**Exp 2**

User

**You**

**import numpy as np**

**import matplotlib.pyplot as plt**

**%matplotlib inline**

ChatGPT

**ChatGPT**

Sure! Here's the meaning of each line:

1. **import numpy as np**: This line imports the NumPy library, which is used for numerical computations in Python. The **as np** part aliases NumPy as **np**, allowing you to refer to its functions and objects using the shorthand **np** instead of typing out **numpy** every time.
2. **import matplotlib.pyplot as plt**: This line imports the **pyplot** module from the Matplotlib library, which is used for creating visualizations in Python. The **as plt** part aliases **pyplot** as **plt**, making it easier to reference its functions and objects.
3. **%matplotlib inline**: This is a magic command in Jupyter notebooks and IPython that allows Matplotlib plots to be displayed directly within the notebook. It's used to ensure that plots appear inline, rather than in separate windows or files.

User

**You**

**import pandas as pd**

ChatGPT

**ChatGPT**

Certainly! Here's the explanation for this line:

1. **import pandas as pd**: This line imports the Pandas library, which is used for data manipulation and analysis in Python. Similar to the previous lines, the **as pd** part aliases Pandas as **pd**, allowing you to refer to its functions and objects using the shorthand **pd** instead of typing out **pandas** every time.

User

**You**

**df = pd.read\_csv(r"D:\College\TE\SEM-2\Practical\DSBDA\2\AcademicPerformance.csv")**

**print(df)**

ChatGPT

**ChatGPT**

Certainly! Here's the breakdown:

1. **df = pd.read\_csv(r"D:\College\TE\SEM-2\Practical\DSBDA\2\AcademicPerformance.csv")**: This line reads a CSV file named "AcademicPerformance.csv" located at the specified path (**D:\College\TE\SEM-2\Practical\DSBDA\2**) using Pandas' **read\_csv()** function. The **r** before the string indicates a raw string, which is useful for specifying file paths to avoid interpreting backslashes as escape characters.
2. **print(df)**: This line prints the contents of the DataFrame **df** to the console. A DataFrame is a 2-dimensional labeled data structure with columns of potentially different types. Printing **df** displays the tabular data from the CSV file in a structured format.

User

**You**

**print(df['math score'])**

ChatGPT

**ChatGPT**

Certainly! Here's the explanation:

1. **print(df['math score'])**: This line prints the contents of the 'math score' column from the DataFrame **df**. It selects the 'math score' column using square brackets and then prints the values of that column to the console. This is useful for examining a specific column of data within the DataFrame.

User

**You**

**print(df['math score'].isnull())**

ChatGPT

**ChatGPT**

Absolutely! Here's what that line does:

1. **print(df['math score'].isnull())**: This line checks for missing (null) values in the 'math score' column of the DataFrame **df**. The **.isnull()** method returns a boolean Series where each element is True if the corresponding element in the 'math score' column is null and False otherwise. The **print()** function then displays this Series, indicating True for missing values and False for non-missing values.

User

**You**

**missing\_values = ["n/a", "na", "--"]**

**df = pd.read\_csv(r"D:\College\TE\SEM-2\Practical\DSBDA\2\AcademicPerformance.csv", na\_values = missing\_values)**

ChatGPT

**ChatGPT**

Here's what that code does:

1. **missing\_values = ["n/a", "na", "--"]**: This line creates a list called **missing\_values** containing strings that represent missing values in the CSV file. These strings are common representations of missing data that Pandas should recognize and convert to NaN (Not a Number) values during the reading process.
2. **df = pd.read\_csv(r"D:\College\TE\SEM-2\Practical\DSBDA\2\AcademicPerformance.csv", na\_values = missing\_values)**: This line reads the CSV file "AcademicPerformance.csv" located at the specified path (**D:\College\TE\SEM-2\Practical\DSBDA\2**) into a DataFrame called **df**, using Pandas' **read\_csv()** function. The **na\_values** parameter is set to the list **missing\_values**, which specifies the strings that should be considered as missing values. Pandas will replace these values with NaN when reading the CSV file.

User

**You**

**print(df['reading score'])**

ChatGPT

**ChatGPT**

Sure! This line prints the contents of the 'reading score' column from the DataFrame **df**. It selects the 'reading score' column using square brackets and then prints the values of that column to the console. This is useful for examining the data in the 'reading score' column.

User

**You**

**dataset = [11,41,20,3,101,55,68,97,99,6]**

ChatGPT

**ChatGPT**

Certainly! This line creates a Python list named **dataset** containing a sequence of numerical values:

1. **dataset = [11,41,20,3,101,55,68,97,99,6]**: This line initializes the **dataset** list with ten numerical values: 11, 41, 20, 3, 101, 55, 68, 97, 99, and 6. This list can be used to store and manipulate the provided data in subsequent code.

User

**You**

**sorted(dataset)**

ChatGPT

**ChatGPT**

This line sorts the elements of the **dataset** list in ascending order:

1. **sorted(dataset)**: This function sorts the elements of the **dataset** list in ascending order and returns a new list containing the sorted elements. However, it doesn't modify the original list **dataset**. If you want to sort the list in place, you can use the **sort()** method like this: **dataset.sort()**.

User

**You**

**quantile1, quantile3 = np.percentile(dataset, [25,75])**

ChatGPT

**ChatGPT**

This line calculates the first quartile (Q1) and the third quartile (Q3) of the dataset using NumPy's **percentile()** function:

1. **quantile1, quantile3 = np.percentile(dataset, [25,75])**: This line computes the values of the first quartile (25th percentile) and the third quartile (75th percentile) of the **dataset** list. The function **np.percentile()** takes two arguments: the dataset (in this case, **dataset**) and a list containing the desired percentiles (in this case, **[25, 75]** for Q1 and Q3). The calculated values are assigned to the variables **quantile1** and **quantile3**, respectively.

User

**You**

**print(quantile1, quantile3**)

ChatGPT

**ChatGPT**

This line prints the calculated first quartile (Q1) and third quartile (Q3) values:

1. **print(quantile1, quantile3)**: This line prints the values of the first quartile (**quantile1**) and the third quartile (**quantile3**) to the console. These values represent the 25th and 75th percentiles of the dataset, respectively, and are useful measures for understanding the spread and distribution of the data.

User

**You**

**iqr\_value = (quantile3 - quantile1)**

ChatGPT

**ChatGPT**

This line calculates the interquartile range (IQR) of the dataset:

1. **iqr\_value = (quantile3 - quantile1)**: This line subtracts the value of the first quartile (**quantile1**) from the value of the third quartile (**quantile3**) to compute the interquartile range (IQR). The IQR is a measure of statistical dispersion and is calculated as the difference between the third quartile and the first quartile. The result is assigned to the variable **iqr\_value**.

User

**You**

**print(iqr\_value)**

ChatGPT

**ChatGPT**

This line prints the calculated interquartile range (IQR) value:

1. **print(iqr\_value)**: This line prints the value of the interquartile range (**iqr\_value**) to the console. The IQR represents the range of the middle 50% of the dataset and is a useful measure for understanding the spread of the data around the median.

User

**You**

**lower\_bound\_value = quantile1 - (1.5\*iqr\_value)**

**upper\_bound\_value = quantile3 + (1.5\*iqr\_value)**

ChatGPT

**ChatGPT**

This code calculates the lower and upper bounds for identifying outliers using the interquartile range (IQR) method:

1. **lower\_bound\_value = quantile1 - (1.5\*iqr\_value)**: This line computes the lower bound value for outlier detection. It subtracts 1.5 times the interquartile range (**iqr\_value**) from the first quartile (**quantile1**). This value is used to identify potential outliers that fall below the lower bound.
2. **upper\_bound\_value = quantile3 + (1.5\*iqr\_value)**: This line calculates the upper bound value for outlier detection. It adds 1.5 times the interquartile range (**iqr\_value**) to the third quartile (**quantile3**). This value is used to identify potential outliers that exceed the upper bound.

User

**You**

**print(lower\_bound\_value, upper\_bound\_value**)

ChatGPT

**ChatGPT**

This line prints the calculated lower and upper bounds for outlier detection:

1. **print(lower\_bound\_value, upper\_bound\_value)**: This line prints the values of the lower bound (**lower\_bound\_value**) and the upper bound (**upper\_bound\_value**) to the console. These bounds are used to identify potential outliers in the dataset based on the interquartile range (IQR) method. Any value below the lower bound or above the upper bound is considered a potential outlier.

User

**You**

**from datetime import date**

**df['age'] = date.today().year - df['Year\_Birth']**

ChatGPT

**ChatGPT**

This code calculates the age of individuals in the DataFrame **df** based on their birth year:

1. **from datetime import date**: This line imports the **date** class from the **datetime** module. This class is used to represent dates in Python.
2. **df['age'] = date.today().year - df['Year\_Birth']**: This line calculates the age of individuals in the DataFrame **df** by subtracting their birth year (stored in the 'Year\_Birth' column) from the current year obtained using **date.today().year**. The resulting age values are assigned to a new column named 'age' in the DataFrame **df**.

User

**You**

**df['Year'] = pd.DatetimeIndex(df['Dt\_Admission']).year**

**df['E\_L'] = date.today().year - df['Year']**

ChatGPT

**ChatGPT**

This code computes the difference in years between the admission year and the current year for each entry in the DataFrame **df**:

1. **df['Year'] = pd.DatetimeIndex(df['Dt\_Admission']).year**: This line extracts the year component from the 'Dt\_Admission' column of the DataFrame **df** using **pd.DatetimeIndex()**. It then creates a new column named 'Year' in **df** to store these extracted years.
2. **df['E\_L'] = date.today().year - df['Year']**: This line calculates the elapsed years (**E\_L**, meaning "Elapsed Years") by subtracting the year values in the 'Year' column from the current year obtained using **date.today().year**. The resulting values represent the number of years that have elapsed since the admission year for each entry in the DataFrame **df**.

User

**You**

**df.head(5)**

ChatGPT

**ChatGPT**

Certainly! Here's what the **df.head(5)** function does:

This function displays the first 5 rows of the DataFrame **df**, showing a preview of the data after the previous manipulations. This allows you to inspect the DataFrame and verify that the operations performed, such as calculating age and elapsed years, were successful.

User

**You**

**df['Fees$'] = df['College\_Fees'].str.replace(',', '').str.replace('$', '').str.replace( df['Fees\_M$'] = df['Fees$'].apply(lambda X:round(X/1000000))**

**df.head(5)**

ChatGPT

**ChatGPT**

This code snippet performs several operations on the 'College\_Fees' column to extract numerical values and convert them into millions:

1. **df['Fees$'] = df['College\_Fees'].str.replace(',', '').str.replace('$', '')**: This line removes commas and dollar signs from the 'College\_Fees' column, converting the fees from a string format to a numeric format.
2. **df['Fees\_M$'] = df['Fees$'].apply(lambda X:round(X/1000000))**: This line creates a new column named 'Fees\_M$' in the DataFrame **df**. It divides the fees by 1,000,000 to convert them to millions and rounds the result to the nearest integer using the **round()** function. The **apply()** method applies the lambda function to each value in the 'Fees$' column.

Finally, **df.head(5)** displays the first 5 rows of the DataFrame **df** after these transformations. This allows you to inspect the DataFrame and verify the changes made to the fees columns.

Top of Form