**Batch: C7 Roll No.: 57**

**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

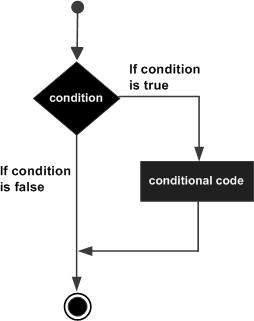
**If statement:**

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:

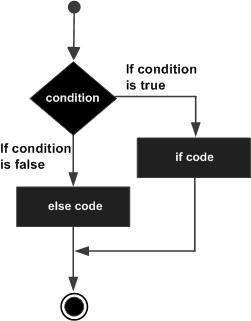
if expression:

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:**

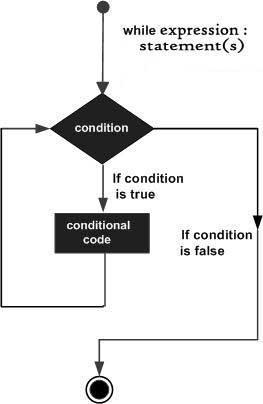
A **while** loop statement in Python programming language repeatedly executes a target statement as long as a given condition is true.

Syntax:

while expression:

statement(s)

While loop flowchart:



**For Loop:**

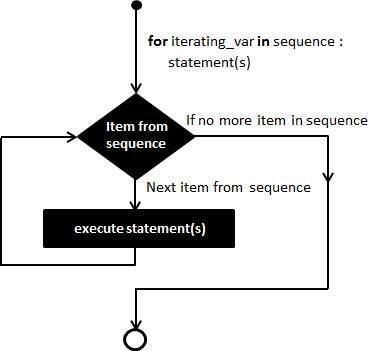
The [**for**](https://docs.python.org/3/reference/compound_stmts.html#for)statement in Python differs a bit from what you may be used to in C. Rather than giving the user the ability to define both the iteration step and halting condition (as C), Python’s **for**statement iterates over the items of any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

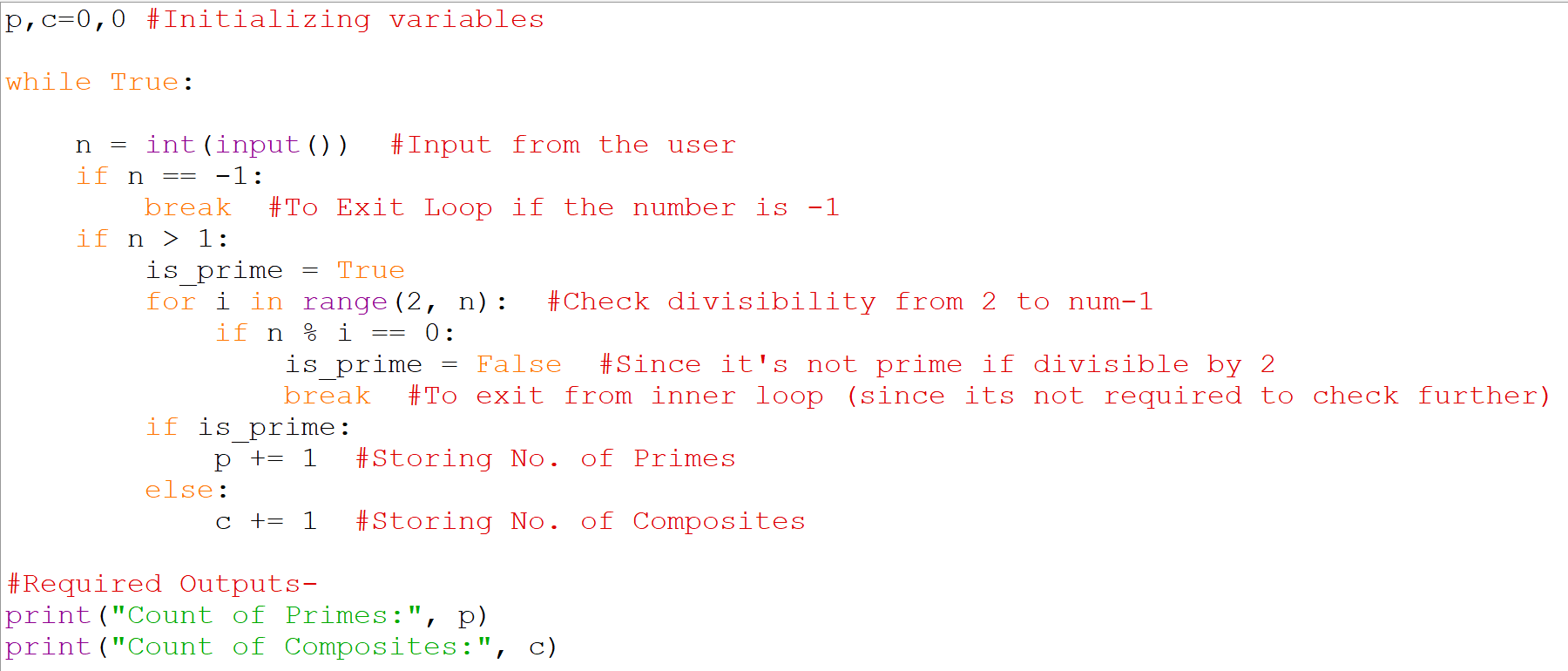
2) Write a program to check whether a number is Armstrong or not.

## (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)**

p,c=0,0 #Initializing variables

while True:

n = int(input()) #Input from the user

if n == -1:

break #To Exit Loop if the number is -1

if n > 1:

is\_prime = True

for i in range(2, n): #Check divisibility from 2 to num-1

if n % i == 0:

is\_prime = False #Since it's not prime if divisible by 2

break #To exit from inner loop (since its not required to check further)

if is\_prime:

p += 1 #Storing No. of Primes

else:

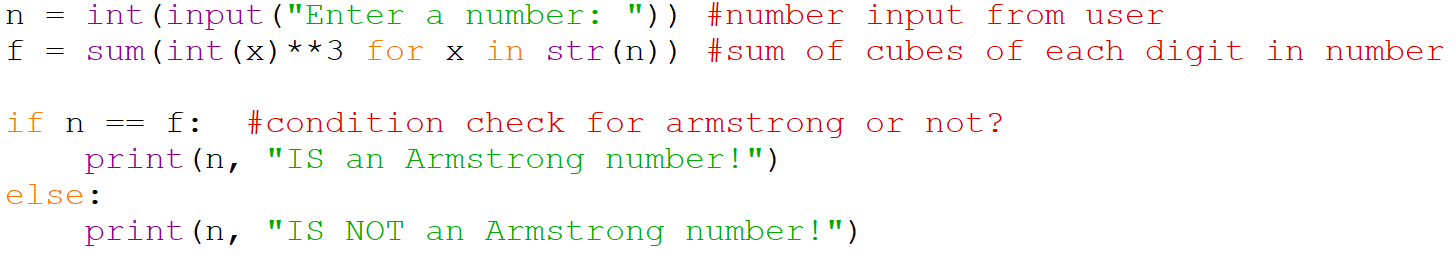
c += 1 #Storing No. of Composites

#Required Outputs-

print("Count of Primes:", p)

print("Count of Composites:", c)

**2)**



n = int(input("Enter a number: ")) #number input from user

f = sum(int(x)\*\*3 for x in str(n)) #sum of cubes of each digit in number

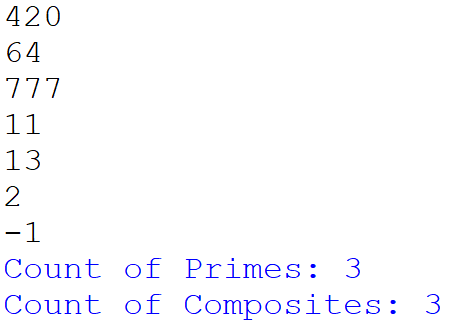
if n == f: #condition check for armstrong or not?

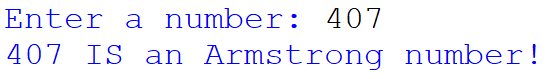
print(n, "IS an Armstrong number!")

else:

print(n, "IS NOT an Armstrong number!")

**Output(s):**

**1)**

  
  
  
  
  
  
  
**2)**

**Conclusion:**

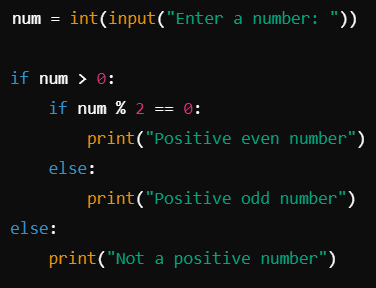
I learned that using loops (for, while) and conditions (if, elif, else) makes solving problems concise and efficient and helps in writing clear, effective code.

**Post Lab Questions:**

1. When should we use **Nested If** Statements? Illustrate your answer with the help of an example.

**Nested if** statements are used when you need to **check multiple conditions** that **depends on one another**. They allow for more specific decision-making based on the outcome of previous checks.

**Example**: To check is a number is “Positive Even Number”, “Positive Odd Number” or “Not a Positive Number”?



**Ans**:

1. Explain the utility of **Break** and **Continue** statements with the help of an example.

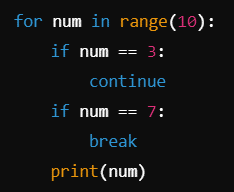
**Ans**:

**Break** Statement: It is used to **terminate** the loop **entirely** and exits it.

**Continue** Statement: It is used to **skip** the **current iteration** and just move to the next iteration of the loop without exiting it.

**Example**: To print numbers from 0 to 9, **skipping** 3 (using **continue**) and **stopping** at 7 (using **break**).

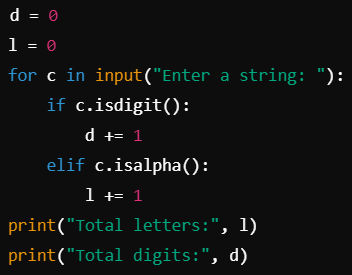
Input: Output:

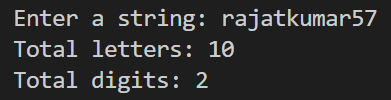




**3)** Write a program that accepts a string from user and calculate the number of digits and letters in string.

**Ans:** Input: Output:





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