# Function to add two numbers

def add(x, y):

return x + y

# Function to subtract two numbers

def subtract(x, y):

return x - y

# Function to multiply two numbers

def multiply(x, y):

return x \* y

# Function to divide two numbers

def divide(x, y):

if y == 0:

raise ValueError("Division by zero is not allowed")

return x / y

# Function to calculate square root of a number

def square\_root(x):

import math

if x < 0:

raise ValueError("Square root of a negative number is not defined")

return math.sqrt(x)

# Function to calculate x to the power of y

def power(x, y):

return x \*\* y

# Function to clear memory (bonus feature)

def clear\_memory():

global memory

memory = 0

# Main function to operate the calculator

def main():

print("Welcome to Simple Calculator!")

while True:

try:

# Menu for operations

print("\nOperations:")

print("1. Addition")

print("2. Subtraction")

print("3. Multiplication")

print("4. Division")

print("5. Square Root")

print("6. Exponentiation")

print("7. Clear Memory")

print("0. Exit")

choice = int(input("\nEnter choice (0-7): "))

if choice == 0:

print("Exiting the calculator. Goodbye!")

break

elif choice == 7:

clear\_memory()

print("Memory cleared.")

continue

if choice not in [5, 7]: # For operations needing two numbers

num1 = float(input("Enter first number: "))

num2 = float(input("Enter second number: "))

if choice == 1:

print(f"Result: {add(num1, num2)}")

elif choice == 2:

print(f"Result: {subtract(num1, num2)}")

elif choice == 3:

print(f"Result: {multiply(num1, num2)}")

elif choice == 4:

try:

print(f"Result: {divide(num1, num2)}")

except ValueError as e:

print(f"Error: {e}")

elif choice == 5:

try:

print(f"Result: {square\_root(num1)}")

except ValueError as e:

print(f"Error: {e}")

elif choice == 6:

print(f"Result: {power(num1, num2)}")

else:

print("Invalid choice. Please enter a number between 0 and 7.")

except ValueError:

print("Invalid input. Please enter a valid number.")

except Exception as e:

print(f"Error: {e}")

if \_\_name\_\_ == "\_\_main\_\_":

main()