**Batch: C4-1 Roll No.: 16010123217**

**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

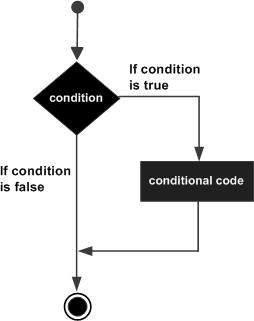
**If statement:**

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:

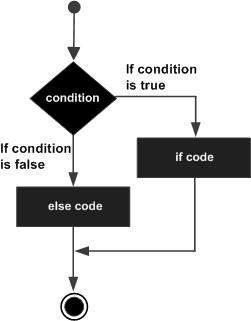
if expression:

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:**

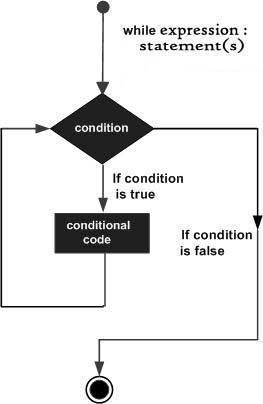
A **while** loop statement in Python programming language repeatedly executes a target statement as long as a given condition is true.

Syntax:

while expression:

statement(s)

While loop flowchart:



**For Loop:**

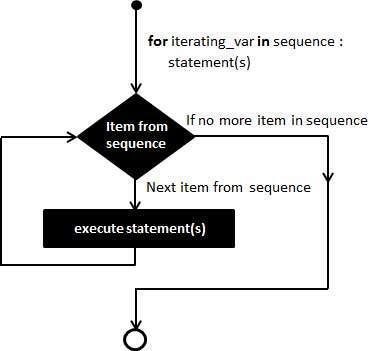
The [**for**](https://docs.python.org/3/reference/compound_stmts.html#for)statement in Python differs a bit from what you may be used to in C. Rather than giving the user the ability to define both the iteration step and halting condition (as C), Python’s **for**statement iterates over the items of any 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. Code:

*# 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*

prime\_count = 0

composite\_count = 0

while True:

    num = int(input("Enter a number (-1 to stop): "))

    if num == -1:

        break

elif num <= 0:

        print("Negative integer cannot be prime\nPlease enter a positive value")

        continue

    elif num > 1:

        for i in range(2, num):

            if (num % i) == 0:

                composite\_count += 1

                break

        else:

            prime\_count += 1

    else:

        print(num," is neither prime nor composite")

print("Number of prime numbers: ", prime\_count)

print("Number of composite numbers: ", composite\_count)

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.)*

num = input("Enter the number\n")

digits = list(num)

sum = 0

for i in range(0, len(digits)):

    sum += int(digits[i])\*\*len(digits)

if int(num) != sum:

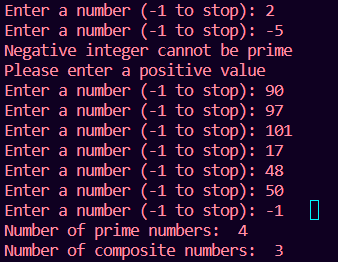
    print(num,"is not an Armstrong number.")

else:

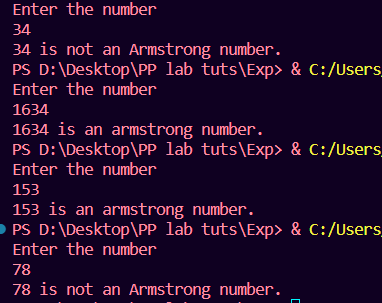
    print(num,"is an armstrong number.")

**Output(s):**

1.



2.



**Conclusion:**

The experiment focused on using different decision-making statements and control flow statements in Python, such as if statements, if-else statements, if-elif-else statements, while loops, for loops and break and continue.

**Post Lab Questions:**

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

Ans. Nested if statements are used when you need to make more than one decision in a program. They allow you to check multiple conditions and perform different actions based on the outcome of those conditions.

In the first question, we first check if the number is greater than 1. If it is, we enter a loop to check for factors. Within this loop, we have another if statement to determine if a factor other than 1 and itself is found. If a factor is found, we print that the number is composite. If no factors are found, we use else to indicate that the number is prime. Finally, if the initial condition (number greater than 1) is not met, we print that the number is neither prime nor composite.

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

Ans. “Break” is used to exit a loop prematurely, while “continue” is used to skip the rest of the loop's code for the current iteration and move to the next iteration. These statements provide more control over the flow of loops in your program.

In the first question

Break:

When the user enters -1, the condition num == -1 is true, and the break statement is executed. This means the loop will be terminated, and the program will move on to the statements after the loop.

Continue:

If the user enters a number less than or equal to 0, the condition num <= 0 is true. The continue statement is executed, which means that the remaining code inside the loop for that iteration (including the loop's increment step) is skipped, and the loop immediately moves on to the next iteration.

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

Code:

string1 = input("Enter your string:\n")

alpha = 0

numeric = 0

for i in range(0,len(string1)):

    if string1[i].isalpha():

        alpha += 1

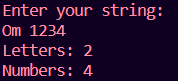
    elif string1[i].isnumeric():

        numeric += 1

print("Letters:", alpha)

print("Digits:", numeric)

Output:



**Date: \_\_\_\_\_\_\_\_\_\_\_\_\_ Signature of faculty in-charge**