Control Flow System

Day 1

Copyright © 2021, ABES Engineering College

Downloaded from pickywallpapers.com

General Guideline

© (2021) ABES Engineering College.

This document contains valuable confidential and proprietary information of ABESEC. Such confidential and proprietary information includes, amongst others, proprietary intellectual property which can be legally protected and commercialized. Such information is furnished herein for training purposes only. Except with the express prior written permission of ABESEC, this document and the information contained herein may not be published, disclosed, or used for any other purpose.

Objective of Control Flow System

To understand the purpose and syntax of decision control statements.

To understand the purpose and syntax of Iterative/looping statements and jump statements.

To develop python programs with conditions and loops.

To apply the knowledge of conditions, loops and jump in solving real time problems.

To create python programs with decision ,looping and jump statements.

Copyright © 2021, ABES Engineering College

Topics Covered

Day 1	Day 2	Day 3	Day 4
2.1 Introduction **2.2 Selection / Decision control statement** •Simple If statement •If Else statement •If-Elif statement •Nested If Else statement	**2.3 Iterative/ Looping Statements** • While loop • Nested While loop	**2.3 Iterative/ Looping Statements** • Range() Function • For loop • Nested for loop	**2.4 Jump Statements** • Break • Continue **2.5 Else with Loop** • While with else • For with else

Copyright © 2021, ABES Engineering College

Session Plan - Day 1

2.1. Introduction to Decision Control Statements

2.2. Selection / Decision control statement

- Simple If Statements
- If Else Statements
- If ..elif..else statements
- Nested If Else Statements
- Examples
- Review questions
- Summary

Copyright © 2021, ABES Engineering College

Introduction

Control flow of the program is:

The order in which the code of the program executes.

Regulated by:

- Conditional statements.
- Iterative/looping statements.

Copyright © 2021, ABES Engineering College

Contd..

- *Conditional statement examples*: Based on certain conditions, we take decisions in our daily life.

  - Darkness in the room: Turn on lights.
  - No Darkness: Do Nothing

  - Weather is *Cold*: Drink Coffee
  - Weather is *not Cold*: Softdrink

Copyright © 2021, ABES Engineering College

Contd..

- *Iterative Looping statements examples:* For taking decisions we perform various actions and repeat the same action many times.

Real Life Scenario:

- Suppose you want to buy a new t-shirt, visit the shop.
- If you do not found a shirt of your choice, visit to the new shop.
- Until your desired shirt is found, action is performed again and again.
- This is called looping.

Copyright © 2021, ABES Engineering College

Selection/decision control statements

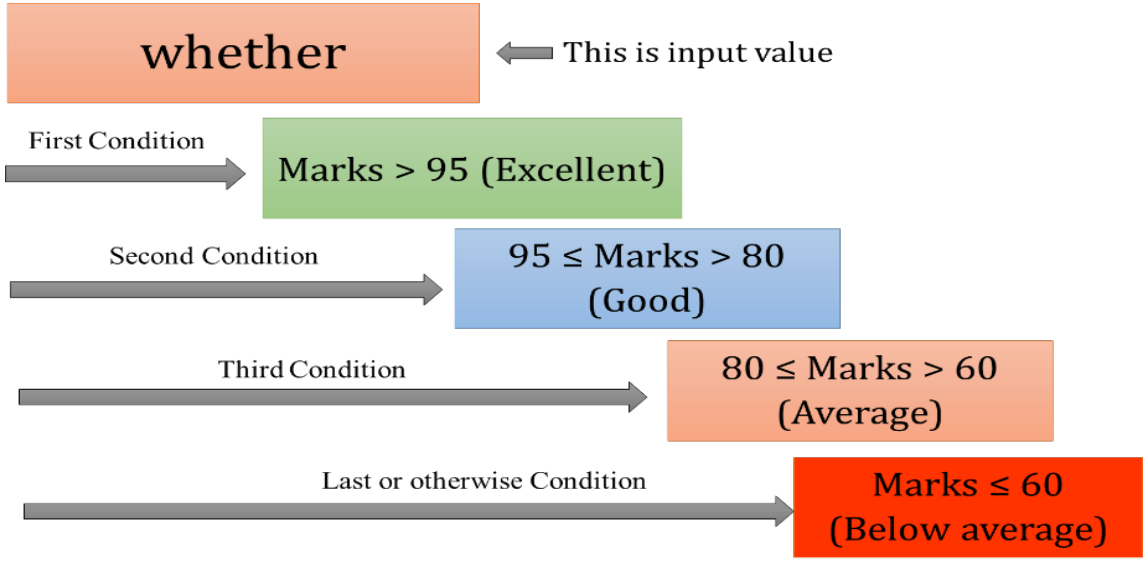
- It is used to control the flow of execution of program depending upon condition:

  - if the condition is true then the code block will execute
  - if the condition is false then code block will not execute.

Real Life Scenario:

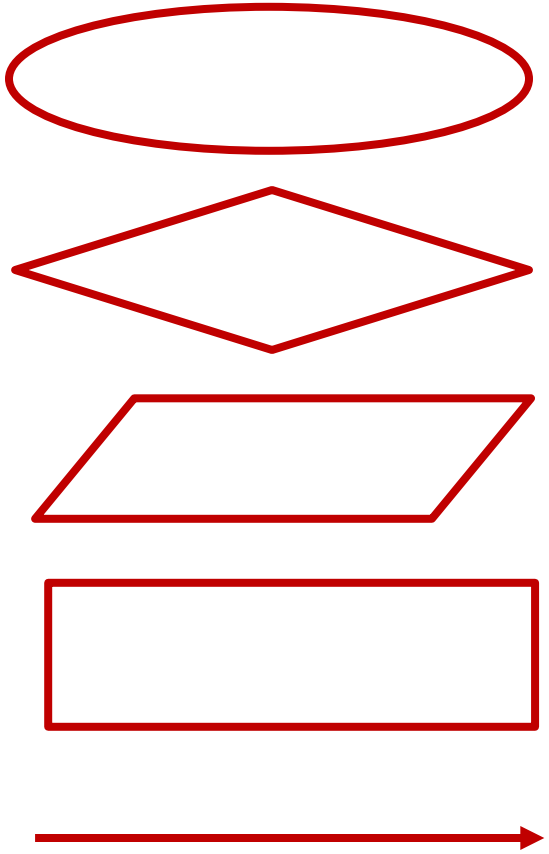
Consider the grading system for students:

- Excellent : Above 95%.
- Good: Greater than 80% & less than equal to 95%.
- Average: Greater than 60% & less than equal to 80%.
- Below average: Less than 60%



Copyright © 2021, ABES Engineering College

Revision of Flowchart



Used for start and stop.

Used for decision making.

Used for input/output.

Used for processing statement.

Connector / flow

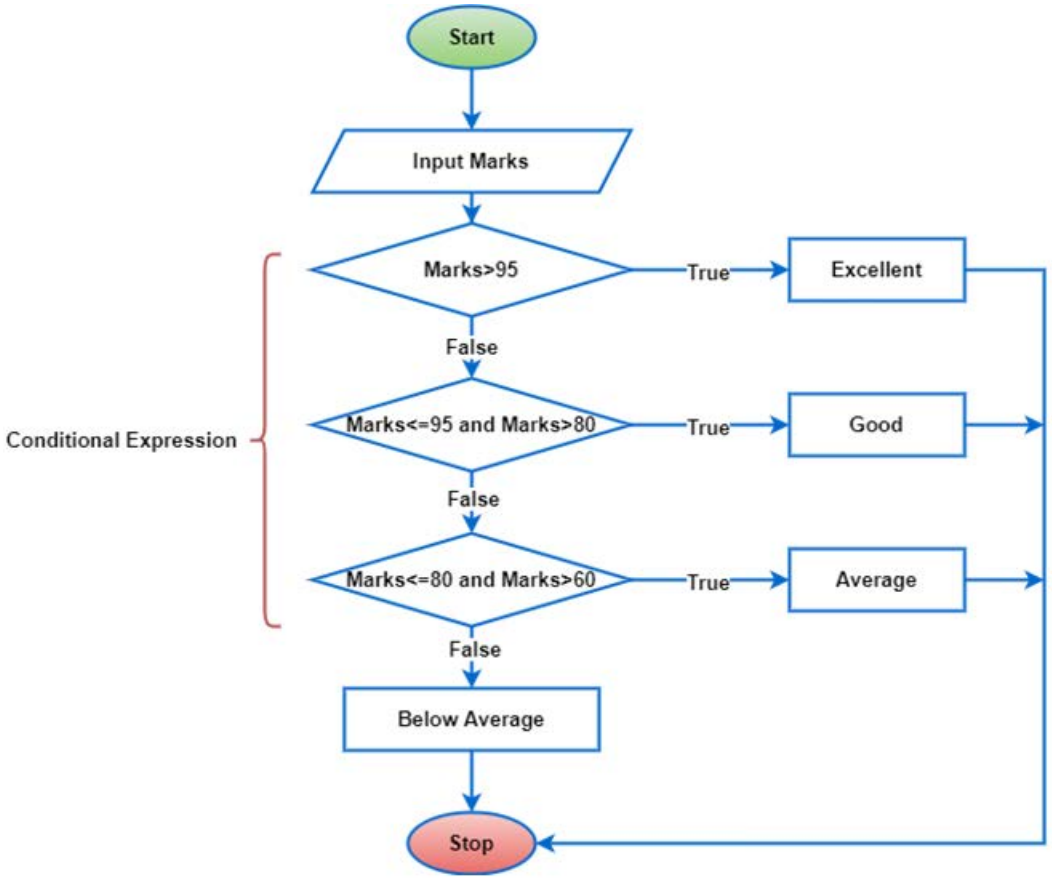
Copyright © 2021, ABES Engineering College

Flowchart of the grading system

On the basis of the conditions, corresponding block of the code will be executed or not.

Consider the grading system for students:

- Excellent :Above 95%.
- Good :Greater than 80% &less than equal t 95%.
- Average : Greater than 60% & less than equal to 80%.
- Below average: Less than 60%



Copyright © 2021, ABES Engineering College

Types of decision/selection control statements:

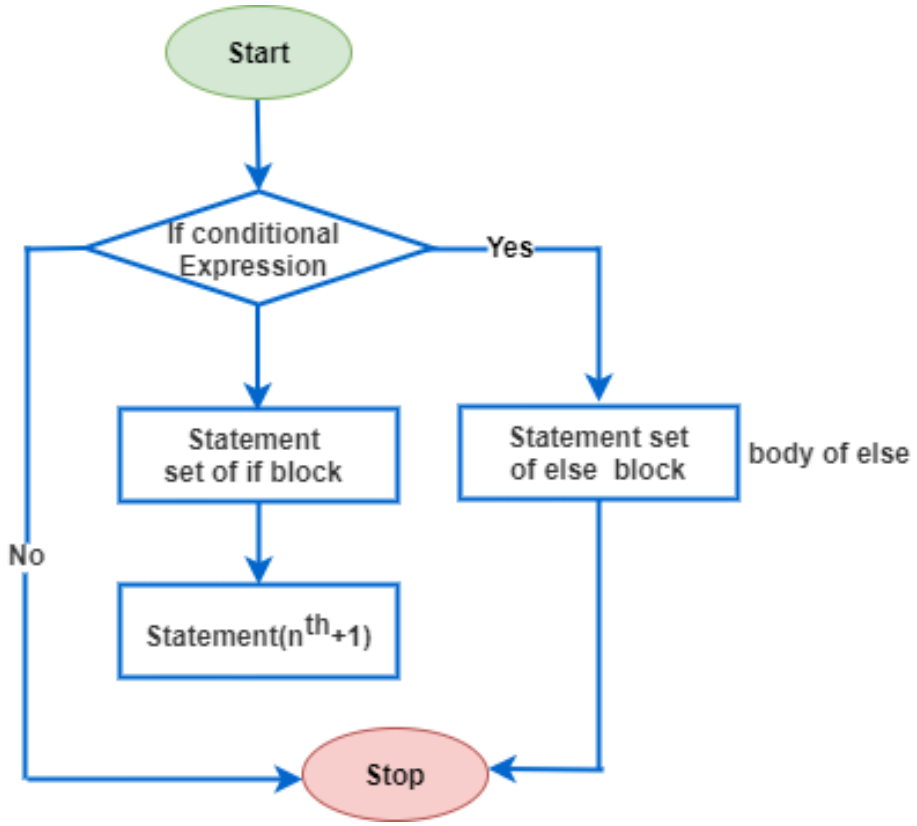
- Simple if
- if-else
- if..elif..else
- Nested...if…else

Copyright © 2021, ABES Engineering College

Simple if statement

- *if* evaluates whether a condition is:
  True
  False.

- *if* block is executed:
  if and only if the value of its condition expression is true.



Note: As we use curly braces in 'C' Language to define the scope of conditions and loops; in the same manner we use colon (:) and indentation in Python.

Copyright © 2021, ABES Engineering College

Syntax of Simple if

Mention the condition to be checked , return true or false

keyword

```
if <conditional expression>:
    statement 1
    statement 2
    .
    .
    .
    statement nth
statement (nth + 1)
```

Colon(:) at the end of if line is required

Indentation

The statements having same indentation represent a code block or group of statements in Python

The block of statements/code executed if condition is TRUE

Copyright © 2021, ABES Engineering College

Examples

- Print whether a number is even or not.

Case-1 : When n is even.

```
n=10 #Initialize the value of n

if(n % 2 == 0):#test condition for n is even

    print("n is an even number")

print("Statement outside if block")
```

Execute only if condition evaluates to True

It always run either if True of False

```
n is an even number
Statement outside if block
```

Output

Case-2 : When n is not even

```
n=11 #Initialize the value of n

if(n % 2 == 0):#test condition for n is even

    print("n is an even number")

print("Statement outside if block")
```

Execute only if condition evaluates to True

It always run either if True of False

```
Statement outside if block
```

Output

Copyright © 2021, ABES Engineering College

Examples

- A program to increment a number if it is positive.

```
x = 10 #Initialize the value of x

if(x > 0): #test the value of x

    x = x+1 # Increment the value of x if it is > 0
    print(x)

print("Statement outside if block")
```

Execute only if condition evaluates to True

It always run either if True of False

```
11
Statement outside if block
```

Output

Explanation:

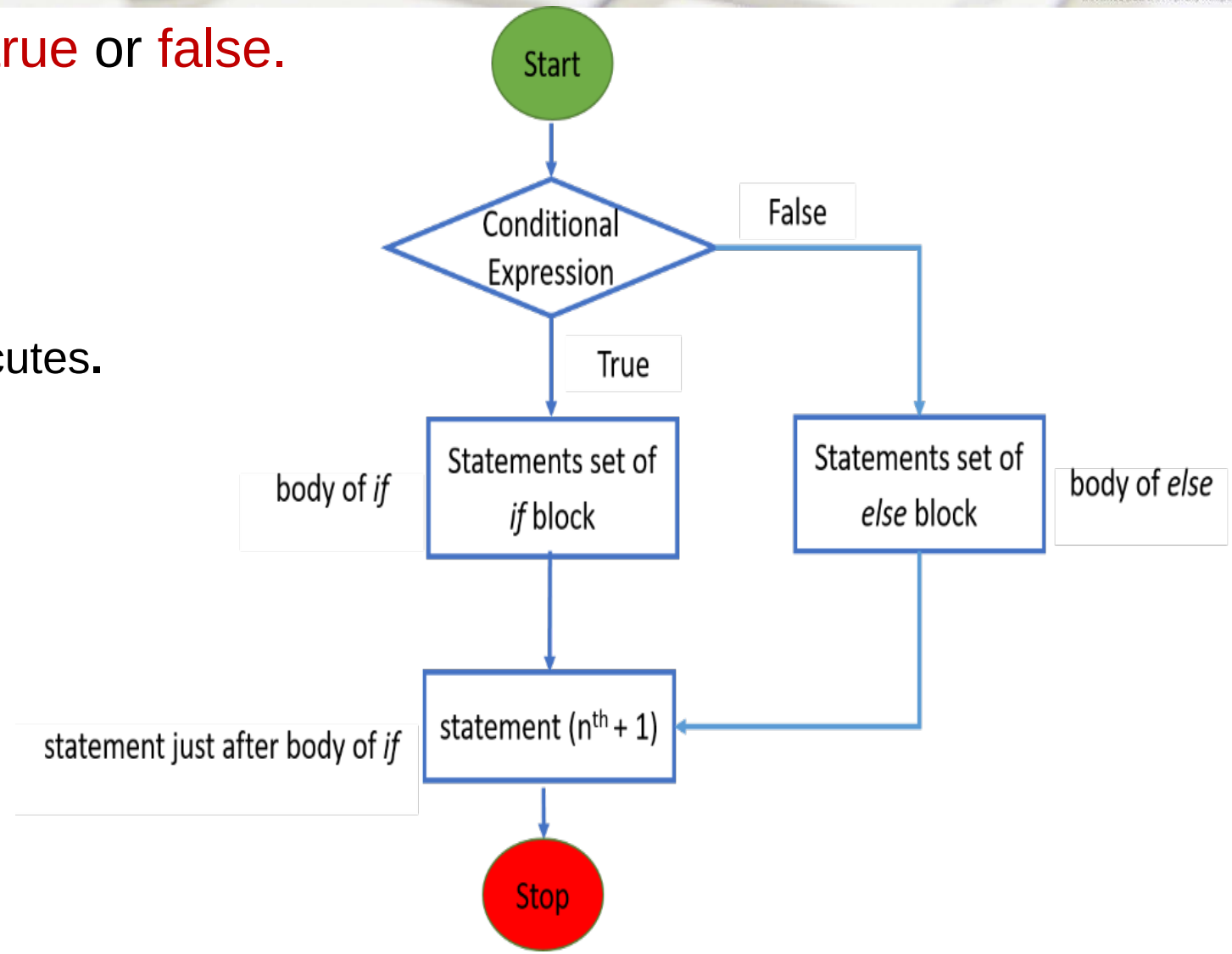
- The condition **x>0** is True for value of **x = 10** so the statements in the block of code will be executed.
- The statement **x = x+1** will increment the value of x by 1 i.e. **x = 10 + 1; x = 11** and statement **print (x) will print x value as 11.**

Copyright © 2021, ABES Engineering College

if else Statement

- It checks whether a condition is true or false.

- If a condition is true,
  - The *if* statement executes
  - Otherwise, the *else* statement executes.



Copyright © 2021, ABES Engineering College

Syntax of if else Statement

Mention the condition to be checked , return true or false

keyword

colon (:) at the end of the **if** line is required

```
if <conditional expression>:
    Body of if
else:
    Body of else
statement just after if-else
```

The block of statements/code executed if condition is TRUE

The block of statements/code executed if condition is False

Copyright © 2021, ABES Engineering College

Example

- A program to print "Excellent" if marks is greater than 95 otherwise print "Not Excellent"

Input marks

```
marks = int(input('Enter your total marks'))
if marks > 95 :
    print("Excellent")
else:
    print("Not Excellent")
```

If TRUE

If FALSE

```
Enter your total marks68
Not Excellent
```

Output

Explanation:

- For input 68, condition marks > 95 is False. Therefore, else statement print ("Not Excellent") is executed and output will be "Not Excellent".

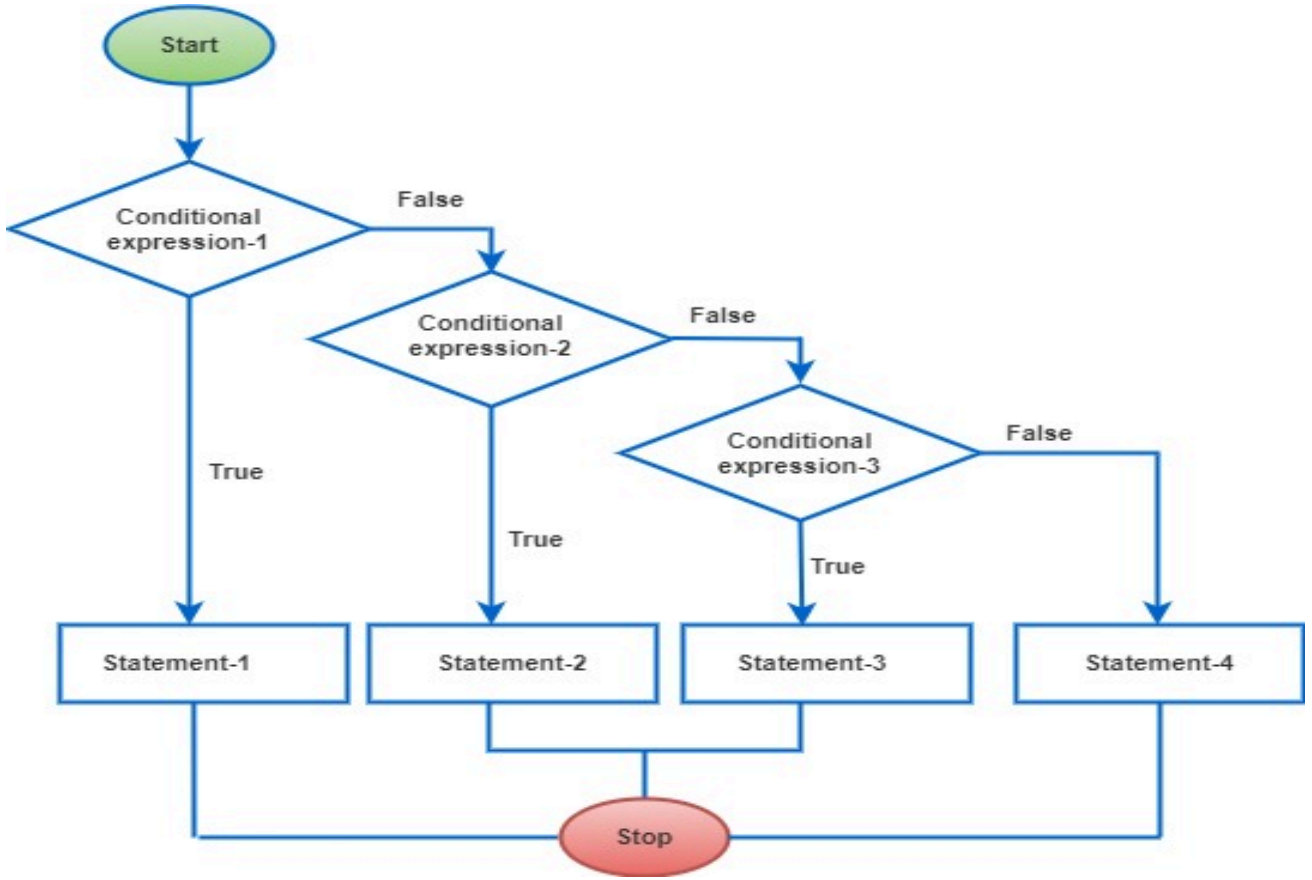
Copyright © 2021, ABES Engineering College

if..elif..else statement

Extension of the if-else statement

For a multiple conditional statements

- We use if-elif statement.



Note: The conditional expression of *if-elif-else* are executed from top to bottom in a sequential manner. *An elif statements are known as elif Ladder.*

Copyright © 2021, ABES Engineering College

Syntax of if..elif..else statement

Mention the condition to be checked , return true or false

keyword

colon (:) at the end of the **if** line is required

```
if <condition expression 1> :
    #Body of if
elif <condition expression 2> :
    #Body of First elif
elif <condition expression 3> :
    #Body of Second elif
.
.
.
else:
    #Body of else
```

TRUE

FALSE

TRUE

FALSE

TRUE

FALSE

Copyright © 2021, ABES Engineering College

Example

```
marks = int(input('Enter your Percentage'))
if marks > 95 :
    print("Excellent")
elif marks > 80 and marks <= 95:
    print("Good")
elif marks > 60 and marks <= 80:
    print("Average")
else:
    print("Below Average")
```

For marks = 88 this will evaluate FALSE

For marks = 88 this will evaluate TURE

Once conditional statement evaluated TRUE, then

All conditional statement after that will not be executed

```
Enter your Percentage88
Good
```

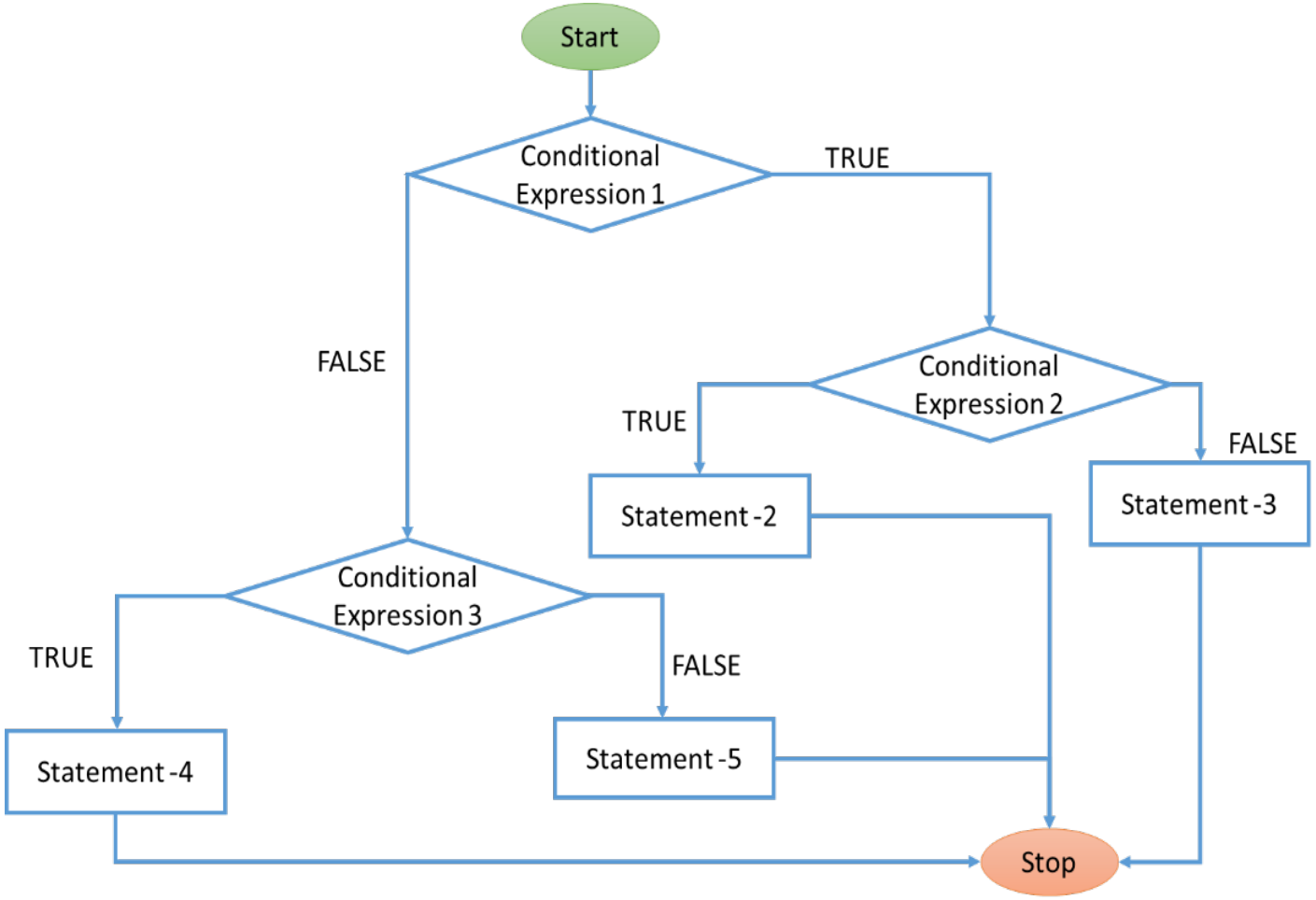
Explanation:

- For input 88, condition marks > 95 is False & control goes to next elif statement (marks>80 and marks<=95), which is True and statement print (“Good”) will be executed.
- The output will be “Good”.

Copyright © 2021, ABES Engineering College

Nested if-else statement

- When an if-else statement is present inside the body of another “if” or “else” then this is called nested if else.



Copyright © 2021, ABES Engineering College

Syntax of Nested if..else statement

```
if <conditional expression 1> :
        if <conditional expression 2> :
                #Statement 2
        else:
                #Statement 3
else :
        if <conditional expression 3> :
                #Statement 4
        else:
                #Statement 5
```

Executes when condition1 is TRUE

Executes when condition1 is FALSE

Copyright © 2021, ABES Engineering College

Example

- A program to take age as input from user and print “You are less than 18” if age is less than 18, otherwise print “You are above 18”.Also check that age could not be less than 0. If so then print “Age could not be less than 0”.

```
age = int(input('Enter your Age in years'))
if age > 0:
    if age < 18:
        print("You are less then 18")
    else:
        print("You are above 18")
else:
    print("Age could not be less than 0")
```

If-else inside if

For age > 0

For age < 0

Output -

```
Enter your Age in years20
You are above 18
```

Copyright © 2021, ABES Engineering College

Example

- A program to take age as input from user and print
  - "You are less than 18" if age is less than 18
  - otherwise print "You are above 18".
  - Also check that age could not be less than 0.
  - If so then print "Age could not be less than 0".

```
age = int(input('Enter your Age in years'))
if age > 0:
    if age < 18:
        print("You are less then 18")
    else:
        print("You are above 18")
else:
    print("Age could not be less than 0")
```

If-else inside if

For age > 0

For age < 0

Output -

```
Enter your Age in years-1
Age could not be less than 0
```

Copyright © 2021, ABES Engineering College

Can you answer these questions?

1. Which one of the following is a valid Python if statement :

A) if a>=2 :

B) if (a >= 2)

C) if (a => 22)

D) if a >= 22

Copyright © 2021, ABES Engineering College

3.Which of following is not a decision-making statement?

A) if-elif statement

B) for statement

C) if -else statement

D) if statement

Copyright © 2021, ABES Engineering College

4.Predict the output of the following code:

A) No output

B) okok

C) ok

D) None of above

```
x=3
if (x>2 or x<5 )and x==5:
    print("ok")
else:
    print ("no output")
```

Copyright © 2021, ABES Engineering College

5. What is the output of the following code snippet?

A) Launch a Missile

B) Let's have peace

C) 0.3

D)None

```
y=0.3
if y!=0.3:
    print("lunch a missile")
else:
    print ("let's have peace")
```

Copyright © 2021, ABES Engineering College

6. Which of the following is true about the code below?

```
x = 3
if (x > 2):
    x = x * 2;
if (x > 4):
    x = 0;
print(x)
```

A) x will always equal 0 after this code executes for any value of x

B) if x is greater than 2, the value in x will be doubled after this code executes

C) if x is greater than 2, x will equal 0 after this code executes

Copyright © 2021, ABES Engineering College

Summary

- Control statement are statements that control the flow of execution of statements so that they can be executed repeatedly and randomly.
- The if statement executes a group of statements depending upon whether a condition is true or false.
- The if..else statement executes a group of statements when a condition is true; Otherwise, it will execute another group of statements.
- The if..elif statement is an extension of the if-else statement. When we have multiple conditional statements, then we use if-elif statement.
- When an if..else statement is present inside the body of another “if” or “else” then this is called nested if else.

Copyright © 2021, ABES Engineering College

Session Plan - Day 2

2.3 Iterative looping Statements

- While loop
- Nested While loop
- Examples
- Review Questions
- Summary

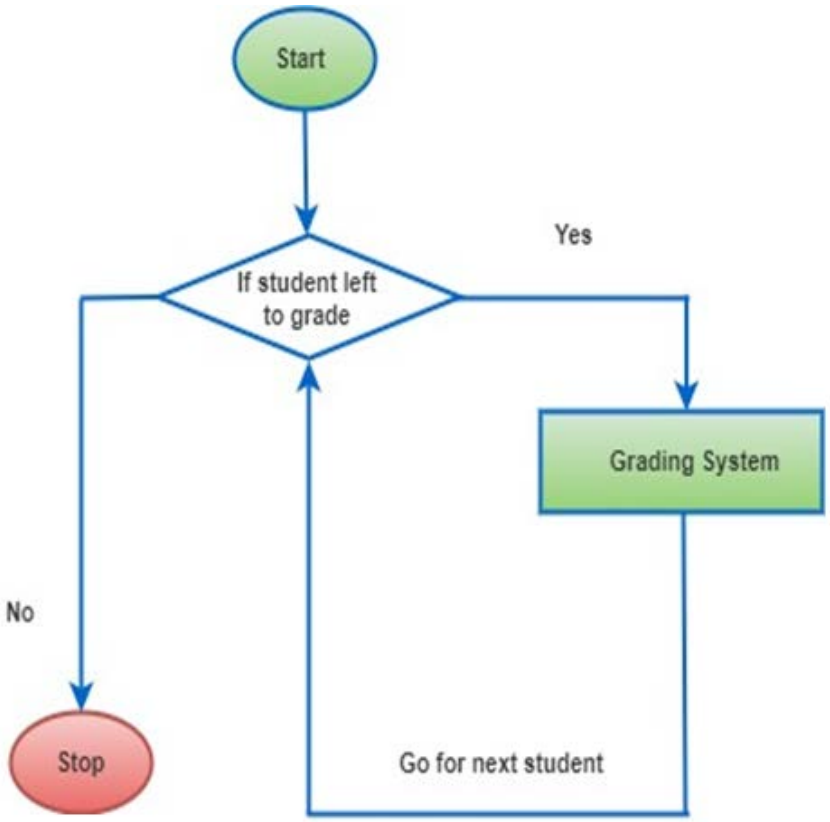
Copyright © 2021, ABES Engineering College

Iterative/Looping Statements

Sometimes we need to perform certain operations again and again.

Real Life Scenario:

- A teacher decide to grade 75 students on the basis of marks.
- He/she wants to do this for whole class.
- Teacher would repeat grading procedure for each student in the class.
- This is called iterative/ looping.



Copyright © 2021, ABES Engineering College

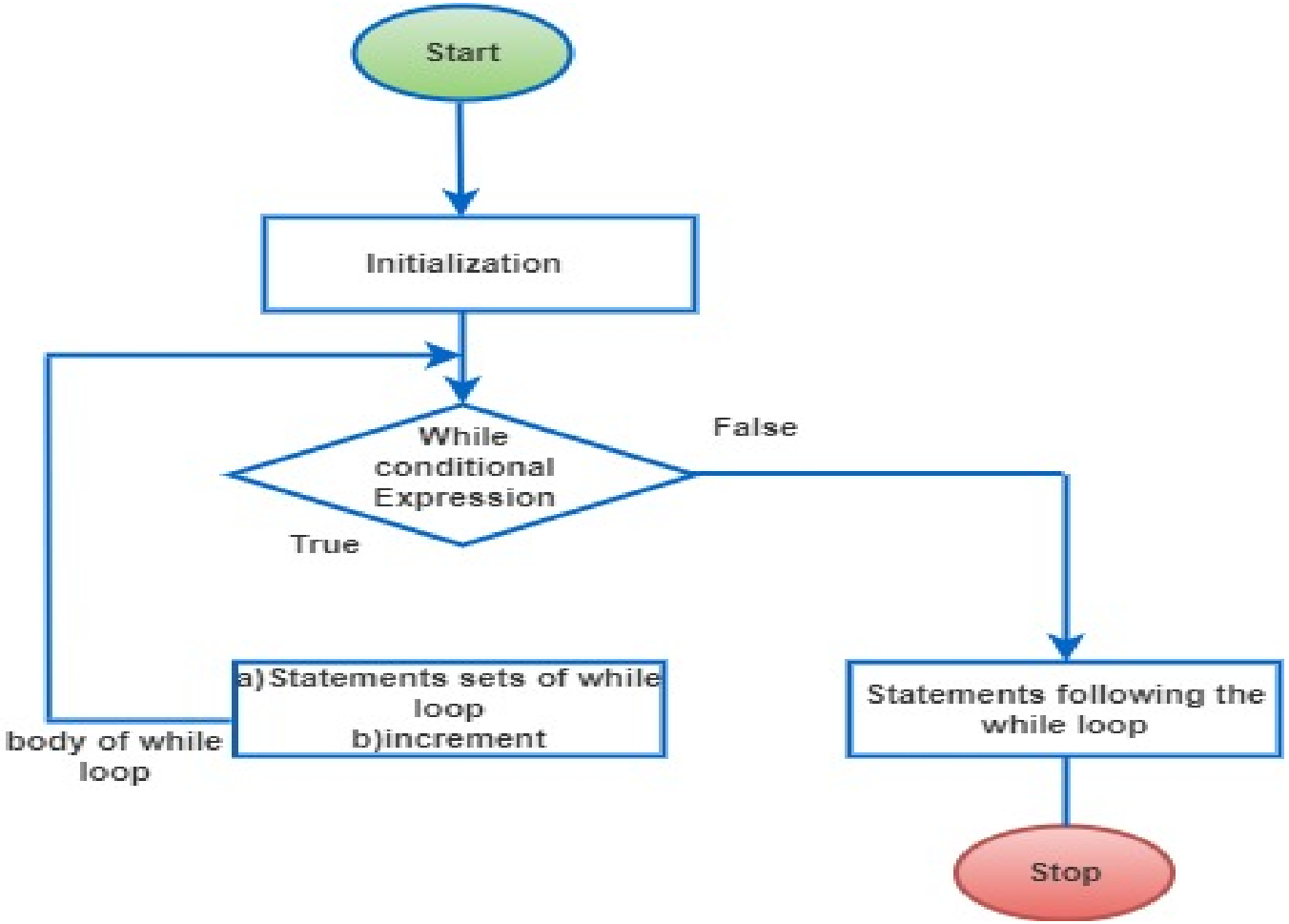
Types of loops in Python

- While loop
- Nested While loop
- For loop
- Nested for loop

Copyright © 2021, ABES Engineering College

While loop

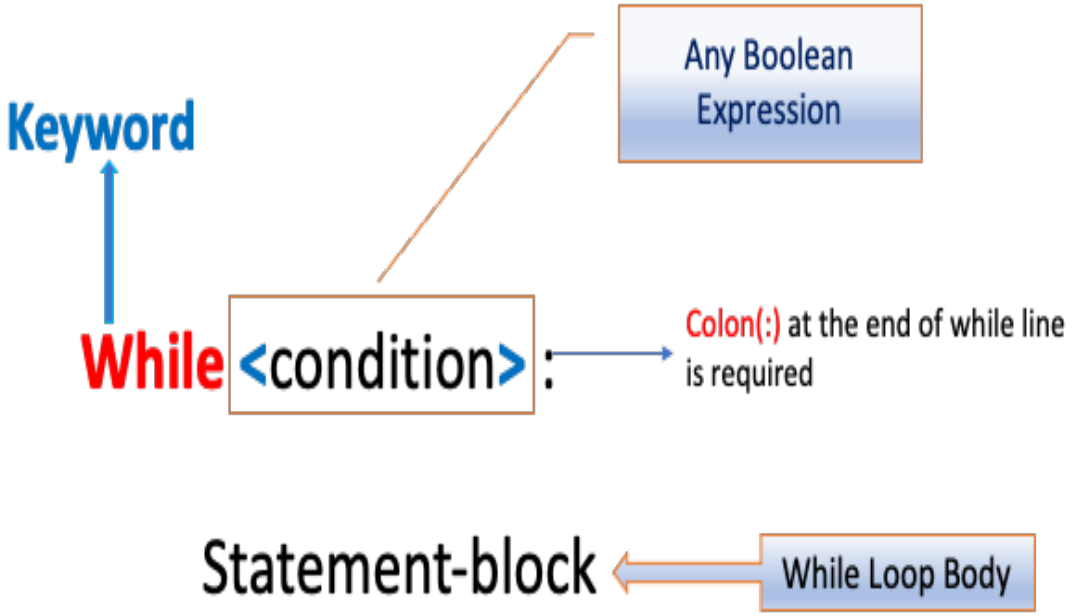
- It is used to repeat a block of code as long as the given condition is true.



Copyright © 2021, ABES Engineering College

Syntax of While loop

- Boolean expression is checked first.
- The body of the loop is entered only if the Boolean expression evaluates to True.
- After one iteration, the Boolean expression is checked again.
- This process continues until the Boolean expression evaluates to False.



Copyright © 2021, ABES Engineering College

Example

Write a Python Program to print 4 Natural Numbers using i.e. 1,2,3,4 using While loop.

i, start value 1

```
i=1   #Intitialization
while (i<5):

    print(i)
    i=i+1      # Increment
```

True

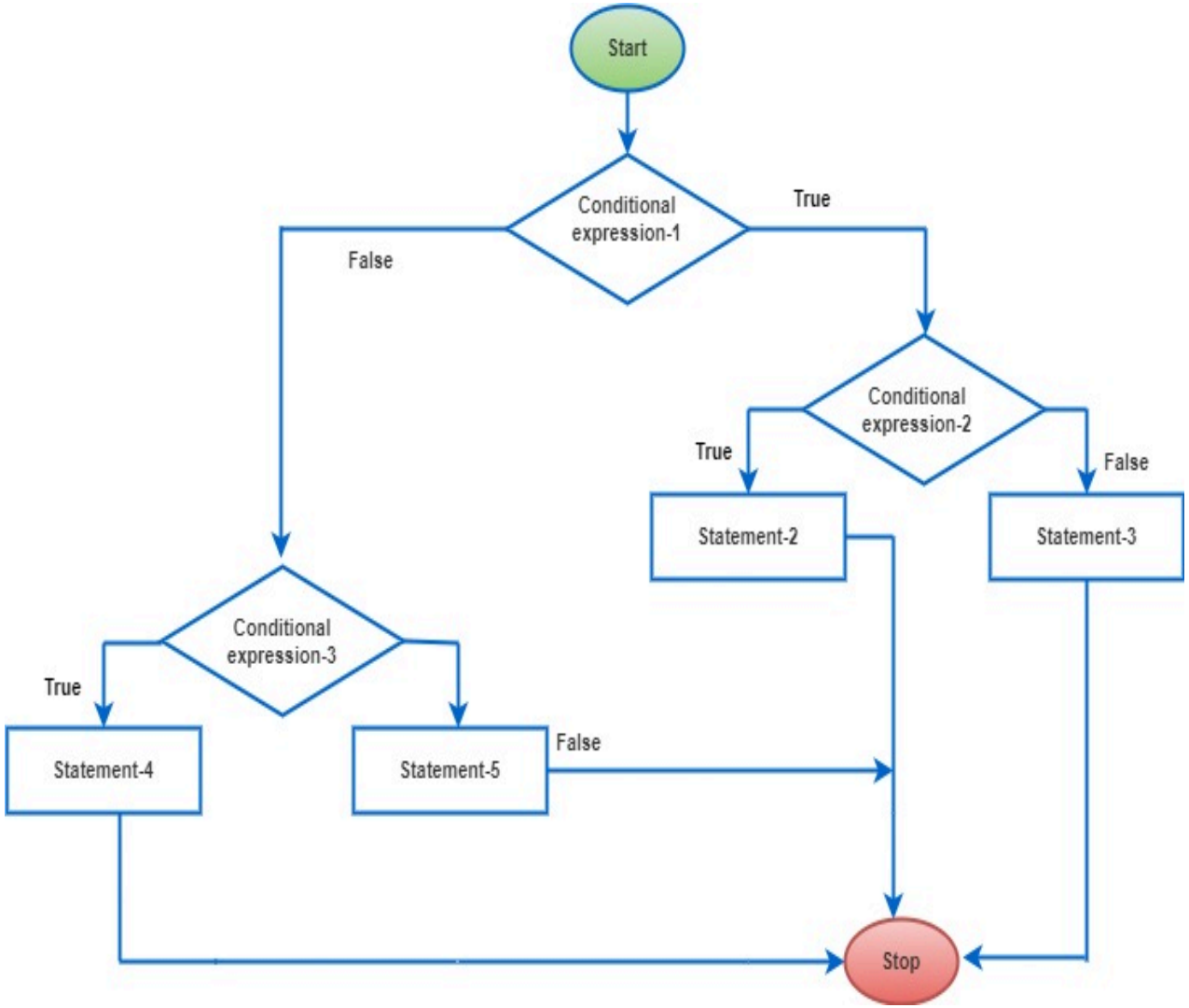
```
1
2
3
4
```

Output

Copyright © 2021, ABES Engineering College

Nested While loop

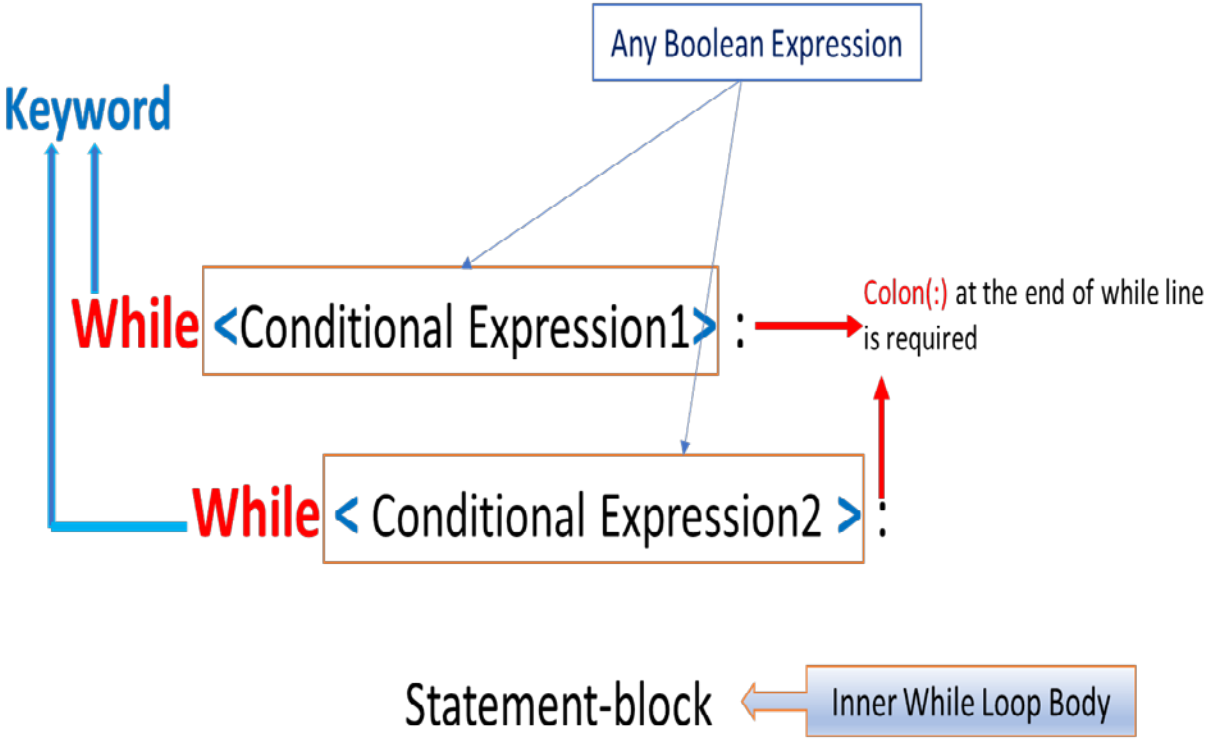
- Nested while loop is called when we use while loop inside a while loop.
- We can use any number of while loop inside a while loop.
- Main while loop as outer while loop and nested while loop as inner while loop.



Copyright © 2021, ABES Engineering College

Syntax of Nested While loop:

- Outer while loop runs m number of times.
- Inner while loop runs n number of times.
- The total no. of iterations would be m*n.



Copyright © 2021, ABES Engineering College

Example of Nested While loop:

- Write a program to demonstrate While loop.

outer while loop

Inner while loop

```
condition1 =1
while condition1<=3 :
     print(condition1,"Outer loop executed")
     condition2=1

     while condition2<=3:
          print(condition2,"Inner loop")
          condition2+=1

     condition1+=1;
```

```
1 Outer loop executed
1 Inner loop
2 Inner loop
3 Inner loop
2 Outer loop executed
1 Inner loop
2 Inner loop
3 Inner loop
3 Outer loop executed
1 Inner loop
2 Inner loop
3 Inner loop
```

Output

Copyright © 2021, ABES Engineering College

Can you answer these Questions?

1. What will be the output of the following code snippet?

```
string1 = "Python"
i = "p"
while i in string1:
    print(i, end =" ")
```

Options

A. None

B. Python

C. Ppppp

D. PPPPP

Copyright © 2021, ABES Engineering College

2. What will be the output of the following code snippet?

```
i = 0
while i < 3:
    print(i)
    i += 1
else:
    print(0)
```

A. 0 1 2

B. 2 3

C. 0 1 2 0

D. None of the above

Copyright © 2021, ABES Engineering College

3. What should be the value of the variables num1 and num2 in the code snippet below if the output expected is 4?

```
num1=?
num2=?
while(num1>=2):
    if(num1>num2):
        num1=num1/2
    else:
        print(num1)
        break
```

A. 16,6

B. 12,5

C. 8,2

D. 16,2

Copyright © 2021, ABES Engineering College

Summary:

- While loop is used to iterate block of codes repeatedly until given condition is True
- While loop present inside another while loop it is called as nested while loop.
- The nested while loop is while statement inside another while statement.

Copyright © 2021, ABES Engineering College

Session Plan - Day 3

2.3 Iterative looping Statements

- Range()
- For loop
- Nested for loop
- Examples
- Review Questions
- Summary

Copyright © 2021, ABES Engineering College

Introduction to Iterative/looping statements:

Copyright © 2021, ABES Engineering College

Downloaded from pickywallpapers.com

Range() Function,

Optional
default value = 0

Required

Optional
default value =
+1

range (start value, stop value, step value)

- Range() is a built in function in Python
- It gives the sequence of numbers.

Copyright © 2021, ABES Engineering College

Range Function Example:

- range (1,5,1)
- It will generate a sequence starting from value 1 up to value 4
- (5 is not included) and step by 1
- So, the output of the above gives us the sequence 1,2,3,4.

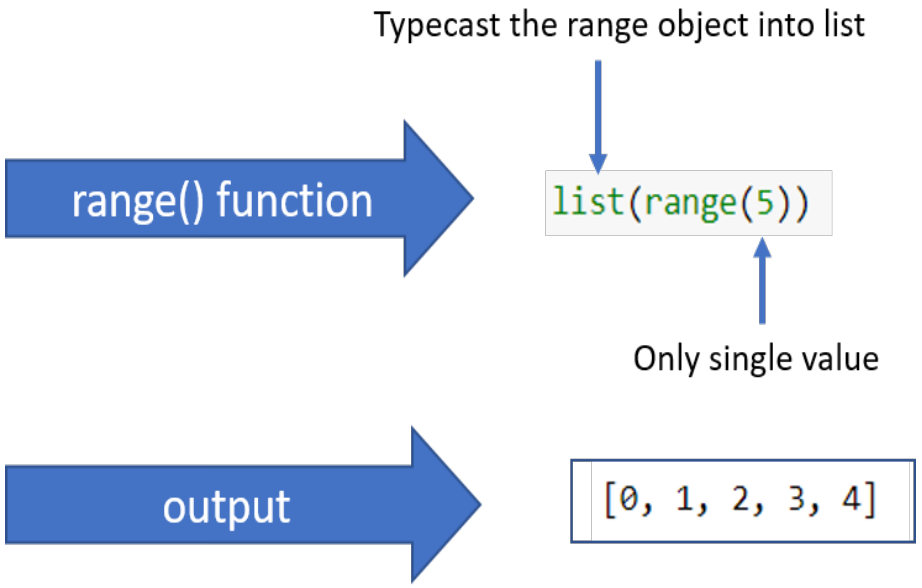
Note: Range()Function returns the range object, you must typecast range function into collections.

Copyright © 2021, ABES Engineering College

Range

- Range () Function with one argument.
- Single value is considered as the stop value.
- It means the start value is considered as 0, and step value is considered as 1.

Program to generate sequence numbers 0 to 4.

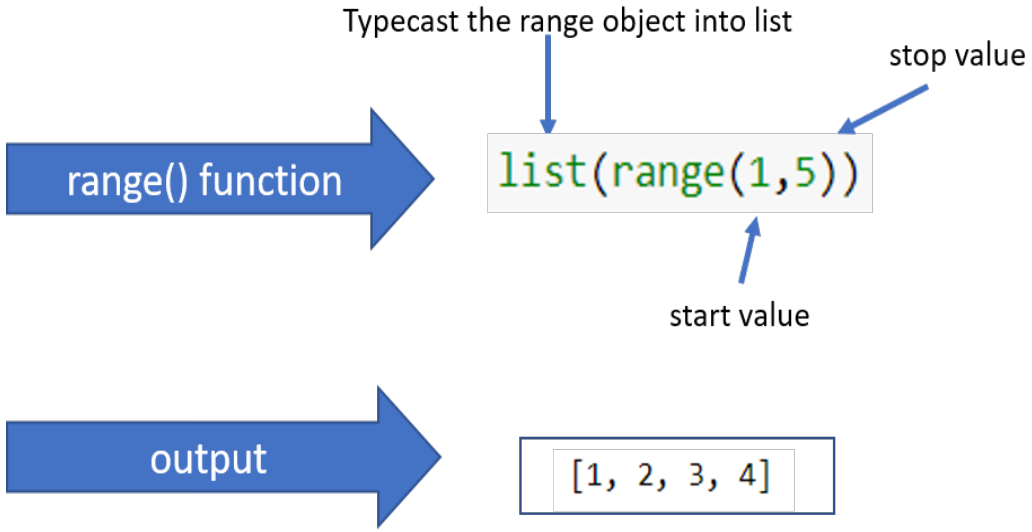


Copyright © 2021, ABES Engineering College

Range

- Range () with two argument:
- This means the start value and stop value is mentioned.
- The step values is considered as 1 by default.

Program to generate sequence of numbers 1 to 4.

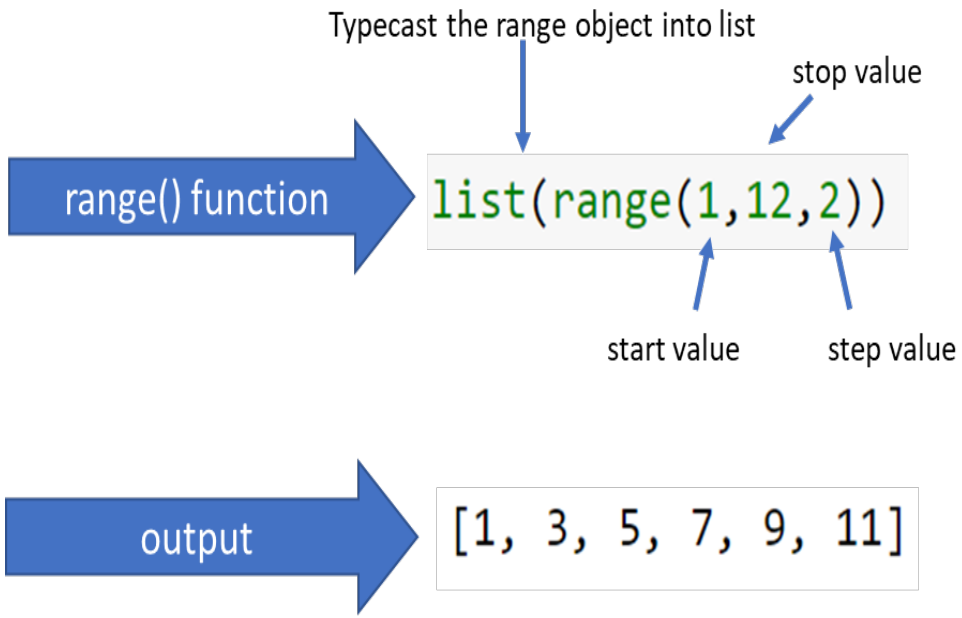


Copyright © 2021, ABES Engineering College

Range

- Range() with three arguments:
- It has start value, stop value and the step value, all three values are given inside range ().
- Here first start value is 1, stop value 12 and step value is 2. That's why the output is [1, 3, 7, 9, 11].

Program to generate a sequence of alternate natural number starting from 1 to 12.



Copyright © 2021, ABES Engineering College

Example

- Write a program to generate a sequence of starting from 10 to 2.

Typecast the range object into list

stop value

range() function

```
list(range(10,1,-1))
```

start value

step value

output

```
[10, 9, 8, 7, 6, 5, 4, 3, 2]
```

Copyright © 2021, ABES Engineering College

For loop

- For loop is used when we want to run a part of our program multiple times.

- It is also used to traverse sequences in python like lists, tuple etc.

Syntax of For loop :

keyword Variable Collection of elements

```
for iter_var in sequence:
    statements
```

Colon required at end of for

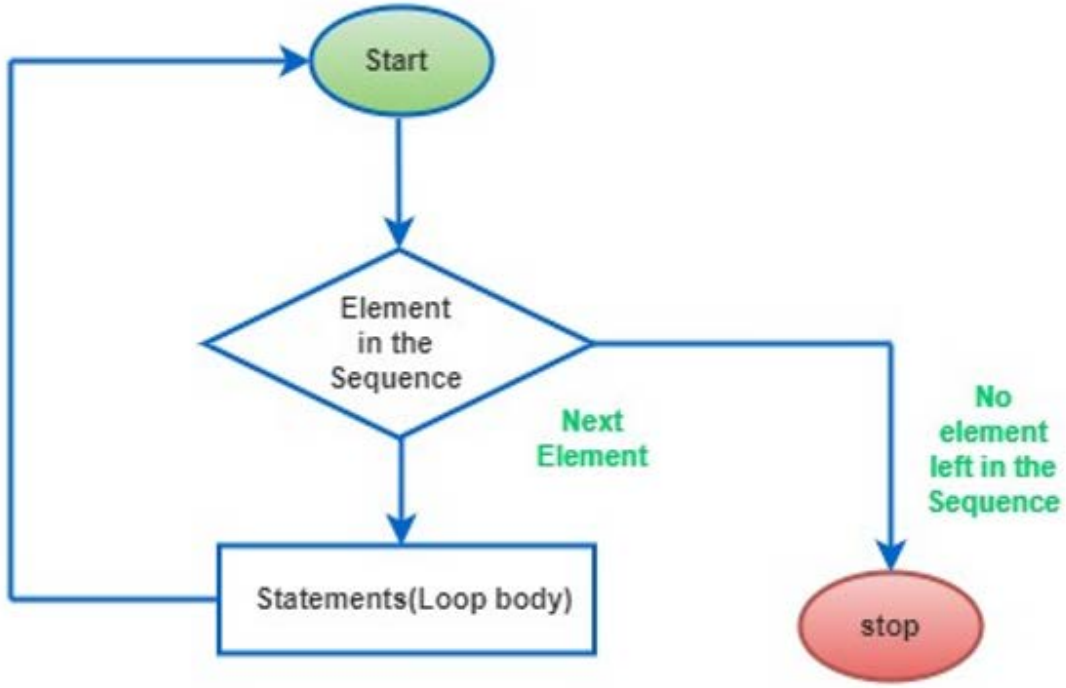
Part of program that need to be run multiple times

Copyright © 2021, ABES Engineering College

For loop:

Flowchart of For loop:

- In this flowchart the statements which we want to run multiple times would keep on running till, we have elements in the sequence.
- As no element left, it will come out from loop.



Copyright © 2021, ABES Engineering College

Example:

Example1:
For loop with range function:

For-loop statement

```
for i in range(10):
    print(i)
```

Statement

```
0
1
2
3
4
5
6
7
8
9
```

Output, number printed from 0 to 9

Example2:
Table of number 5 is printed using for loop

Number for which table is to be created

for-loop statement

```
num=5
for i in range(1,11):
    print(i,"* 5 =",i*5)
```

```
1 * 5 = 5
2 * 5 = 10
3 * 5 = 15
4 * 5 = 20
5 * 5 = 25
6 * 5 = 30
7 * 5 = 35
8 * 5 = 40
9 * 5 = 45
10 * 5 = 50
```

Output, table of 5

Copyright © 2021, ABES Engineering College

Nested For loop:

- Nested for loop is called when we use for loop inside a for loop.
- In python we can use any number of for loop inside a for loop.
- Generally, we call main for loop as outer for loop and nested for loop as inner for loop.
- If outer loop running m times and inner loop running n times, then total iterations would be m*n.

Syntax of **Nested For loop**:

keyword Variable Collection of elements

```
for iter_var1 in sequence1:
    for iter_var2 in sequence2:
        statements
```

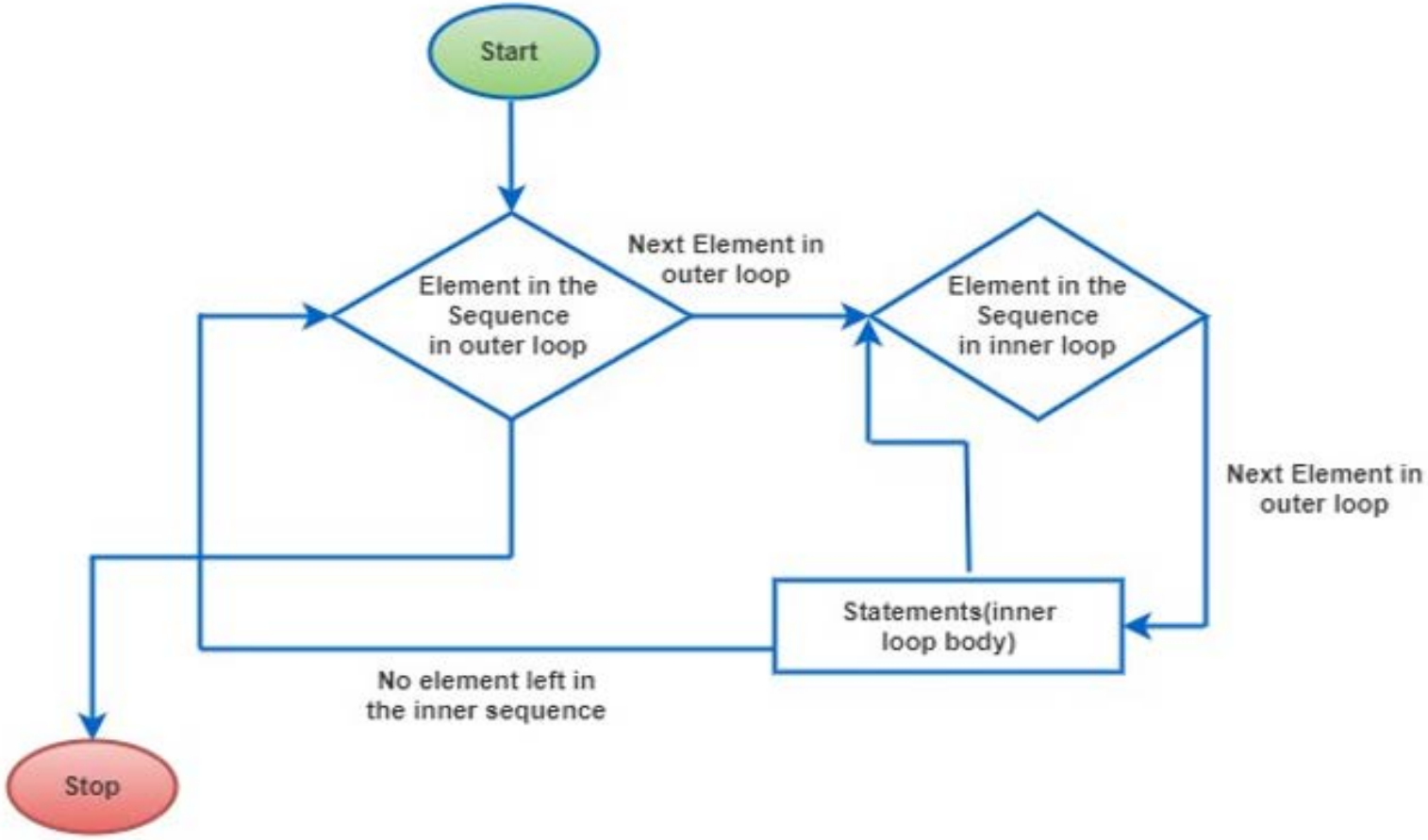
Outer Loop

Inner Loop

Part of program that need to be run multiple times

Copyright © 2021, ABES Engineering College

Flowchart of Nested For loop:



Copyright © 2021, ABES Engineering College

Example

- We have used two for loops.
- Outer loop is running for *two* values-0 and 1 and inner loop is running for *three* values-0,1 and 2 and total print statements we are getting *2*3=6.*

Outer Loop

```
for i in range(2):
    for j in range(3):
        print('value of i-',i,' value of j-',j)
```

Inner Loop

Inner Loop statements

```
value of i- 0 | value of j- 0
value of i- 0 | value of j- 1
value of i- 0 | value of j- 2
value of i- 1 | value of j- 0
value of i- 1 | value of j- 1
value of i- 1 | value of j- 2
```

Output coming from inner loop for 0,1 and 2 while outer loop value is 0

Output coming from inner loop for 0,1 and 2 while outer loop value is 1

Copyright © 2021, ABES Engineering College

Can you answer these questions?

1. What is the output of the following code snippet??

a.
```
string3 = 'Technology'
for i in range(len(string3)):
    string3[i].upper()
print (string3[4])
```

A. N

B. n

C. H

D. h

b.
```
num1 = 5
num2 = 4
while ( num2>=1):
    print ('&')
    for index in range (1,num1+1):
        print ('+')
        num2= num2-1
        print ('*')
```

How many time special character print?

A. 11

B. 8

C. 10

D. 6

Copyright © 2021, ABES Engineering College

3. Which of the below programs will print all integers that aren't divisible by either 2 or 3 and lies between 1 and 50?

A.
```
for i in range(0,50):
    if(i%4!= 0 and i%6!=0):
        print(i)
```

B.
```
for i in range(0,51):
    if(i%4!= 0 and i%6!=0):
        print(i)
```

C.
```
for i in range(0,51):
    if(i%4!= 0 and i%3!=0):
        print(i)
```

D.
```
for i in range(0,51):
    if(i%2!= 0 and i%3!=0):
        print(i)
```

Copyright © 2021, ABES Engineering College

Summary:

- Range function () returns a sequence of numbers.
- A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).
- Nested for loop allows us to create one for loop inside another for loop.

Copyright © 2021, ABES Engineering College

Session Plan - Day 5

2.4 Jump statements

- Break
- Continue

2.5 Else With Loop

- While with else statement
- For with else statement
- Examples
- Review questions
- Summary

Copyright © 2021, ABES Engineering College

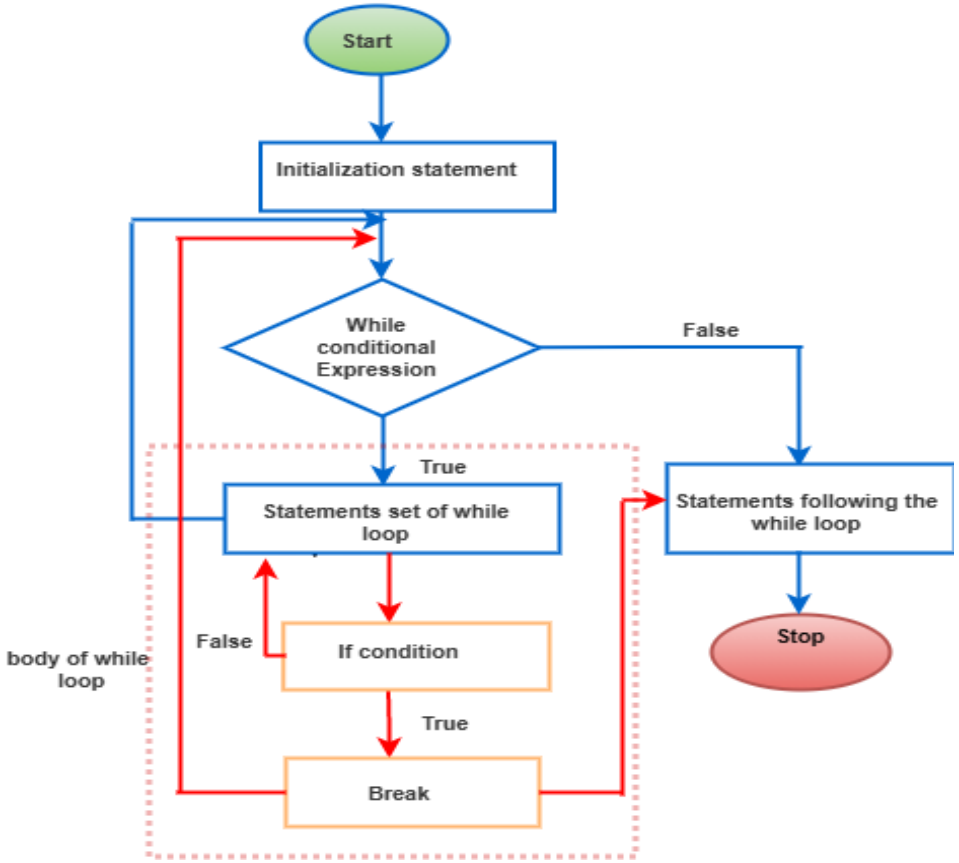
Jump Statements

- It is used to :-
  - Jump, skip or terminate the iteration or loop from the running program
  - Interrupt loops
- Also known as early exit from loop.
- Jump statements are of two types:-
  - Break
  - Continue

Copyright © 2021, ABES Engineering College

Break

- It terminates the execution of loop immediately
- Execution will jump to the next statements.



Copyright © 2021, ABES Engineering College

Syntax

Use of while loop

```
while <condition1>:

    <statement1>

    if <condition2>:
        break
```

If condition with break

scope of while loop

Outside the scope of while loop

```
<statement2>
```

Copyright © 2021, ABES Engineering College

Example

Use of while loop

If condition with break

```
i=1

while i<5:

    if i == 3:
        break

    print(i)
    i+=1

print("All done")
```

scope of while loop

Outside the scope of while loop

output

```
1
2
All done
```

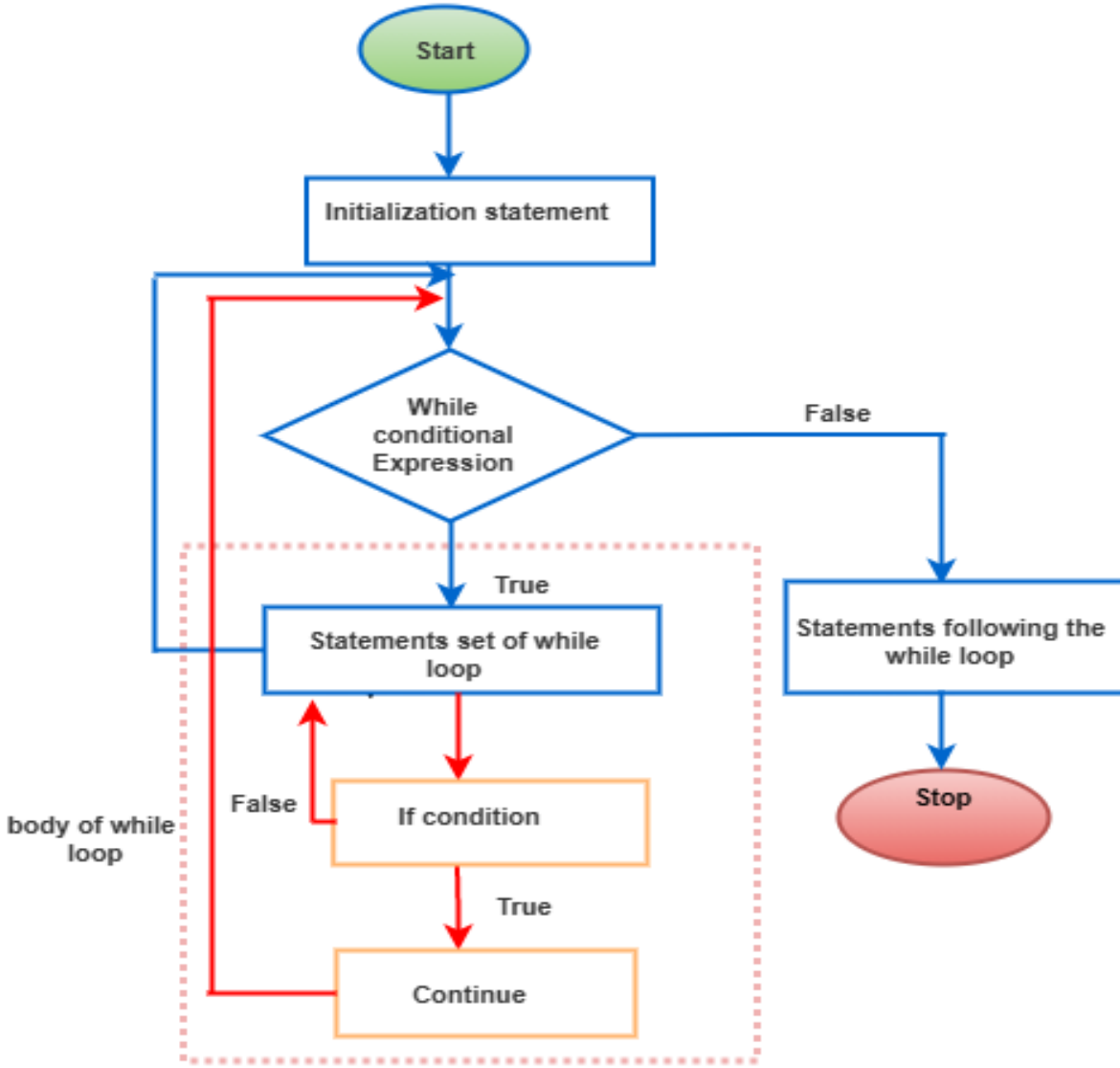
Explanation :-

- when value of I would be 3 break statement will be executed and loop will be terminated and control moves outside the while loop to print “All done”.
- If there would have no break, then loop would had run for i=1 to 4.

Copyright © 2021, ABES Engineering College

Continue

- It is used to :-
  - Skip ongoing iteration
  - continues remaining iterations of loop.
  - Jump to the next iteration by skipping the existin



Copyright © 2021, ABES Engineering College

Syntax

Use of while loop

If condition with continue

```
while <condition1>:
    if <condition2>:
        continue
    <statement1>
    <statement2>
```

scope of while loop

Outside the scope of while loop

```
<statement3>
```

Note: In break statement, the whole loop will end; and in continue statement, only the specific iteration will end.

Copyright © 2021, ABES Engineering College

Example

Use of while loop

If condition with continue

Outside the scope of while loop

```
i=1

while i<5:

    if i == 3:
        i+=1
        continue
    print(i)
    i+=1

print("All done")
```

scope of while loop

output

```
1
2
4
All done
```

Explanation:-

- The continue statement will interrupt the on-going iteration for i=3 and skip all the statements of current iteration which are mentioned after the continue statement like print(i).
- The control goes back to the while loop for next iteration and loop moves on.

Copyright © 2021, ABES Engineering College

Else with Loop

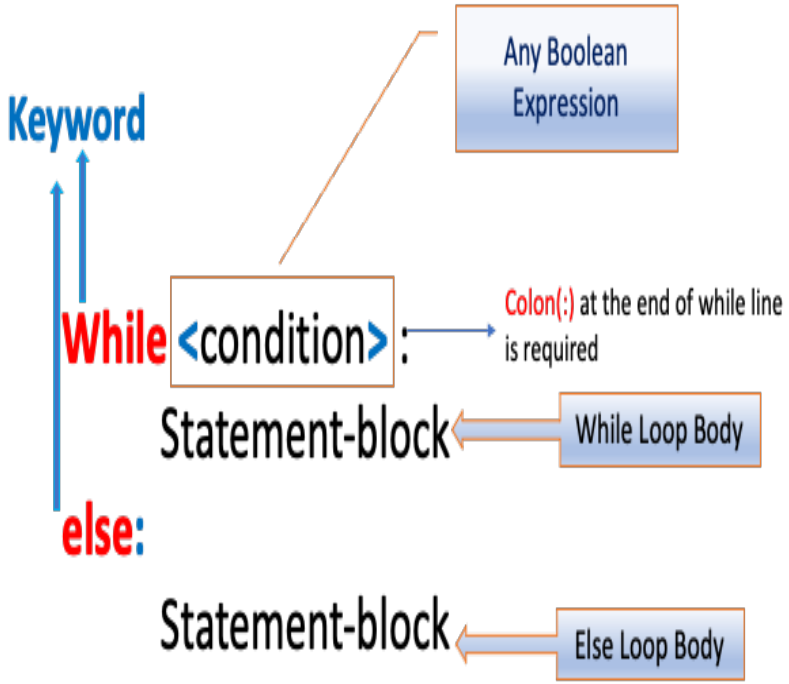
- While with else statement
- For with else statement

Copyright © 2021, ABES Engineering College

While with else Statement

In python the while statement may have an exceptional else clause.

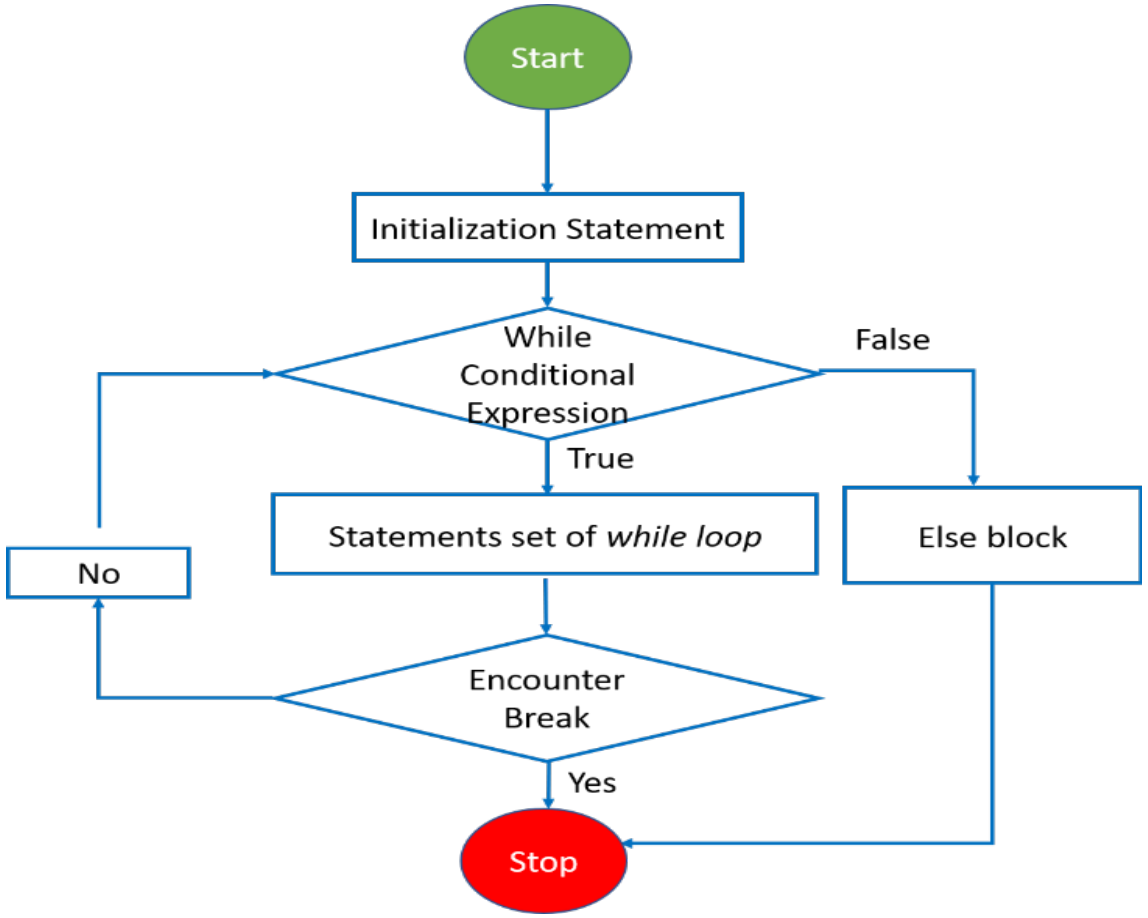
- If the condition in the while loop evaluates to False, the else portion of the code runs.
- if break statement is used in while loop then else section is not taken into consideration.



Copyright © 2021, ABES Engineering College

Flowchart of While with else loop

- The else clause will be executed when the condition becomes False and the loop runs normally.
- The else clause, on the other hand, will not execute if the loop is terminated early by a break or return statement.



Copyright © 2021, ABES Engineering College

Example

- Without Break Statement

```
condition =1
while (condition <5):

  print(condition)

  condition+=1
else:

  print("Else part is executed")
```

True

False

The **else clause** is executed when the while loop condition becomes False

```
1
2
3
4
Else part is executed
```

Output

- While loop will execute normally up to a given condition. No early exit is there so else block will be executed.

Copyright © 2021, ABES Engineering College

Example

- With Break Statement

```
condition =1
while (condition <5):

  print(condition)

  condition+=1

  if (condition==3):
    break
else:

  print("Else part is executed")
```

True

The **else clause** will not executed because of **break** statement

```
1
2
```

Output

- While loop will get terminated when value of condition variable = 3, So else block will not be executed.

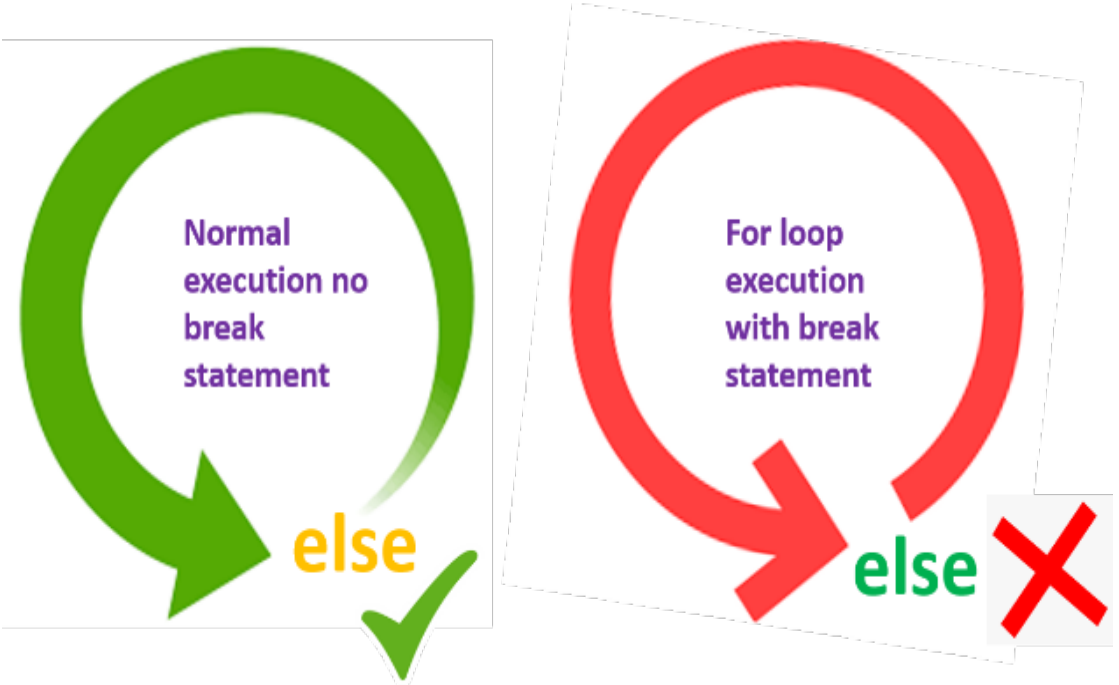
Copyright © 2021, ABES Engineering College

For with else statement

Python allows **else with for loop**.

- When all the iteration of **for loop** are finished normally, then control goes to else part.

- If loop has a break statement, then else part would not execute.



Copyright © 2021, ABES Engineering College

Syntax of *for with else*

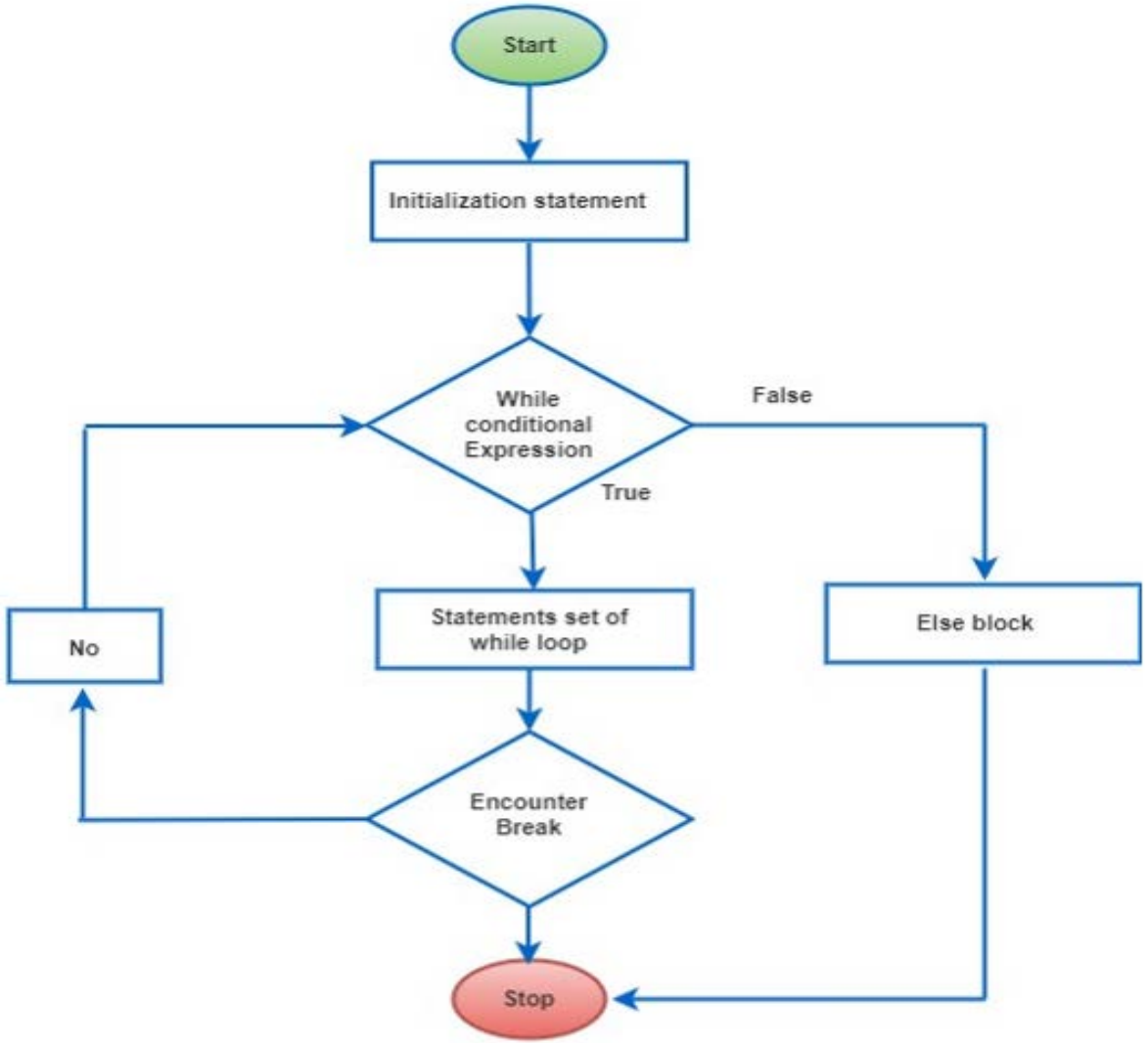
keyword Variable Collection of elements

```
for iter_var in sequence:
        statements
else:
        statements
```

Statement that would run after completion of for loop

Copyright © 2021, ABES Engineering College

Flow chart of *for with else*



Copyright © 2021, ABES Engineering College

Example

- Program to demonstrate for loop with else.

```
for i in range(5):
    print(i)
else:
    print("Loop finished, i am in else part")
```

for-loop with statement

Body of else

```
0
1
2
3
4
Loop finished, i am in else part
```

Output from for-loop

Else statement executing after for loop execution finished

- For loop has been used with else, we can clearly see that, first all the elements of sequence get executed and when all values from 0 to 4 gets printed, control goes to else part of the program.

Copyright © 2021, ABES Engineering College

Example

- Program to demonstrate for loop with break and else.

```
for i in range(11):
    if(i==5):
        break
    print(i)
else:
    print("Loop finished, i am in else part")
```

for loop with sequence

Break with condition for i=5

else body

```
0
1
2
3
4
```

Output from 0 to 4 then break executed and program terminated and no else part executed

.

- We have used break statement with the condition for i=5, so in the output we are getting values from 0 to 4 and then no else part executed.

Copyright © 2021, ABES Engineering College

Copyright © 2021, ABES Engineering College

Example

- Write a program to search a given color name in list of colors. Use Linear/Sequential Search Techniques.

```
color_name = ['Red','Green','Blue','Black','Pink']
name = input("Enter the color name to be search ")
for c in color_name:
    if name==c:
        print(f"Color name {c} found in list")
        break
else:
    print("Color name not found in list")
```

Test case - 1 For the input value **'Blue',** output will be –

Enter the color name to be search Blue
Color name Blue found in list

Test case - 2 For the input value **'Yellow',** output will be –

Enter the color name to be search Yellow
Color name not found in list

Copyright © 2021, ABES Engineering College

Example

- Write a program to find a given number is prime or not. A number is prime number if it is divisible only by itself and 1.

Approach:

- Every number is divisible by 1 and itself, so in the below code we check that a given number is divisible by 2 to n-1 or not.

- If divisible means not a prime number otherwise a prime number.

.

```
n = int(input("Enter any positive number"))
for i in range(2,n):
    if n%i==0:
        print(f"{n} is not a prime number")
        break
else:
    print(f"{n} is a prime number")
```

Test case - 1 For the input value 11, output will be –

Enter any positive number11
11 is a prime number

Test case - 2 For the input value 12, output will be –

Enter any positive number12
12 is not a prime number

Copyright © 2021, ABES Engineering College

Can you answer these questions?

1:-what will be the output of following code:-

```
for i in range(10):
    print(i)
    if(i == 7):
        print('break')
        break
```

A)
```
0
1
2
3
4
5
6
7
BREAK
```

B)
```
0
1
2
3
4
5
6
```

C)
```
0
1
2
3
4
5
6
7
```

D) **NONE**

Copyright © 2021, ABES Engineering College

2:- What will be the output of following code:-

```
for i in range(6):
    if(i==3):
        continue
    print(i)
```

A)
```
0
1
2
3
4
5
continue
```

B)
```
0
1
2
4
5
```

C)
```
0
1
2
3
4
5
6
continue
```

D) **NONE**

Copyright © 2021, ABES Engineering College

Summary

- Jump statements in python are used to alter the flow of a loop like you want to skip a part of a loop or terminate a loop.
- Break Statement in Python is used to terminate the loop.
- Continue Statement in Python is used to skip all the remaining statements in the loop and move controls back to the top of the loop.

Copyright © 2021, ABES Engineering College

References

- https://docs.python.org/3/tutorial/controlflow.html
- Think Python: An Introduction to Software Design, Book by Allen B. Downey
- Head First Python, 2nd Edition, by Paul Barry
- Python Basics: A Practical Introduction to Python, by David Amos, Dan Bader, Joanna Jablonski, Fletcher Heisler
- https://fresh2refresh.com/python-tutorial/python-jump-statements/
- https://tutorialsclass.com/python-jump-statements/

Copyright © 2021, ABES Engineering College

Thank You

Copyright © 2021, ABES Engineering College