Conditional Statements

Introduction to Conditional Statements

- Statements in Python programs are executed sequentially in the order in which they are written ,this is called sequential execution.
- This involves a kind of decision making to see whether a particular condition has occurred or not and then direct the computer to execute certain statements accordingly .
- We can achieve these through the use of control or decision making statements.
- These statements alter the flow of execution of programs and the control statements control our program's flow of execution, as such they form the backbone of our programs.
- The statement which transfers control from one statement to another statement breaking normal sequence of the execution is knows as control structures or statements.

Introduction to Conditional Statements

- Decision making is required when we want to execute a code only if a certain condition is satisfied.
- Python provides three types of control structures. They are
  - Conditional control
  - Loop or Iteration control

Conditional Control Structures

- The conditional control statements are mainly used for decision making purpose and required when we want to execute a code only if a certain condition is satisfied.
- Python provides the following constructs to make decision or condition.
  - **if** Statement
  - **if – else** Statement
  - Nested **if – else** Statements
  - The **if – else if – else** or **if - elif - else** Statements

if Statement

- The **if** statement is called a single –selection structure because it selects or ignores a single action.
- if is also python keyword and as its name implies ,is used to make decisions .
- The if statement takes a logical condition and evaluates it, Based on whether the condition evaluates to true or false
- The simple form of the **if** statements is

  if expression or condition: or **if** expression or condition: statement

  statement

- The expression may be any valid python expression and it must be enclosed with colon(:).
- **if** the expression evaluates as true, the statement will be executed , **if** the expression evaluates not true, then the statement is not executed.
- instead of that the statement is bypassed and next statement order is executed.

if Statement

- In Python, the body of the if statement is indicated by the indentation. Body starts with an indentation and the first unindented line marks the end.
- Python interprets non-zero values as True. None and 0 are interpreted as False.
  - Example : **if** marks<40:

    print("failed")

Multiple Statements within **if**

- If the programmer want to more than one statement to execute following if statement.

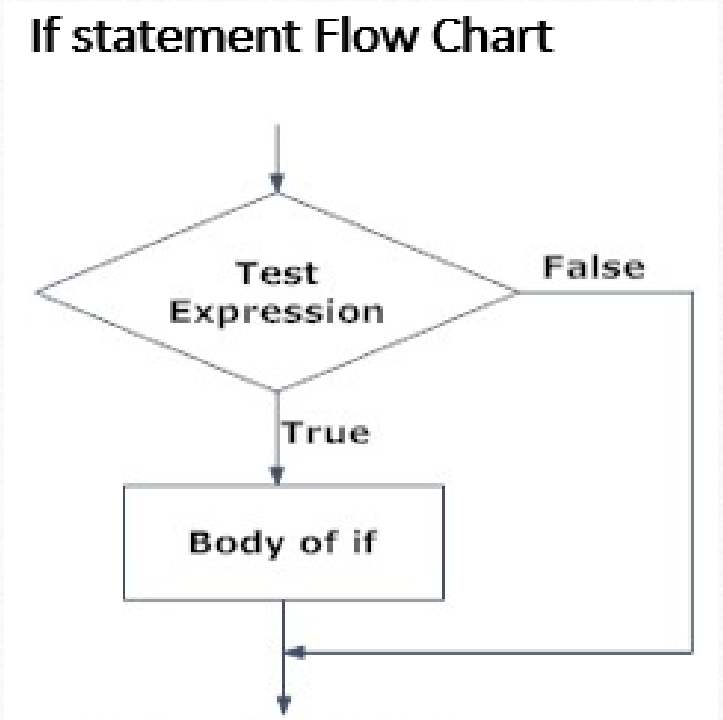
  General form : **if** condition :

  statement 1

  statement 2

  …….

  statement n



If statement Examples

Example of If statement

```
# If the number is positive, we print an appropriate message

num = 3
if num > 0:
    print(num, "is a positive number.")
print("This is always printed.")

num = -1
if num > 0:
    print(num, "is a positive number.")
print("This is also always printed.")
```

- Write a program to determine whether an integer of two numbers entered through the keyboard is swapped without using third number or variable
- Write a program to determine whether an integer number entered through the keyboard is even or odd.
- Write a program to find the largest of three numbers.
- Write a program to find the total marks ,average and percentage of the student per his/her semester 1 and declare pass or fail.

If – else statement

- The **if** statement lets you do something **if** condition is true, the statement will be executed , **if** the expression evaluates not true, then the statement is not executed.
- In such cases, we will use the **if – else** statement.
- It consists of **if** statement ,followed by a statement or block of statements ,followed by the keyword **else**, followed by another statement of block of statements.
- The simple form of the **if** statements is

  if expression or condition:

  (body of if) statement 1

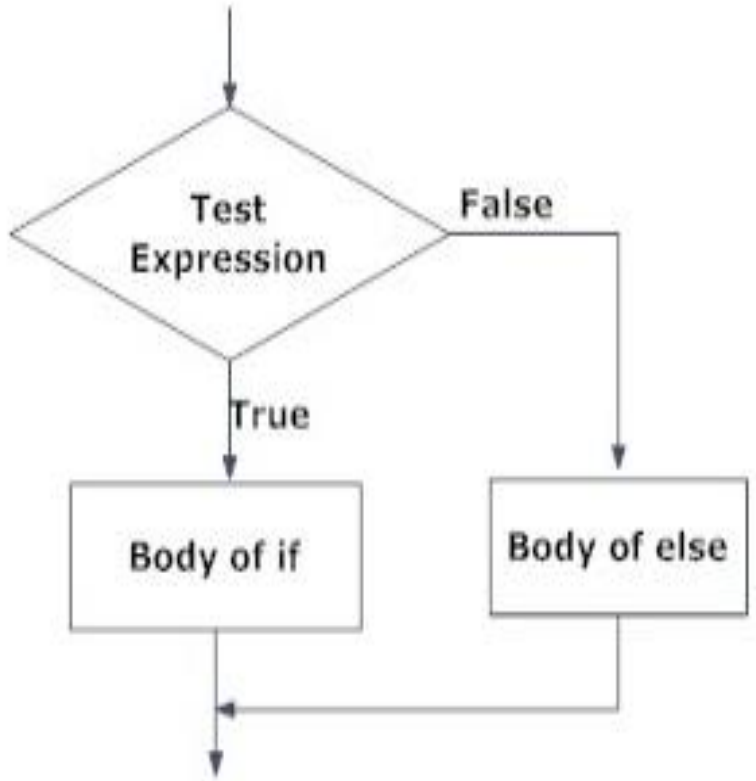
  else:

  (body of if) statement 2

- The **if- else** statement evaluates test expression and will execute body of if only when test condition is True.
- If the condition is False, body of else is executed. Indentation is used to separate the blocks

If – else Example and Flowchart

if - else statement Flowchart



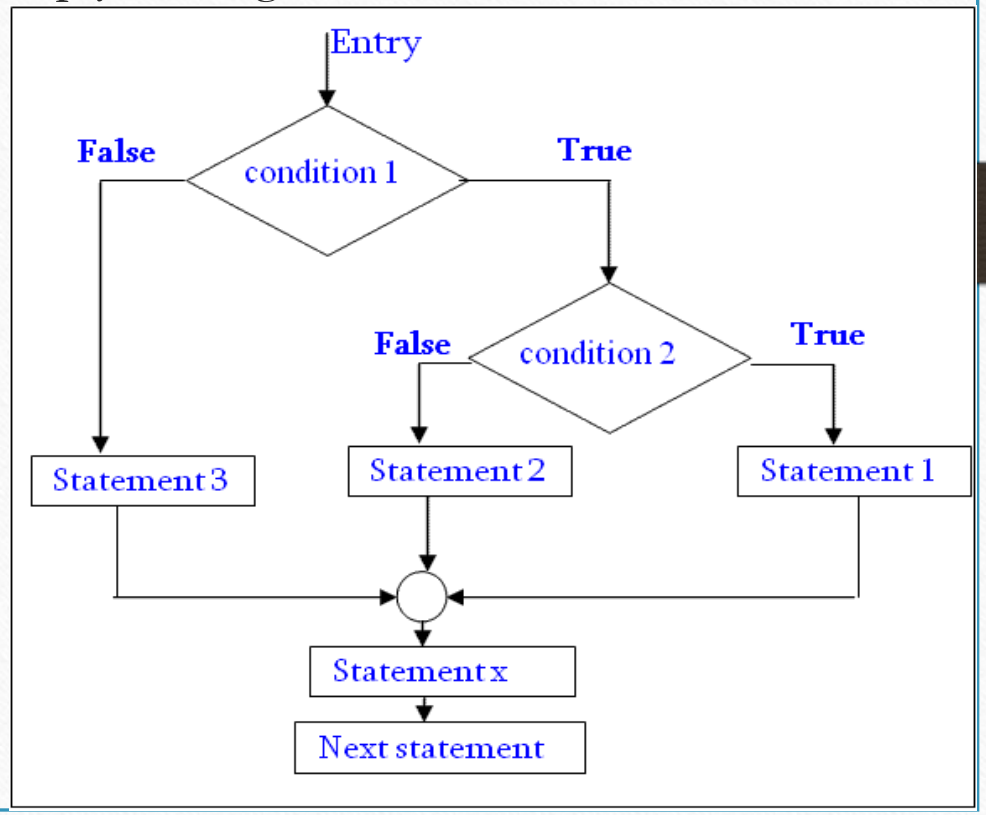
Example of **if - else** statement

```
# Program checks if the number is positive or negative
# And displays an appropriate message

num = 3

# Try these two variations as well.
# num = -5
# num = 0

if num >= 0:
    print("Positive or Zero")
else:
    print("Negative number")
```

Nested if – else statement

- An **if** statement may contain another **if** statement.
- If either the if –block or else –block or both contains another **if** statements or **if-else** statement, then the construct is known as nested conditional statement or simply nesting of **if** statements.
- This is useful when a series of decisions are involved
  - The syntax is as follows

```
if expression 1:
        if expression 2:
                statement – 1
        else:
                statement – 2
else:
        if expression 3:
                statement – 3
        else:
                statement – 4
```



Nested if – else Example

- Find the largest among 3 numbers ?

```
a=int(input("Enter a value : "))
b=int(input("Enter b value : "))
c=int(input("Enter c value : "))
if a>b:
    if a>c:
        print("%d is largest value among 3 numbers"%a)
    else:
        print("%d is largest value among 3 numbers"%c)
else:
    if b>c:
        print("%d is largest value among 3 numbers"%b)
    else:
        print("%d is largest value among 3 numbers"%c)
```

- Write a Python program, we input a number check if the number is positive or negative or zero and display an appropriate message ?

The **if – else – if (elif)-else** statement

- Multi –way decisions arise when there are multiple conditions and different statements are to be executed under each condition.
- The if block can have only one else block. But it can have multiple elif blocks.
- There is another way of putting **if**s together when multiple decisions are involved .
- The syntax and Flowchart of **if – elif -else**

```
if expression 1:
        statement – 1
elif expression 2
        statement – 2
else:
        statement – 3
```

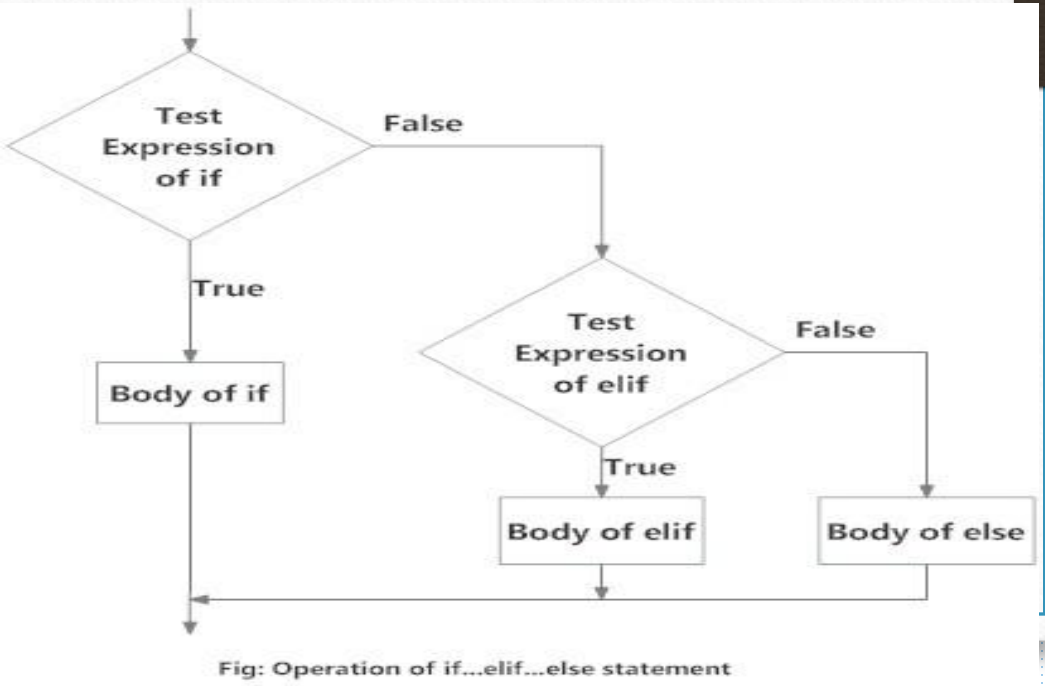


Fig: Operation of if...elif...else statement

Example of **if- elif- else**

```
#Use of if..else..if ladder structure
#To identify the stastus of an individual
age=int(input("Enter the age :"))
if age>=1 and age<=18:
    print("The individual is a minor")
elif age>=19 and age<=40:
    print("The individual is a major")
elif age>41 and age<=100:
    print("The individual is a adult")
else:
    print("Invalid Number")
```

Extension of the if - elif- else Statement(switch)

- We have seen that when one of many alternatives is to be selected, we can design a program using **if** statements to control the selection.
- Designing a program using **if** statements becomes cumbersome(Difficult to handle) when several alternatives needed.
- C has a built – in multiway decision statement known as a **switch.**
- The **switch** statement is in effect an extension of the **if …else** statement and it allows the user to select one of several alternatives.
- The General form of **switch** in C (it consists of several **case** labels)

```
switch( expression ){
  case 1:Statement 1;
            Break
  case  2 :Statement 2;
            Break;
 default :Statement 3;
}
```

Note -: In Python we can create switch case using dictionary or if –elif –else statements

- Python does not have a switch or case statement. To get around this fact, we use dictionary mapping:

```
if n == 0:
    print "You typed zero.\n"
elif n== 1 or n == 9 or n == 4:
    print "n is a perfect square\n"
elif n == 2:
    print "n is an even number\n"
elif  n== 3 or n == 5 or n == 7:
    print "n is a prime number\n"
```

- Python's dictionaries allow a simple one-to-one matching of a key and a value.

```
num=int(input("Enter a digit :"))
digit={0:"Zero",1:"One",2:"Two",3:"Three",
       4:"Four",5:"Five",6:"Six",7:"Seven",
       8:"Eight",9:"Nine"}
print("Entered Digit is ",num,"in Word -",digit[num])
```