Batch: G3	**Roll No.:** 16010421063
Experiment / assignment / tutorial No.	
Grade: AA / AB / BB / BC / CC / CD /DD	
Signature of the Staff In-charge with date	

TITLE: Decision Making Statements

AIM: 1) Write a program to count the number of prime numbers and composite numbers entered by the user.
2) Write a program to check whether a given number is Armstrong or not.

Expected OUTCOME of Experiment: Use different Decision Making statements in Python.

Resource Needed: Python IDE

Theory:

Decision Control Statements

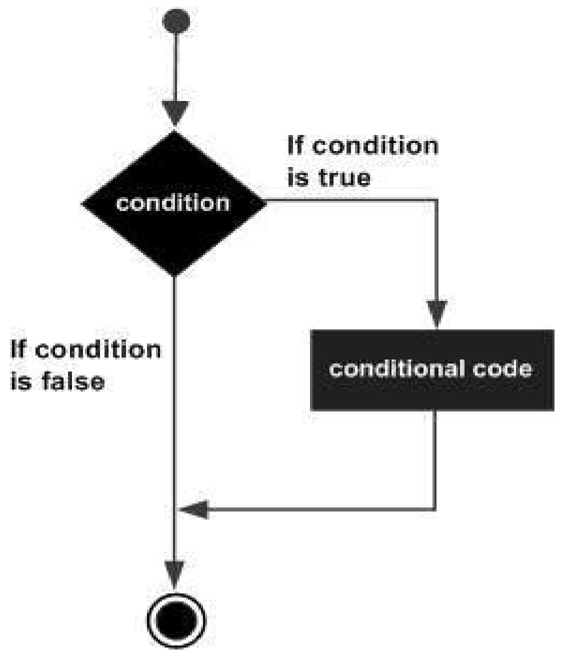
1) **Selection/Conditional branching statements**
   a) if statement
   b) if-else statement
   c) if-elif-else statement
2) **Basic loop Structures/Iterative statement**
   a) while loop
   b) for loop

If statement:

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

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

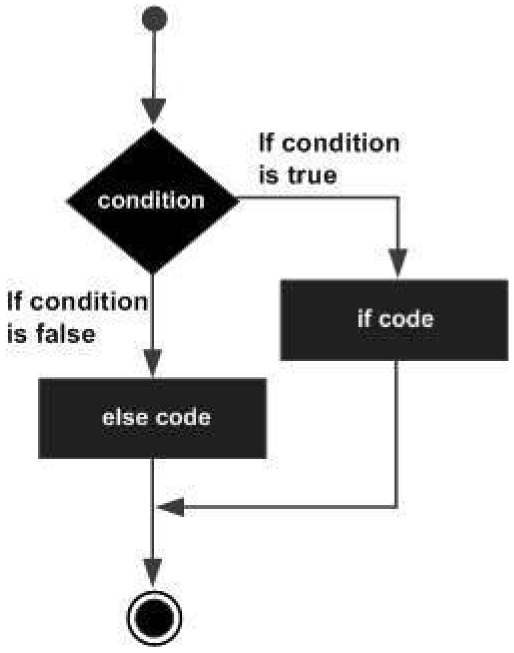
If flowchart:



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:

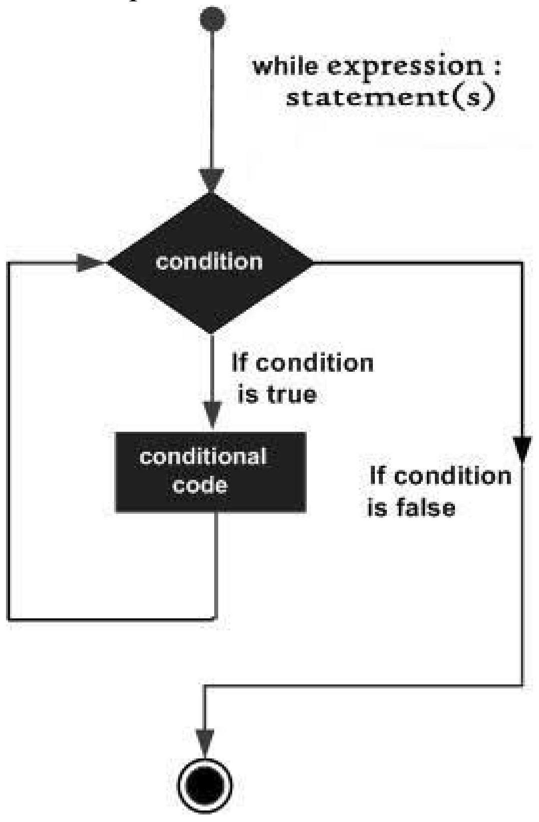
The **elif** statement allows you to check multiple expressions for TRUE and execute a block of code as soon as one of the conditions evaluates to TRUE.
Similar to the else, the **elif** statement is optional. However, unlike **else**, for which there can be at most one statement, there can be an arbitrary number of **elif** statements following an **if.**

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

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)
```

While loop flowchart:

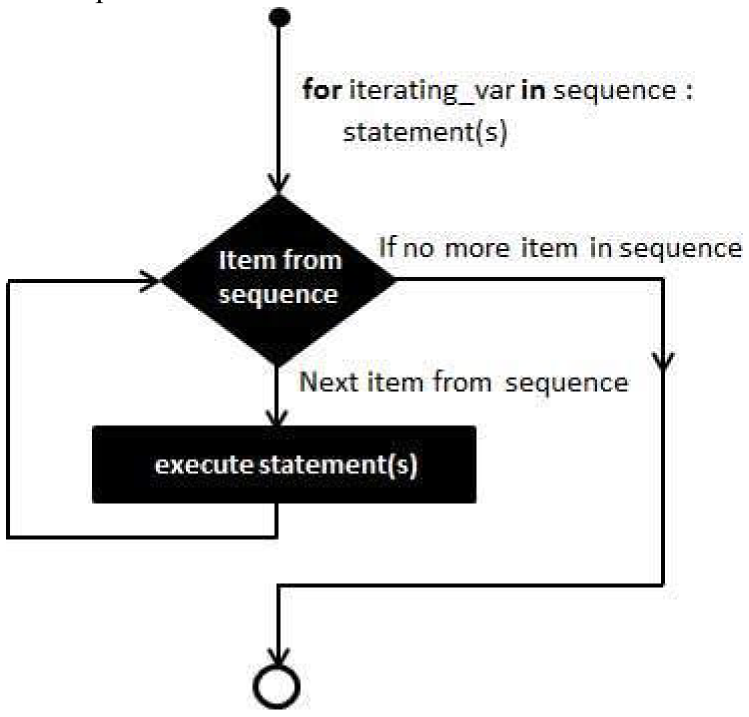


For Loop:
The **for** statement in Python differs a bit from what you may be used to in C. Rather than giving the user the ability to define both the iteration step and halting condition (as C), Python's **for** statement iterates over the items of any sequence (a list or a string), in the order that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

2) Write a program to check whether a number is Armstrong or not.
(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.)

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://docs.python.org/3/tutorial/controlflow.html#for-statements

---

Implementation details:

1.

```python
def checkPrime(num):
    for i in range(2, num):
        if num % i == 0:
            return False
    return True


num=0
prime_count=0
composite_count=0

while num!=-1:
    num=int(input())
    if num==-1:
        pass

    elif checkPrime(num):
        prime_count+=1
    else:
        composite_count+=1

print("Prime: ",prime_count)
print("Composite: ",composite_count)
```

2.

```python
num=input()
```

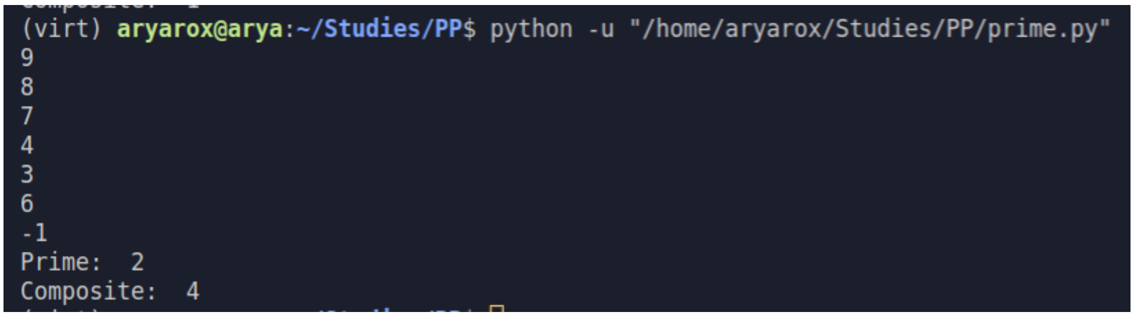
```
l=len(num)
sum=0

for i in num:
   sum=sum+ int(i)**l

if sum==int(num):
   print("It is armstrong")
else:
   print("It is not armstrong")
```

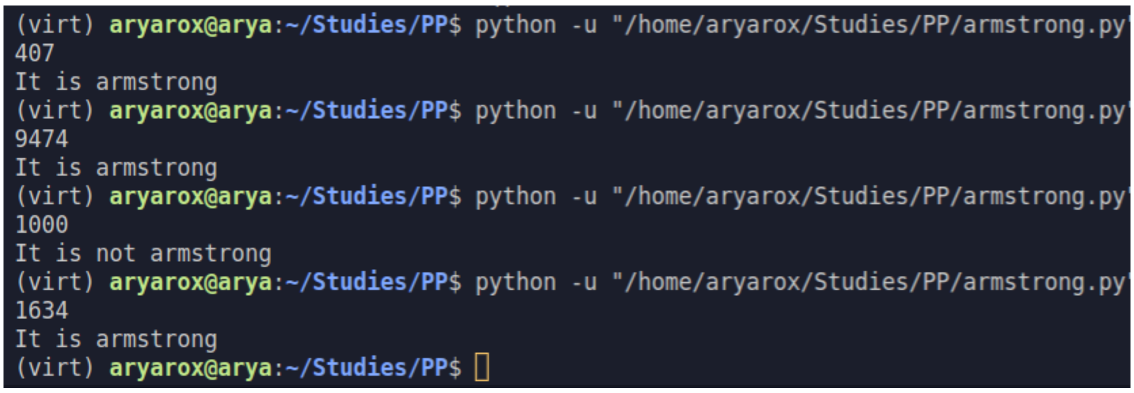
Output(s):

1.



```
(virt) aryarox@arya:~/Studies/PP$ python -u "/home/aryarox/Studies/PP/prime.py"
9
8
7
4
3
6
-1
Prime:  2
Composite:  4
```

2.



```
(virt) aryarox@arya:~/Studies/PP$ python -u "/home/aryarox/Studies/PP/armstrong.py
407
It is armstrong
(virt) aryarox@arya:~/Studies/PP$ python -u "/home/aryarox/Studies/PP/armstrong.py
9474
It is armstrong
(virt) aryarox@arya:~/Studies/PP$ python -u "/home/aryarox/Studies/PP/armstrong.py
1000
It is not armstrong
(virt) aryarox@arya:~/Studies/PP$ python -u "/home/aryarox/Studies/PP/armstrong.py
1634
It is armstrong
(virt) aryarox@arya:~/Studies/PP$ 
```

Conclusion:
Nested if statement were executed and also use of function was understood and a user

defined function was executed for binary search.

Post Lab Questions:

1) When should we use nested if statements? Illustrate your answer with the help of an example.

When we are given if more than one condition that needs to be checked and all condition returning different values.
Eg: Grade of student according to his/her marks

```
if(marks>=75):
   print("Excellent")
elif (marks>50 and marks<75):
   print("good")
elif(marks<50):
   print("could be better")
```

2) Explain the utility of break and continue statements with the help of an example.

Break statement ends the loop when the condition given is true
Eg:

```
for i in range(n):
        If(i>10):
                break
```

Continue statement skips the current iteration of the loop and continues with next Iteration if condition is true
Eg.

```
for i in range(n):
        If(i == 10):
                continue
```

3) Write a program that accepts a string from user and calculate the number of digits and letters in string.

Truck/Bus- 20 Rs. per hour
Car- 10 Rs. per hour
Scooter/Cycle/Motor cycle- 5 Rs. per hour

```
a=input("Give type of vehicle")
b=float(input("enter number of hours"))
if(a==' t' or a==' b'):
   pay=20*b
   print("Total pay=",pay)
else:
   pay=10*b
   print("Total pay=",pay)
```

Date: ______________ **Signature of faculty in-charge**