**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%.

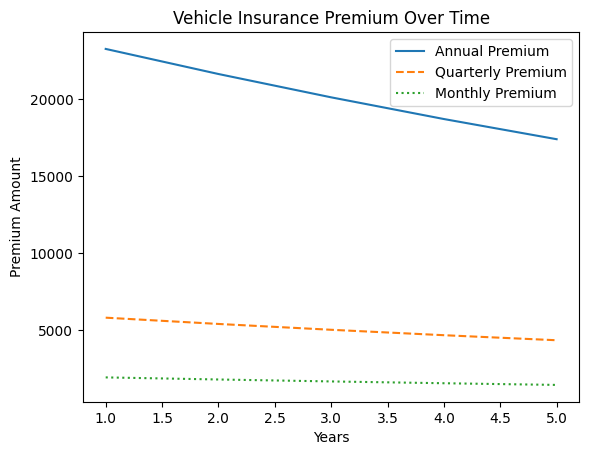
**Solution:** Insurance Model with Depreciation and Premium Calculation

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:**



**Q:4** Generate a model to represent interest calculations of a Bank account where the process of calculating interest for 6 months is a. Find minimum balance for each month b. Make a total of all minimum balances c. Calculate interest based on interest rate d. Divide interest by 12 to find one-month interest e. Multiply interest by 6 to show interest in the account. Generate a model to represent transactions and interest calculations for 6 months.

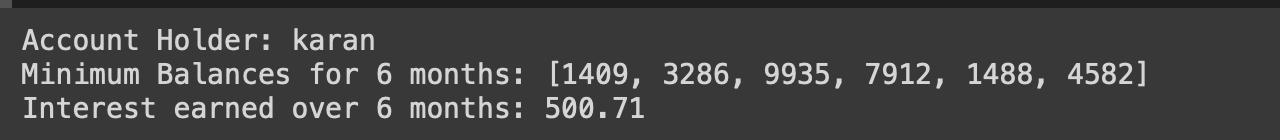
**Solution:** Bank Interest Calculation Model

Here’s a Python model to represent a bank account with interest calculations based on minimum balances over six months.

**Code:**

|  |
| --- |
| import random    class BankAccount:  def \_\_init\_\_(self, account\_holder, interest\_rate):  self.account\_holder = account\_holder  self.interest\_rate = interest\_rate  self.transactions = {month: [] for month in range(1, 7)}  self.minimum\_balances = []    def add\_transaction(self, month, amount):  self.transactions[month].append(amount)    def calculate\_minimum\_balances(self):  for month, trans in self.transactions.items():  balance = sum(trans)  if balance:  self.minimum\_balances.append(min(trans))  else:  self.minimum\_balances.append(0)    def calculate\_interest(self):  total\_min\_balance = sum(self.minimum\_balances)  interest\_per\_year = (total\_min\_balance \* self.interest\_rate) / 100  monthly\_interest = interest\_per\_year / 12  six\_month\_interest = monthly\_interest \* 6  return six\_month\_interest    def display\_account(self):  print(f"Account Holder: {self.account\_holder}")  print(f"Minimum Balances for 6 months: {self.minimum\_balances}")  print(f"Interest earned over 6 months: {self.calculate\_interest():.2f}")    # Example usage  bank\_account = BankAccount(account\_holder="karan", interest\_rate=3.5)    # Random transactions for 6 months  random.seed(42)  for month in range(1, 7):  for \_ in range(random.randint(1, 5)):  bank\_account.add\_transaction(month, random.randint(1000, 10000))    bank\_account.calculate\_minimum\_balances()  bank\_account.display\_account() |

**Output:**

****