Chapter-
Flow of Control

Introduction

- The order of execution of the statements in a program is known as flow of control. The flow of control can be implemented using control structures.
- Python supports two types of control structures—

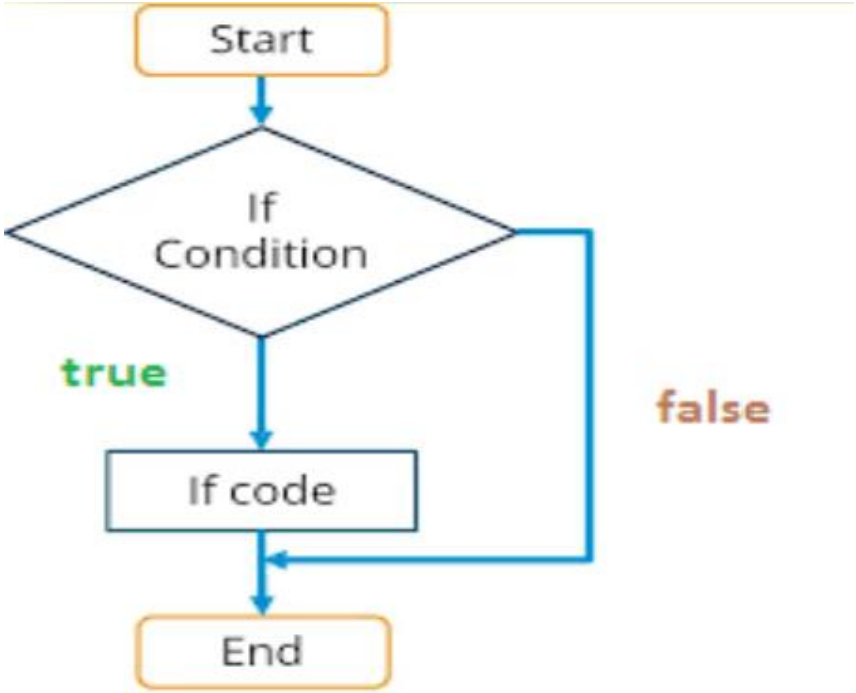
a. selection
b. repetition.

Selection

- It is also know as decision making.
- Decision making statement used to control the flow of execution of program depending upon condition.
- There are three types of decision making statement.
  1. if statements
  2. if-else statements
  3. Nested if-else statement

Selection (contd.)

- If- statements: An if statement is a programming conditional statement that, if proved true, performs a function or displays information.



Selection (contd.)

- **Syntax:**
  if(condition):
  statement
  [statements]

- **e.g.**:

```
noofbooks = 2
if (noofbooks == 2):
 print('You have ')
  print('two books')
  print('outside of if statement')
```

Output: _____

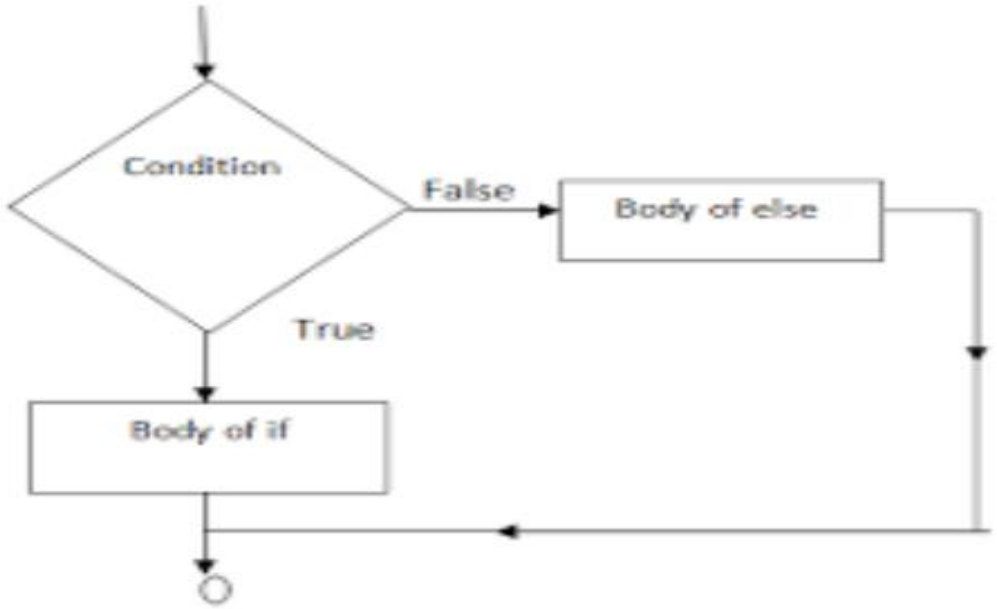
Using logical operator in if statement

```
x=1
y=2
if(x==1 and y==2):
print('condition matching the criteria')
```

Output :- _____

Selection (contd.)

- **if-else Statements**
  If-else statement executes some code if the test expression is true (nonzero) and some other code if the test expression is false.



Selection (contd.)

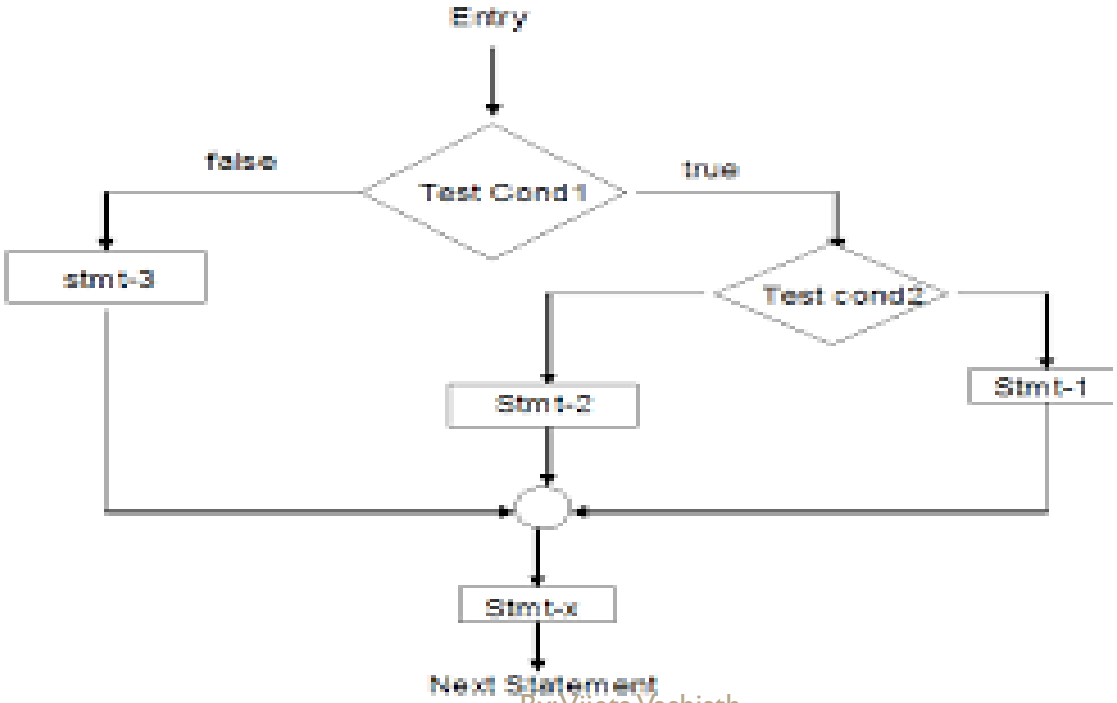
- **Syntax:**
  if(condition):
  statements
  else:
  statements
- **e.g.**

```
a=10
if(a < 100):
        print('less than 100')
else:
        print('more than equal 100')
```

- **OUTPUT:** _____
- Write a program in python to check that entered number is even or odd?

Selection (contd.)

- **Nested if-else statement**
  The nested if...else statement allows you to check for multiple test expressions and execute different codes for more than two conditions.



Selection (contd.)

- **Syntax**
  If (condition):
  statements
  elif (condition):
  statements
  else:
  statements
- **E.g.**
  num = float(input("Enter a number: "))
  if num >= 0:
  if num == 0:
  print("Zero")
  else:
  print("Positive number")
  else:
  print("Negative number")

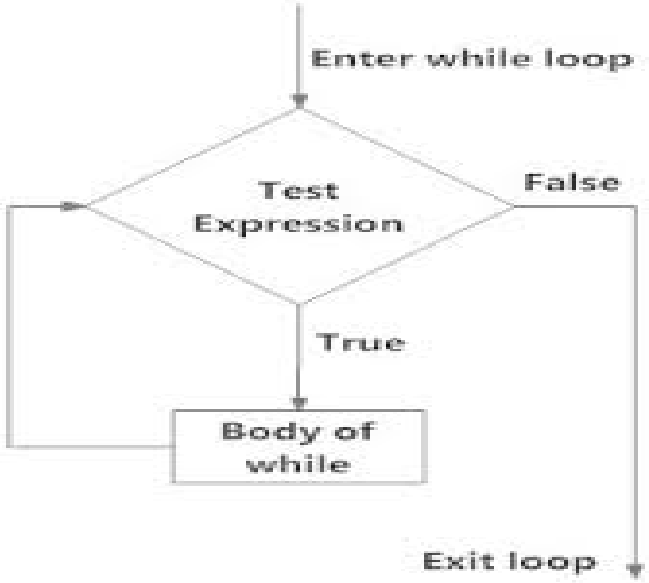
- **OUTPUT:** Enter a number: 5
  Positive number
- **Write python program to find out largest of 3 numbers.**

Repetition

- Also known as loops or iteration statements
- They are used to execute a block of statements as long as the condition is true. Loops statements are used when we need to run same code again and again.
- Python Iteration (Loops) statements are of three type :-
  1. While loop
  2. For loop
  3. Nested For loops

Repetition(contd.)

- **While Loop**
  It is used to execute a block of statement as long as a given condition is true. And when the condition become false, the control will come out of the loop. The condition is checked every time at the beginning
  loop.
  Syntax
  while (condition):
  statement
  [statements]



- Example:
- count = 1
  while count <= 5:
  print(count)
  count += 1
- WAP to print first n natural numbers.

Repetition(contd.)

- **While Loop With Else**

  e.g.

```
x = 1
while (x < 3):
print('inside while loop value of x is ',x)
x = x + 1
else:
print('inside else value of x is ', x)
```

  OUTPUT:

```
inside while loop value of x is 1
inside while loop value of x is 2
inside else value of x is 5
```

- WAP to find the sum of all the positive numbers entered by the user.

Repetition(contd.)

- **For Loop**
  It is used to iterate over items of any sequence, such as a list or a string.
- ***Syntax:***
  for <control-variable> in <sequence/
  items in range>:
  <statements inside body of the
  loop>
- Example: count = [10,20,30,40,50]
  for num in count:
  print(num) output: _____
- for letter in 'PYTHON':
  print(letter) output:_____

Nested Loop

- A loop may contain another loop inside it. A loop inside another loop is called a nested loop.
- Example: Program to print the pattern for a number input by the user
  #The output pattern to be generated is
  #1
  #1 2
  #1 2 3
  #1 2 3 4
  #1 2 3 4 5

```
num = int(input("Enter a number to generate its pattern = "))
for i in range(1,num + 1):
for j in range(1,i + 1):
print(j, end = " ")
print()
```

- WAP to find prime numbers between 2 to 50 using nested for loops?

Range Function

- The range() is a built-in function in Python. Syntax of range() function is:
  range([start], stop[, step])
  It is used to create a list containing a sequence of integers from the given start value upto stop value (excluding stop value), with a difference of the given
  step value.
- Example: #start and step not specified (default value: start=0; step=1 in each iteration)
  >>> list(range(10)-)
  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

- >>> list(range(2, 10))

- >>> list(range(0, 30, 5))
- Program to print the multiples of 10 for numbers in a given range.

Jump Statements

Jump statements are used to transfer the **program's** control from one location to another. Means these are used to alter the flow of a loop like – to skip a part of a loop or terminate a loop.

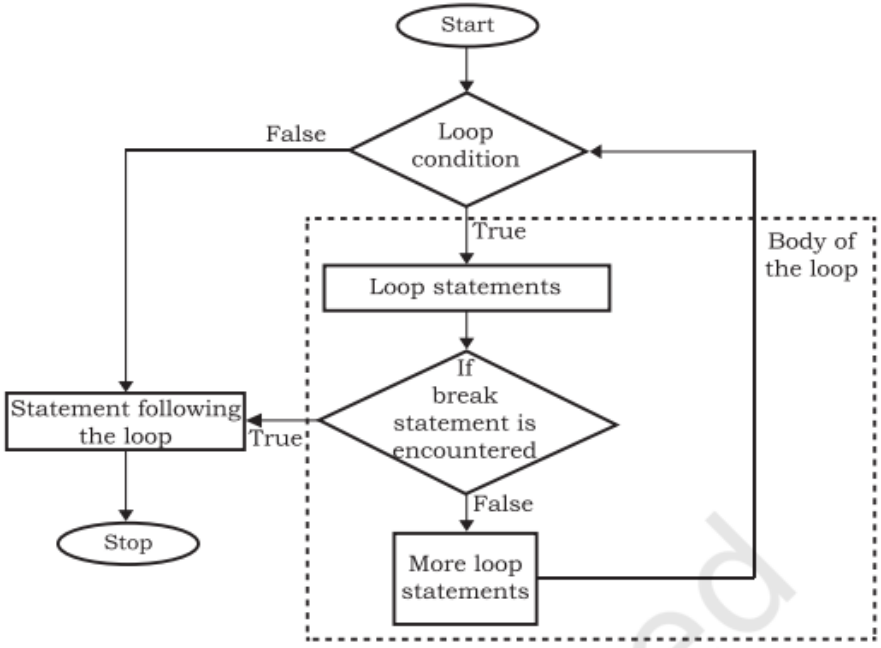
There are three types of jump statements used in
python.

1.break
2.continue
3.pass

Break statement

- **Break**
  The break statement alters the normal flow of execution as it terminates the current loop and resumes execution of the statement following that loop.
  e.g.

```
for val in "string":
if val == "i":
break
print(val)
print("The end")
```

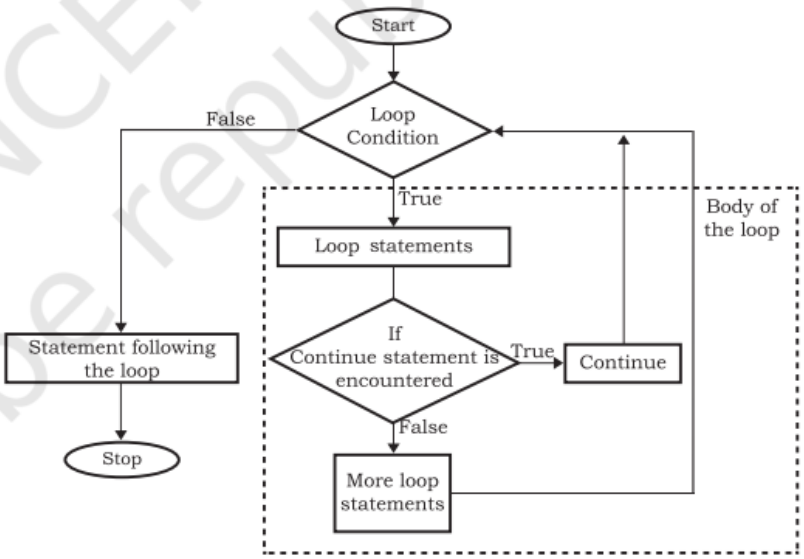


Continue statement

- When a continue statement is encountered, the control skips the execution of remaining statements inside the body of the loop for the current iteration and jumps to the beginning of the loop for the next iteration. If the loop's condition is still true, the loop is entered again, else the control is transferred to the statement immediately following the loop.
- Example: Prints values from 0 to 6 except 3

```
num = 0
for num in range(6):
num = num + 1
if num == 3:
continue
print('Num has value ' +
```

- str(num))
  print('End of loop')



Pass statement

- This statement does nothing. It can be used when a statement is required syntactically but the program requires no action.
- Use in loop:-
  while True:
  pass # Busy-wait for keyboard interrupt (Ctrl+C)
  In function:-
  It makes a controller to pass by without executing any code.
- Example:
  for i in 'initial':
  if(i == 'i'):
  pass
  else:
  print(i)
- NOTE : **continue** forces the loop to start at the next iteration while **pass** means "there is no code to execute here" and will **continue** through the remainder or the loop body.

Question Session