K. J. Somaiya College of Engineering, Mumbai-77

Batch:C7-2	**Roll No.:27**
Experiment / assignment / tutorial No.	
Grade: AA / AB / BB / BC / CC / CD /DD	
Signature of the Staff In-charge with date	

TITLE: Write a program to demonstrate the use of Decision Making Statements in Python

AIM: 1) Write a program to read the numbers until -1 is encountered. Also, count the number of prime and composite numbers entered by the user.
2) Write a program to check whether a given number is Armstrong.

OUTCOME: Students will be able to

CO1: Formulate a problem statement and develop the logic (algorithm/flowchart) for its solution.

CO3: Use different Decision-Making statements and Functions in Python.

Use of input-output function, Use different Decision-Making statements in Python.

Resource Needed: Python IDE

Books/ Journals/ Websites referred:

1. Reema Thareja, *Python Programming: Using Problem-Solving Approach*, Oxford University Press, First Edition 2017, India
2. Sheetal Taneja and Naveen Kumar, *Python Programming: A modular Approach*, Pearson India, Second Edition 2018, India
3. https://www.geeksforgeeks.org/python-strings/?ref=lbp

Theory:

Decision Control Statements

1) Selection/Conditional branching statements

a) if statement
b) if-else statement
c) if-elif-else statement

2)Basic loop Structures/Iterative statement

a) while loop
b) for loop

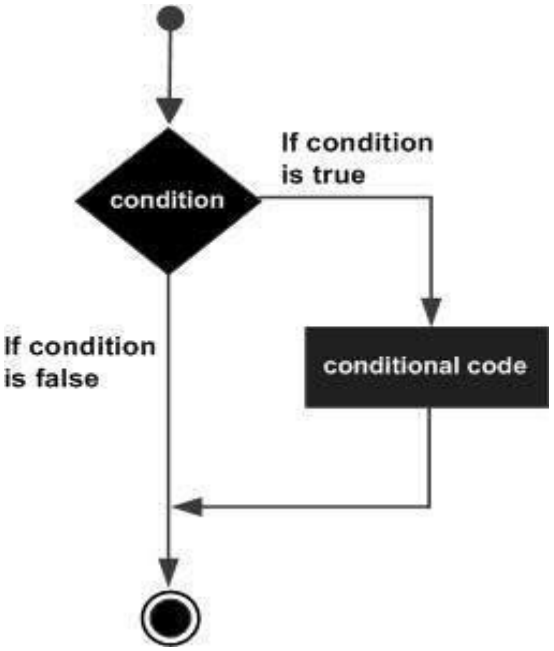
If statement:

In Python **if** statement is used for decision-making operations. It contains a body of code that runs only when the condition given in the **if** statement is true.

Syntax:

```
if condition:
   statement(s)
```

If flowchart:



If-else Statement:

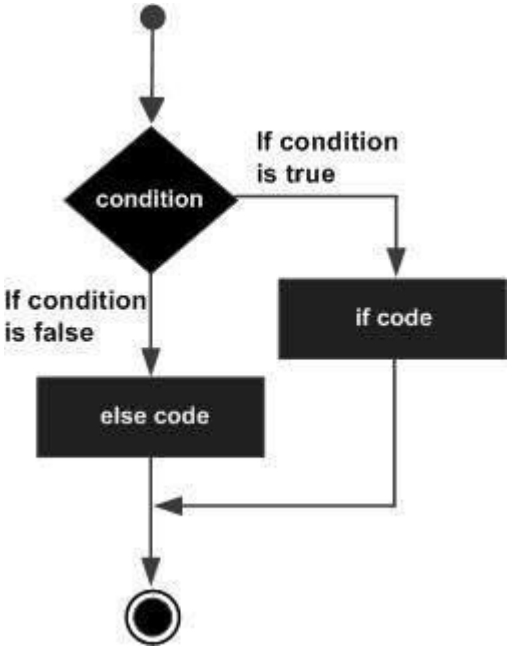
An **else** statement can be combined with an **if** statement. An **else** statement contains the block of code that executes if the conditional expression in the **if** statement resolves to 0 or a FALSE value.

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

Syntax:

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

If-else flowchart:



If-elif-else Statement:

The **elif** statement allows you to check multiple expressions for TRUE and execute a code block 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)
```

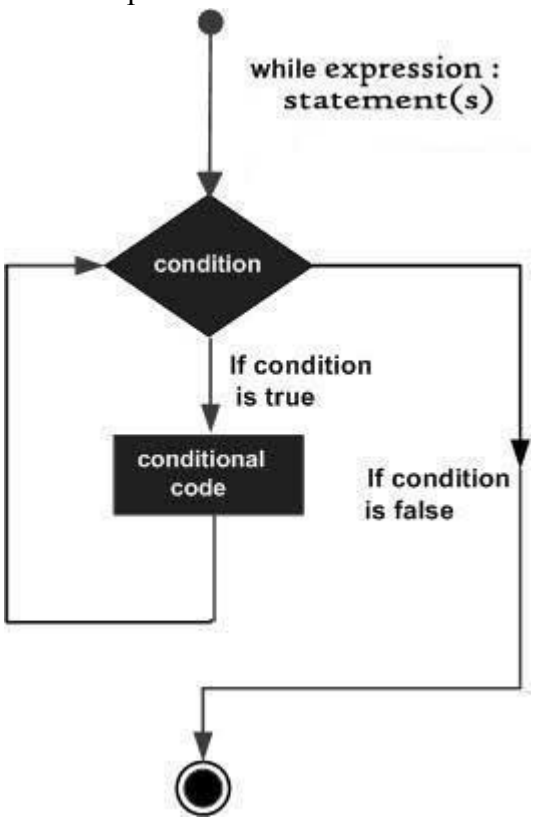
While loop:

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

Syntax:

```
while expression:
   statement(s)
```

While loop flowchart:



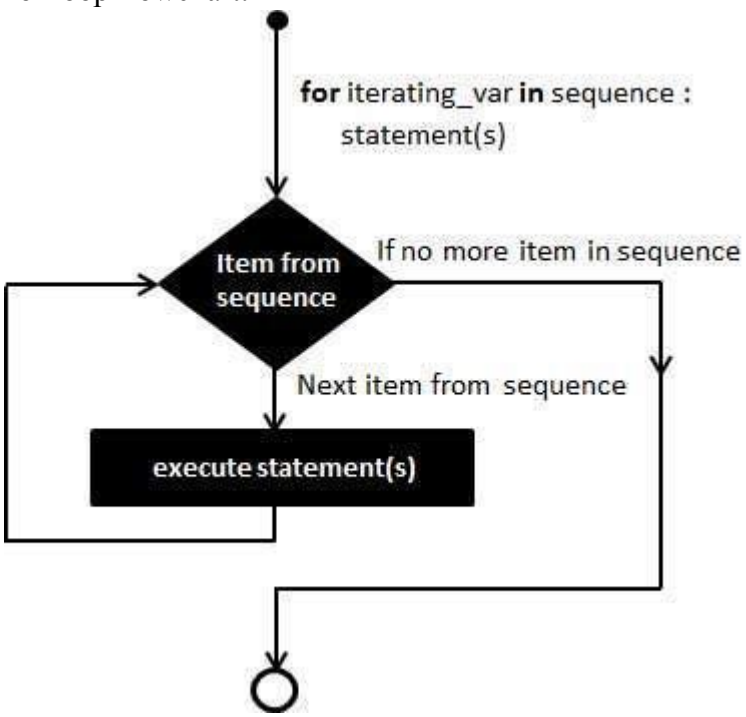
For Loop:

The **for** statement in Python differs a bit from what you may be used to in C. Rather than giving the user the ability to define both the iteration step and halting condition (as C), Python's **for** statement iterates over the items of any sequence (a list or a string), in the order that they appear in the sequence.

Syntax:

```
for iterating_var in sequence:
   statements(s)
```

For loop flowchart:



Problem Definition:

1) Write a program to read the numbers until -1 is encountered. Also, count the number of prime and composite numbers entered by the user.

2) Write a program to check whether a number is Armstrong or not.
(An Armstrong number is a number that is equal to the sum of cubes of its digits; for example, 153 = 1^3 + 5^3 + 3^3.)

Implementation details:

1) Write a program to read the numbers until -1 is encountered. Also, count the number of prime and composite numbers entered by the user.

```
print("enter numbers")
num=0
pcount=0
ccount=0

while(num!=-1):
    num=int(input())
    if(num==-1):
        break
    for x in range(2,num):
        if(num%x==0):
            ccount+=1
            break
    else:
        pcount+=1

print("prime count: ",pcount)
print("composite count ", ccount)
```

2) Write a program to check whether a number is Armstrong or not.
(An Armstrong number is a number that is equal to the sum of cubes of its digits; for example, 153 = 1^3 + 5^3 + 3^3.)

```
num=int(input("enter a number"))
on=num
str=str(num)
count=len(str)
sum=0
while(num>0):
    rem=num%10
    pwr=rem**count
    sum=sum+pwr
    num=num//10
if(on==sum):
    print(on, "is an armstrong number")
else:
    print(on, "is not an armstrong number")
```

Output(s):

1)

```
enter numbers
15
11
13
17
-1
prime count:  3
composite count  1
```

2)

```
enter a number153
153 is an armstrong number
```

Conclusion:

Post Lab Questions:

1) When should we use nested if statements? Illustrate your answer with an example.
2) Explain the utility of break and continue statements with the help of an example.
3) Write a program that accepts a string from the user and calculates the number of digits and letters in the string.