PROGRAM:-

```
# Get input values
a = int(input("Enter the value for a :"))
b = int(input("Enter the 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)
# Relational Expressions
Print("In Relational 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("In Logical 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("In Assignment Expressions:")
```

30/07/25

Ex. No: 1

1.1 PYTHON PROGRAMS TO EXECUTE VARIOUS 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:-

Step 1:- Start

Step 2:- Accept input values for two variables (eg., a and b) from the user.

Step 3:- Perform and display Arithmetic Expressions:-

- Addition, Subtraction, Multiplication, Divisions, Modulus, Floor Division, Exponentiation

Step 4:- Perform and display Relational Expressions:

- Greater than, Lesser than, Equal to, Not equal to, Greater than or equal to, Less than or equal to.

Step 5:- Perform and display Logical Expressions:

- and, or, not

Step 6:- Perform and display Assignment Expressions:

- =, +=, -=, *=, /=, etc.

Step 7:- Print results of all evaluate expressions.

Step 8:- End.

```
print ("\n Assignment 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
```

VEL TECH	
EX No.	
PERFORMANCE (5)	
RESULT AND ANAL	
VIVA VOCE (5)	
RECORD (5)	
TOTAL (20)	
SIGN WITH DATE	

RESULT :-

Thus a python program that executes and displays the result of various types of expressions such as arithmetic, relational, logical & assignment expressions is successfully executed.

PROGRAMS :-

```
# 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"\n Simple Interest = Rs. {SI:.2f}")
```

1.2 SIMPLE INTEREST CALCULATOR

AIM:-

To write a Python Program that calculate the Simple Interest for a customer based on the Principal amount, rate of interest, and time Period.

ALGORITHM :-

Step 1 :- Start

Step 2 :- Input the principal amount (P) from the user.

Step 3 :- Input the rate of interest (R) from the user (per annum)

Step 4 :- Input the time period (T) in years from the user.

Step 5 :- Calculate Simple Interest (SI) using the formula :-

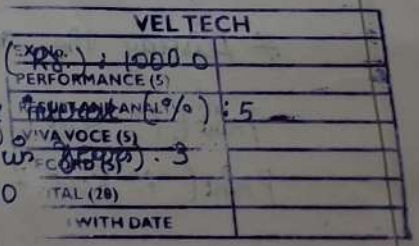
$$SI = P \times R \times T / 100$$

Step 6 :- Display the Simple Interest

Step 7 :- End.

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
```



RESULT :-

Thus a Python Program to calculate Simple Interest is compiled and executed successfully.

PROGRAM :-

```
import cmath
# To handle both real and complex roots
# Input coefficients
a = float( input (" Enter coefficients 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"\n Discriminant (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 = {root 1}")
Print( f" Root 2 = {root 2}")
```

30/7/25

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 :-

Step 1 :- Start
Step 2 :- Input the coefficients a, b, and c from the user.
Step 3 :- Calculate the discrimination using :

$$D = b^2 - 4ac$$

Step 4 :- Check the nature of the roots :

- If $D > 0$, roots are real & distinct
- If $D == 0$, roots are real & equal
- If $D < 0$, roots are complex

Step 5 :- Use the quadratic formula to compute roots :

$$x = -b \pm \sqrt{D} / 2a$$

Step 6 :- Display the roots accordingly
Step 7 :- End

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
Discriminate (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)

RESULT :-

Thus a Python program to find the roots of a quadratic equation using the quadratic formula is compiled and executed succesfully.

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}%")
```

30/7/25

1.4 GAIN PERCENTAGE

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 is entered by the user.

ALGORITHM:-

Step 1:- Start
Step 2:- Take input X - the cost price of the old scooter.
Step 3:- Take input Y - the amount spent on repairs
Step 4:- Take input Z - the selling price of the scooter
Step 5:- Compute the total cost price using:

total_cost = X + Y

Step 6:- Compute the total cost price using

gain = Z - total_cost

Step 7:- Compute the gain percentage using:

gain_Percent = (gain / total_cost) * 100

Step 8:- Print the gain percentage rounded to 2 decimal places.

Step 9:- End.

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%.
```

RESULT :-

Thus a Python Program to Calculate a gain percentage is compiled and executed succesfully.

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")
```

30/7/25

1.5 BRANCH WISE SYSTEM COUNT

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 :-

Step 1:- Start

Step 2:- Input the total number of systems in the lab. (eg. total = 100)

Step 3:- Define the percentage of each brand:

- Dell : 36 %.
- Lenovo : 34 %.
- Acer : 28 %.
- Samsung : 2 %.

Step 4:- Calculate the brand-wise count using the formula :

brand_count = (percentage / 100) * total

Step 5:- Use the print() function and the sep operator to display results in a readable format

Step 6:- End

Sample Input / Output:-

Enter the total number of Systems in the lab: 150
Total systems : 150
Dell : 54
Lenovo : 51
Acer : 42
Samsung : 3

VEL TECH	
X No.	1
PERFORMANCE (5)	5
RESULT AND ANALY	5
VIVA VOCE (5)	5
ECORD (5)	5
TAL (20)	20
N WITH DATE	

30/7

RESULT:-

Thus a Python Program that calculates and Print the total number of systems in the lab & brandwise count of the system is compiled and executed successfully.