Batch: C5_1	**Roll No.: 19**
Experiment / assignment / tutorial No. 3	
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

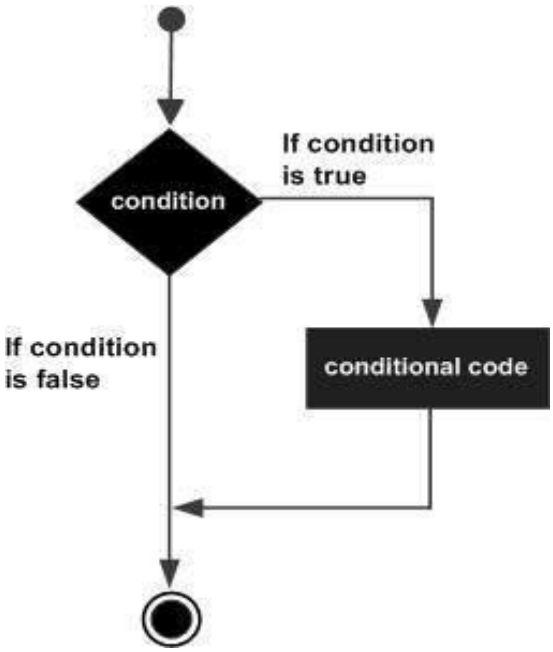
2)Basic loop Structures/Iterative statement
- a) while loop
- b) for loop

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:

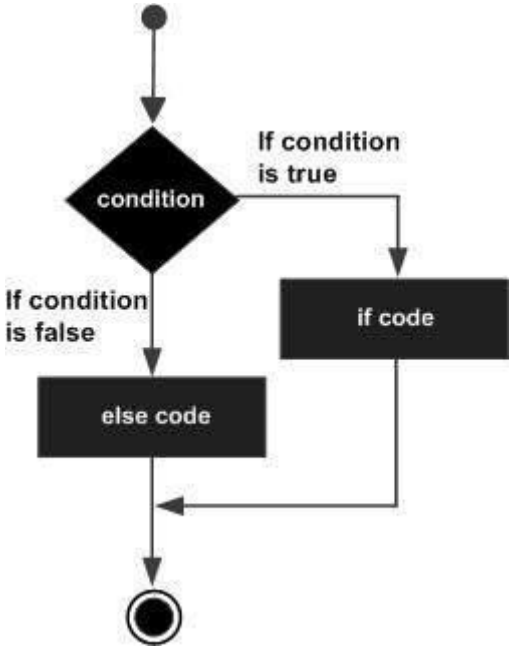
An **else** statement can be combined with an **if** statement. An **else** statement contains the block of code that executes if the conditional expression in the **if** statement resolves to 0 or a FALSE value.

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)
```

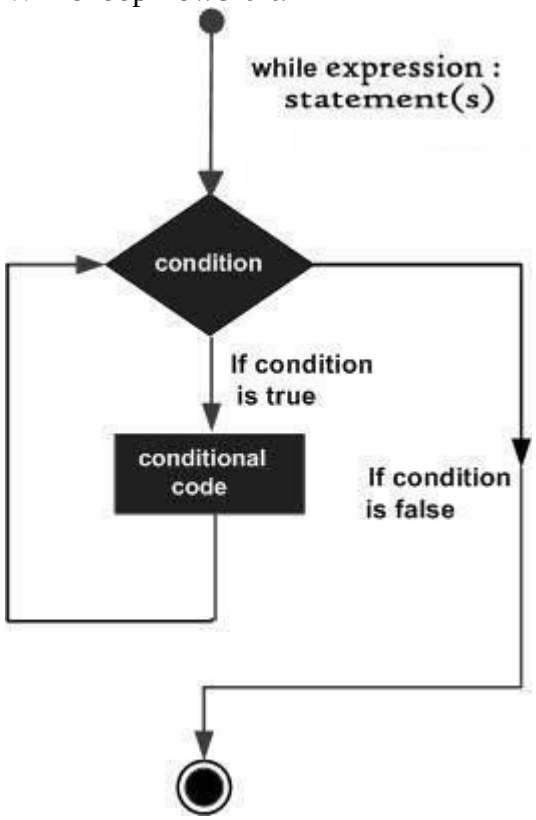
While loop:

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:



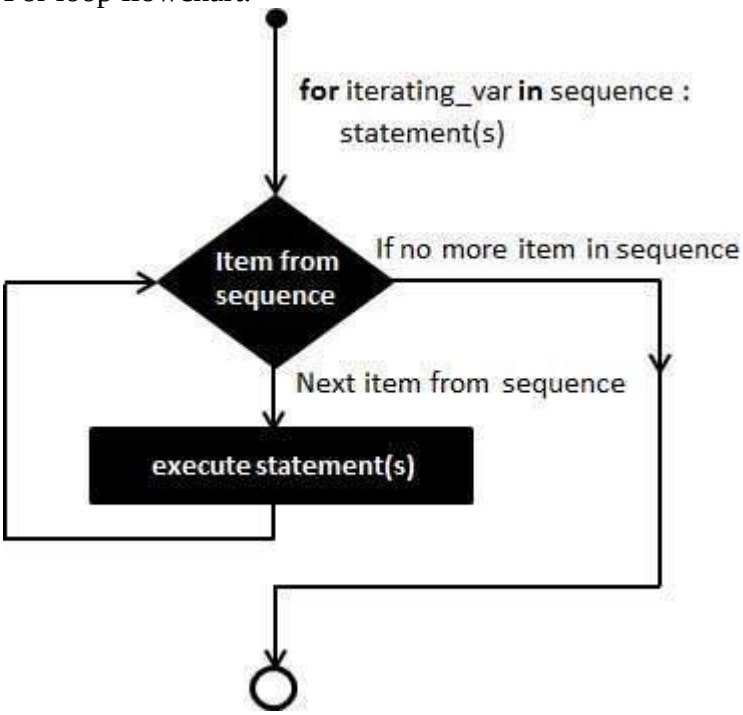
For Loop:

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:

```
#Q.1
n = int(input("Enter a number: "))
prime_no = 0
composite_no = 0
while(n!=-1):
    no_of_factors = 0
    for i in range(1,n+1):
        if(n%i==0):
            no_of_factors+=1
    if(no_of_factors==2):
        prime_no += 1
    else:
        composite_no += 1
    n = int(input("Enter a number: "))
print("Number of prime numbers:",prime_no)
print("Number of composite numbers:",composite_no)
```

```
#Q.2

def is_armstrong_number(number):
    num_digits = len(number)
    number = int(number)
    a = number
    sum =0
    while(a>0):
        t = a%10
        sum+=t**num_digits
        a//=10
    if(sum == number):
        return True
```

```
    else:
        return False

number = input("Enter a number: ")
if is_armstrong_number(number):
    print(number,"is an Armstrong number.")
else:
    print(number," is not an Armstrong number.")
```

Output(s):

Q.1

Enter a number: 3
Enter a number: 67
Enter a number: 334
Enter a number: 12
Enter a number: 888
Enter a number: 64
Enter a number: 3
Enter a number: 25
Enter a number: 0
Enter a number: -1
Number of prime numbers: 3
Number of composite numbers: 6

Q.2

Enter a number: 153
153 is an Armstrong number.

Conclusion:

In this experiment I've done the two given question using different decision making statement in Python.

.

Post Lab Questions:

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

Ans: Each "if" statement is nested inside another "if" or "else" statement. This allows you to create a more complex decision-making structure. Example-

```
marks = float(input("Enter your total marks: "))
if marks>=90:
    grade = 'A'
    if marks>=96:
        grade+='+'
elif marks>=80:
    grade = 'B'
    if marks>=85:
        grade+='+'
else:
    grade = 'C'

print("The grade at",marks,"marks is",grade)
```

2) Explain the utility of break and continue statements with the help of an example.
Ans: "break" is a statement that exits a loop when a certain condition is met. For example, it can be used to stop iterating through a list once a specific element is found.
"continue" is a statement that skips the loop's code for the current iteration and proceeds to the next iteration. Both break and continue enhance the control flow within loops in the programming.

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

```
#Write a program that accepts a string from user and calculate the number
of digits and letters in string.
str1 = (input("Enter a string: "))
dc = 0
lc = 0
for i in str1:
    if i.isdigit():
        dc+=1
    elif i.isalpha():
        lc+=1
print("No of digits: ",dc)
print("No of letters: ",lc)
```

OUTPUT:

```
Enter a string: 12 af 44 hrg6
No of digits:  5
```

Date: 4/10/2023 **Signature of faculty in-charge**