Name : Prateek Nayyar

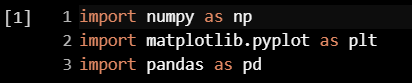
PRN : 21070521053

Subject : Generative AI (CA2)

Section : A (A2)

Problem Statement : Generate a model in Python to represent a Housing loan scheme and create a chart to display the Emi based on rate of interest and reducing balance for a given period. If a customer wishes to close the loan earlier, print the interest lost distributed over the remaining no. Of months. Assume suitable data and inputs as necessary.

**1. Importing Libraries**

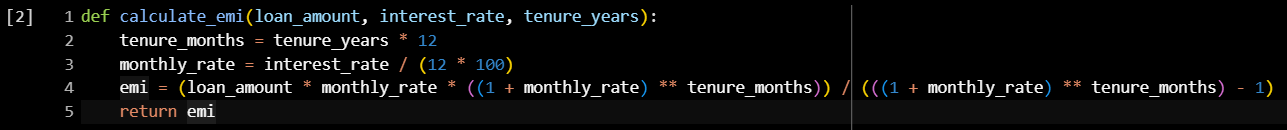


 **numpy (np):** Provides support for large, multi-dimensional arrays and matrices, along with mathematical functions.

 **matplotlib.pyplot (plt):** A plotting library used to create visualizations like line plots, bar charts, etc.

 **pandas (pd):** A data manipulation and analysis library. It's not used in the current code but might be useful for future data processing tasks.

**2. Function to Calculate EMI**



 **Purpose**: This function calculates the Equated Monthly Installment (EMI) for a given loan.

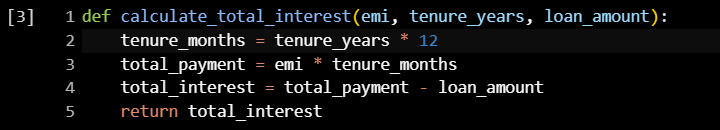
 **Parameters**:

* **loan\_amount:** Total loan amount.
* **interest\_rate:** Annual interest rate (in percentage).
* **tenure\_years:** The loan tenure (in years).

 **How it works**:

* **tenure\_months:** Converts the loan tenure from years to months.
* **monthly\_rate:** Converts the annual interest rate to a monthly interest rate by dividing by 12 (for months) and 100 (to convert percentage into a decimal).
* The EMI formula is then applied to calculate the monthly installment.
* **Returns**: The EMI amount.

**3. Function to Calculate Simple Interest**



 **Purpose**: This function calculates the total interest paid over the entire loan tenure.

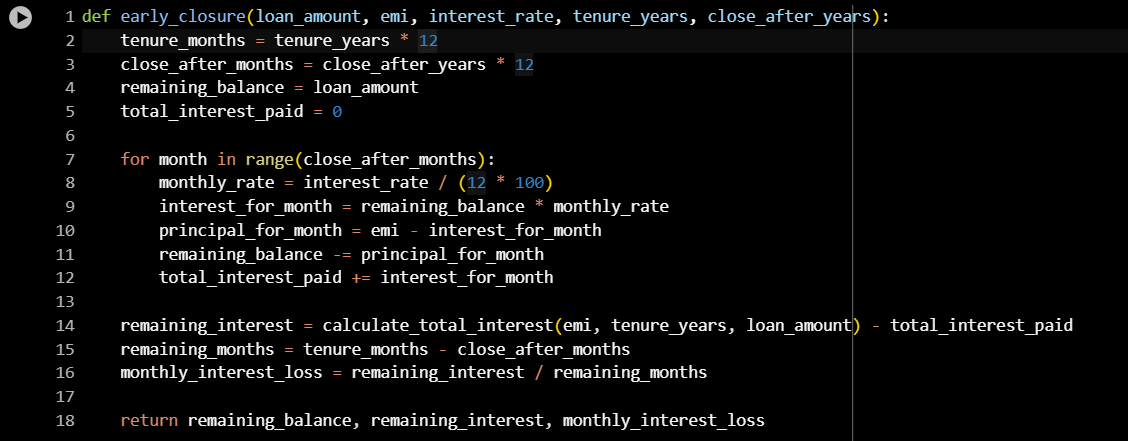
 **Parameters**:

* **emi:** The EMI value (calculated earlier).
* **tenure\_years:** The loan tenure in years.
* **loan\_amount:** Total loan amount.

 **How it works**:

* **tenure\_months:** Converts the tenure from years to months.
* **total\_payment:** Total amount paid over the entire loan period (EMI \* number of months).
* **total\_interest:** Subtracting the initial loan amount from the total payment gives the total interest paid.
* **Returns**: The total interest paid over the loan's lifetime.

**4. Function to Handle Early Loan Closure**



 **Purpose**: This function calculates the remaining loan balance and interest if the loan is closed earlier than the full tenure.

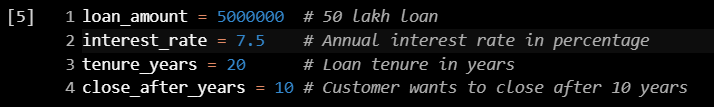
 **Parameters**:

* **loan\_amount:** Total loan amount.
* **emi:** Monthly EMI payment.
* **interest\_rate:** Annual interest rate.
* **tenure\_years:** Loan tenure in years.
* **close\_after\_years:** After how many years the loan will be closed.

 **How it works**:

* The function calculates how much of the loan has been paid off and the remaining balance after close\_after\_years years.
* It also calculates how much interest has been paid until this point and the remaining interest after closure.
* **Returns**:
  + **remaining\_balance:** Loan balance after closing early.
  + **remaining\_interest:** The total interest that would have been paid if the loan were not closed early.
  + **monthly\_interest\_loss:** The average monthly interest loss after early closure.

**5. Input Data:**



* **These are the inputs for the loan details:**
* Loan amount: ₹50,00,000.
* Interest rate: 7.5% annually.
* Tenure: 20 years.
* The customer wants to close the loan after 10 years.

 **Displaying the Sorted Data**:

* The sorted DataFrame is printed in full, showing all rows as configured, so users can view the entire sorted dataset.

**6. Calculating EMI and Total Interest**



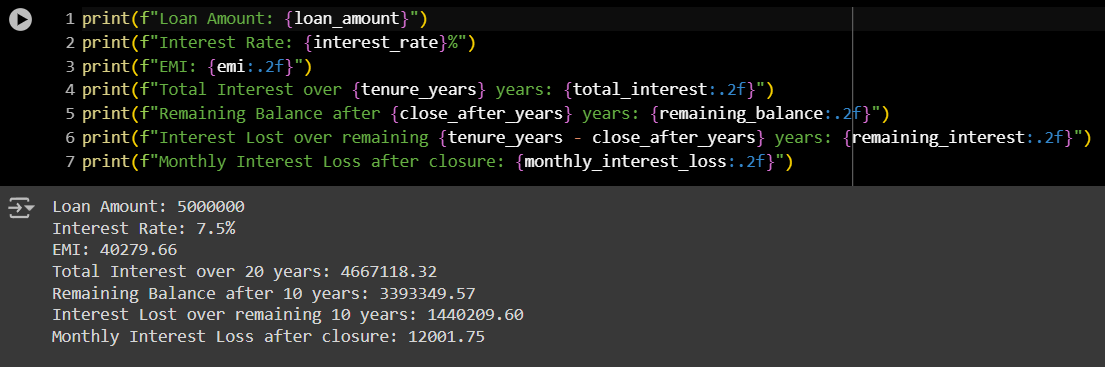
* Here, the code calculates the EMI and total interest for the given loan details using the earlier-defined functions.

**7. Calculating Early Closure Details:**



* This step calculates the remaining balance, remaining interest, and monthly interest loss if the loan is closed after 10 years.

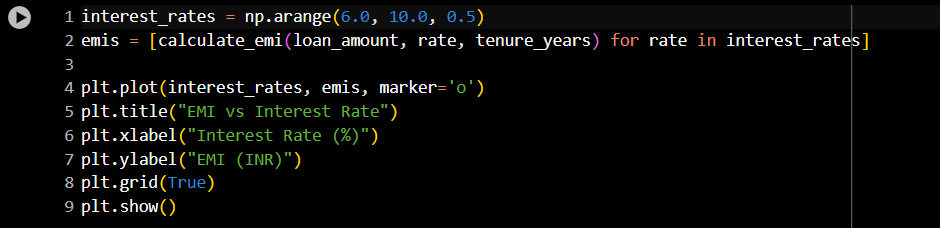
**8. Displaying the Results:**



**This section prints the calculated values for:**

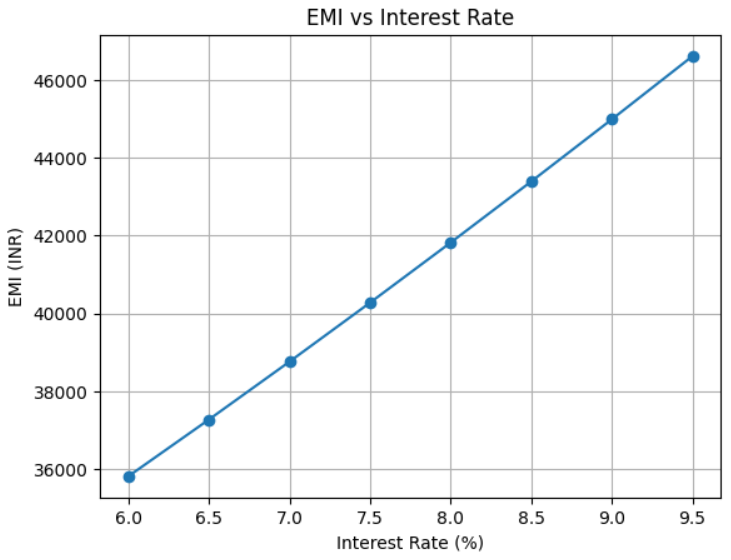
* EMI
* Total interest paid
* Remaining balance after early closure
* Interest lost after closure
* Monthly interest loss

**9. Plotting EMI vs Interest Rate:**

****

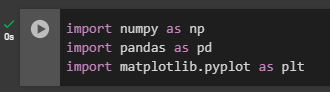
* **Purpose**: This segment plots how the EMI changes with different interest rates.
* **Steps**:
* **interest\_rates:** Creates an array of interest rates from 6.0% to 10.0% (in steps of 0.5%).
* **emis:** Calculates EMI for each interest rate.
* **plt.plot:** Plots the interest rates vs EMI.
* The plot is displayed using plt.show() with appropriate labels and a grid.

**10. Graph:**

****

Problem Statement : Generate a model for an insurance company to hold information on the insurer's vehicle, and create a chart of monthly, yearly, and qtrly premiums based on no. of years of insurance where in each year, the value of the vehicle depreciates by 7%.

**1. Importing Libraries:**

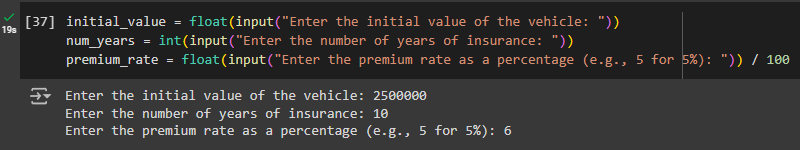


 **numpy (np)**: For numerical operations and generating random data.

 **pandas (pd)**: For data manipulation and creating DataFrames.

 **matplotlib.pyplot (plt)**: For creating visualizations and plots.

**2. User Input for Insurance Model:**

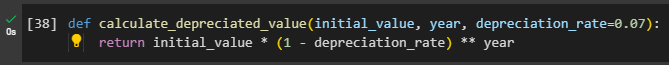


 **initial\_value**: Prompts the user to enter the initial value of the vehicle. This value represents the starting value of the vehicle for calculating depreciation.

 **num\_years**: Asks the user for the number of years the vehicle will be insured. This value is used to determine the duration over which the premiums will be calculated.

 **premium\_rate**: Requests the premium rate as a percentage (e.g., enter 5 for 5%). This rate is converted to a decimal by dividing by 100, and is used to calculate the insurance premiums

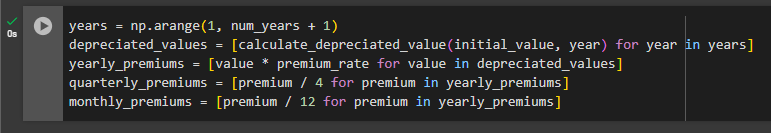
**3. Depreciation Calculation Function:**



**calculate\_depreciated\_value**: Computes the value of a vehicle after depreciation over a specified number of years.

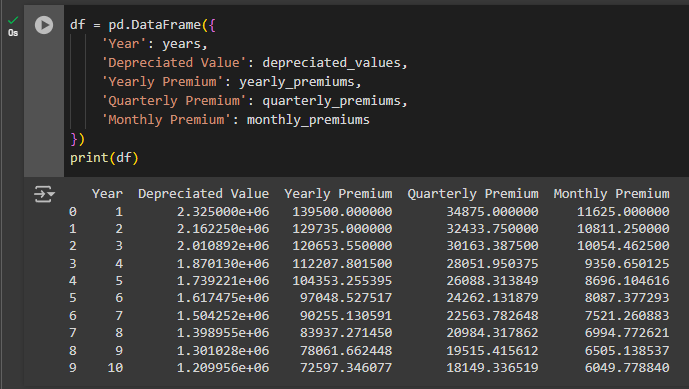
* **Parameters**:
  + **initial\_value**: The vehicle’s initial value.
  + **year**: The number of years over which depreciation is calculated.
  + **depreciation\_rate**: The annual depreciation rate (default is 7% or 0.07).
* **Returns**: The vehicle's value after applying the depreciation formula

**4. Calculating Premiums Based on Depreciation:**



* **years**: An array of integers representing each year of the insurance period, from 1 to num\_years.
* **depreciated\_values**: A list of the vehicle’s depreciated values for each year, calculated using the calculate\_depreciated\_value function.
* **yearly\_premiums**: The insurance premiums for each year, computed as the product of the depreciated values and the premium rate.
* **quarterly\_premiums**: The premiums divided by 4 to determine the amount due each quarter.
* **monthly\_premiums**: The premiums divided by 12 to find the monthly premium amount.

**5. Creating and Displaying the Premium DataFrame:**

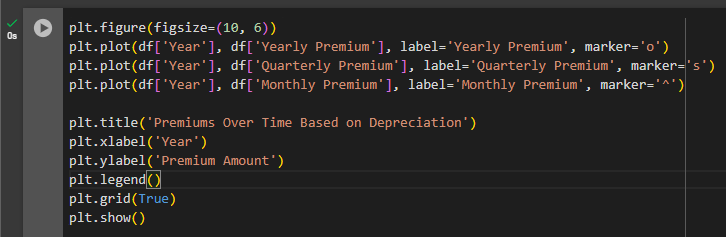


 **df**: A pandas DataFrame that organizes the calculated insurance data into a tabular format.

* **Columns**:
  + **Year**: The year of the insurance period.
  + **Depreciated Value**: The vehicle’s value after depreciation for each year.
  + **Yearly Premium**: The total annual insurance premium.
  + **Quarterly Premium**: The premium amount due each quarter.
  + **Monthly Premium**: The premium amount due each month.

 **print(df)**: Displays the DataFrame, showing the calculated premiums and depreciated values for each year of insurance.

**6. Plotting Premiums Over Time:**



 **plt.figure(figsize=(10, 6))**: Sets up the figure for the plot with a size of 10x6 inches.

 **plt.plot(df['Year'], df['Yearly Premium'], label='Yearly Premium', marker='o')**: Plots the yearly premiums as a line with circle markers.

 **plt.plot(df['Year'], df['Quarterly Premium'], label='Quarterly Premium', marker='s')**: Plots the quarterly premiums as a line with square markers.

 **plt.plot(df['Year'], df['Monthly Premium'], label='Monthly Premium', marker='^')**: Plots the monthly premiums as a line with triangle markers.

 **plt.title('Premiums Over Time Based on Depreciation')**: Sets the title of the plot.

 **plt.xlabel('Year')**: Labels the x-axis as 'Year'.

 **plt.ylabel('Premium Amount')**: Labels the y-axis as 'Premium Amount'.

 **plt.legend()**: Adds a legend to the plot to identify each line.

 **plt.grid(True)**: Adds a grid to the plot for better readability.

 **plt.show()**: Displays the plot, showing how premiums change over time based on vehicle depreciation.

**7. Plotted Graph:**

