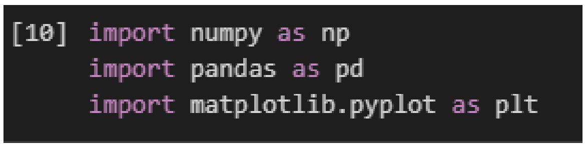
**Name: Om Kuhikar**
**PRN: 22070521504**
**Section: B**

**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%.

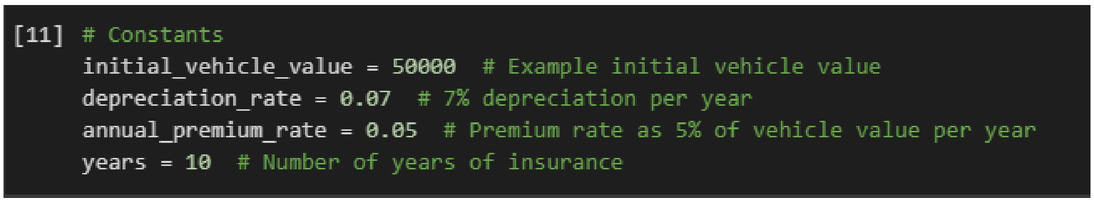
1. **Importing Required Libraries**



```
[10] import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
```

- NumPy (np) is used for numerical operations, such as generating random numbers or performing mathematical calculations.
- pandas (pd) is used for handling structured data, such as storing and manipulating tabular data.
- matplotlib.pyplot (plt) is a popular library for creating visualizations and plotting graphs in Python.

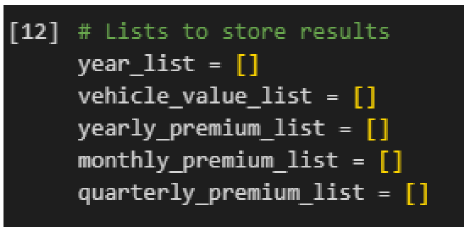
2. **Defining Constants**



```
[11] # Constants
     initial_vehicle_value = 50000  # Example initial vehicle value
     depreciation_rate = 0.07  # 7% depreciation per year
     annual_premium_rate = 0.05  # Premium rate as 5% of vehicle value per year
     years = 10  # Number of years of insurance
```

- initial_vehicle_value: This variable stores the initial value of the vehicle, which is $50,000 in this example.
- depreciation_rate: The annual depreciation rate is set to 7%. This means the vehicle's value will decrease by 7% every year.
- annual_premium_rate: This is the insurance premium rate, set at 5% of the current vehicle value each year.
- years: The number of years for which we want to calculate the vehicle's depreciation and premiums is set to 10.

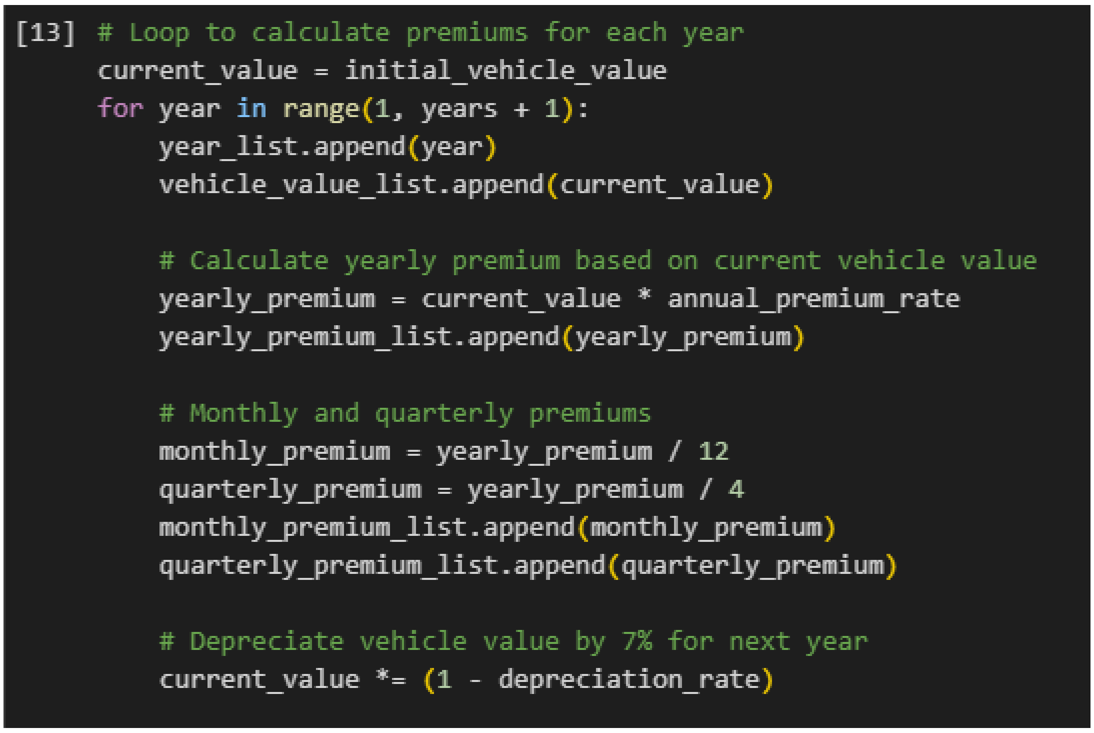
3. **Initializing Lists for Storing Results**



```
[12] # Lists to store results
     year_list = []
     vehicle_value_list = []
     yearly_premium_list = []
     monthly_premium_list = []
     quarterly_premium_list = []
```

- We are initializing five empty lists to store the results for each year:
  - year_list: To store the year number (1 through 10).
  - vehicle_value_list: To store the vehicle's depreciated value at the end of each year.
  - yearly_premium_list: To store the calculated yearly insurance premium based on the vehicle's value.
  - monthly_premium_list: To store the monthly insurance premium.
  - quarterly_premium_list: To store the quarterly insurance premium.

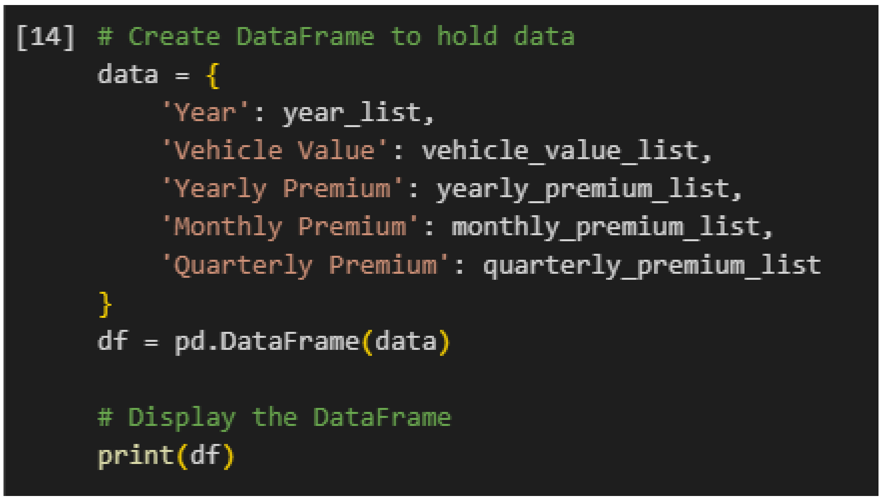
4. **Looping to Calculate Premiums and Depreciation**



```
[13] # Loop to calculate premiums for each year
     current_value = initial_vehicle_value
     for year in range(1, years + 1):
         year_list.append(year)
         vehicle_value_list.append(current_value)

         # Calculate yearly premium based on current vehicle value
         yearly_premium = current_value * annual_premium_rate
         yearly_premium_list.append(yearly_premium)

         # Monthly and quarterly premiums
         monthly_premium = yearly_premium / 12
         quarterly_premium = yearly_premium / 4
         monthly_premium_list.append(monthly_premium)
         quarterly_premium_list.append(quarterly_premium)

         # Depreciate vehicle value by 7% for next year
         current_value *= (1 - depreciation_rate)
```

- We start with the current_value as the initial value of the vehicle ($50,000).
- A for loop runs for each year (from 1 to 10).
  - In each iteration, the current year is appended to the year_list, and the current value of the vehicle is added to the vehicle_value_list.
  - The yearly premium is calculated as 5% of the vehicle's current value and is added to the yearly_premium_list.
  - The monthly premium is simply the yearly premium divided by 12, and the quarterly premium is the yearly premium divided by 4. These values are stored in the respective lists.
  - After calculating the premiums for the current year, the vehicle's value is depreciated by 7% for the next year.

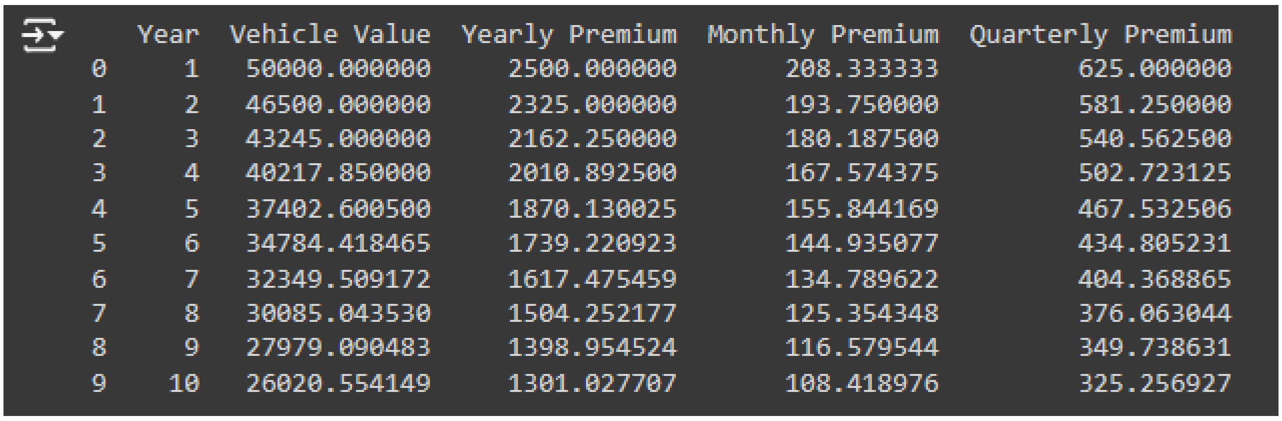
5. **Creating a DataFrame to Hold the Data**



```
[14] # Create DataFrame to hold data
     data = {
         'Year': year_list,
         'Vehicle Value': vehicle_value_list,
         'Yearly Premium': yearly_premium_list,
         'Monthly Premium': monthly_premium_list,
         'Quarterly Premium': quarterly_premium_list
     }
     df = pd.DataFrame(data)

     # Display the DataFrame
     print(df)
```

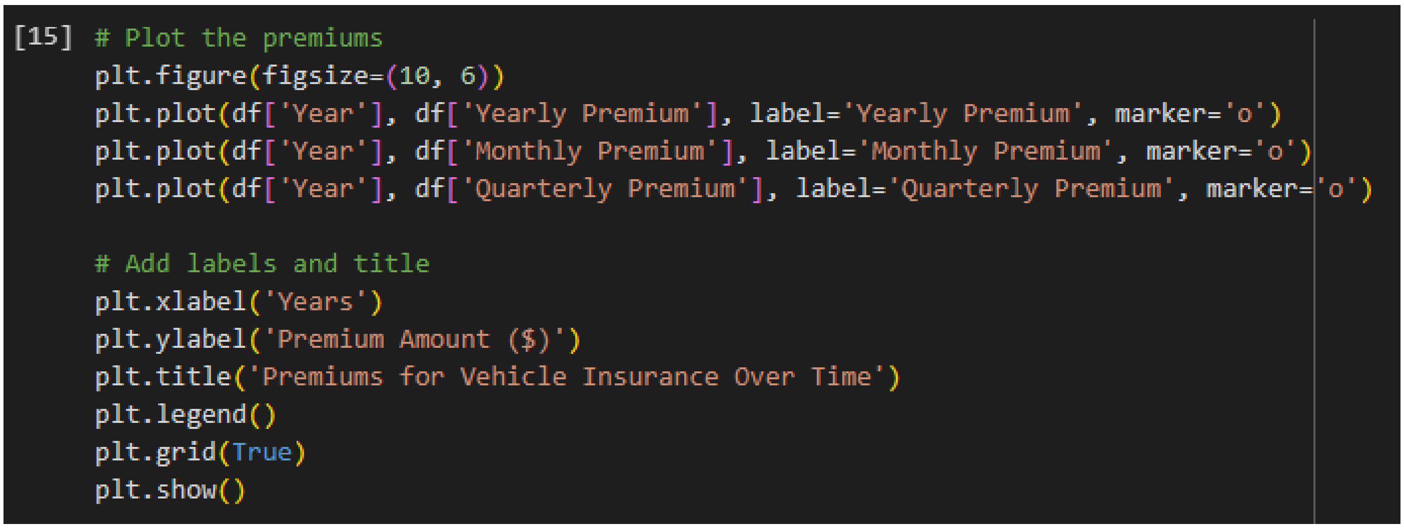
- We are using pandas to create a DataFrame, which organizes our data in tabular form with columns for:
  - Year
  - Vehicle Value
  - Yearly Premium
  - Monthly Premium
  - Quarterly Premium

- The DataFrame makes it easier to view, manipulate, and export the data. Finally, we print the DataFrame to display the results.



```
   Year  Vehicle Value  Yearly Premium  Monthly Premium  Quarterly Premium
0     1   50000.000000     2500.000000       208.333333         625.000000
1     2   46500.000000     2325.000000       193.750000         581.250000
2     3   43245.000000     2162.250000       180.187500         540.562500
3     4   40217.850000     2010.892500       167.574375         502.723125
4     5   37402.600500     1870.130025       155.844169         467.532506
5     6   34784.418465     1739.220923       144.935077         434.805231
6     7   32349.509172     1617.475459       134.789622         404.368865
7     8   30085.043530     1504.252177       125.354348         376.063044
8     9   27979.090483     1398.954524       116.579544         349.738631
9    10   26020.554149     1301.027707       108.418976         325.256927
```

**6. Plotting the Premiums**



```
[15] # Plot the premiums
     plt.figure(figsize=(10, 6))
     plt.plot(df['Year'], df['Yearly Premium'], label='Yearly Premium', marker='o')
     plt.plot(df['Year'], df['Monthly Premium'], label='Monthly Premium', marker='o')
     plt.plot(df['Year'], df['Quarterly Premium'], label='Quarterly Premium', marker='o')

     # Add labels and title
     plt.xlabel('Years')
     plt.ylabel('Premium Amount ($)')
     plt.title('Premiums for Vehicle Insurance Over Time')
     plt.legend()
     plt.grid(True)
     plt.show()
```

- We use Matplotlib to visualize the premiums for each year.
  - plt.plot(df['Year'], ...): This line plots the year on the x-axis and the premiums (yearly, monthly, and quarterly) on the y-axis.
  - label: Labels are added for each type of premium so they can be differentiated in the graph.
  - We then add labels to the axes, a title, and a legend for the graph.
  - plt.grid(True): Adds a grid for better readability of the graph.
  - plt.show(): Displays the plot.

**Conclusion:** This code calculates the yearly, monthly, and quarterly premiums for a vehicle insurance model where the vehicle value depreciates by 7% each year. The premiums are calculated based on a 5% yearly premium rate applied to the depreciated vehicle value. Finally, the data is plotted to visualize how the premiums change over time.

**Q:6** Generate a model to represent a mathematical equation, write a program to parse the equation, and ask for input for each parameter.

1. **Import Required Libraries**

```python
import sympy as sp
```

- We use sympy to handle symbolic mathematics, which allows us to parse and evaluate mathematical equations.

2. **Define and Parse the Equation**

```python
# Define the equation as a string
equation_str = "a * x**2 + b * x + c"

# Parse the equation using sympy
x = sp.Symbol('x')
a, b, c = sp.symbols('a b c')
equation = eval(equation_str)

# Display the parsed equation
print(f"Parsed Equation: {equation}")
```

- equation_str is a string representation of the equation a·x$^2$ + b·x + c.
- We use sympy.symbols to create symbolic variables and eval to convert the string into a symbolic expression.
- This block prints the parsed equation for verification.

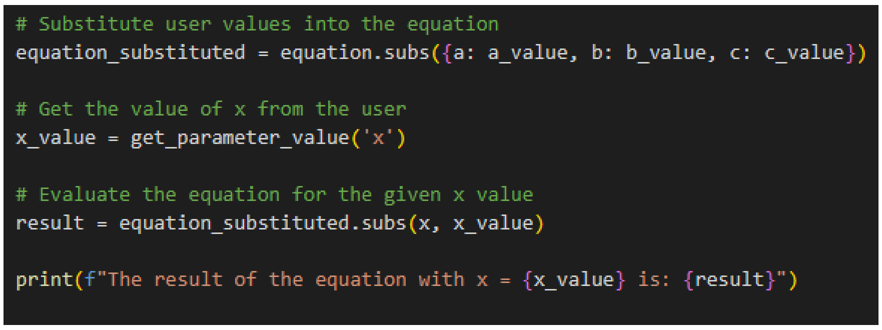
3. **Prompt User for Input**

```python
# Function to get user input for each parameter
def get_parameter_value(param_name):
    while True:
        try:
            value = float(input(f"Enter the value for {param_name}: "))
            return value
        except ValueError:
            print("Invalid input. Please enter a numeric value.")

# Get values for parameters
a_value = get_parameter_value('a')
b_value = get_parameter_value('b')
c_value = get_parameter_value('c')
```

- get_parameter_value prompts the user to enter a numeric value for each parameter.
- This function ensures that the input is valid (numeric) and repeats the prompt if invalid input is provided.
- Values for parameters aaa, bbb, and ccc are collected using this function.

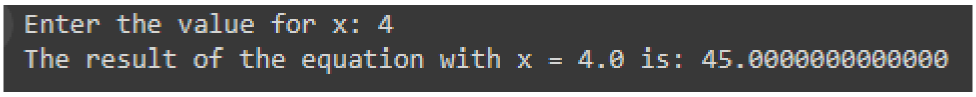
```
Enter the value for a: 2
Enter the value for b: 3
Enter the value for c: 1
```

## 4. Evaluate the Equation



```
# Substitute user values into the equation
equation_substituted = equation.subs({a: a_value, b: b_value, c: c_value})

# Get the value of x from the user
x_value = get_parameter_value('x')

# Evaluate the equation for the given x value
result = equation_substituted.subs(x, x_value)

print(f"The result of the equation with x = {x_value} is: {result}")
```

- equation_substituted replaces the symbolic parameters in the equation with the user-provided values.
- The user is prompted to enter a value for x.
- The equation is evaluated using the substituted values, and the result is printed



```
Enter the value for x: 4
The result of the equation with x = 4.0 is: 45.0000000000000
```