S.P INSTITUTE OF COMPUTRONICS
SULTANPUR
विद्या ददाति विनयम्

SP Group of Institute

UNIT 2 : OPERATORS - Arithmetic & Relational

Operators are special symbols that perform operations on variables and values.

1. Arithmetic Operators

These operators are used to perform mathematical operations.

Operator	Symbol	Concept	Code Example	Output
Addition	+	Adds two values	a = 10; b = 5; c = a + b;	c = 15
Subtraction	-	Subtracts right value from left value	a = 10; b = 5; c = a - b;	c = 5
Multiplication	*	Multiplies two values	a = 10; b = 5; c = a * b;	c = 50
Division	/	Divides left value by right value	a = 10; b = 5; c = a / b;	c = 2
Modulus	%	Returns remainder of division	a = 10; b = 3; c = a % b;	c = 1

2. Relational Operators

These operators are used to compare two values.

Operator	Symbol	Concept	Code Example	Output
Equal to	==	Checks if two values are equal	a = 5; b = 5; c = (a == b);	c = true
Not equal to	!=	Checks if two values are not equal	a = 5; b = 3; c = (a != b);	c = true
Greater than	>	Checks if left value is greater	a = 7; b = 3; c = (a > b);	c = true
Less than	<	Checks if left value is smaller	a = 2; b = 9; c = (a < b);	c = true
Greater than or equal to	>=	Checks if left value is greater or equal	a = 5; b = 5; c = (a >= b);	c = true
Less than or equal to	<=	Checks if left value is less or equal	a = 4; b = 7; c = (a <= b);	c = true

Note : Relational operators return boolean values: true (1) or false (0).

★ Key Takeaway:
Arithmetic operators perform calculations, while relational operators perform comparisons.

SP Group of Institute

UNIT 2 : OPERATORS – Logical & Assignment

Operators are special symbols that perform operations on variables and values.

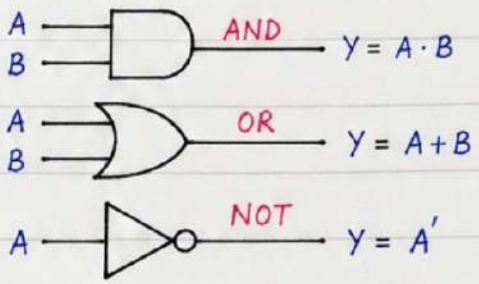
* LOGICAL OPERATORS

Logical operators are used to combine two or more conditions. They return either True (1) or False (0).

Truth Table

A	B	A AND B	A OR B	NOT A
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Logic Gate Representation



Note: In C/C++, 0 is False and non-zero value is True.

* ASSIGNMENT OPERATORS

Assignment operators are used to assign values to variables. The left operand must be a variable.

Common Assignment Operators

Operator	Example	Equivalent To
=	a = b	a = b
+=	a += b	a = a + b
-=	a -= b	a = a - b
*=	a *= b	a = a * b
/=	a /= b	a = a / b
%=	a %= b	a = a % b

Code Examples

```
// Example 1
int a = 10, b = 3;
a += b;      // a = a + b
cout << a;   // Output: 13
```

```
// Example 2
int x = 20;
x -= 5;      // x = x - 5
cout << x;   // Output: 15
```

Remember:
- Use && for AND, || for OR, ! for NOT in C/C++.
- Assignment operators make code short and easy.

Example (Logical Operators in C++)

```
int p = 5, q = 10;
if (p < q && q != 0)
    cout << "Both conditions are True";
else
    cout << "At least one condition is False";
```

→

Output:
Both conditions are True

SP Group of Institute

UNIT 2 : OPERATORS – Bitwise & Membership

✳ BITWISE OPERATORS

Bitwise operators work on individual bits of integers.
They treat numbers in binary form.

Operator	Name	Symbol	Example (a = 5, b = 3)	Binary	Result
AND	Bitwise AND	&	5 & 3 = 1	0101 0011	0001 (1)
OR	Bitwise OR	|	5 | 3 = 7	0101 0011	0111 (7)
XOR	Bitwise XOR	^	5 ^ 3 = 6	0101 0011	0110 (6)
NOT	Bitwise NOT	~	~5 = -6	0101 →	...1010 (-6)
Left Shift	Shift Left	<<	5 << 1 = 10	0101 1010	1010 (10)
Right Shift	Shift Right	>>	5 >> 1 = 2	0101 0010	0010 (2)

Interview Question :

Q. What is the result of 12 & 5 ?

12 = 1100 , 5 = 0101

```
  1100
& 0101
------
  0100 = 4
```

Answer : 4

✳ MEMBERSHIP OPERATORS

Membership operators are used to test whether a value or variable is found in a sequence (string, list, tuple, set, etc.).

Operators :

- in → True if present
- not in → True if not present

Note :

The 'in' and 'not in' operators improve code readability and are widely used in loops and conditional statements.

Examples :

1.
```
x = 'Computronics'
print ('Comp' in x)        # Substring check
print ('tron' not in x)    # Not present
```
Output : True, False

2.
```
lst = [10, 20, 30, 40]
print (20 in lst)          # Element check
print (50 not in lst)      # Not present
```
Output : True, True

3.
```
s = {1, 2, 3, 4}
print (3 in s)
print (5 not in s)
```
Output : True, True

✳ **Summary :** Bitwise operators work at the bit level, while membership operators check existence of elements in a sequence. ☺

SP Group of Institute

UNIT 2: OPERATORS - Identity & Precedence

This unit explains Identity Operators and Operator Precedence in C programming with examples.

1. IDENTITY OPERATORS

It is used to compare the memory location of two operands.

Operators:

== (is equal to)

!= (is not equal to)

Example:

```
int a = 10, b = 10;
int *p = &a, *q = &b;
if (p == q)
      ↳ Same address
else
      ↳ Different address
```

Common Mistakes:

- Confusing == (equality) with = (assignment).
- Comparing values instead of addresses.

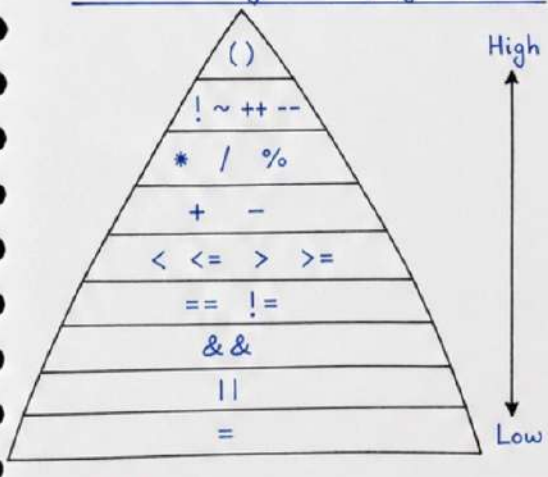
Example:

```
if (a == b)     // compares values
if (&a == &b)   // compares addresses
```

2. PRECEDENCE & ASSOCIATIVITY

Operators are evaluated according to their precedence.
Higher precedence operators are evaluated first.

Precedence Pyramid (High → Low)



Example 1:

```
int a = 10, b = 5, c = 2;
int result = a + b * c;
printf("%d", result);
```

Output: 20

Explanation:

First b * c = 5 * 2 = 10

Then a + 10 = 10 + 10 = 20

Example 2:

```
int a = 10, b = 5, c = 2;
int result = (a + b) * c;
printf("%d", result);
```

Output: 30

Explanation:

First (a+b) = 10 + 5 = 15

Then 15 * c = 15 * 2 = 30

Notes:

- Precedence decides which operator is evaluated first.
- Operators in same precedence group are evaluated according to associativity.
- Most binary operators are left to right associative.
- Assignment operator (=) is right to left associative.

Practice Makes Perfect! ☆

SP Group of Institute

UNIT 3: CONDITIONAL STATEMENTS - if & if-else

Conditional Statements are used to make decisions in a program based on certain conditions.

1. if STATEMENT

→ **Concept:**

The if statement is used to execute a block of code only if a given condition is True.

→ **Syntax:**

```
if (condition):
    # block of code
```

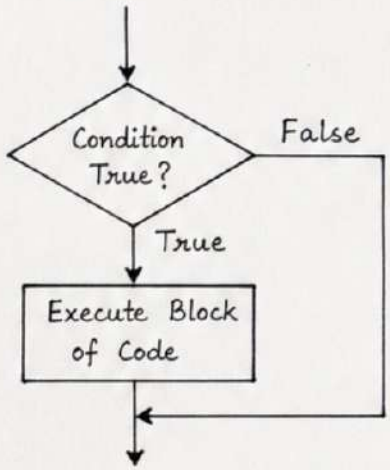
→ **Example:**

```
x = 10
if x > 5:
    print ("x is greater than 5")
```

→ **Output:**

```
x is greater than 5
```

Flowchart:



2. if-else STATEMENT

→ **Concept:**

The if-else statement is used to execute one block of code if the condition is True and another block if the condition is False.

→ **Syntax:**

```
if (condition):
    # block of code if condition is True
else:
    # block of code if condition is False
```

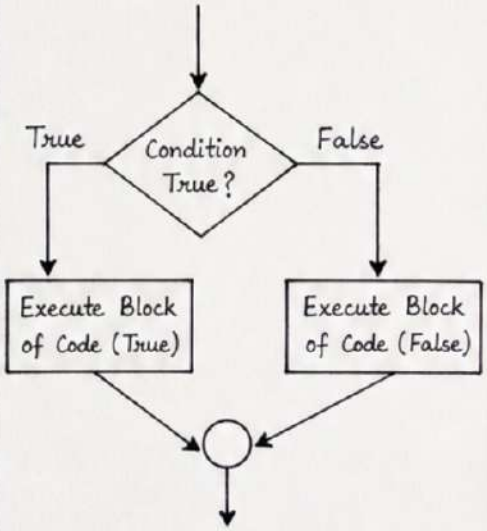
→ **Example:**

```
x = 3
if x >= 5:
    print ("x is greater than or equal to 5")
else:
    print ("x is less than 5")
```

→ **Output:**

```
x is less than 5
```

Flowchart:



* **Note:** Use proper indentation in Python. It is very important.

S.P INSTITUTE OF COMPUTRONICS SULTANPUR

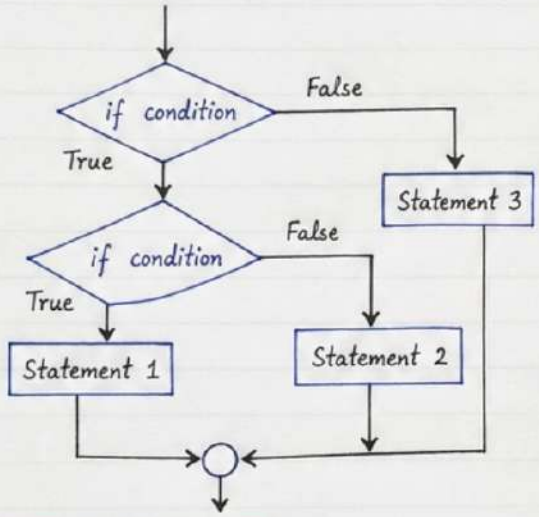
SP Group of Institute

UNIT 3 : CONDITIONAL STATEMENTS - Nested if & elif

1. NESTED IF

When an if or else block contains another if or else block, it is called Nested if.

FLOWCHART :



COMMON MISTAKES :

- Missing indentation
- Using assignment (=) instead of comparison (==)
- Forgetting the inner block
- Confusing multiple conditions

TIP : Always use proper indentation and test each condition carefully.

2. ELIF LADDER

Used to check multiple conditions. The conditions are checked in order. Once one is True, the rest are skipped.

SYNTAX :

```
if condition1 :
    statement 1
elif condition2 :
    statement 2
elif condition3 :
    statement 3
    ......
else :
    statement n
```

EXAMPLE :

```
marks = 85
if marks >= 90 :
    print ("Grade : A+")
elif marks >= 75 :
    print ("Grade : A")
elif marks >= 60 :
    print ("Grade : B")
elif marks >= 40 :
    print ("Grade : C")
else :
    print ("Fail")
```

OUTPUTS :

If marks = 92
Output :
Grade : A+

If marks = 78
Output :
Grade : A

If marks = 55
Output :
Grade : B

If marks = 35
Output :
Fail

KEY POINTS :

- Nested if is used inside another if/else block.
- elif is used for multiple conditions.
- Python checks conditions from top to bottom.
- Use else for the default case.

☆ Practice more examples to become confident !

CODE. PRACTICE. DEBUG. SUCCEED. ☺

S.P INSTITUTE OF COMPUTRONICS SULTANPUR
विद्या ददाति विनयम्
SP Group of Institute

UNIT 3 : CONDITIONAL STATEMENTS – match-case & Practice

1. match-case Statement (Python 3.10+)

The match-case statement is used to compare a value against multiple patterns. It is more readable and efficient than multiple if-elif-else statements.

Syntax:

```
match expression:
    case value1:
        # code block
    case value2:
        # code block
    ...
    case _:
        # default code block
```

Example:

```
x = int(input("Enter a number (1-7): "))
match x:
    case 1:
        print("Monday")
    case 2:
        print("Tuesday")
    case 3:
        print("Wednesday")
    case 4:
        print("Thursday")
    case 5:
        print("Friday")
    case 6:
        print("Saturday")
    case 7:
        print("Sunday")
    case _:
        print("Invalid input! Please enter 1-7.")
```

Output (Example):

```
Enter a number (1-7): 3
Wednesday
```

Interview Question:

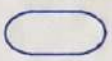
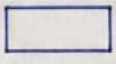
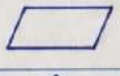
Q. What is the difference between if-elif-else ladder and match-case statement?

Ans: match-case is more readable, supports pattern matching, and can handle multiple data types.

2. Practice Programs

1. Write a program to find the day of the week using match-case.
2. Write a program to perform basic calculator operations using match-case.
3. Write a program to grade a student based on marks (A, B, C, D, F) using match-case.
4. Write a program to check the type of shape (circle, square, triangle, rectangle) and display its name using match-case.

3. Flowcharts – Common Symbols

Symbol	Name	Use
	Terminator	Start / End of a program
	Process	Instruction or operation
	Input / Output	Read input or Display output
	Decision	Condition check (Yes / No)
	Flow Line	Direction of flow

* **Note:** The case _ (underscore) acts as a default case, similar to else in if-else.

* **Tip:** Use meaningful variable names and proper indentation for better readability. ☺