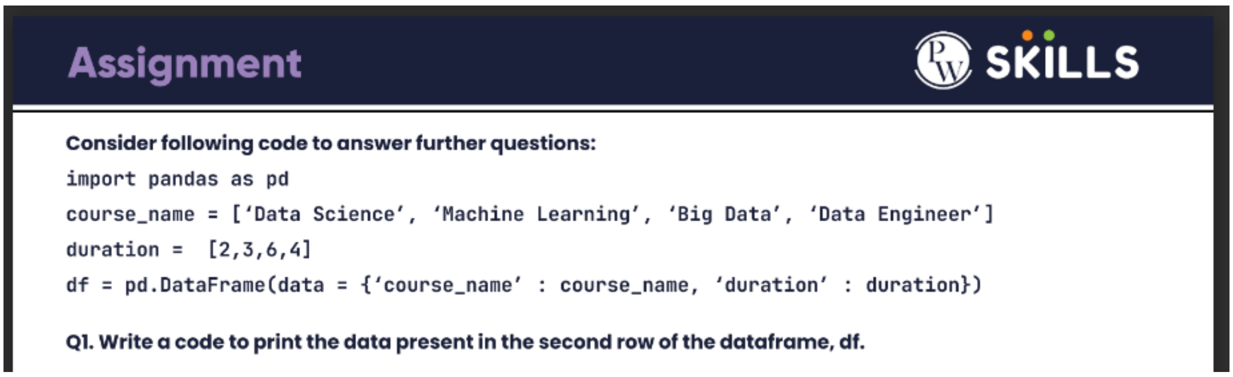

Assignment

PW SKILLS

Consider following code to answer further questions:

```
import pandas as pd
course_name = ['Data Science', 'Machine Learning', 'Big Data', 'Data Engineer']
duration =  [2,3,6,4]
df = pd.DataFrame(data = {'course_name' : course_name, 'duration' : duration})
```

Q1. Write a code to print the data present in the second row of the dataframe, df.

Answer. `df.loc[1]`

OUTPUT:

```
course_name    Machine Learning

duration                      3
```

Q2. What is the difference between the functions loc and iloc in pandas.DataFrame?

Answer:

*** loc() function**

The loc() function is label based data selecting method which means that we must pass the name of the row or column which we want to select. This method includes the last element of the range passed in it, unlike iloc(). loc() can accept the Boolean data unlike iloc(). Many operations can be performed using the loc() method like

Example:

```
import pandas as pd
course_name = ['Data Science', 'Machine Learning', 'Big Data', 'Data Engineer']
duration = [2,3,6,4]
df = pd.DataFrame(data = {'course_name' : course_name, 'duration' : duration})
df.loc[0:1]
```

OUTPUT:

```
        course_name    duration

0       Data Science   2

1       Machine Learning      3
```

***iloc() function**

The iloc() function is an indexed-based selecting method which means that we have to pass an integer index in the method to select a specific row/column. This method does not include the last element of the range passed in it unlike loc(). iloc() does not accept the boolean data unlike loc().

Example:

```
import pandas as pd
course_name = ['Data Science', 'Machine Learning', 'Big Data', 'Data Engineer']
duration = [2,3,6,4]
df = pd.DataFrame(data = {'course_name' : course_name, 'duration' : duration})
df.iloc[1]
```

OUTPUT:

```
course_name    Machine Learning
duration                      3
Name: 1, dtype: object
```

Q3. Reindex the given dataframe using a variable, reindex = [3,0,1,2] and store it in the variable, new_df then find the output for both new_df.loc[2] and new_df.iloc[2].

Did you observe any difference in both the outputs? If so then explain it.

Consider the below code to answer further questions:

```
import pandas as pd
import numpy as np
columns = ['column_1', 'column_2', 'column_3', 'column_4', 'column_5', 'column_6']
indices = [1,2,3,4,5,6]
#Creating a dataframe:
df1 = pd.DataFrame(np.random.rand(6,6), columns = columns, index = indices)
```

Answer:

```
import pandas as pd
import numpy as np
columns = ['column_1', 'column_2', 'column_3', 'column_4', 'column_5', 'column_6']
indices = [1,2,3,4,5,6]
df9 = pd.DataFrame(np.random.rand(6,6), columns = columns, index = indices)

reindex = [3,0,1,2]
new_df=df.reindex(reindex)
new_df.loc[2]
new_df.iloc[2]
```

OUTPUT:

```
       column_1      column_2      column_3      column_4      column_5      column_6
1      0.852745      0.240424      0.677123      0.391806      0.101129      0.331626
2      0.913653      0.289365      0.547524      0.128570      0.214530      0.106578
3      0.918250      0.348758      0.319199      0.112775      0.961572      0.410263
4      0.096239      0.369263      0.180898      0.464039      0.061873      0.443576
5      0.018651      0.100308      0.712073      0.771588      0.192403      0.131127
6      0.015661      0.003203      0.988978      0.666025      0.038504      0.568360

       course_name    duration
3      Data Engineer         4
0      Data Science          2
1      Machine Learning      3
2      Big Data              6

course_name    Big Data
duration              6
Name: 2, dtype: object

course_name    Machine Learning
duration                      3
Name: 1, dtype: object
```

Q4. Write a code to find the following statistical measurements for the above dataframe df1:

(i) mean of each and every column present in the dataframe.

(ii) standard deviation of column, 'column_2'

Answer:

1. Mean=df9.mean()

OUTPUT:

```
column_1    0.500788
column_2    0.291213
column_3    0.731096
column_4    0.539491
column_5    0.436392
column_6    0.238888
Mean             NaN
dtype: float64
```

```
    2.  STD=df9['column_2'].std()
        STD
```

OUTPUT:

```
0.3562730725423985
```

Q5. Replace the data present in the second row of column, 'column_2' by a string variable then find the mean of column, column_2.

If you are getting errors in executing it then explain why.

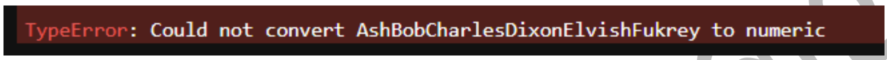
[Hint: To replace the data use df1.loc[] and equate this to string data of your choice.]

Answer:

```
df9['column_2']=['A','B','C','D','E','F']
df9
Mean=df9['column_2'].mean()
```

OUTPUT:

```
       column_1       column_2       column_3       column_4       column_5       column_6
1      0.145041       Ash     0.931205       0.612652       0.221380       0.121833       NaN
2      0.526049       Bob     0.828166       0.877997       0.417871       0.001056       NaN
3      0.196658       Charles 0.910823       0.019863       0.629237       0.713036       NaN
4      0.522347       Dixon   0.588695       0.663229       0.207949       0.245245       NaN
5      0.956943       Elvish  0.483529       0.738243       0.714197       0.129350       NaN
6      0.657689       Fukrey  0.644161       0.324963       0.427719       0.222805       NaN
```



```
TypeError: Could not convert AshBobCharlesDixonElvishFukrey to numeric
```

The error TypeError occurs when calculating the mean of the String, as the mean can only be calculated for the numeric values.

Q6. What do you understand about the windows function in pandas and list the types of windows functions?

Answer: Pandas Window functions are functions where the input values are taken from a "window" of one or more rows in a series or a table and calculation is performed over them. The word window means the number of rows between the two boundaries by which we perform calculations including the boundary rows.

Pandas provide window functions for the following 4 types of windowing operations.

a) Rolling window operations
b) Weighted window operations
c) Expanding window operations
d) Exponentially Weighted window

Q7. Write a code to print only the current month and year at the time of answering this question.

[Hint: Use pandas.datetime function]

Answer:

```
df7 = pd.DataFrame({"date":['2023-06-17']})
df7
df7['year'] = pd.DatetimeIndex(df7['date']).year
df7['month'] = pd.DatetimeIndex(df7['date']).month
print(df7)
```

OUTPUT:

```
date
0      2023-06-17


         date   year   month
0  2023-06-17   2023       6
```

Q8. Write a Python program that takes in two dates as input (in the format YYYY-MM-DD) and calculates the difference between them in days, hours, and minutes using Pandas time delta. The program should prompt the user to enter the dates and display the result.

Answer:

```
import pandas as pd
date1 = input("Enter the first date (YYYY-MM-DD): ")
date2 = input("Enter the second date (YYYY-MM-DD): ")
date1 = pd.to_datetime(date1)
date2 = pd.to_datetime(date2)
delta = date2 - date1
days = delta.days
hours = delta.seconds // 3600
minutes = (delta.seconds % 3600) // 60
print(f"The difference between {date1.date()} and {date2.date()} is {days} days, {hours} hours, and {minutes}
minutes.")
```

OUTPUT:

```
Enter the first date (YYYY-MM-DD):  2023-01-01
Enter the second date (YYYY-MM-DD):  2023-02-02
The difference between 2023-01-01 and 2023-02-02 is 32 days, 0 hours, and 0 minutes.
```

Q9. Write a Python program that reads a CSV file containing categorical data and converts a specified column to a categorical data type. The program should prompt the user to enter the file path, column name, and category order, and then display the sorted data.

Answer:

```
import pandas as pd
Path_name= str(input("Enter the path name"))
Col_name=str(input("Enter the Column name"))
df= pd.read_csv(Path_name)
df[Col_name]=pd.Categorical(df[Col_name],categories=None,ordered=True)
print(df.sort_values(by=Col_name))
```

OutPut:

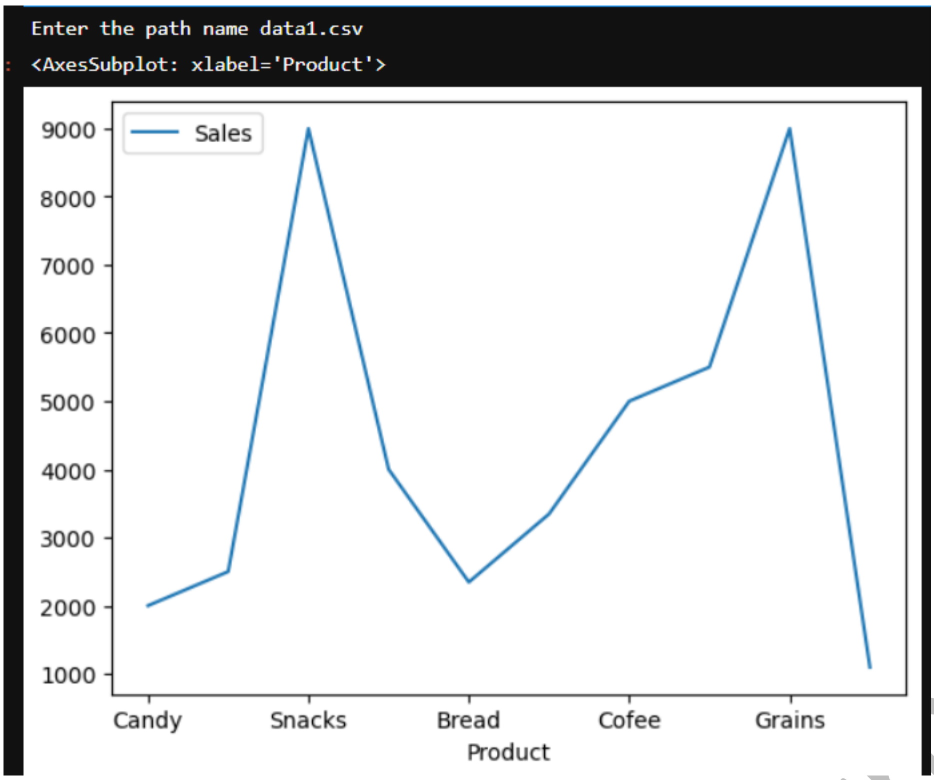
```
Enter the path name data.csv
Enter the Column name Age
   Unnamed: 0       Name             email       Phone   Address Age
3           4  Rao Shabh  Koibhi@gmail.com  9087654324      Agra  16
2           3     Koibhi  Koibhi@gmail.com  9087654323      Agra  22
4           5     Kim  Koibhi@gmail.com  9087654325      Agra  22
0           1     Ashish       ashish@gmail   776546888     Delhi  24
1           2      Lucky   Lucky@gmail.com  8317040193  Banglore  27
6           7   Abhishek  Koibhi@gmail.com  9087654327      Agra  33
8           9      Palak  Koibhi@gmail.com  9087654329      Agra  40
7           8    Manisha  Koibhi@gmail.com  9087654328      Agra  45
5           6        Jia  Koibhi@gmail.com  9087654326      Agra  66
```

Q10. Write a Python program that reads a CSV file containing sales data for different products and visualizes the data using a stacked bar chart to show the sales of each product category over time. The program should prompt the user to enter the file path and display the chart.

Answer:

```
import pandas as pd
Path_name= str(input("Enter the path name"))
df= pd.read_csv(Path_name)
df.plot('Product','Sales')
```

OUTPUT:



```
Enter the path name data1.csv
<AxesSubplot: xlabel='Product'>
```

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.

Answer:

```
import pandas as pd
data={'Date':["1st July","2nd July","3rd July","4th July","5th July","6th July","7th July"],
      "Sales":[10,20,50,30,55,65,25]
      }
df=pd.DataFrame(data)
df['MovingAverage'] = df['Sales'].rolling(window=7, min_periods=1).mean()
print(df)
```

OUTPUT:

```
        Date  Sales  MovingAverage
0   1st July     10      10.000000
1   2nd July     20      15.000000
2   3rd July     50      26.666667
3   4th July     30      27.500000
4   5th July     55      33.000000
5   6th July     65      38.333333
6   7th July     25      36.428571
```

Q11. You are given a CSV file containing student data that includes the student ID and their test score. Write a Python program that reads the CSV file, calculates the mean, median, and mode of the test scores, and displays the results in a table.

The program should do the following:

- Prompt the user to enter the file path of the CSV file containing the student data.
- Read the CSV file into a Pandas DataFrame.
- Calculate the mean, median, and mode of the test scores using Pandas tools.
- Display the mean, median, and mode in a table.

Assume the CSV file contains the following columns:

- Student ID: The ID of the student.
- Test Score: The score of the student's test.

Example usage of the program:
Enter the file path of the CSV file containing the student data: student_data.csv

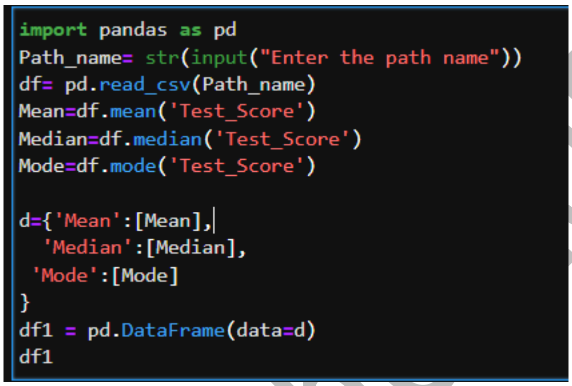
```
+-----------+--------+
| Statistic | Value |
+-----------+--------+
| Mean     | 79.6  |
| Median   | 82    |
| Mode     | 85, 90 |
+-----------+--------+
```

Assume that the CSV file student_data.csv contains the following data:

```
Student ID,Test Score
1,85
2,90
3,80
4,75
5,85
6,82
7,78
8,85
9,90
10,85
```

The program should calculate the mean, median, and mode of the test scores and display the results in a table.

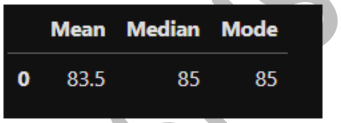
Answer:



```
import pandas as pd
Path_name= str(input("Enter the path name"))
df= pd.read_csv(Path_name)
Mean=df.mean('Test_Score')
Median=df.median('Test_Score')
Mode=df.mode('Test_Score')

d={'Mean':[Mean],
   'Median':[Median],
  'Mode':[Mode]
}
df1 = pd.DataFrame(data=d)
df1
```

Output:



	Mean	Median	Mode
0	83.5	85	85