Lab-02
To Get familiar with concepts in Python

Objectives:

- Learn the basic concepts
- Conditional Statements: If,Else,Elif Statement In Python
- Loops In Python (For Loop ,While Loop Nested Loops In Python

Apparatus:

- Hardware Requirement
  Personal computer.
- Software Requirement
  Anaconda.

Theory:

CONTROL STRUCTURES IF ELSE ELIF:

An else statement can be combined with an if statement. An else statement contains the block of code that executes if the conditional expression in the if statement resolves to 0 or a FALSE value.

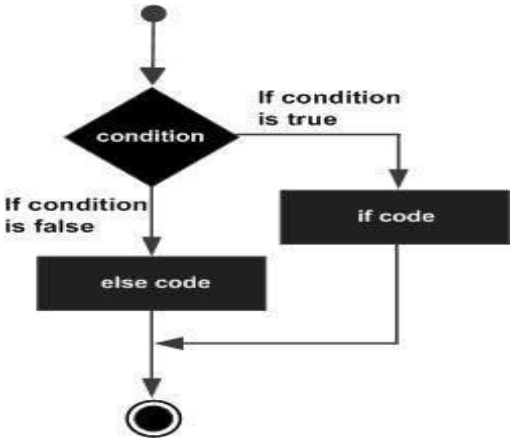
The else statement is an optional statement and there could be at most only one else statement following if.

Syntax

The syntax of the if...else statement is −

```
if expression:
   statement(s)
else:
   statement(s)
```

Flow Diagram:



The elif Statement

The elif statement allows you to check multiple expressions for TRUE and execute a block of code as soon as one of the conditions evaluates to TRUE.

Similar to the else, the elif statement is optional. However, unlike else, for which there can be at most one statement, there can be an arbitrary number of elif statements following an if.

syntax

```
if expression1:
   statement(s)
elif expression2:
   statement(s)
elif expression3:
   statement(s)
else:
   statement(s)
```

Core Python does not provide switch or case statements as in other languages, but we can use if..elif...statements to simulate switch case as follows −

Demo Code:

```
#!/usr/bin/python

var = 100
if var == 200:
   print "1 - Got a true expression value"
   print var
elif var == 150:
   print "2 - Got a true expression value"
   print var
elif var == 100:
```

```
   print "3 - Got a true expression value"
   print var
else:
   print "4 - Got a false expression value"
   print var

print "Good bye!"
```

Example: 01
Print square root of negative or positive number using if and operators.

```
In [2]: a = int(input("Enter a number and I'll get its square root: "))
        if (a > 0):
                print("The number you entered is greater than 0, so I can calculate it!")
                root = a ** (1/2)
                print("The square root of %d is %f" % (a, root))
        if (a <= 0):
                print("I can't calculate the square root of a negative number!")
        print("Thanks for the input!")
```

While we are looking at operators, there are another set used in writing conditions, these are called logical operators. There is an example of a logical operator.

```
if country == "England" or country == "england":
```

The logical operator used here is the word or. The interpreter evaluates the expressions either side of the **or** operator and if one or the other, or both expressions are true the result is true. There are several logical operators you can use in Python: **and, or, not.**
Conditions can be composed using two basic logical operators:

Operator	**Syntax**
Logical AND	And
Logical OR	Or

Example: 02
Write conditional statements to print value of 0 to 1 and 1 to 0 and numbers in between.

```
a = int(input()) # the variable is initialized with a value of 0

if (a == 1): # if the value is 1, we change its value to 0
        a = 0
if (a == 0): # if the value is 0, we change its value to 1
        a = 1
if (a >2 or a < 0):
       print("You have entered number between")
print(a)
```

```
0
1
```

Example: 03

Let's look at another example:

```
a = int(input("Enter a number between 10-20: "))
if a >= 10 and a <= 20:
        print("The condition has been met.")
else:
        print("You did it wrong.")
```

```
Enter a number between 10-20: 28
You did it wrong.
```

Example: 04

Of course, operators can also be combined using parentheses:

```
a = int(input("Enter a number between 10-20 or 30-40: "))
if (a >= 10 and a <= 20) or (a >= 30 and a <= 40):
        print("The condition has been met.")
else:
        print("You did it wrong.")
```

```
Enter a number between 10-20 or 30-40: 1
You did it wrong.
```

CONTROL STRUCTURES LOOP

In general, statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on. There may be a situation when you need to execute a block of code several number of times.

Programming languages provide various control structures that allow for more complicated execution paths.

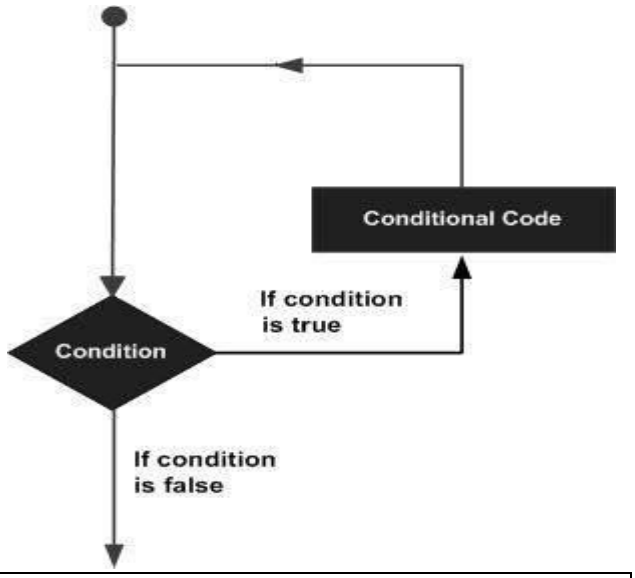
A loop statement allows us to execute a statement or group of statements multiple times.

For Loop:

syntax:

for iterating_var in sequence:

statements(s)



While - Loop

Syntax:

[1]initialization

[2]**while (Condition):**
    #body of the loop/statement goes here
    [3]**increment/decrement**

Example #1

Print Karachi Pakistan 100 times in a separate line

```
i=1
while(i<=100):
    print("Karachi Pakistan")
    i+=1
```

Another way to solve this problem with decrement operator

```
i=100
while(i>=1):
    print("Karachi Pakistan")
    i-=1
```

Example # 2

Take collection of number input from user. Print the total positive and negative number.

```
#counting positive and negative numbers
pcount=0
ncount=0
count=int(input("how many numbers you want? "))
i=1 #initialization
while(i<=count): #condition
    num=int(input("enter number "))
    if(num>=0):
        pcount+=1
    else:
        ncount+=1
    i+=1
print("positive: ",pcount)
print("negative: ",ncount)
```

Example # 3

Fixed a Letter from a to e and then ask the user to guess that letter until correct letter entered.

```
value='c'
userValue=input("guess a number from a to e    ")
while(userValue!=value):
    print("Incorrect")
    userValue=input("guess a number from a to e    ")
print("welcome user")
```

```
guess a number from a to e    a
Incorrect
guess a number from a to e    i
Incorrect
guess a number from a to e    b
Incorrect
guess a number from a to e    c
welcome user
```

LOOP TYPE AND DESCRIPTION

Sr.No.	Loop Type & Description
1	while loop Repeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body.
2	for loop Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.

3	nested loops You can use one or more loop inside any another while, for or do..while loop.

LAB EXERCISES

Exercise 1:

(I)Cabinets and Boxes are objects that are mostly in cubic shape. Make a program that takes inputs like height, width and depth from user and then calculate volume of the cube:
volume = height ∗ width ∗ depth
After calculating volume of cube, compare it with following ranges and print the relevant label:

Volume Range	Label
1 cm^3 – 10 cm^3	Extra Small
11 cm^3 – 25 cm^3	Small
26 cm^3 – 75 cm^3	Medium
76 cm^3 – 100 cm^3	Large
101 cm^3 – 250 cm^3	Extra Large
251 cm^3 and above	Extra-Extra Large

(II)In a company ,worker efficiency is determined on the basis of the time required for a worker to complete a particular job.If the time taken by the worker is between 2-3 hours then the worker is said to be highly efficient. If the time required by the worker is between 3-4hours,then the worker is ordered to improve speed. If the time taken is between 4-5 hours ,the worker is given training to improve his speed ,and if the time taken by the worker is more than 5 hours ,then the worker haas to leave the company, If the time taken by the worker is input through the keyboard,find the efficiency of the worker.

(iii)The program must prompt the user for a username and password. The program should compare the password given by the user to a known password. If the password matches, the program should display “Welcome!” If it doesn’t match, the program should display “I don’t know you.

Note: the password should not be case sensitive and it’s value is abc$123 or ABC$123

```
What is the password? 12345
I don't know you.

Or

What is the password? abc$123
Welcome!
```

What is the password? ABC$123
Welcome!

Exercise 2:
(i)What Would Python Print?

```
>>> n = 3
>>> while n >= 0:
...     n -= 1
...     print(n)
...
```

_____

The code block will continue to run until n becomes < 0, since 0 is not greater than or equal to 0.

(ii): What Would Python Print?

```
>>> # typing Ctrl-C will stop infinite loops
>>> n = 4
>>> while n > 0:
...     n += 1
...     print(n)
...
```

_____

Make sure your while loop condition eventually becomes false, or it'll never stop!

(ii)Try the scenrio below:

Make a program that lists the countries in the set

```
clist =
['Canada','USA','Mexico','Australia']
```

1. Create a loop that counts from 0 to 100
2. Make a multiplication table using a loop
3. Output the numbers 1 to 10 backwards using a loop
4. Create a loop that counts all even numbers to 10
5. Create a loop that sums the numbers from 100 to 200

(iii) Try the exercise below:

1. Make a program that lists the countries in the set below using a while loop.

```
clist =
["Canada","USA","Mexico"]
```

2. What's the difference between a while loop and a for loop?
3. Can you sum numbers in a while loop?
4. Can a for loop be used inside a while loop?

Exercise 2:
(i)What Would Python Print?

```
>>> n = 3
>>> while n >= 0:
...     n -= 1
...     print(n)
...
```

_____
The code block will continue to run until n becomes < 0, since 0 is not greater than or equal to 0.
(ii): What Would Python Print?

```
>>> # typing Ctrl-C will stop infinite loops
>>> n = 4
>>> while n > 0:
...     n += 1
...     print(n)
...
```

_____
Make sure your while loop condition eventually becomes false, or it'll never stop!
(ii)Try the scenrio below:
Make a program that lists the countries in the set

```
clist =
['Canada','USA','Mexico','Australia']
```

1. Create a loop that counts from 0 to 100
2. Make a multiplication table using a loop
3. Output the numbers 1 to 10 backwards using a loop
4. Create a loop that counts all even numbers to 10
5. Create a loop that sums the numbers from 100 to 200

(iii) Try the exercise below:
1. Make a program that lists the countries in the set below using a while loop.

```
clist =
["Canada","USA","Mexico"]
```

2. What's the difference between a while loop and a for loop?
3. Can you sum numbers in a while loop?
4. Can a for loop be used inside a while loop?