pandas advance assginment

June 30, 2023

Q1. List any five functions of the pandas library with execution.

—> functions of the pandas library are given below :

1)head() 2)tail() 3)dtype() 4)drop() 6)dropna() 7)describe() 8)DataFrame() 9)Series() 10)info() 11).t 12)apply() 13)fillna() 14)concate() 15)iloc() 16)loc() 17)min() 18)max()

executon with sample data set

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

```
[2]: df=pd.read_csv('taxonomy.csv.xls')
```

```
[3]: df.head()
```

```
[3]:   taxonomy_id                      name parent_id     parent_name
     0         101                 Emergency       NaN             NaN
     1      101-01         Disaster Response       101       Emergency
     2      101-02            Emergency Cash       101       Emergency
     3   101-02-01         Help Pay for Food    101-02  Emergency Cash
     4   101-02-02   Help Pay for Healthcare    101-02  Emergency Cash
```

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

```
[4]:     taxonomy_id                           name parent_id           parent_name
     285   111-01-07               Workplace Rights    111-01  Advocacy & Legal Aid
     286      111-02                      Mediation       111                 Legal
     287      111-03                         Notary       111                 Legal
     288      111-04                 Representation       111                 Legal
     289      111-05  Translation & Interpretation       111                 Legal
```

```
[5]: df.dtypes
```

```
[5]: taxonomy_id    object
     name           object
     parent_id      object
     parent_name    object
     dtype: object
```

[6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 290 entries, 0 to 289
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   taxonomy_id  290 non-null    object
 1   name         290 non-null    object
 2   parent_id    279 non-null    object
 3   parent_name  279 non-null    object
dtypes: object(4)
memory usage: 9.2+ KB
```

[7]:
```
df.describe()
```

[7]:
```
       taxonomy_id          name  parent_id       parent_name
count          290           290        279               279
unique         290           183         60                50
top            101  Nursing Home  106-06-07  Health Education
freq             1             4         11                15
```

[8]:
```
df.min()
```

```
/tmp/ipykernel_1257/3962516015.py:1: FutureWarning: The default value of
numeric_only in DataFrame.min is deprecated. In a future version, it will
default to False. In addition, specifying 'numeric_only=None' is deprecated.
Select only valid columns or specify the value of numeric_only to silence this
warning.
  df.min()
```

[8]:
```
taxonomy_id        101
name           12-Step
dtype: object
```

[9]:
```
df.max()
```

```
/tmp/ipykernel_1257/1299571182.py:1: FutureWarning: The default value of
numeric_only in DataFrame.max is deprecated. In a future version, it will
default to False. In addition, specifying 'numeric_only=None' is deprecated.
Select only valid columns or specify the value of numeric_only to silence this
warning.
  df.max()
```

[9]:
```
taxonomy_id               111-05
name           Workplace Rights
dtype: object
```

```
[10]: df.drop('name', axis=1)
```

```
[10]:     taxonomy_id parent_id           parent_name
      0           101       NaN                   NaN
      1        101-01       101             Emergency
      2        101-02       101             Emergency
      3     101-02-01    101-02        Emergency Cash
      4     101-02-02    101-02        Emergency Cash
      ..          …         …                   …
      285   111-01-07    111-01  Advocacy & Legal Aid
      286      111-02       111                 Legal
      287      111-03       111                 Legal
      288      111-04       111                 Legal
      289      111-05       111                 Legal

      [290 rows x 3 columns]
```

Q2. Given a Pandas DataFrame df with columns ‘A’, ‘B’, and ‘C’, write a Python function to re-index the DataFrame with a new index that starts from 1 and increments by 2 for each row.

```
[11]: import pandas as pd
      df  = pd.DataFrame({'A' : [3,8,6,2,9],
                    'B' : [5,2,9,3,1],
                    'C' : [1,7,4,5,2]})
      def reindex_dataframe(df):
          new_index = pd.Index(range(1, len(df) * 2, 2))
          df = df.reset_index(drop=True)
          df.index = new_index
          return df
```

```
[12]: reindexed_df = reindex_dataframe(df)
```

```
[13]: reindex_dataframe(df)
```

```
[13]:    A  B  C
      1  3  5  1
      3  8  2  7
      5  6  9  4
      7  2  3  5
      9  9  1  2
```

Q3. You have a Pandas DataFrame df with a column named ‘Values’. Write a Python function that iterates over the DataFrame and calculates the sum of the first three values in the ‘Values’ column. The function should print the sum to the console.

```
[1]: import pandas as pd

     def calculate_sum(df):
```

```
    values_column = df['Values']
    sum_of_first_three = sum(values_column[:3])
    print("Sum of the first three values:", sum_of_first_three)
```

[3]:
```
import pandas as pd

# Create a sample DataFrame
data = {'Values': [10, 20, 30, 40, 50]}
df = pd.DataFrame(data)
```

[5]:
```
df
```

[5]:
```
   Values
0      10
1      20
2      30
3      40
4      50
```

[6]:
```
calculate_sum(df)
```

```
Sum of the first three values: 60
```

Q4. Given a Pandas DataFrame df with a column 'Text', write a Python function to create a new column 'Word_Count' that contains the number of words in each row of the 'Text' column.

[10]:
```
import pandas as pd

def count_words(df):
    df['Word_Count'] = df['Text'].apply(lambda x: len(str(x).split()))
```

[11]:
```
import pandas as pd

# Create a sample DataFrame
data = {'Text': ['Hello, how are you?', 'I am doing well.', 'Python is awesome!
 ↪']}
df = pd.DataFrame(data)
```

[12]:
```
count_words(df)
df
```

[12]:
```
                  Text  Word_Count
0  Hello, how are you?           4
1     I am doing well.           4
2   Python is awesome!           3
```

Q5. How are DataFrame.size() and DataFrame.Shape() different? —> Both DataFrame.size and DataFrame.shape are attributes of a pandas DataFrame, but they return different values

1.DataFrame.size return the total number of elements in the DataFrame,which is the product of the number of rows and columns it is equivalent to the size of the underlying Numpy arry

2.DataFrame.shape retun a tuple containing the number of rows and columns in the DataFrame,respectively. it is a convenient way to check the dimensions of the DataFrame

Q6.Which functions of pandas do we use to read an excel file?

pd.read_excel(): This function is used to read an Excel file into a pandas DataFrame. It can read both .xls and .xlsx file formats.

Q7.You have a pandas DataFrame df that contains a column named 'Email' that contains email addresses in the format 'username@domain.com' write a python funciton that creates a new column 'Username' in df that contains only the username part of each email address

The username is the part of the email address that appears before the @ symbol for example if the email address is 'john.do@example.com', the 'Username' column should contain 'john.doe' Your function shoul extract the username from each email address and store it in the new 'username' column.

[15]:

```python
import pandas as pd

def extract_username(df):
    df['Username'] = df['Email'].str.split('@').str[0]
```

[19]:

```python
import pandas as pd

# Create a sample DataFrame
data = {'Email': ['abc@gmail.com', 'xyz@gmail.com', 'myskill@gmail.com']}
df = pd.DataFrame(data)

# Call the function
extract_username(df)
```

[20]:

```python
df
```

[20]:

```
               Email Username
0      abc@gmail.com      abc
1      xyz@gmail.com      xyz
2  myskill@gmail.com  myskill
```

Q8. You have a Pandas DataFrame df with columns 'A', 'B', and 'C'. Write a Python function that selects all rows where the value in column 'A' is greater than 5 and the value in column 'B' is less than 10. The function should return a new DataFrame that contains only the selected rows. For example, if df contains the following values:

[25]:

```python
def select_rows(df):
    mask = (df['A'] > 5) & (df['B'] < 10)

    df_new = df[mask].copy()
    return df_new
```

```
[ ]: df = pd.DataFrame({'A':[3,8,6,2,9],'B':[5,2,9,3,1],'C':[1,7,4,5,2]})
     df
```

```
[ ]:    A  B  C
     0  3  5  1
     1  8  2  7
     2  6  9  4
     3  2  3  5
     4  9  1  2
```

```
[ ]: df_new = select_rows(df)
     df_new
```

```
[ ]:    A  B  C
     1  8  2  7
     2  6  9  4
     4  9  1  2
```

Q9. Given a Pandas DataFrame df with a column ‘Values’, write a Python function to calculate the mean, median, and standard deviation of the values in the ‘Values’ column.

```
[30]: df1  = pd.DataFrame({"Values" : [11,18,36,45,67]})
```

```
[32]: df1
```

```
[32]:    Values
      0      11
      1      18
      2      36
      3      45
      4      67
```

```
[33]: df1.mean()
```

```
[33]: Values    35.4
      dtype: float64
```

```
[34]: df1.median()
```

```
[34]: Values    36.0
      dtype: float64
```

```
[35]: df1.std()
```

[35]:
```
Values    22.300224
dtype: float64
```

Q10 Given a Pandas DataFrame df with a column 'Sales' and a column 'Date', write a Python function to create a new column 'MovingAverage' that contains the moving average of the sales for the past 7 days for each row in the DataFrame. The moving average should be calculated using a window of size 7 and should include the current day.

[37]:
```
import pandas as pd

def calculate_moving_average(df):

    df = df.sort_values(by='Date')

    df['MovingAverage'] = df['Sales'].rolling(window=7, min_periods=1).mean()

    return df
```

[38]:
```
df = pd.DataFrame({
    'Date': ['2022-01-01', '2022-01-02', '2022-01-03', '2022-01-04',␣
 ↪'2022-01-05', '2022-01-06', '2022-01-07', '2022-01-08', '2022-01-09',␣
 ↪'2022-01-10', '2022-01-11', '2022-01-12'],
    'Sales': [11, 15, 35, 41, 42, 64, 72, 51, 65, 81, 95, 111]
})

df = calculate_moving_average(df)
```

[41]:
```
df_new = calculate_moving_average(df)
df_new
```

[41]:
```
          Date  Sales  MovingAverage
0   2022-01-01     11      11.000000
1   2022-01-02     15      13.000000
2   2022-01-03     35      20.333333
3   2022-01-04     41      25.500000
4   2022-01-05     42      28.800000
5   2022-01-06     64      34.666667
6   2022-01-07     72      40.000000
7   2022-01-08     51      45.714286
8   2022-01-09     65      52.857143
9   2022-01-10     81      59.428571
10  2022-01-11     95      67.142857
11  2022-01-12    111      77.000000
```

Q11 You have a Pandas DataFrame df with a column 'Date'. Write a Python function that creates a new column 'Weekday' in the DataFrame. The 'Weekday' column should contain the weekday name (e.g. Monday, Tuesday) corresponding to each date in the 'Date' column.

For example, if df contains the following values: Date 0 2023-01-01 1 2023-01-02 2 2023-01-03 3 2023-01-04 4 2023-01-05

[43]:
```
def return_weekdays(df):

    df['Date']=pd.to_datetime(df['Date'])

    df['Weekday'] = df['Date'].dt.day_name()
    return df
```

[44]:
```
df = pd.DataFrame({'Date':
  ↪['2023-01-01','2023-01-02','2023-01-03','2023-01-04','2023-01-05']})
df
```

[44]:
```
         Date
0  2023-01-01
1  2023-01-02
2  2023-01-03
3  2023-01-04
4  2023-01-05
```

[45]:
```
df_new = return_weekdays(df)
df_new
```

[45]:
```
        Date    Weekday
0 2023-01-01     Sunday
1 2023-01-02     Monday
2 2023-01-03    Tuesday
3 2023-01-04  Wednesday
4 2023-01-05   Thursday
```

Q12.Given a Pandas DataFrame df with a column ‘Date’ that contains timestamps, write a Python function to select all rows where the date is between ‘2023-01-01’ and ‘2023-01-31’.

[46]:
```
def select_january_rows(df):

    df['Date'] = pd.to_datetime(df['Date'])



    january_rows = df[df['Date'].between('2023-01-01', '2023-01-31')]

    return january_rows
```

[47]:
```
df = pd.DataFrame({'Date':pd.date_range(start='12/1/2022',end='3/1/
  ↪2023',freq='D')})
df
```

[47]:
```
          Date
0   2022-12-01
1   2022-12-02
2   2022-12-03
3   2022-12-04
4   2022-12-05
..         …
86  2023-02-25
87  2023-02-26
88  2023-02-27
89  2023-02-28
90  2023-03-01

[91 rows x 1 columns]
```

[48]:
```
select_january_rows(df)
```

[48]:
```
          Date
31  2023-01-01
32  2023-01-02
33  2023-01-03
34  2023-01-04
35  2023-01-05
36  2023-01-06
37  2023-01-07
38  2023-01-08
39  2023-01-09
40  2023-01-10
41  2023-01-11
42  2023-01-12
43  2023-01-13
44  2023-01-14
45  2023-01-15
46  2023-01-16
47  2023-01-17
48  2023-01-18
49  2023-01-19
50  2023-01-20
51  2023-01-21
52  2023-01-22
53  2023-01-23
54  2023-01-24
55  2023-01-25
56  2023-01-26
57  2023-01-27
58  2023-01-28
59  2023-01-29
```

```
60 2023-01-30
61 2023-01-31
```

Q13. To use the basic functions of pandas, what is the first and foremost necessary library that needs to be imported?

[49]:
```python
import pandas as pd
```

this statement imports the pandas luibrary and givens it an alias pd which is common convention in the python community

[ ]:
```
```