Batch: C-7(3) **Roll No : 58**

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

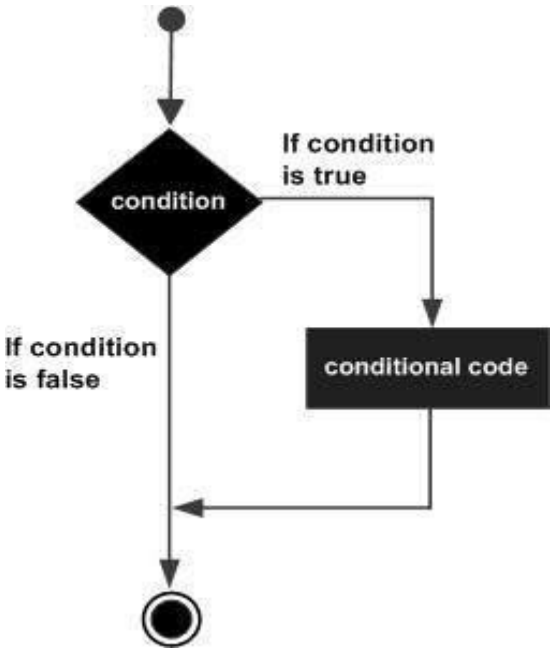
2)Basic loop Structures/Iterative statement
- a) while loop
- b) for loop

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:

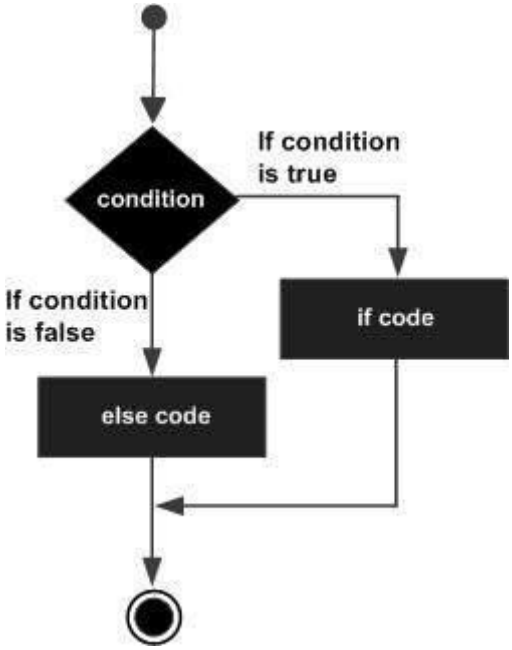
An **else** statement can be combined with an **if** statement. An **else** statement contains the block of code that executes if the conditional expression in the **if** statement resolves to 0 or a FALSE value.

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)
```

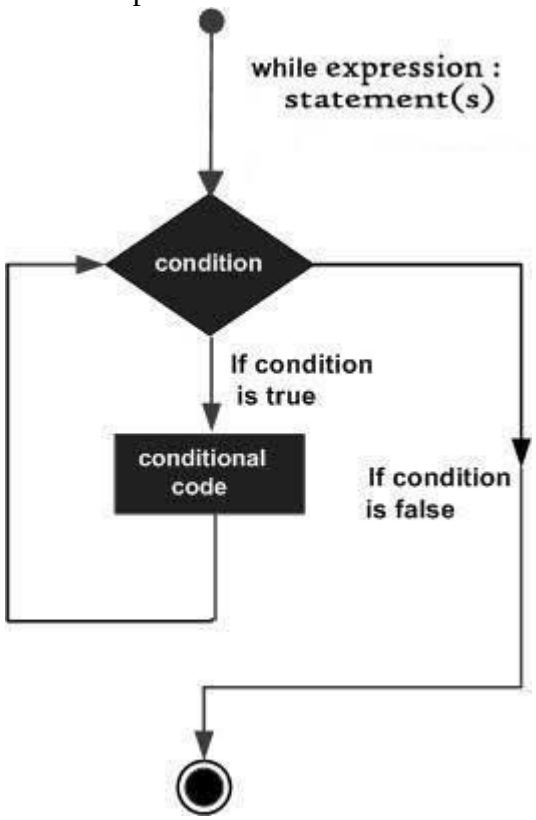
While loop:

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:



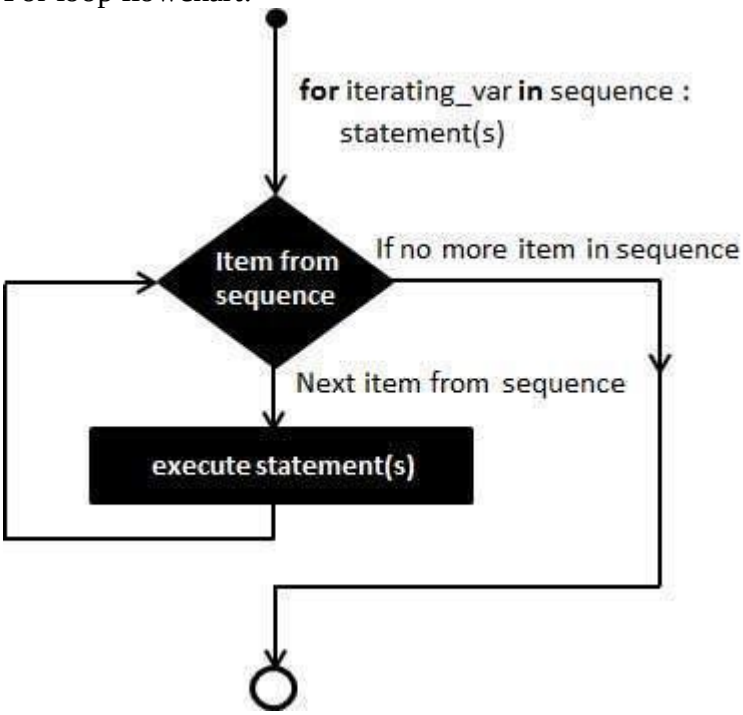
For Loop:

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:

```python
# Question 1
def is_prime(n):                                          # Defining a function
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

def main():                                               # Defining a function
    prime_count = 0                                       # Initialising the count
    composite_count = 0

    while True:
        num = int(input("Enter a number (-1 to stop): "))   # To take input from user until -1 is encountered
        if num == -1:
            break                                         # Break statement
        if num > 1:
            if is_prime(num):
                prime_count += 1
            else:
                composite_count += 1

    print(f"Prime numbers count: {prime_count}")
    print(f"Composite numbers count: {composite_count}")

if __name__ == "__main__":
    main()                                              # Function call
```

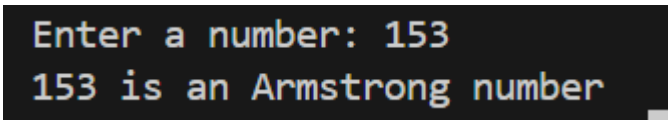
```python
# Question 3

num = int(input("Enter a number: "))            # take input from user
sum = 0
temp = num
while temp > 0:                                 # while loop
    digit = temp % 10
    sum += digit ** 3
    temp //= 10
if num == sum:                                  # if else condition
    print(num,"is an Armstrong number")
else:
    print(num,"is not an Armstrong number")
```

Output(s):

Question 1: output

```
Enter a number (-1 to stop): 4
Enter a number (-1 to stop): 9
Enter a number (-1 to stop): 1
Enter a number (-1 to stop): 2
Enter a number (-1 to stop): 5
Enter a number (-1 to stop): 7
Enter a number (-1 to stop): 10
Enter a number (-1 to stop): -1
Prime numbers count: 3
Composite numbers count: 3
```

Question 2: output



Conclusion:

Post Lab Questions:

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

Ans: *The nested if statements in Python are the nesting of an if statement inside another if statement with or without an else statement.*

For example: To find the greatest number among the three entered numbers

```
a = int(input("Enter first number: "))
b = int(input("Enter second number: "))
c = int(input("Enter third number: "))
if a > b:                                          # if a is greater than both b and c
    if a > c:
        print(f"{a} is the greatest number among the three")
if b > c:                                          # if b is greater than both a and c
   if b > a:
        print(f"{b} is the greatest number among the three")
if c > a:                                          # if c is greater than both a and b
    if c > b:
        print(f"{c} is the greatest number among the three")
```

```
Enter first number: 10
Enter second number: 20
Enter third number: 30
30 is the greatest number among the three
```

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

Ans: *The break statement is used within a loop or switch statement to terminate the loop or exit the switch. It is typically used within conditional statements. The continue statement is used within a loop to skip the rest of the code in the current iteration and begin the next iteration.*

```
for i in range(1, 5):
    for j in range(2, 6):
        if j%i == 0:
            break
        print(i, " ", j)
```

```
for i in range(5):
    if i == 3:
        continue
    print(i)
```

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

```
s = input("Input a string")              # to input string from the user
d = l = 0
for c in s:
    if c.isdigit():                      # to check if c is digit
        d = d + 1
    elif c.isalpha():                    # to check if c is a letter/alphabet
        l = l + 1
    else:
        pass
print("Letters", l)
print("Digits", d)
```

```
Input a string Mayuri C7
Letters 7
Digits 1
```

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