrame 'd2' with city and Temperature:
          city  Temperature
0    Bengaluru           24
1   Coimbatore           35
2    Srirangam           36
3  Pondicherry           39
```

print the shape of this dataframe

In [35]:
```python
d2.shape
```

Out[35]:
```
(4, 2)
```

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

In [36]:
```python
# d3 =
d3 = pd.concat([d1, d2], ignore_index=True)
print("Merged DataFrame 'd3':")
print(d3)
```

```
Merged DataFrame 'd3':
          city  Temperature
0   Chandigarh           15
1        Delhi           22
2       Kanpur           20
3      Chennai           26
4       Manali           -2
5    Bengaluru           24
6   Coimbatore           35
7    Srirangam           36
8  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 [37]:
```python
selected_cities = d3[d3['Temperature'] <= 20]
print("Cities where temperature is less than or equal to 20:")
print(selected_cities)
num_cities = len(selected_cities)
print("Number of cities where temperature is less than or equal to 20:", num_cities)
```

```
Cities where temperature is less than or equal to 20:
         city  Temperature
0  Chandigarh           15
2      Kanpur           20
4      Manali           -2
Number of cities where temperature is less than or equal to 20: 3
```

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

In [38]:
```python
selected_cities_high_temp = d3[d3['Temperature'] >= 35]
print("Cities where temperature is greater than or equal to 35:")
print(selected_cities_high_temp)
num_cities_high_temp = len(selected_cities_high_temp)
print("Number of cities where temperature is greater than or equal to 35:", num_cities_high_temp)
```

```
Cities where temperature is greater than or equal to 35:
          city  Temperature
6   Coimbatore           35
7    Srirangam           36
8  Pondicherry           39
Number of cities where temperature is greater than or equal to 35: 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.

In [54]:
```python
def is_ut(city):
    union_territories = ['Chandigarh', 'Pondicherry', 'Delhi']
    if city in union_territories:
        return True
    else:
        return False
d3['is_ut'] = d3['city'].apply(is_ut)
```

In [55]:
```python
# print d3
print("DataFrame 'd3' with is_ut column:")
print(d3)
```

```
DataFrame 'd3' with is_ut column:
          city  Temperature  is_ut
0   Chandigarh           15   True
1        Delhi           22   True
2       Kanpur           20  False
3      Chennai           26  False
4       Manali           -2  False
5    Bengaluru           24  False
6   Coimbatore           35  False
7    Srirangam           36  False
8  Pondicherry           39   True
```

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 [56]:
```python
# write function here

def c_to_f(celsius_temp):
    fahrenheit_temp = (celsius_temp * 9/5) + 32
    return fahrenheit_temp
d3['Temperature_F'] = d3['Temperature'].apply(c_to_f)
print("DataFrame 'd3' with Temperature_F column (in Fahrenheit):")
print(d3)
```

```
DataFrame 'd3' with Temperature_F column (in Fahrenheit):
          city  Temperature  is_ut  Temperature_F
0   Chandigarh           15   True           59.0
1        Delhi           22   True           71.6
2       Kanpur           20  False           68.0
3      Chennai           26  False           78.8
4       Manali           -2  False           28.4
5    Bengaluru           24  False           75.2
6   Coimbatore           35  False           95.0
7    Srirangam           36  False           96.8
8  Pondicherry           39   True          102.2
```

In [57]:
```python
# check function c_to_f(10)
def c_to_f(celsius_temp):
    fahrenheit_temp = (celsius_temp * 9/5) + 32
    return fahrenheit_temp

print("Temperature in Fahrenheit for 10 Celsius:", c_to_f(10))
```

```
Temperature in Fahrenheit for 10 Celsius: 50.0
```

In [58]:
```python
# apply function c_to_f to d3 to create a column 'temp_farenhiet'
d3['temp_fahrenheit'] = d3['Temperature'].apply(c_to_f)
print("DataFrame 'd3' with temp_fahrenheit column (in Fahrenheit):")
print(d3)
```

```
DataFrame 'd3' with temp_fahrenheit column (in Fahrenheit):
          city  Temperature  is_ut  Temperature_F  temp_fahrenheit
0   Chandigarh           15   True           59.0             59.0
1        Delhi           22   True           71.6             71.6
2       Kanpur           20  False           68.0             68.0
3      Chennai           26  False           78.8             78.8
4       Manali           -2  False           28.4             28.4
5    Bengaluru           24  False           75.2             75.2
6   Coimbatore           35  False           95.0             95.0
7    Srirangam           36  False           96.8             96.8
8  Pondicherry           39   True          102.2            102.2
```

Indexing and selecting rows in DataFrame

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

In [61]:
```python
# List of union territories
union_territories = ['Chandigarh', 'Pondicherry', 'Delhi']
subset_d1_ut = d1[d1['city'].isin(union_territories)]
print("Subset of DataFrame d1 with cities that are union territories:")
print(subset_d1_ut)
```

```
Subset of DataFrame d1 with cities that are union territories:
         city  Temperature
0  Chandigarh           15
1       Delhi           22
```

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

In [62]:
```python
subset_d1_above_90F = d1[d1['Temperature'] > 90]
print("Subset of DataFrame d1 with cities having temperatures above 90 Fahrenheit:")
print(subset_d1_above_90F)
```

```
Subset of DataFrame d1 with cities having temperatures above 90 Fahrenheit:
Empty DataFrame
Columns: [city, Temperature]
Index: []
```

Select only the first three rows of the dataframe d1.

In [63]:
```python
subset_d1_first_three_rows = d1.iloc[:3]
print("First three rows of DataFrame d1:")
print(subset_d1_first_three_rows)
```

```
First three rows of DataFrame d1:
         city  Temperature
0  Chandigarh           15
1       Delhi           22
2      Kanpur           20
```

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

In [64]:
```python
subset_d1_last_two_columns = d1.iloc[:, -2:]
print("All rows and last two columns of DataFrame d1:")
print(subset_d1_last_two_columns)
```

```
All rows and last two columns of DataFrame d1:
         city  Temperature
0  Chandigarh           15
1       Delhi           22
2      Kanpur           20
3     Chennai           26
4      Manali           -2
```

Groupby

In [65]:
```python
# Create a dataframe using dictionary of your choice
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David'],
    'Age': [25, 30, 35, 40],
    'City': ['New York', 'Los Angeles', 'Chicago', 'Houston']
}

# Create a DataFrame using the dictionary
df = pd.DataFrame(data)

# Print the DataFrame
print("DataFrame created from dictionary:")
print(df)
```

```
DataFrame created from dictionary:
      Name  Age         City
0    Alice   25     New York
1      Bob   30  Los Angeles
2  Charlie   35      Chicago
3    David   40      Houston
```

In [66]:
```python
# Use Groupby of single column with aggregate sum()
# Sample DataFrame
data = {
    'Category': ['A', 'B', 'A', 'B', 'A'],
    'Value': [10, 20, 30, 40, 50]
}

df = pd.DataFrame(data)
result = df.groupby('Category').aggregate({'Value': 'sum'})
print(result)
```

```
          Value
Category       
A            90
B            60
```

In [67]:
```python
# Use Groupby of single column with aggregate count()

result = df.groupby('Category').aggregate({'Value': 'count'})
print(result)
```

```
          Value
Category       
A             3
B             2
```

In [68]:
```python
# Use Groupby of single column with aggregate min() and max()
result = df.groupby('Category').aggregate({'Value': ['min', 'max']})
print(result)
```

```
         Value    
           min max
Category          
A           10  50
B           20  40
```

In [69]:
```python
# Use Groupby of any 2 columns with aggregate mean()
data = {
    'Category': ['A', 'B', 'A', 'B', 'A'],
    'Subcategory': ['X', 'Y', 'X', 'Y', 'X'],
    'Value': [10, 20, 30, 40, 50]
}

df = pd.DataFrame(data)
result = df.groupby(['Category', 'Subcategory']).aggregate({'Value': 'mean'})

# Print the result
print(result)
```

```
                      Value
Category Subcategory       
A        X             30.0
B        Y             30.0
```

In [70]:
```python
# Use Groupby of any 2 columns with aggregate min() and max()
data = {
    'Category': ['A', 'B', 'A', 'B', 'A'],
    'Subcategory': ['X', 'Y', 'X', 'Y', 'X'],
    'Value': [10, 20, 30, 40, 50]
}

df = pd.DataFrame(data)
result = df.groupby(['Category', 'Subcategory']).aggregate({'Value': ['min', 'max']})

print(result)
```

```
                     Value    
                       min max
Category Subcategory          
A        X              10  50
B        Y              20  40
```

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 [71]:
```python
a = pd.date_range(start='2020-01-01', end='2021-04-01')
```

print a

In [72]:
```python
print(a)
```

```
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',
               ...
               '2021-03-23', '2021-03-24', '2021-03-25', '2021-03-26',
               '2021-03-27', '2021-03-28', '2021-03-29', '2021-03-30',
               '2021-03-31', '2021-04-01'],
              dtype='datetime64[ns]', length=457, freq='D')
```

What is the len of a?

In [73]:
```python
length_of_a = len(a)
print("Length of 'a':", length_of_a)
```

```
Length of 'a': 457
```

What is the type of a?

In [74]:
```python
type_of_a = type(a)
print("Type of 'a':", type_of_a)
```

```
Type of 'a': <class 'pandas.core.indexes.datetimes.DatetimeIndex'>
```