## PROGRAM No. 5

## Aim:

# Write Python program to demonstrate use of looping statements: ‘while’ loop, ‘for’ loop

# and Nested loop

## Theory:

While developing a computer application one need to identify all possible situation. One situation is to repeat execution of certain code block/ line of code. Such situation can be taken care by looping system. Like all other high level programming language, Python also supports all loop structure. To keep a computer doing useful work we need repetition, looping back over the same block of code again and again. This practical will describe the different kinds of loops in Python.

A loop statement allows us to execute a statement or group of statements multiple times. Python programming language provides following types of loops to handle looping requirements.

1. **while loop**: A while loop statement in Python programming language repeatedly executes a target statement as long as a given condition is true.

## Syntax

**while expression:**

**statement(s)**

Here, statement(s) may be a single statement or a block of statements. The condition may be any expression, and true is any non-zero value. The loop iterates while the condition is true. When the condition becomes false, program control passes to the line immediately following the loop.

**Example** #!/usr/bin/Python count = 0

while (count < 5):

print 'The count is ', count

count = count + 1

print "Good bye!"

## Output:

The count is 0 The count is 1 The count is 2 The count is 3 The count is 4 Good bye!

1. **for loop:** It has the ability to iterate over the items of any sequence, such as a list or a string.

## Syntax

for iterating\_ var in sequence: statements(s)

If a sequence contains an expression list, it is evaluated first. Then, the first item in the sequence is assigned to the iterating variable iterating\_var. Next, the statements block is executed. Each item in the list is assigned to iterating\_var, and the statement(s) block is executed until the entire sequence is exhausted.

**Example**

#!/usr/bin/Python

for letter in 'Python': # First Example print 'Current Letter :', letter

fruits = ['banana', 'apple', 'mango']

for fruit in fruits: # Second Example

print 'Current fruit :', fruit

print "Good bye!"

**Output:**

Current Letter : P Current Letter : y

Current Letter : t

Current Letter : h Current Letter : o

Current Letter : n

Current fruit : banana

Current fruit : apple

Current fruit : mango

Good bye!

1. **nested loops:** Python programming language allows to use one loop inside another loop. Following section shows few examples to illustrate the concept.

## Syntax

for iterating\_var in sequence:

for iterating\_var in sequence:

statements(s) statements(s)

The syntax for a nested while loop statement in Python programming language is as follows –

while expression:

**while expression:** statement(s) statement(s)

A final note on loop nesting is that you can put any type of loop inside of any other type of loop. For example a for loop can be inside a while loop or vice versa.

**Example**

The following program uses a nested for loop to find the prime numbers from 2 to 20

#!/usr/bin/Python

i = 2

while(i < 20):

j = 2

while(j <= (i/j)):

if not(i%j): break j = j + 1

if (j > i/j) : print i, " is prime" i = i + 1

print "Good bye!"

## Output:

2 is prime

3 is prime

5 is prime

7 is prime

11 is prime

13 is prime

17 is prime

19 is prime

Good bye!

### **Example1**: Print first 10 numbers using a for loop

### for num in range(10):

### print(num)

### Example2:

## # **Program to find the sum of all numbers stored in a list**

## # List of numbers

## numbers = [6, 5, 3, 8, 4, 2, 5, 4, 11]

## # variable to store the sum

## sum = 0

## # iterate over the list

## for val in numbers:

## sum = sum+val

## print("The sum is", sum)

## Example 3:

## # **program to display student's marks from record**

## student\_name = 'Soyuj'

## marks = {'James': 90, 'Jules': 55, 'Arthur': 77}

## for student in marks:

## if student == student\_name:

## print(marks[student])

## break

## else:

## print('No entry with that name found.')

## Example4:

## # Use of break statement inside the loop

## for val in "string":

## if val == "i":

## break

## print(val)

## print("The end")

Example 5:**Write a Python program to construct the stars(\*) pattern, using a nested for loop**

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

\* \* \* \*

\* \* \*

\* \*

\*

'

n=5;

for i in range(n):

for j in range(i):

print ('\* ', end="")

print('')

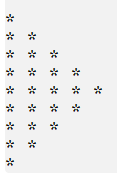
for i in range(n,0,-1):

for j in range(i):

print('\* ', end="")

print('')

## Output:



## Relevant Program Outcomes(POs):

* + **Engineering knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
  + **Problem analysis:** Apply Computer engineering discipline - specific knowledge to solve core computer engineering related problems
  + **Design/development of solutions:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.
  + **Individual and team work:** Function effectively as a leader and team member in diverse/ multidisciplinary teams.
  + **Life-long learning:** Engage in independent and life-long learning activities in the context of technological changes in the Computer engineering field and allied industry.

**CONCLUSION:**

|  |  |
| --- | --- |
| **COURSE OUTCOME** | **PROGRAM OUTCOME Attained** |
| **CO2** | **PO(1,2,3,9,12)** |

## PROGRAM No. 6

## Aim:

# Write Python program to perform following operations on Lists: Create list, Access list, Update list (Add item, Remove item), Delete list

## Theory:

A list can be defined as a collection of values or items of different types. The items in the list are separated with the comma (,) and enclosed with the square brackets []. The elements or items in a list can be accessed by their positions i.e. indices. This practical will make student acquainted with use of list and operations on list in Python.

* + A list is a collection which is ordered and changeable.
  + A list is a collection of items or elements; the sequence of data in a list is ordered.
  + The elements or items in a list can be accessed by their positions i.e. indices
  + In Python lists are written with square brackets. A list is created by placing all the items (elements) inside a square brackets [ ], separated by commas.
  + Lists are mutable. The value of any element inside the list can be changed at any point of time.
  + The index always starts with 0 and ends with n-1, if the list contains n elements.

1. **Creating List:** Creating a list is as simple as putting different comma-separated values between square brackets.

## Example:

>>>List1=[‘Java’,’Python’,’Perl’]

>>>List2=[10,20,30,40,50]

1. **Accessing List:** To access values in lists, use the square brackets for slicing along with the index or indices to obtain value available at that index.

Example:

>>>List2 [10,20,30,40,50]

>>>List2[1] 20

>>>List2[1:3]

[20,30]

>>>List2[5]

Traceback (most recent call last):

File "<pyshell#71>", line 1, in <module> List2[5]

IndexError: list index out of range

1. **Updating List:** You can update single or multiple elements of lists by giving the slice on the left-hand side of the assignment operator, and you can add to elements in a list with the append() method.

>>>List2 [10,20,30,40,50]

>>>List2[0]=60

#Updating first item [60,20,30,40,50]

>>>List2[3:4]=70,80

>>>[60,20,30,70,80,50]

## We can add one item to a list using append() method or add several items using extend() method.

>>> list1=[10,20,30]

>>> list1

[10, 20, 30]

>>> list1.append(40)

>>> list1

[10, 20, 30, 40]

>>> list1.extend([60,70])

>>> list1

[10, 20, 30, 40, 60, 70]

## We can also use + operator to combine two lists. This is also called concatenation

>>> list1=[10,20,30]

>>> list1

[10, 20, 30]

>>> list1+[40,50,60]

[10, 20, 30, 40, 50, 60]

iii) The \* operator repeats a list for the given number of times.

>>> list2 ['A', 'B']

>>> list2 \*2

['A', 'B', 'A', 'B']

## We can insert one item at a desired location by using the method insert()

>>> list1

[10, 20]

>>> list1.insert(1,30)

>>> list1

[10, 30, 20]

1. Deleting List: To remove a list element, you can use either the del statement if you know exactly which element(s) you are deleting or the remove() method if you do not know.
2. Del Operator: We can delete one or more items from a list using the keyword del. It can even delete the list entirely. But it does not store the value for further use.

**Example:**

>>> list=[10,20,30,40,50]

>>> del list[2]

>>> list

[10, 20, 40, 50]

1. Remove Operator: We use the remove operator if we know the item that we want to remove or delete from the list (but not the index)

## Example:

>>> list=[10,20,30,40,50]

>>> list.remove(30)

>>> list

[10, 20, 40, 50]

**Write a program to create, append, and remove lists in python.**

## Source Code:

pets = ['cat', 'dog', 'rat', 'pig', 'tiger']

snakes=['python','anaconda','fish','cobra','mamba']

print("Pets are :",pets)

print("Snakes are :",snakes)

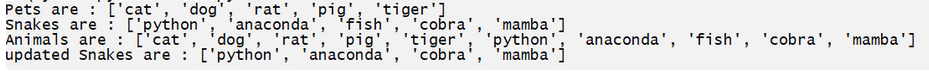
animals=pets+snakes

print("Animals are :",animals)

snakes.remove("fish")

print("updated Snakes are :",snakes)

Output:



## Relevant Program Outcomes(POs):

* + **Engineering knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
  + **Problem analysis:** Apply Computer engineering discipline - specific knowledge to solve core computer engineering related problems
  + **Design/development of solutions:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.
  + **Individual and team work:** Function effectively as a leader and team member in diverse/ multidisciplinary teams.
  + **Life-long learning:** Engage in independent and life-long learning activities in the context of technological changes in the Computer engineering field and allied industry.

**CONCLUSION:**

|  |  |
| --- | --- |
| **COURSE OUTCOME** | **PROGRAM OUTCOME Attained** |
| **CO3** | **PO(1,2,3,4,9,12)** |

## Practical related Questions

1. When to used list
2. Describe various list functions.
3. Write syntax for a method to sort a list
4. Write syntax for a method to count occurrences of a list item in Python
5. How to concatenate list
6. Justify the statement “Lists are mutable”
7. Describe the use pop operator in list