**10211CS213- Python Programming**

**Task 1 : Python program to execute various expressions**

**1.1** Arithmetic, relational, logical, and assignment expressions.

**Aim:**

To write a Python program that executes and displays the result of various types of expressions such as arithmetic, relational, logical, and assignment expressions.

**Algorithm:**

1. Start
2. Accept input values for two variables (e.g., a and b) from the user.
3. Perform and display **Arithmetic Expressions**:
   * Addition, Subtraction, Multiplication, Division, Modulus, Floor Division, Exponentiation
4. Perform and display **Relational Expressions**:
   * Greater than, Less than, Equal to, Not equal to, Greater than or equal to, Less than or equal to
5. Perform and display **Logical Expressions**:
   * and, or, not
6. Perform and display **Assignment Expressions**:
   * =, +=, -=, \*=, /=, etc.
7. Print results of all evaluated expressions.
8. End

**Program:**

# Get input values

a = int(input("Enter value for a: "))

b = int(input("Enter value for b: "))

# Arithmetic Expressions

print("Arithmetic Expressions:")

print("a + b =", a + b)

print("a - b =", a - b)

print("a \* b =", a \* b)

print("a / b =", a / b)

print("a % b =", a % b)

print("a // b =", a // b)

print("a \*\* b =", a \*\* b)

# Relational Expressions

print("\nRelational Expressions:")

print("a > b:", a > b)

print("a < b:", a < b)

print("a == b:", a == b)

print("a != b:", a != b)

print("a >= b:", a >= b)

print("a <= b:", a <= b)

# Logical Expressions

print("\nLogical Expressions:")

print("a > 0 and b > 0:", a > 0 and b > 0)

print("a > 0 or b > 0:", a > 0 or b > 0)

print("not(a > 0):", not(a > 0))

# Assignment Expressions

print("\nAssignment Expressions:")

x = a

print("Initial x =", x)

x += b

print("x += b:", x)

x -= b

print("x -= b:", x)

x \*= b

print("x \*= b:", x)

x /= b

print("x /= b:", x)

**Sample Input/Output:**

Enter value for a: 10

Enter value for b: 5

Arithmetic Expressions:

a + b = 15

a - b = 5

a \* b = 50

a / b = 2.0

a % b = 0

a // b = 2

a \*\* b = 100000

Relational Expressions:

a > b: True

a < b: False

a == b: False

a != b: True

a >= b: True

a <= b: False

Logical Expressions:

a > 0 and b > 0: True

a > 0 or b > 0: True

not(a > 0): False

Assignment Expressions:

Initial x = 10

x += b: 15

x -= b: 10

x \*= b: 50

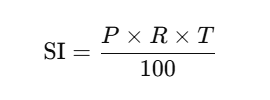
x /= b: 10.0

**1.2. Simple Interest Calculator**

**AIM:** To write a Python program that calculates the Simple Interest for a customer based on the principal amount, rate of interest, and time period.

**Algorithm:**

1. Start
2. Input the principal amount (P) from the user.
3. Input the rate of interest (R) from the user (per annum).
4. Input the time period (T) in years from the user.
5. Calculate Simple Interest (SI) using the formula:



1. Display the Simple Interest.
2. End

**Program**

#Input from the user

P = float(input("Enter the principal amount (Rs.): "))

R = float(input("Enter the annual rate of interest (%): "))

T = float(input("Enter the time period (in years): "))

# Calculate Simple Interest

SI = (P \* R \* T) / 100

# Display the result

print(f"\nSimple Interest = Rs. {SI:.2f}")

**Sample Input/Output**

Enter the principal amount (Rs.): 10000

Enter the annual rate of interest (%): 5

Enter the time period (in years): 3

Simple Interest = Rs. 1500.00

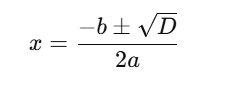
**1.3. Quadratic Equation**

**Aim:**

To write a Python program to find the roots of a **quadratic equation** using the quadratic formula. The coefficients **a**, **b**, and **c** are entered by the user.

**Algorithm:**

1. Start
2. Input the coefficients a, b, and c from the user.
3. Calculate the discriminant using:  
   D = b² - 4ac
4. Check the nature of the roots:
   * If D > 0, roots are **real and distinct**
   * If D == 0, roots are **real and equal**
   * If D < 0, roots are **complex**
5. Use the **quadratic formula** to compute roots:

​​

1. Display the roots accordingly.
2. End

Program

import cmath # To handle both real and complex roots

# Input coefficients

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

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

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

# Calculate discriminant

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

# Compute roots using quadratic formula

root1 = (-b + cmath.sqrt(D)) / (2 \* a)

root2 = (-b - cmath.sqrt(D)) / (2 \* a)

# Display results

print(f"\nDiscriminant (D) = {D}")

# Check nature of roots

if D > 0:

print("The roots are real and distinct.")

elif D == 0:

print("The roots are real and equal.")

else:

print("The roots are complex.")

# Display the roots

print(f"Root 1 = {root1}")

print(f"Root 2 = {root2}")

**Sample Input/Output**

a. Enter coefficient a: 1

Enter coefficient b: -4

Enter coefficient c: 4

Discriminant (D) = 0.0

The roots are real and equal.

Root 1 = (2+0j)

Root 2 = (2+0j)

b. Enter coefficient a: 1

Enter coefficient b: -5

Enter coefficient c: 6

Discriminant (D) = 1.0

The roots are real and distinct.

Root 1 = (3+0j)

Root 2 = (2+0j)

c. Enter coefficient a: 1

Enter coefficient b: 2

Enter coefficient c: 5

Discriminant (D) = -16.0

The roots are complex.

Root 1 = (-1+2j)

Root 2 = (-1-2j)

**1.4. Gain percentage**

Alfred buys an old scooter for Rs. X and spends Rs. Y on its repairs. If he sells the scooter for Rs. Z (Z>X+Y). Write a program to help Alfred to find his gain percent. Get all the above-mentioned values through the keyboard and find the gain percent.

Input Format:

The first line contains the Rs X

The second line contains Rs Y

The third line contains Rs Z

Sample Input:

10000

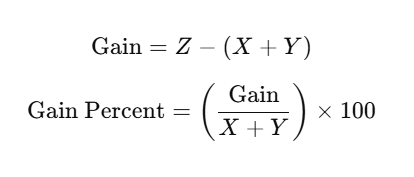
250

15000

Sample Output:

46.34 is the gain percent.

Formula



**Aim:**

To write a Python program to calculate the gain percentage Alfred earns after buying and repairing a scooter and then selling it. The values for purchase cost, repair cost, and selling price are entered by the user.

**Algorithm:**

1. Start
2. Take input X – the cost price of the old scooter.
3. Take input Y – the amount spent on repairs.
4. Take input Z – the selling price of the scooter.
5. Compute the **total cost price** using:  
   total\_cost = X + Y
6. Compute the **gain** using:  
   gain = Z - total\_cost
7. Compute the **gain percentage** using:  
   gain\_percent = (gain / total\_cost) \* 100
8. Print the **gain percentage** rounded to 2 decimal places.
9. End

Program

X = float(input("Enter the cost price of the scooter (Rs.): "))

Y = float(input("Enter the repair cost (Rs.): "))

Z = float(input("Enter the selling price (Rs.): "))

# Calculate total cost price

cost\_price = X + Y

# Ensure selling price is greater than cost price

if Z <= cost\_price:

print("No gain. Selling price must be greater than total cost.")

else:

gain = Z - cost\_price

gain\_percent = (gain / cost\_price) \* 100

# Display results

print(f"Total Cost Price = Rs. {cost\_price}")

print(f"Gain = Rs. {gain}")

print(f"Gain Percent = {gain\_percent:.2f}%")

**Sample Input/Output**

Enter the cost price of the scooter (Rs.): 50000

Enter the repair cost (Rs.): 3000

Enter the selling price (Rs.): 58000

Total Cost Price = Rs. 53000.0

Gain = Rs. 5000.0

Gain Percent = 9.43%

**1.5. Branch wise system count**

In a Lab 36% are Dell and 34% Lennovo and 28% are Acer and 2% are Samsung. Write a python code to print total systems and brand wise count in the specific format using sep operator.

**Aim:**

To write a Python program that calculates and prints the total number of systems in a lab and the count of systems for each brand (Dell, Lenovo, Acer, Samsung) based on percentage data using the sep operator for formatted output.

**Algorithm:**

1. **Start**
2. Input the **total number of systems** in the lab (e.g., total = 100)
3. Define the **percentage** of each brand:
   * Dell: 36%
   * Lenovo: 34%
   * Acer: 28%
   * Samsung: 2%
4. Calculate the **brand-wise count** using the formula:  
   brand\_count = (percentage / 100) \* total
5. Use the print() function and the sep operator to display results in a readable format.
6. **End**

Program

total\_systems = int(input("Enter the total number of systems in the lab: "))

# Percent distribution

dell\_percent = 36

lenovo\_percent = 34

acer\_percent = 28

samsung\_percent = 2

# Count per brand using integer rounding

dell\_count = total\_systems \* dell\_percent // 100

lenovo\_count = total\_systems \* lenovo\_percent // 100

acer\_count = total\_systems \* acer\_percent // 100

samsung\_count = total\_systems \* samsung\_percent // 100

# Print results using sep

print("Total Systems:", total\_systems)

print("Dell:", dell\_count, sep="\t")

print("Lenovo:", lenovo\_count, sep="\t")

print("Acer:", acer\_count, sep="\t")

print("Samsung:", samsung\_count, sep="\t")

or

print("Total Systems:", total\_systems)

print("Dell:", dell\_count, "\nLenovo:", lenovo\_count, "\nAcer:", acer\_count, "\nSamsung:", samsung\_count, sep="\t")

**Sample input/output**

Enter the total number of systems in the lab: 150

Total Systems: 150

Dell: 54

Lenovo: 51

Acer:42

Samsung: 3