Name : Sadaf Farooqui

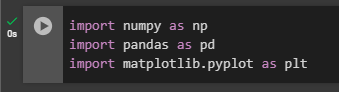
PRN : 21707521118

Subject : Generative AI (CA2)

Section : B (B2)

Question : 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. Libraries Used**

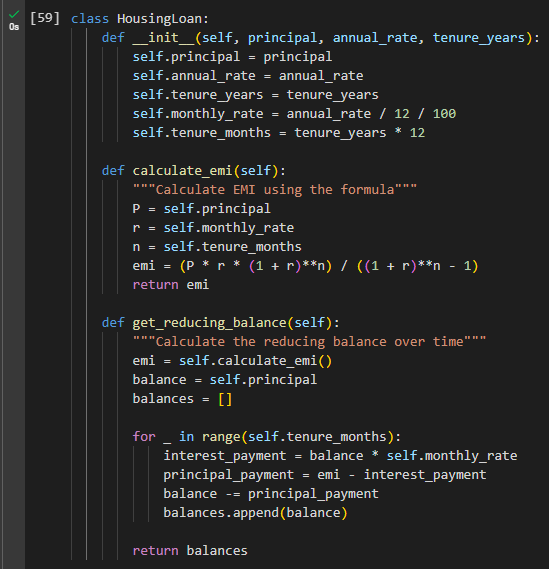


 **NumPy**: A library for numerical operations and handling arrays. It was used to perform mathematical calculations and generate random data.

 **Pandas**: A data manipulation and analysis library. It was utilized to create and manage data structures like DataFrames.

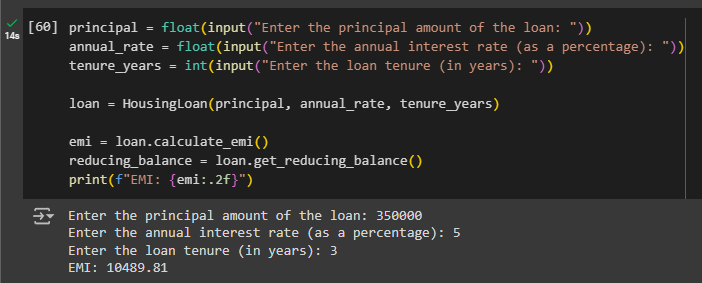
 **Matplotlib**: A plotting library for visualizing data. It was used to create charts and graphs to represent loan data and EMI calculations.

**2. HousingLoan Class**



* **Initialization (\_\_init\_\_ Method)**:
  + **Purpose**: Initializes the loan with key details:
    - **Principal**: The total amount borrowed.
    - **Annual Interest Rate**: The interest rate charged annually.
    - **Tenure**: The length of time, in years, over which the loan will be repaid.
  + **Calculations**: Computes monthly interest rate and total number of months for the loan term.
* **Calculate EMI (calculate\_emi Method)**:
  + **Purpose**: Determines the fixed monthly payment amount (EMI) required to repay the loan over its term.
  + **How It Works**: The EMI is calculated based on the loan amount, interest rate, and duration, ensuring equal monthly payments.
* **Get Reducing Balance (get\_reducing\_balance Method)**:
  + **Purpose**: Provides a monthly breakdown of the remaining loan balance.
  + **How It Works**:
    - Calculates the interest and principal portions of each monthly payment.
    - Updates the loan balance after each payment to reflect the reducing balance over time.

**3. User Input and Calculation**



 **User Inputs**:

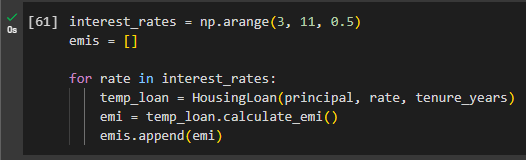
* **Principal Amount**: The total loan amount entered by the user.
* **Annual Interest Rate**: The yearly interest rate, provided as a percentage.
* **Loan Tenure**: The length of the loan in years.

 **Process**:

* **Loan Initialization**: Creates an instance of the HousingLoan class with the user-provided values.
* **Calculate EMI**: Computes the monthly installment amount using the calculate\_emi method.
* **Calculate Reducing Balance**: Determines the loan balance over time using the get\_reducing\_balance method.

 **Output**:

* **EMI**: Displays the calculated Equated Monthly Installment, rounded to two decimal places.

**4. EMI Calculation for Varying Interest Rates**  
  


 **Interest Rates Range**:

* **Definition**: A range of interest rates from 3% to 10.5%, incremented by 0.5%.

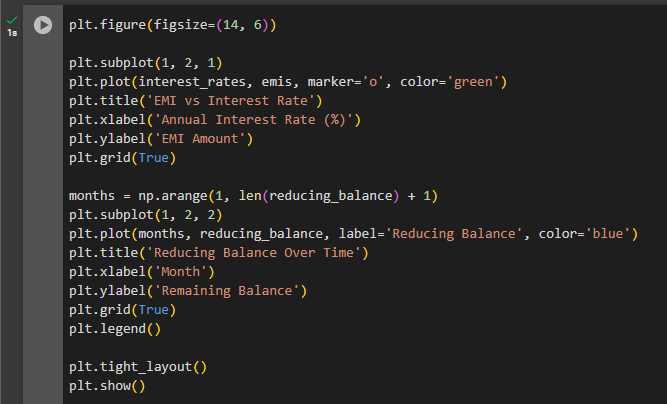
 **Process**:

* **Create Loan Instances**: For each interest rate in the defined range:
  + **Initialize**: Create a temporary HousingLoan object with the current interest rate.
  + **Calculate EMI**: Compute the EMI for the current interest rate.
  + **Store EMI**: Append the calculated EMI to a list for later analysis.

 **Purpose**:

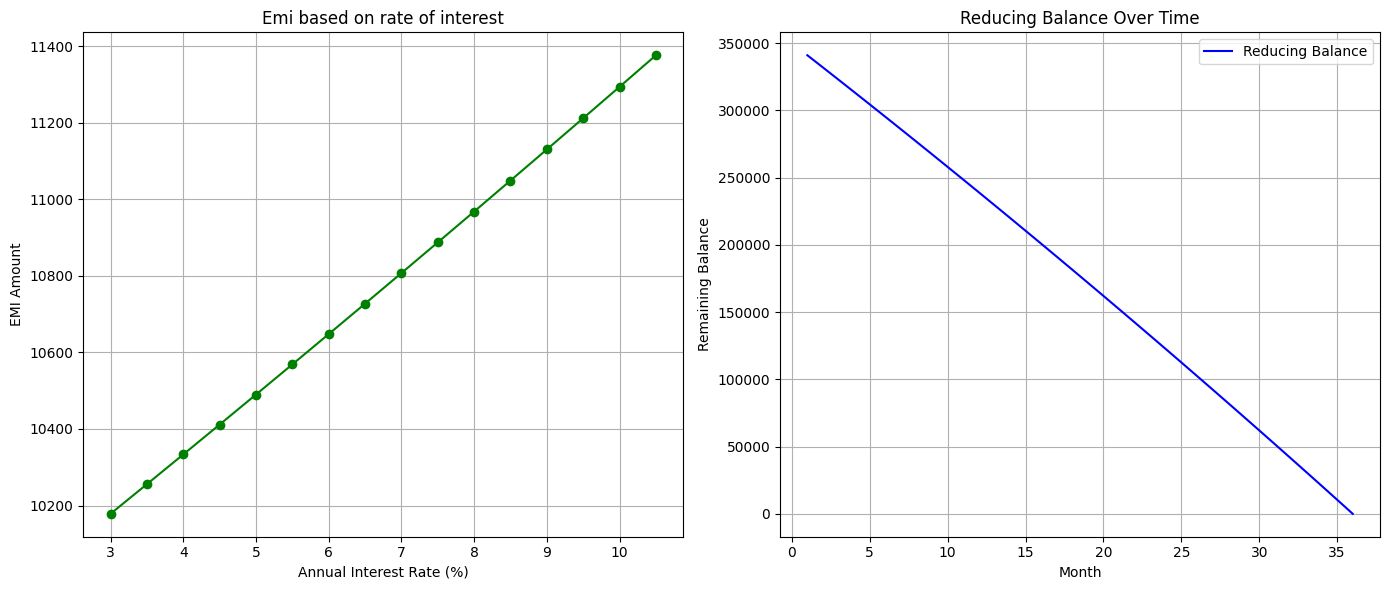
* To analyze how the EMI changes with different interest rates over the same principal and loan tenure.

**5. Visualization of EMI and Reducing Balance**

****

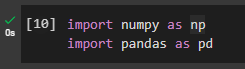
* **Emi based on rate of interest**:
  + **Chart**: Displays how the Equated Monthly Installment (EMI) varies with different annual interest rates.
  + **X-axis**: Annual Interest Rate (%) ranging from 3% to 10.5%.
  + **Y-axis**: EMI Amount.
  + **Details**: A line plot shows the relationship between interest rates and EMI, with markers highlighting individual data points.
* **Reducing Balance Over Time**:
  + **Chart**: Illustrates the change in the loan balance over the repayment period.
  + **X-axis**: Month, representing the time elapsed.
  + **Y-axis**: Remaining Balance.
  + **Details**: A line plot shows the reducing balance over time, reflecting how the outstanding loan amount decreases with each payment.
* **Layout**:
  + **Arrangement**: Both charts are displayed side by side for easy comparison.
  + **Grid**: Enabled for better readability.
  + **Legend**: Included for the reducing balance chart to identify the data series.

**6. Output Graphs**

****

Question : Generate a model for Covid 19 with symptoms of parameters like fever, cold, shivering, weight loss, generate 100 model data with random values for each parameter and order by parameter lowest to highest in display based on the input parameter.

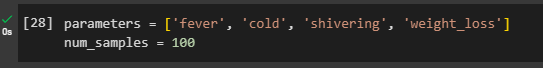
**1. Importing Libraries**



 **numpy (np)**: This library is used for generating random numerical data. In this code, it helps create random values representing symptoms for each parameter.

 **pandas (pd)**: This library is used for creating and managing a tabular dataset (DataFrame), which is a structured way of organizing and manipulating the data.

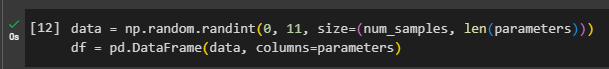
**2. Define Parameters and Number of Samples**



 **parameters**: This list contains four different symptoms (fever, cold, shivering, and weight\_loss), which will be the column names in the dataset.

 **num\_samples**: Specifies that the code will generate 100 records (rows) representing 100 individuals, each having random values for these symptoms.

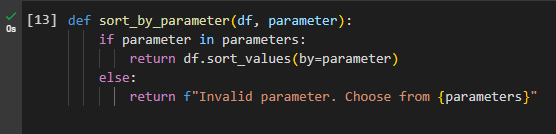
**3. Generating and Structuring Random Symptom Data**



 **Generating Random Data**: The first line creates a 2D array with random integer values between 0 and 10. Each row represents an individual, and each column represents a different symptom, based on the number of samples and parameters specified.

 **Creating a DataFrame**: The second line converts this 2D array into a pandas DataFrame, where each column is labeled according to the symptom names (fever, cold, shivering, weight\_loss), providing a structured and easily accessible format for data analysis.

**4. Sorting Data by a Specified Parameter**



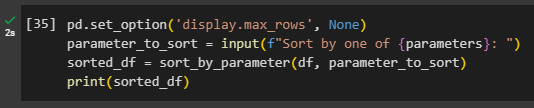
 **Function Definition**: sort\_by\_parameter(df, parameter):

* **Purpose**: To sort the DataFrame (df) by a column specified by the parameter.

 **Parameter Check**:

* **if parameter in parameters**: Checks if the provided parameter is a valid column name in the DataFrame, based on the predefined list of parameters (parameters).
  + **Valid Parameter**: If the parameter is valid, the function sorts the DataFrame by this column in ascending order using df.sort\_values(by=parameter) and returns the sorted DataFrame.
  + **Invalid Parameter**: If the parameter is not found in the list, the function returns an error message indicating the valid parameters that can be chosen.

**5. Configuring Display Options and Sorting Data**



 **Setting Display Options**:

* pd.set\_option('display.max\_rows', None) ensures that all rows of the DataFrame are displayed, preventing truncation in the console output.

 **Prompting User for Input**:

* The input() function asks the user to specify a column (e.g., fever, cold, shivering, weight\_loss) to sort the DataFrame by, and stores this input in parameter\_to\_sort.

 **Sorting the DataFrame**:

* The sort\_by\_parameter(df, parameter\_to\_sort) function sorts the DataFrame based on the selected column and returns the sorted DataFrame.

 **Displaying the Sorted Data**:

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

**5. Configuring Display Options and Sorting Data**

