**Name: Vedant Shrirao SEM: VII**

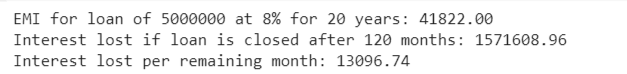
**PRN: 21070521091 Section:B**

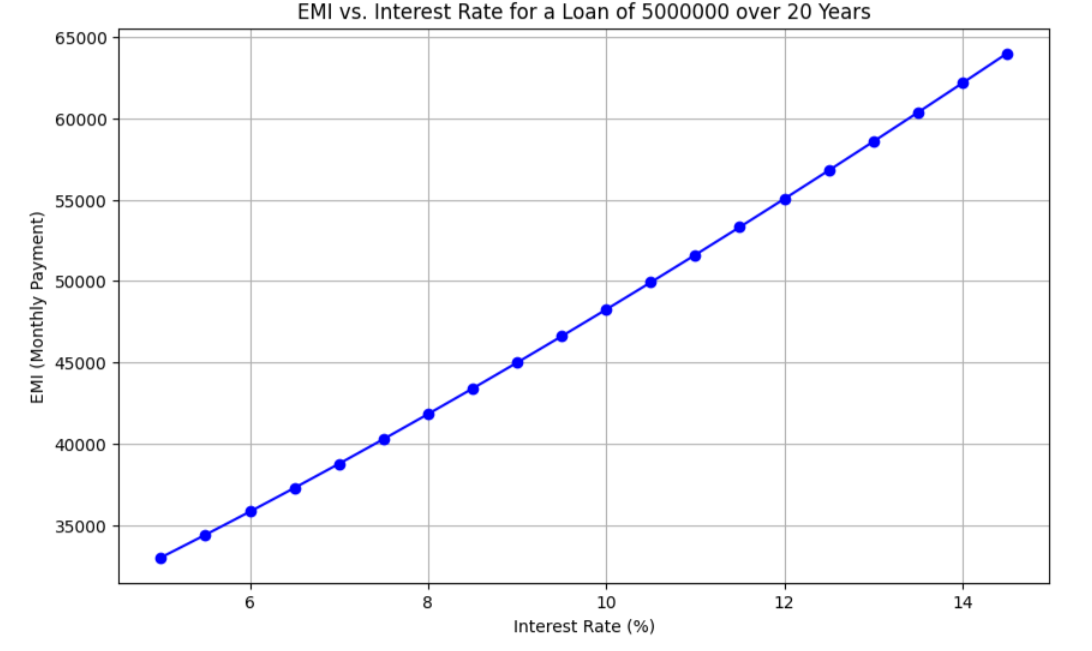
**Q:2** Generate a model in Python to represent a Housing loan scheme and create a chart to display the Emi based on rate of interest and reducing balance for a given period. If a customer wishes to close the loan earlier, print the interest lost distributed over the remaining no. Of months. Assume suitable data and inputs as necessary.

**Solution:** A housing loan helps individuals purchase property, repaid through Equated Monthly Installments (EMIs) over a set tenure. The EMI includes both principal and interest, where the reducing balance method applies interest on the outstanding loan balance, reducing the interest burden over time. The EMI amount is influenced by the interest rate, with higher rates increasing the EMI. If the borrower chooses early loan closure, they save on the interest that would have been paid on the remaining tenure, reducing the overall cost of the loan.

**Code:**

|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt  class HousingLoan:  def \_\_init\_\_(self, principal, rate, years):  self.principal = principal  self.annual\_rate = rate  self.monthly\_rate = rate / (12 \* 100)  self.num\_years = years  self.num\_months = years \* 12  def calculate\_emi(self):  P = self.principal  r = self.monthly\_rate  n = self.num\_months  emi = P \* r \* (1 + r)\*\*n / ((1 + r)\*\*n - 1)  return emi    def calculate\_interest\_for\_period(self, months=None):  if months is None:  months = self.num\_months  balance = self.principal  total\_interest = 0  emi = self.calculate\_emi()    for \_ in range(months):  interest = balance \* self.monthly\_rate  total\_interest += interest  principal\_paid = emi - interest  balance -= principal\_paid  return total\_interest    def close\_loan\_early(self, early\_months):  full\_interest = self.calculate\_interest\_for\_period()  interest\_till\_early\_close = self.calculate\_interest\_for\_period(early\_months)  interest\_lost = full\_interest - interest\_till\_early\_close  return interest\_lost, interest\_lost / (self.num\_months - early\_months)  def plot\_emi\_vs\_interest(principal, years):  rates = np.arange(5, 15, 0.5)  emis = []    for rate in rates:  loan = HousingLoan(principal, rate, years)  emi = loan.calculate\_emi()  emis.append(emi)    plt.figure(figsize=(10, 6))  plt.plot(rates, emis, marker='o', color='b')  plt.title(f'EMI vs. Interest Rate for a Loan of {principal} over {years} Years')  plt.xlabel('Interest Rate (%)')  plt.ylabel('EMI (Monthly Payment)')  plt.grid(True)  plt.show()  principal = 5000000  rate = 8  years = 20  loan = HousingLoan(principal, rate, years)  emi = loan.calculate\_emi()  print(f"EMI for loan of {principal} at {rate}% for {years} years: {emi:.2f}")  early\_months = 120  interest\_lost, lost\_per\_month = loan.close\_loan\_early(early\_months)  print(f"Interest lost if loan is closed after {early\_months} months: {interest\_lost:.2f}")  print(f"Interest lost per remaining month: {lost\_per\_month:.2f}")  plot\_emi\_vs\_interest(principal, years) |

**Output:**

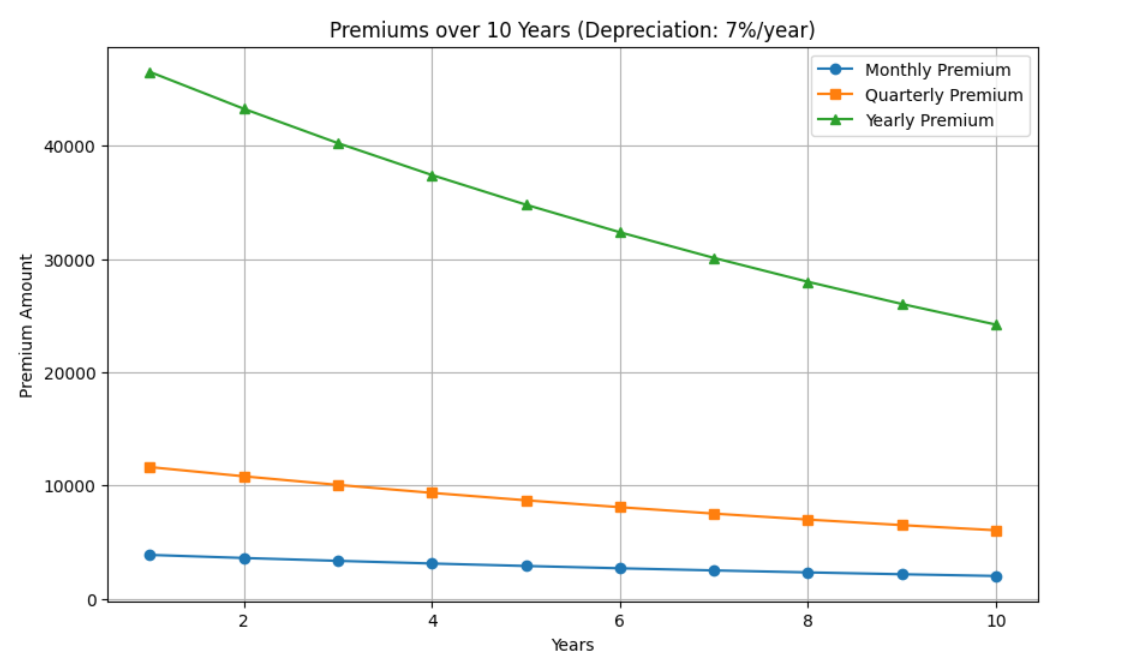


**Q:2** Generate a model for an insurance company to hold information on the insurer's vehicle, and create a chart of monthly, yearly, and quterly premiums based on no. of years of insurance where in each year, the value of the vehicle depreciates by 7%.

**Solution:** The Vehicle Insurance Model calculates the premiums based on the vehicle's depreciating value, which decreases by 7% each year. The insurance premium is determined as a percentage of the vehicle's value and is calculated for monthly, quarterly, and yearly intervals. Over time, as the vehicle's value reduces, the premium amount also decreases. This model helps in understanding how depreciation impacts the cost of insurance and allows visualizing the premiums over multiple years. The premium is lower in later years due to the reduced vehicle value.

**Code:**

|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt  class VehicleInsurance:  def \_\_init\_\_(self, initial\_value, yearly\_rate):  self.initial\_value = initial\_value  self.yearly\_rate = yearly\_rate  self.depreciation\_rate = 0.07    def depreciated\_value(self, years):  return self.initial\_value \* ((1 - self.depreciation\_rate) \*\* years)    def calculate\_premium(self, years):  value = self.depreciated\_value(years)  yearly\_premium = value \* self.yearly\_rate  quarterly\_premium = yearly\_premium / 4  monthly\_premium = yearly\_premium / 12  return monthly\_premium, quarterly\_premium, yearly\_premium  def plot\_premiums\_over\_years(initial\_value, yearly\_rate, max\_years):  years = np.arange(1, max\_years + 1)  monthly\_premiums = []  quarterly\_premiums = []  yearly\_premiums = []    insurance = VehicleInsurance(initial\_value, yearly\_rate)    for year in years:  monthly, quarterly, yearly = insurance.calculate\_premium(year)  monthly\_premiums.append(monthly)  quarterly\_premiums.append(quarterly)  yearly\_premiums.append(yearly)    plt.figure(figsize=(10, 6))  plt.plot(years, monthly\_premiums, label="Monthly Premium", marker='o')  plt.plot(years, quarterly\_premiums, label="Quarterly Premium", marker='s')  plt.plot(years, yearly\_premiums, label="Yearly Premium", marker='^')  plt.title(f'Premiums over {max\_years} Years (Depreciation: 7%/year)')  plt.xlabel('Years')  plt.ylabel('Premium Amount')  plt.grid(True)  plt.legend()  plt.show()  initial\_value = 1000000  yearly\_rate = 0.05  max\_years = 10  plot\_premiums\_over\_years(initial\_value, yearly\_rate, max\_years) |

**Output:**