**Symbiosis Institute of Technology, Nagpur**

**CA-II GenAI**

**Sub: GenAI Sem: VII**

**Name: RISHIKESH KUMBHALKAR PRN: 21070521059**

**Q:2 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.**

**Ans:**

In this question need to make a model on housing loan scheme and calculate the EMI (Equated Monthly Installment) based on the rate of interest and a reducing balance method. The solution follows these steps:

1. Loan Inputs: The user inputs the principal amount, annual interest rate, and the loan term (in months).
2. EMI Calculation: Using the reducing balance formula, the EMI is computed. This method considers that interest is applied to the outstanding loan balance after each monthly payment, reducing the interest component as the balance decreases over time.
3. Simulating Loan Repayment: Each month, the program calculates:
4. Interest for that month based on the outstanding balance.
5. The principal repayment as the EMI minus the interest.
6. The outstanding balance is reduced by the principal payment.
7. Early Loan Closure: The program also allows for early closure of the loan. If the loan is closed early, the interest lost is calculated as the difference between the total interest paid with early closure and what would have been paid over the full term.
8. EMI Breakdown Chart: A chart is displayed showing the interest and principal components of the EMI over time, illustrating how the interest decreases as the outstanding loan balance reduces.

This solution dynamically calculates and visualizes the EMI and total interest payments, accounting for both regular and early loan closure scenarios.

CODE:

import math

import matplotlib.pyplot as plt

# Step 1: Define function to calculate EMI

def calculate\_emi(principal, rate\_of\_interest, loan\_term):

    monthly\_rate = rate\_of\_interest / (12 \* 100)  # Monthly interest rate

    emi = (principal \* monthly\_rate \* math.pow(1 + monthly\_rate, loan\_term)) / (math.pow(1 + monthly\_rate, loan\_term) - 1)

    return emi

# Step 2: Simulate loan repayment and calculate EMI and interest for each month

def simulate\_loan\_repayment(principal, rate\_of\_interest, loan\_term):

    emi = calculate\_emi(principal, rate\_of\_interest, loan\_term)

    remaining\_principal = principal

    interest\_paid = []

    principal\_paid = []

    for month in range(1, loan\_term + 1):

        monthly\_interest = remaining\_principal \* rate\_of\_interest / (12 \* 100)

        monthly\_principal = emi - monthly\_interest

        remaining\_principal -= monthly\_principal

        # Track interest and principal components

        interest\_paid.append(monthly\_interest)

        principal\_paid.append(monthly\_principal)

        if remaining\_principal <= 0:

            break

    return emi, interest\_paid, principal\_paid

# Step 3: Calculate interest lost if loan is closed early

def calculate\_early\_closure\_loss(interest\_paid, early\_closure\_month):

    total\_interest\_paid\_early = sum(interest\_paid[:early\_closure\_month])

    total\_interest\_paid\_full = sum(interest\_paid)

    interest\_lost = total\_interest\_paid\_full - total\_interest\_paid\_early

    return interest\_lost

# Step 4: Plot EMI chart showing interest and principal components

def plot\_emi\_chart(interest\_paid, principal\_paid):

    months = list(range(1, len(interest\_paid) + 1))

    plt.plot(months, interest\_paid, label='Interest Component')

    plt.plot(months, principal\_paid, label='Principal Component')

    plt.xlabel('Month')

    plt.ylabel('Amount Paid')

    plt.title('EMI Breakdown: Interest vs Principal')

    plt.legend()

    plt.grid(True)

    plt.show()

# Step 5: Main function to handle loan details and early closure

def main():

    # Input loan details

    principal = float(input("Enter the loan amount: "))

    rate\_of\_interest = float(input("Enter the annual interest rate (in %): "))

    loan\_term = int(input("Enter the loan term (in months): "))

    # Calculate EMI and simulate loan repayment

    emi, interest\_paid, principal\_paid = simulate\_loan\_repayment(principal, rate\_of\_interest, loan\_term)

    print(f"Monthly EMI: {emi:.2f}")

    # Plot the EMI breakdown chart

    plot\_emi\_chart(interest\_paid, principal\_paid)

    # Input for early loan closure

    early\_closure\_month = int(input("Enter the month when loan is closed early (0 for full term): "))

    if early\_closure\_month > 0:

        # Calculate interest loss for early closure

        interest\_lost = calculate\_early\_closure\_loss(interest\_paid, early\_closure\_month)

        print(f"Interest lost due to early closure: {interest\_lost:.2f}")

    else:

        print("Loan was repaid over the full term.")

# Run the main function

if \_\_name\_\_ == "\_\_main\_\_": # Corrected the typo in the if statement

    main() # Added indentation to ensure the main function is called within the if block

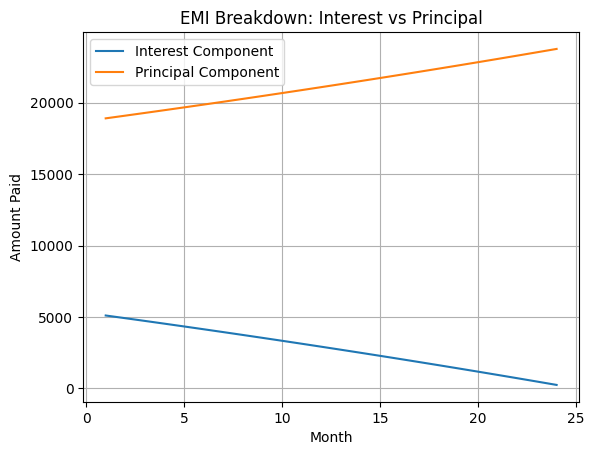
OUTPUT:

Enter the loan amount: 510000

Enter the annual interest rate (in %): 12

Enter the loan term (in months): 24

Monthly EMI: 24007.47



**Q.3. 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%.**

**Ans**: Here, we model an insurance system that calculates the monthly, quarterly, and yearly premiums for a vehicle based on its depreciating value over time. The solution follows these steps:

1. **Initial Vehicle Value:**

- The user inputs the initial value of the vehicle, the insurance term (in years), and the premium rate as a percentage of the vehicle’s value.

2. **Depreciation Calculation**:

- Each year, the vehicle's value depreciates by 7%. The depreciation is calculated using the formula:

`Depreciated Value = Previous Value × (1 - 0.07)`

- This process repeats annually for the entire insurance term.

3. **Premium Calculation**:

- Insurance premiums are calculated based on the depreciated value for each year. The yearly premium is a fixed percentage of the vehicle’s value.

- From the yearly premium, the quarterly and monthly premiums are derived as follows:

- `Quarterly Premium = Yearly Premium ÷ 4`

- `Monthly Premium = Yearly Premium ÷ 12`

4. **Premium and Depreciation Visualization**:

- A chart is generated to display:

- The depreciating value of the vehicle over the insurance term.

- The corresponding yearly, quarterly, and monthly premiums over time.

This approach models the impact of depreciation on the vehicle’s value and the resulting insurance premiums, providing a dynamic and visual understanding of how premiums change as the vehicle’s value declines over the years.

CODE:

import matplotlib.pyplot as plt

# Step 1: Define a function to calculate the depreciated value of the vehicle each year

def calculate\_depreciation(initial\_value, years, depreciation\_rate=0.07):

    values = [initial\_value]

    for year in range(1, years + 1):

        depreciated\_value = values[-1] \* (1 - depreciation\_rate)  # Depreciate by 7%

        values.append(depreciated\_value)

    return values

# Step 2: Calculate premiums based on the depreciated value of the vehicle

def calculate\_premiums(depreciated\_values, premium\_rate):

    yearly\_premiums = [value \* premium\_rate for value in depreciated\_values]

    quarterly\_premiums = [yearly / 4 for yearly in yearly\_premiums]

    monthly\_premiums = [yearly / 12 for yearly in yearly\_premiums]

    return yearly\_premiums, quarterly\_premiums, monthly\_premiums

# Step 3: Plot the depreciation and premium charts

def plot\_depreciation\_and\_premiums(years, values, yearly\_premiums, quarterly\_premiums, monthly\_premiums):

    x = list(range(years + 1))

    plt.figure(figsize=(10, 5))

    # Plot depreciated value

    plt.subplot(1, 2, 1)

    plt.plot(x, values, label='Depreciated Value')

    plt.xlabel('Years')

    plt.ylabel('Value of Vehicle')

    plt.title('Depreciation of Vehicle')

    plt.grid(True)

    plt.legend()

    # Plot premiums

    plt.subplot(1, 2, 2)

    plt.plot(x, yearly\_premiums, label='Yearly Premium')

    plt.plot(x, quarterly\_premiums, label='Quarterly Premium')

    plt.plot(x, monthly\_premiums, label='Monthly Premium')

    plt.xlabel('Years')

    plt.ylabel('Premium Amount')

    plt.title('Premiums Over Time')

    plt.grid(True)

    plt.legend()

    plt.tight\_layout()

    plt.show()

# Step 4: Main function to handle user input and calculation

def main():

    # Input vehicle details

    initial\_value = float(input("Enter the initial value of the vehicle: "))

    years = int(input("Enter the number of years for insurance: "))

    premium\_rate = float(input("Enter the insurance premium rate (as a percentage): ")) / 100  # Convert percentage to decimal

    # Calculate depreciation over the years

    depreciated\_values = calculate\_depreciation(initial\_value, years)

    # Calculate premiums based on depreciated values

    yearly\_premiums, quarterly\_premiums, monthly\_premiums = calculate\_premiums(depreciated\_values, premium\_rate)

    # Display the premiums

    for year in range(years + 1):

        print(f"Year {year}: Vehicle Value = {depreciated\_values[year]:.2f}, Yearly Premium = {yearly\_premiums[year]:.2f}, Quarterly Premium = {quarterly\_premiums[year]:.2f}, Monthly Premium = {monthly\_premiums[year]:.2f}")

    # Plot the depreciation and premium charts

    plot\_depreciation\_and\_premiums(years, depreciated\_values, yearly\_premiums, quarterly\_premiums, monthly\_premiums)

# Run the main function

if \_\_name\_\_ == "\_\_main\_\_":

    main()

OUTPUT:

Enter the initial value of the vehicle: 2500000

Enter the number of years for insurance: 5

Enter the insurance premium rate (as a percentage): 10

Year 0: Vehicle Value = 2500000.00, Yearly Premium = 250000.00, Quarterly Premium = 62500.00, Monthly Premium = 20833.33

Year 1: Vehicle Value = 2325000.00, Yearly Premium = 232500.00, Quarterly Premium = 58125.00, Monthly Premium = 19375.00

Year 2: Vehicle Value = 2162250.00, Yearly Premium = 216225.00, Quarterly Premium = 54056.25, Monthly Premium = 18018.75

Year 3: Vehicle Value = 2010892.50, Yearly Premium = 201089.25, Quarterly Premium = 50272.31, Monthly Premium = 16757.44

Year 4: Vehicle Value = 1870130.02, Yearly Premium = 187013.00, Quarterly Premium = 46753.25, Monthly Premium = 15584.42

Year 5: Vehicle Value = 1739220.92, Yearly Premium = 173922.09, Quarterly Premium = 43480.52, Monthly Premium = 14493.51

