**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.

**Solution:** EMI calculation and loan schedule plot.

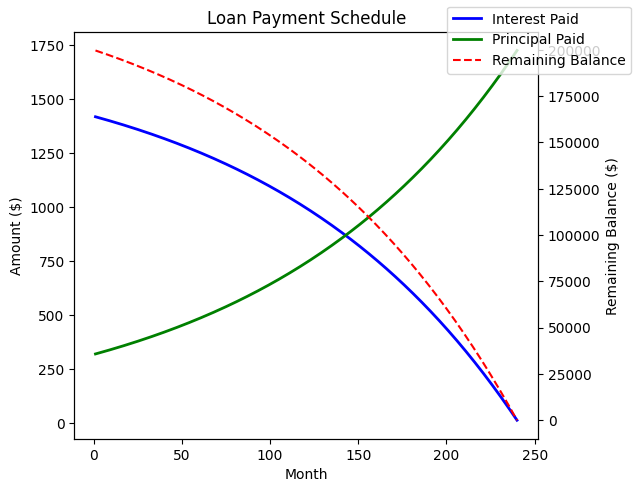
Here’s Python code calculates EMI for a housing loan, generates a repayment schedule, and plots a chart showing interest, principal paid, and remaining balance over the loan term.

**Code:**

|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt  def calculate\_emi(principal, annual\_rate, months):      monthly\_rate = annual\_rate / 12 / 100  # converting annual rate to monthly rate      emi = principal \* monthly\_rate \* ((1 + monthly\_rate) \*\* months) / (((1 + monthly\_rate) \*\* months) - 1)      return emi  def generate\_loan\_schedule(principal, annual\_rate, months):      schedule = []      monthly\_rate = annual\_rate / 12 / 100      remaining\_balance = principal        for month in range(1, months + 1):          interest = remaining\_balance \* monthly\_rate          principal\_payment = calculate\_emi(principal, annual\_rate, months) - interest          remaining\_balance -= principal\_payment          schedule.append((month, interest, principal\_payment, remaining\_balance))        return schedule  def plot\_loan\_schedule(schedule):      months = [entry[0] for entry in schedule]      interest\_paid = [entry[1] for entry in schedule]      principal\_paid = [entry[2] for entry in schedule]      remaining\_balance = [entry[3] for entry in schedule]        fig, ax1 = plt.subplots()      ax1.set\_xlabel('Month')      ax1.set\_ylabel('Amount ($)')      ax1.plot(months, interest\_paid, 'b-', label='Interest Paid', linewidth=2)      ax1.plot(months, principal\_paid, 'g-', label='Principal Paid', linewidth=2)      ax1.tick\_params(axis='y')        ax2 = ax1.twinx()      ax2.set\_ylabel('Remaining Balance ($)')      ax2.plot(months, remaining\_balance, 'r--', label='Remaining Balance')      ax2.tick\_params(axis='y')        fig.tight\_layout()      fig.legend(loc='upper right')      plt.title('Loan Payment Schedule')      plt.show()  # Example usage  principal = 200000  # principal amount of the loan  annual\_rate = 8.5  # annual interest rate (in percent)  months = 12 \* 20  # 20 years loan term  emi = calculate\_emi(principal, annual\_rate, months)  print(f"EMI: ${emi:.2f} per month")  loan\_schedule = generate\_loan\_schedule(principal, annual\_rate, months)  plot\_loan\_schedule(loan\_schedule) |

**Output:**

**EMI: $1735.65 per month**

****

**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 quterly premiums based on no. of years of insurance where in each year, the value of the vehicle depreciates by 7%.

**Solution:** Insurance Model with Depreciation and Premium Calculation

Here’s a Python model that holds information about an insurer’s vehicle and calculates monthly, quarterly, and yearly premiums based on the vehicle's depreciation over the years.

**Code:**

|  |
| --- |
| import matplotlib.pyplot as plt    class VehicleInsurance:  def \_\_init\_\_(self, owner\_name, vehicle\_value, insurance\_years, annual\_premium\_rate):  self.owner\_name = owner\_name  self.initial\_value = vehicle\_value  self.current\_value = vehicle\_value  self.insurance\_years = insurance\_years  self.annual\_premium\_rate = annual\_premium\_rate  self.premiums = {}    def calculate\_depreciation(self):  for year in range(1, self.insurance\_years + 1):  depreciation = self.current\_value \* 0.07 # 7% depreciation each year  self.current\_value -= depreciation  annual\_premium = self.current\_value \* self.annual\_premium\_rate / 100  monthly\_premium = annual\_premium / 12  quarterly\_premium = annual\_premium / 4  self.premiums[year] = {  "Annual": annual\_premium,  "Quarterly": quarterly\_premium,  "Monthly": monthly\_premium  }    def display\_premiums(self):  print(f"Premiums for {self.owner\_name}'s vehicle:")  for year, premium in self.premiums.items():  print(f"Year {year}: Annual: {premium['Annual']:.2f}, Quarterly: {premium['Quarterly']:.2f}, Monthly: {premium['Monthly']:.2f}")    def plot\_premiums(self):  years = list(self.premiums.keys())  annual\_premiums = [self.premiums[year]['Annual'] for year in years]  quarterly\_premiums = [self.premiums[year]['Quarterly'] for year in years]  monthly\_premiums = [self.premiums[year]['Monthly'] for year in years]    plt.plot(years, annual\_premiums, label='Annual Premium')  plt.plot(years, quarterly\_premiums, label='Quarterly Premium', linestyle='--')  plt.plot(years, monthly\_premiums, label='Monthly Premium', linestyle=':')  plt.xlabel('Years')  plt.ylabel('Premium Amount')  plt.title('Vehicle Insurance Premium Over Time')  plt.legend()  plt.show()    # Example usage  vehicle\_insurance = VehicleInsurance(owner\_name="John Doe", vehicle\_value=500000, insurance\_years=5, annual\_premium\_rate=5)  vehicle\_insurance.calculate\_depreciation()  vehicle\_insurance.display\_premiums()  vehicle\_insurance.plot\_premiums() |

**Output:**

