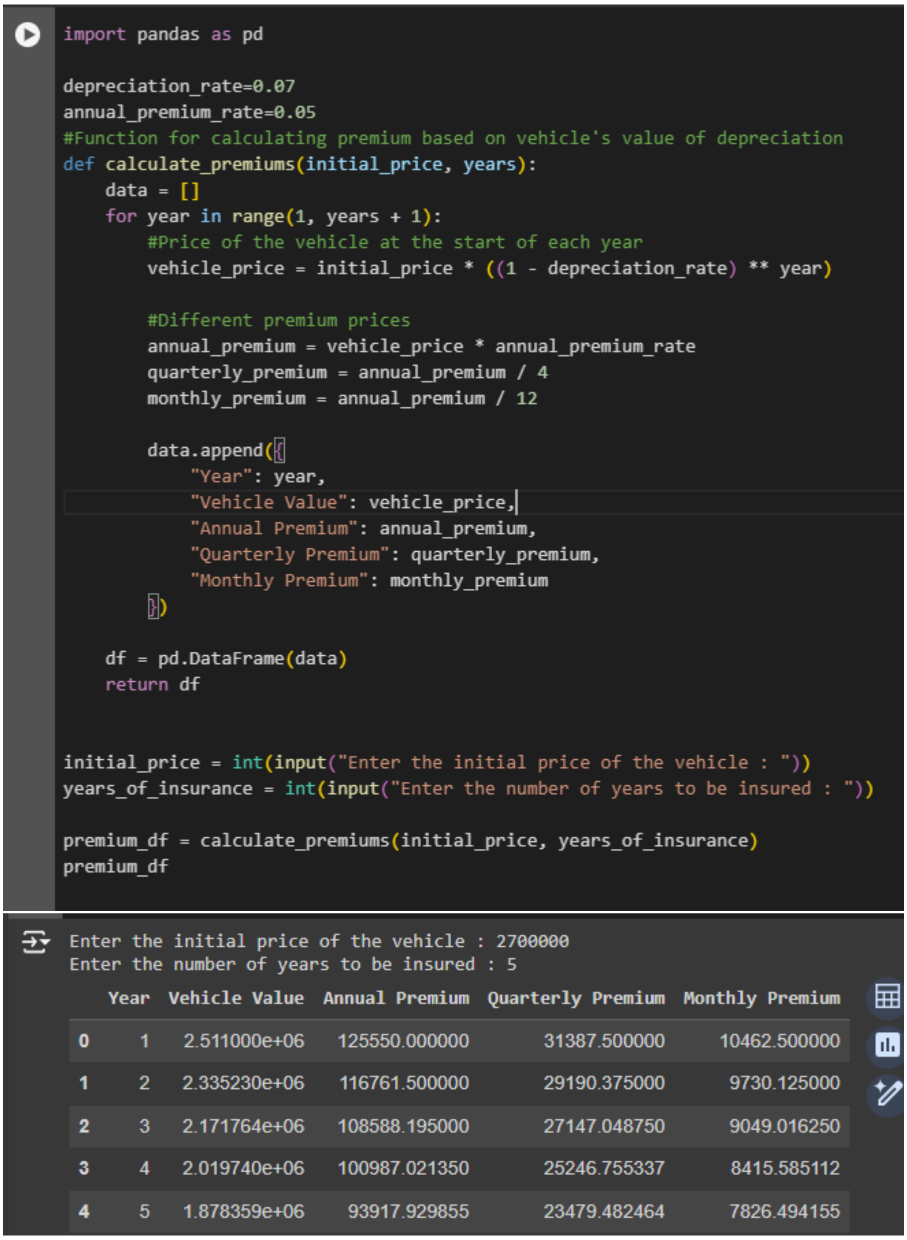
SYMBIOSIS INSTITUTE OF TECHNOLOGY, NAGPUR
Constituent of Symbiosis International (Deemed University), Pune
(Established under Section 3 of the UGC Act of 1956 vide notification number F-9-12/2001-U-3 of the Government of india)
॥वसुधैव कुटुम्बकम्॥ Re-Accredited by NAAC with 'A' Grade

# GENERATIVE ARTIFICIAL INTELLIGENCE(GEN AI)

## CA-2 Assignment

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



```python
import pandas as pd

depreciation_rate=0.07
annual_premium_rate=0.05
#Function for calculating premium based on vehicle's value of depreciation
def calculate_premiums(initial_price, years):
    data = []
    for year in range(1, years + 1):
        #Price of the vehicle at the start of each year
        vehicle_price = initial_price * ((1 - depreciation_rate) ** year)

        #Different premium prices
        annual_premium = vehicle_price * annual_premium_rate
        quarterly_premium = annual_premium / 4
        monthly_premium = annual_premium / 12

        data.append({
            "Year": year,
            "Vehicle Value": vehicle_price,
            "Annual Premium": annual_premium,
            "Quarterly Premium": quarterly_premium,
            "Monthly Premium": monthly_premium
        })

    df = pd.DataFrame(data)
    return df


initial_price = int(input("Enter the initial price of the vehicle : "))
years_of_insurance = int(input("Enter the number of years to be insured : "))

premium_df = calculate_premiums(initial_price, years_of_insurance)
premium_df
```

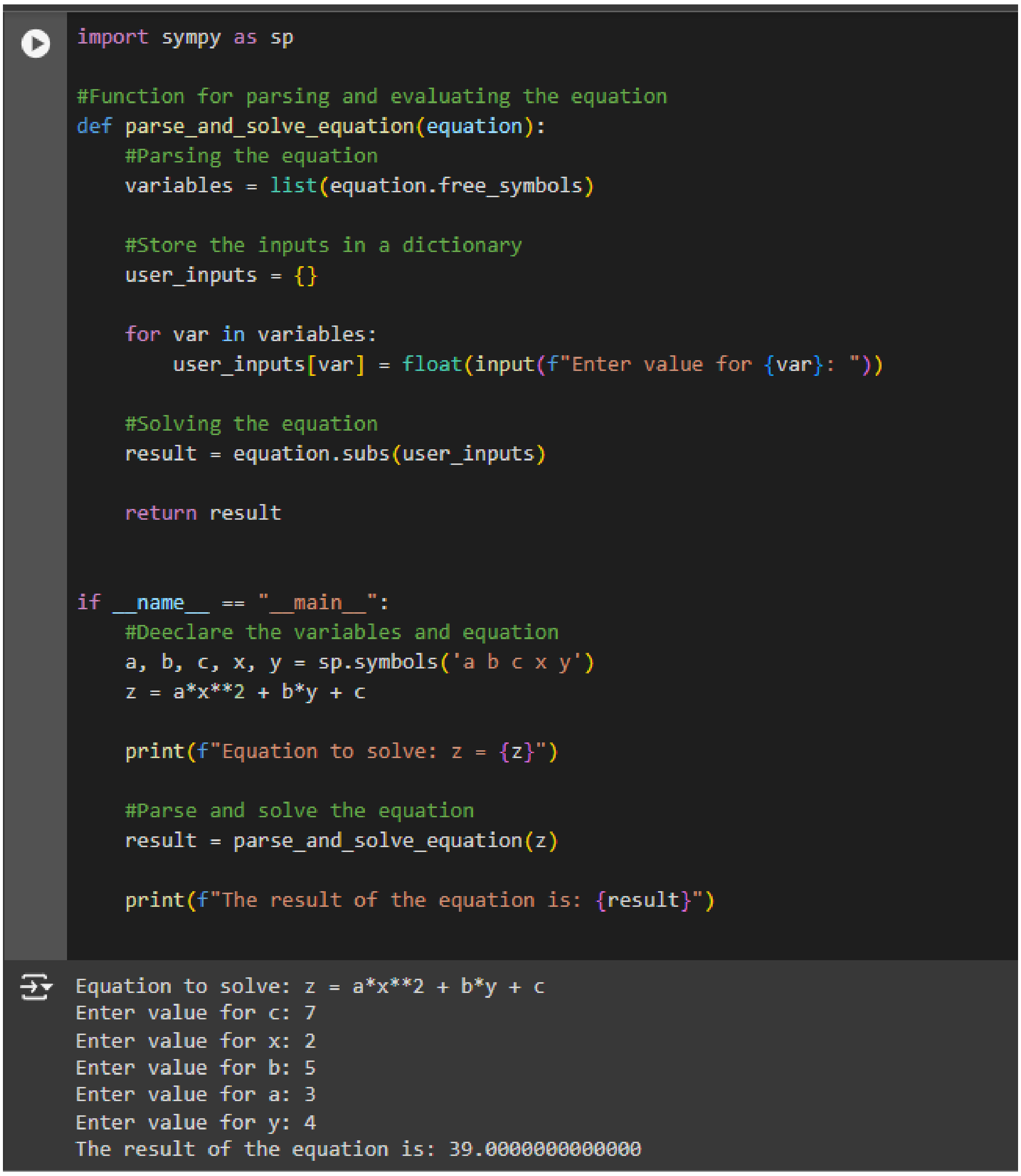
```
Enter the initial price of the vehicle : 2700000
Enter the number of years to be insured : 5
```

| | Year | Vehicle Value | Annual Premium | Quarterly Premium | Monthly Premium |
|---|---|---|---|---|---|
| **0** | 1 | 2.511000e+06 | 125550.000000 | 31387.500000 | 10462.500000 |
| **1** | 2 | 2.335230e+06 | 116761.500000 | 29190.375000 | 9730.125000 |
| **2** | 3 | 2.171764e+06 | 108588.195000 | 27147.048750 | 9049.016250 |
| **3** | 4 | 2.019740e+06 | 100987.021350 | 25246.755337 | 8415.585112 |
| **4** | 5 | 1.878359e+06 | 93917.929855 | 23479.482464 | 7826.494155 |

**Explanation:**

- It helps a user to budget the total cost that it will take to insure a certain car for a given number of years while weighing the depreciation cost.
- The code also imports the pandas data analysis master to cope with and print the tabular form of data.
- Constants for depreciation and annual premium rates are defined:
    - It is also established that the value of a vehicle depreciates by 7% per annum.
    - In its approximation to the current market or models, the cost of the annual premium is pegged at 5 percent cost of the vehicle.
- The main function, calculate_premiums(), takes two inputs:
    - The cost of the car as it was bought by the owner at the first instance.
    - In addition, what period of years do you wish to cover it?
- A loop inside the function gives the value of the vehicle at the start of every year using depreciation.
- The function then computes:
    - Annual premium based on the depreciated value.
    - Quarterly premium (1/4 of the annual premium).
    - Monthly premium (1/12 of the annual premium).
- The values of each year are its lists including the vehicle value list, the annual premium list, the quarterly premium list, and the monthly premium list.
- This list is then converted into pandas DataFrame for better format to display and easy manipulation.
- The user is prompted to input:
    - The initial price of the vehicle.
    - The number of years to insure the vehicle.
- The final data frame is returned and displays how the value of the vehicle or the price of the insurance premiums decreases every year due to depreciation.

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



```python
import sympy as sp

#Function for parsing and evaluating the equation
def parse_and_solve_equation(equation):
    #Parsing the equation
    variables = list(equation.free_symbols)

    #Store the inputs in a dictionary
    user_inputs = {}

    for var in variables:
        user_inputs[var] = float(input(f"Enter value for {var}: "))

    #Solving the equation
    result = equation.subs(user_inputs)

    return result


if __name__ == "__main__":
    #Deeclare the variables and equation
    a, b, c, x, y = sp.symbols('a b c x y')
    z = a*x**2 + b*y + c

    print(f"Equation to solve: z = {z}")

    #Parse and solve the equation
    result = parse_and_solve_equation(z)

    print(f"The result of the equation is: {result}")
```

```
Equation to solve: z = a*x**2 + b*y + c
Enter value for c: 7
Enter value for x: 2
Enter value for b: 5
Enter value for a: 3
Enter value for y: 4
The result of the equation is: 39.0000000000000
```

**Explanation:**

- This Python script uses the SymPy library to solve symbolic equations, allowing the user to input specific values for the variables.
- The parse_and_solve_equation(equation) function is key to the process:
    - It extracts the variables from the symbolic equation using equation.free_symbols.
    - The user is prompted to input values for each variable, which are stored in the user_inputs dictionary.

    - The subs() method replaces the variables with the user-provided values, and the result is returned.
- In the main block:
    - Five symbolic variables (a, b, c, x, y) are declared using sp.symbols().
    - The equation z = a*x**2 + b*y + c is defined, representing a quadratic expression.
    - The equation is printed for the user to view.
- Next, the script:
    - Calls the parse_and_solve_equation() function.
    - Prompts the user to input values for each of the variables (a, b, c, x, y).
    - Substitutes these values into the equation and solves it numerically.
- Example:
    - The user inputs c = 7, x = 2, a = 3, b = 5, and y = 4.
    - The equation z = a*x**2 + b*y + c becomes z = 3*2**2 + 5*4 + 7, which evaluates to 39.0.
- In summary, this script is a flexible tool for solving symbolic equations by letting users input variable values, which makes it adaptable for different mathematical expressions.