Batch:	**A3**	**Roll No.:**	**16010121051**

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

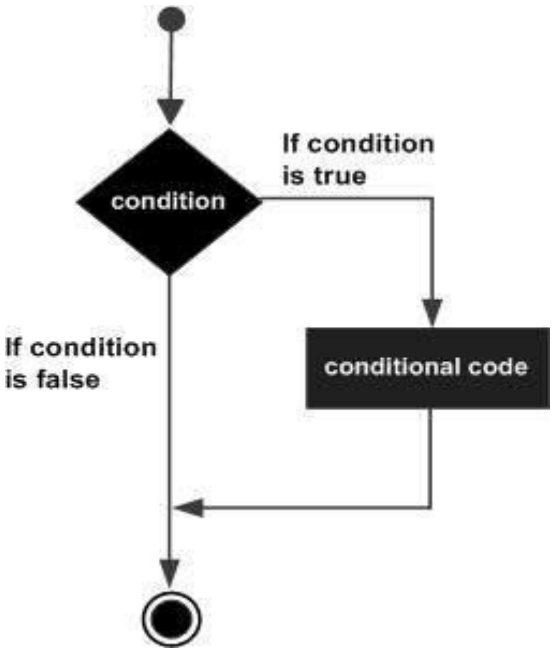
2)Basic loop Structures/Iterative statement
- a) while loop
- b) for loop

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:

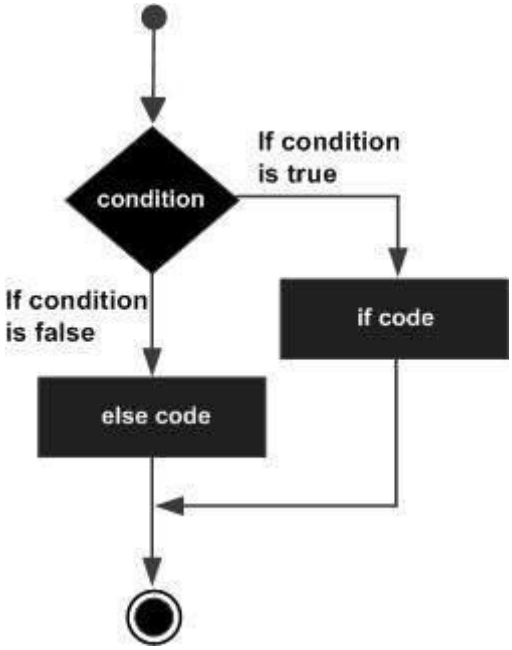
An **else** statement can be combined with an **if** statement. An **else** statement contains the block of code that executes if the conditional expression in the **if** statement resolves to 0 or a FALSE value.

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)
```

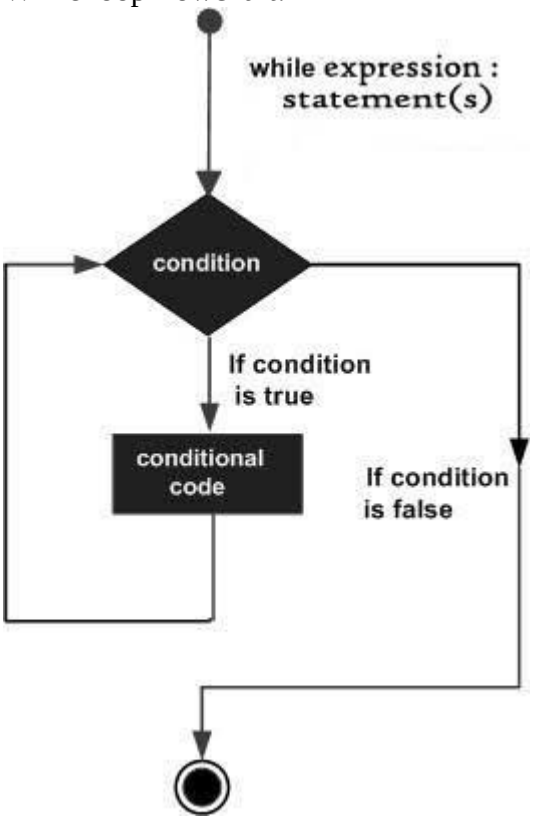
While loop:

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:



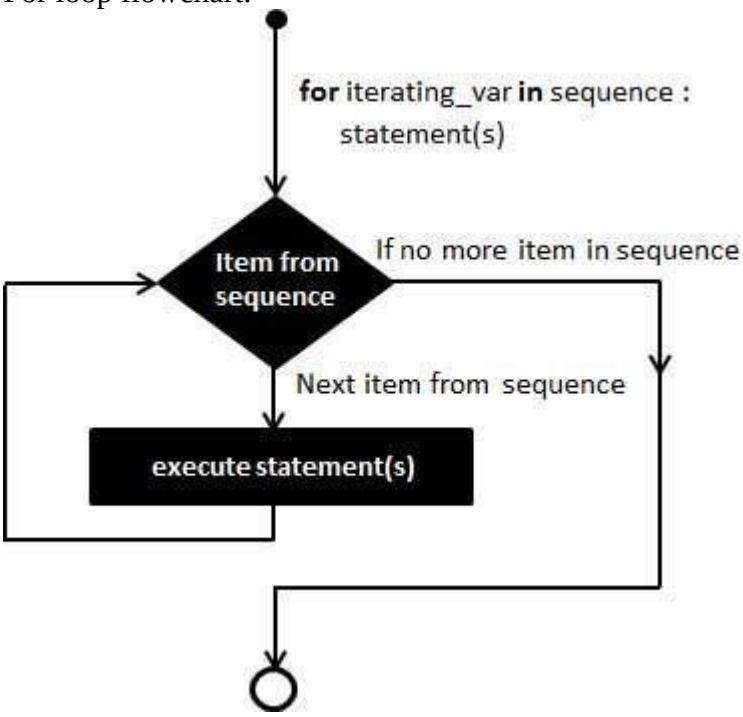
For Loop:

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)

```
#  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

# initial count (defining variables)
count = 0
com_count = 0
prm_count = 0
while True:
    num =int(input("Enter a number\n"))
    # condition for -1
    if num == -1:
        print("Break")
        break
    elif num == 1:
        print("neither composite nor prime")
        # 1 being exception
    elif num == 2:
        prm_count += 1
        print("2 is a prime number")
    else:
        for a in range(2,num):
            # prime checking
            if num%a==0:
                count+=1
        if count>=1:
            # else composite
            print(num, "is Composite Number")
            com_count += 1
        else:
            print(num, "is Prime Number")
            prm_count += 1

# printing final count
print("No of composite numbers till now are: ",com_count)
print("No of prime numbers till now are: ",prm_count)
```

2)

```python
# Write a program to check whether a number is Armstrong or not.

# taking input from user
number = input("enter number\n ")
# length
dig = len(number)
sum = 0

for i in range(dig):
    # condition for number to be armstrong
    sum += int(number[i])**3

if int(number) == sum:
    print("yes")

else :
    print("no")
```

Output(s):

1)

```
PS C:\Academics\SEM2\PP> & C:/Users/Lenovo/AppData/Local/Programs/Python/Python310/python.exe c:/Academics/SEM2/PP/PythonProgramming/exp3.py
Enter a number
1
neither composite nor prime
Enter a number
2
2 is a prime number
Enter a number
3
3 is Prime Number
Enter a number
4
4 is Composite Number
Enter a number
5
5 is Composite Number
Enter a number
6
6 is Composite Number
Enter a number
7
7 is Composite Number
Enter a number
-2
-2 is Composite Number
Enter a number
-1
Break
No of composite numbers till now are:  5
No of prime numbers till now are:  2
PS C:\Academics\SEM2\PP>
```

2)

```
PS C:\Academics\SEM2\PP> & C:/Users/Lenovo/AppData/Local/Programs/Python/Python310/python.exe c:/Academics/SEM2/PP/PythonProgramming/exp3b.py
enter number
 153
yes
PS C:\Academics\SEM2\PP> & C:/Users/Lenovo/AppData/Local/Programs/Python/Python310/python.exe c:/Academics/SEM2/PP/PythonProgramming/exp3b.py
enter number
 234
no
PS C:\Academics\SEM2\PP>
```

Conclusion:

Different Decision Making statements in Python were used successfully.

Post Lab Questions:

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

Ans :- There come situations in real life when we need to make some decisions and based on these decisions, we decide what should we do next. Similar situations arise in programming also where we need to make some decisions and based on these decisions, we will execute the next block of code.

Example:-

```
num = 15
if num >= 0:
        if num == 0:
                print("Zero")
        else:
                print ("Positive number")
else:
        print ("Negative number")
```

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

Ans :- Break Statement breaks the execution and exits the loop.

```
        n = 123.456
sum_digits = 0
for element in str(n):
                try:
            sum_digits += int(element)
                    except:
                break
print(sum_digits)
```

Continue Statement Skips The Remaining Code & Jumps To The Next Iteration.

for example: -

```
phone= ["mobile", "walkie-talkie", "landline"]
for x in phone:
if (x== "walkie-talkie"):
continue
print(x)
```

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

Implementation :-

```
data = input()
d=0
a =0
for x in data :
    if x.isdigit():
        d=d+1
    elif x.isalpha():
        a=a+1
    else:
        pass
print("Letters", a)
print("Digits", d)
```

output :-

```
PS C:\Academics\SEM2\PP> & C:/Users/Lenovo/AppData/Local/Programs/Python/Python310/python.exe c:/Academics/SEM2/PP/PythonProgramming/exp3c.py
3m2s1n88b67brb
Letters 7
Digits 7
PS C:\Academics\SEM2\PP>
```

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