PRN: 21070521048

Name: Omkar Khadakkar

GenAI Assignment

# Question 1: Housing Loan EMI Model

This model calculates the EMI (Equated Monthly Installment) for a housing loan based on a reducing balance method. The user provides the loan principal, annual interest rate, and tenure (in months). The program also computes the interest loss if the loan is closed early.

import matplotlib.pyplot as plt  
def calculate\_emi(principal, annual\_rate, tenure\_months):  
 monthly\_rate = annual\_rate / 12 / 100  
 emi = principal \* monthly\_rate \* (1 + monthly\_rate) \*\* tenure\_months / ((1 + monthly\_rate) \*\* tenure\_months - 1)  
 return emi  
  
def calculate\_reducing\_balance\_emi(principal, annual\_rate, tenure\_months):  
 monthly\_rate = annual\_rate / 12 / 100  
 balance = principal  
 emi = calculate\_emi(principal, annual\_rate, tenure\_months)  
  
 payments = []  
 for month in range(tenure\_months):  
 interest = balance \* monthly\_rate  
 principal\_paid = emi - interest  
 balance -= principal\_paid  
 payments.append((emi, balance if balance > 0 else 0, interest))  
 return payments  
  
def plot\_emi\_chart(payments, tenure\_months):  
 months = list(range(1, tenure\_months + 1))  
 balances = [payment[1] for payment in payments]  
  
 plt.plot(months, balances, label="Remaining Balance")  
 plt.xlabel("Month")  
 plt.ylabel("Remaining Balance")  
 plt.title("EMI and Reducing Balance over Time")  
 plt.legend()  
 plt.show()  
  
principal = float(input("Enter loan principal amount: "))  
annual\_rate = float(input("Enter annual interest rate (in %): "))  
tenure\_months = int(input("Enter loan tenure in months: "))  
  
payments = calculate\_reducing\_balance\_emi(principal, annual\_rate, tenure\_months)  
plot\_emi\_chart(payments, tenure\_months)  
  
def early\_closure\_interest\_loss(payments, close\_after\_months):  
 interest\_paid = sum([payment[2] for payment in payments[:close\_after\_months]])  
 total\_interest\_if\_full = sum([payment[2] for payment in payments])  
 interest\_loss = total\_interest\_if\_full - interest\_paid  
 return interest\_loss  
  
close\_after\_months = int(input(f"Enter the month after which you plan to close the loan early (1-{tenure\_months}): "))  
  
if close\_after\_months <= tenure\_months:  
 interest\_loss = early\_closure\_interest\_loss(payments, close\_after\_months)  
 print(f"Interest lost if closed early after {close\_after\_months} months: {interest\_loss:.2f}")  
else:  
 print(f"Invalid input. The month should be within the tenure of {tenure\_months} months.")

## Example Output:

Enter loan principal amount: 800000  
Enter annual interest rate (in %): 9  
Enter loan tenure in months: 48  
Enter the month after which you plan to close the loan early (1-48): 24  
Interest lost if closed early after 24 months: 42022.95

# A graph with a line Description automatically generated

# Question 2: Mathematical Equation Parser

This model solves a quadratic equation of the form ax^2 + bx + c = 0. The program calculates the roots using the quadratic formula.

import math

def parse\_quadratic\_equation(a, b, c):

# Calculate the discriminant

discriminant = b\*\*2 - 4\*a\*c

if discriminant > 0:

# Two real and distinct solutions

x1 = (-b + math.sqrt(discriminant)) / (2 \* a)

x2 = (-b - math.sqrt(discriminant)) / (2 \* a)

return x1, x2

elif discriminant == 0:

# One real solution

x = -b / (2 \* a)

return x,

else:

# No real solutions

return "No real solutions"

# Example usage

a = float(input("Enter coefficient a: "))

b = float(input("Enter coefficient b: "))

c = float(input("Enter coefficient c: "))

solutions = parse\_quadratic\_equation(a, b, c)

print(f"Solutions of the quadratic equation {a}x^2 + {b}x + {c} = 0 are: {solutions}")

## Example Output:

Enter coefficient a: 1

Enter coefficient b: -3

Enter coefficient c: 2

Solutions of the quadratic equation 1x^2 + -3x + 2 = 0 are: (2.0, 1.0)