**Q2) 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.  
  
Code:**

import matplotlib.pyplot as plt

class HousingLoan:

    def \_\_init\_\_(self, principal, interest\_rate, tenure):

        self.principal = principal

        self.interest\_rate = interest\_rate / 100

        self.tenure = tenure  # in months

    def calculate\_emi(self):

        r = self.interest\_rate / 12

        emi = (self.principal \* r \* ((1 + r) \*\* self.tenure)) / (((1 + r) \*\* self.tenure) - 1)

        return emi

    def early\_closure(self, remaining\_months):

        total\_interest\_saved = self.calculate\_emi() \* remaining\_months - (self.principal \* remaining\_months / self.tenure)

        return total\_interest\_saved

loan = HousingLoan(500000, 7, 240)  # Principal, interest rate, tenure (in months)

emi = loan.calculate\_emi()

print(f"Monthly EMI: {emi:.2f}")

remaining\_months = 120

interest\_lost = loan.early\_closure(remaining\_months)

print(f"Interest saved if closed in {remaining\_months} months: {interest\_lost:.2f}")

interest\_rates = range(5, 11)

emis = [HousingLoan(500000, rate, 240).calculate\_emi() for rate in interest\_rates]

plt.plot(interest\_rates, emis)

plt.title('EMI vs Interest Rate')

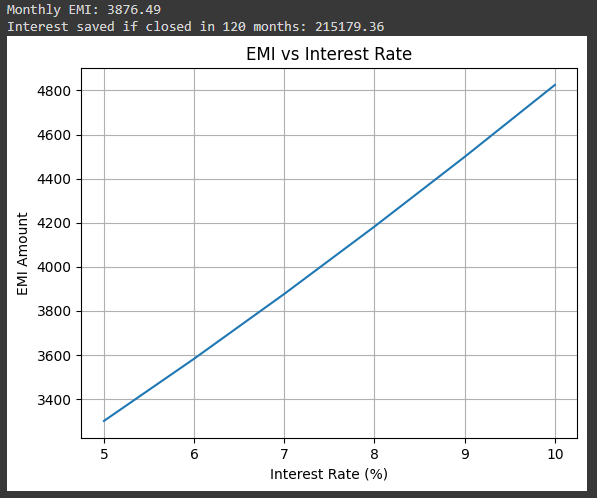
plt.xlabel('Interest Rate (%)')

plt.ylabel('EMI Amount')

plt.grid(True)

plt.show()

**Output:**



**Q6 Generate a model to represent a mathematical equation, write a program to parse the**

**equation, and ask for input for each parameter.**

**Code:**

import sympy as sp

def parse\_equation(equation\_str):

    equation = sp.sympify(equation\_str)

    return equation

equation\_str = input("Enter a mathematical equation: ")

equation = parse\_equation(equation\_str)

variables = equation.free\_symbols

values = {}

for var in variables:

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

result = equation.subs(values)

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

**Output:**

