

| **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:**

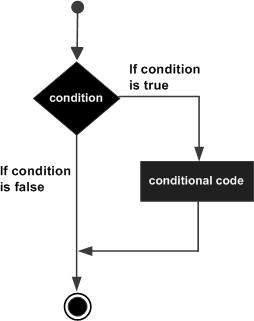
In Python **if** statement is used for decision-making operations. It contains a body of code which runs only when the condition given in the **if** statement is true.



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:

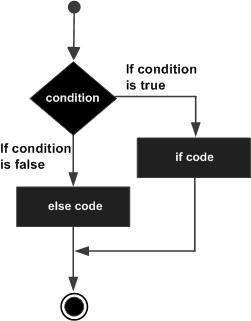
if expression:

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.

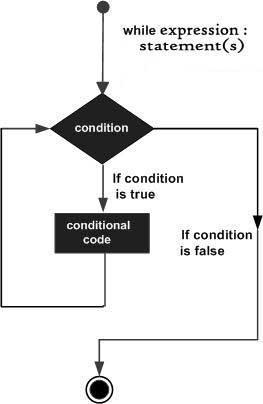


Syntax:

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.

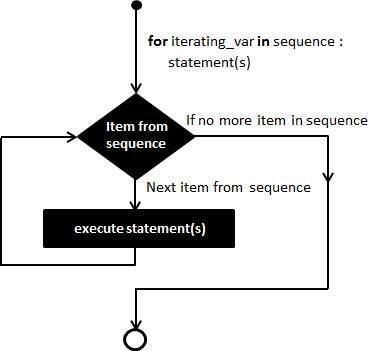


Syntax:

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.**

# Check if a number is prime

def check(num):

    if num <= 1:

        return False

    for i in range(2, int(num)):

        if num % i == 0:

            return False

    return True

# Initialize counters

pcount = 0  # Count of prime numbers

ccount = 0  # Count of composite numbers

onecount = 0 #Count of the number 'one'

# Input loop until -1 is entered

while True:

    num = int(input("Enter a number : "))

    if(num == -1):

        break

    elif(num <= 0):

        print("\nInvalid Input")

        break

    # Count one,prime and composite numbers

    if(num == 1):

      onecount+=1

    else:

      if check(num):

        pcount+=1

      else:

        ccount+=1

# Print counts

print("\nPrime numbers entered :", pcount)

print("\nComposite numbers entered :", ccount)

print("\nNumber of times '1' entered :", onecount)

#The program will break if three types of values are entered

#They are -1 , 0,-ve numbers

2.

def check(number):

  #Store number for later comparision

    og\_num = number

  #Calcuate number of digits

    numofdigits = len(str(number))

  #Initialize a variable to store sum of cubes of all the digits

    sumofcubes = 0

#Loop until the numbers become equal to zero

    while number > 0:

#Find out the last digit of the number by using modulo 10

        digit = number % 10

#Add the cube of digit to sum

        sumofcubes += digit \*\* numofdigits

#Remove the last digit of the number to continue loop

        number //= 10

# Check if the sum of cubes is equal to the original number

    return sumofcubes == og\_num

num = int(input("Enter a number: "))

#Calls the earlier defined function

if check(num):

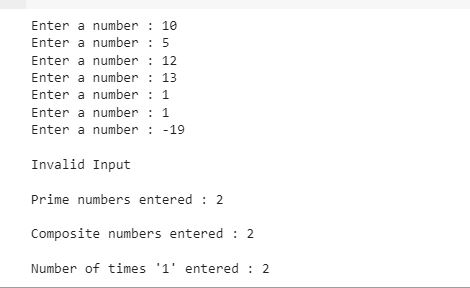
    print(num," is an Armstrong number.")

else:

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

**Output(s):**

1.



2.

OP2.JPG

OP3.JPG



**Conclusion:**

Learn the use of different Decision Making statements in Python.

Used conditional statements(such as if statement ,if-else statement ,if-elif-else statement) to execute the given aims of the experiment.

Also understood the working of various loops(such as for loop & while loop) in the process of completing the experiment.

**Post Lab Questions:**

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

A) We should use nested if statements when we want to check multiple conditions in a step wise manner & when the working of one condition affects the working of the next subsequent conditions.

Eg. A program which checks if the number entered is greater than 0 i.e positive and if its positive then check if its greater than 25.

In the example, the program will only check for number greater than 50 when if condition is false i.e. the number is not negative.

num = int(input("Enter a Number : "))

if num < 0:

  print("The Entered Number is Negative")

else:

  if num > 50:

    print("The Entered Number is Greater than 50")

  else:

    print("The Number is Positive")

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

‘Break’ is used to exit a loop prematurely when a certain condition is met.

for i in range(1, 10):

    if i % 5 == 0:

        break  # Prints till loop is broken

    print(i)

Prints 1 to 4 and then exits the loop when condition is fulfilled.

‘Continue’ is used to skip the current iteration of a loop when a specific condition is met, allowing the loop to continue with the next iteration.

for i in range(1, 10):

    if i % 5 == 0:

        continue  # Skips when condition satisfied

    print(i)      #Print everything else

Prints 1 to 9 but does not print 5 as at that point the condition is fulfilled.

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

n = input("Enter a String : ")

ncount = 0

lcount = 0

for i in n:

  if i.isdigit():

    ncount += 1

  elif i.isalpha():

    lcount += 1

print("Number of Digits = ",ncount)

print("Number of Letters = ",lcount)

Output:

OP5.JPG

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