

| **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:**

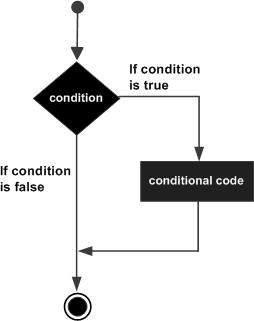
In Python **if** statement is used for decision-making operations. It contains a body of code which runs only when the condition given in the **if** statement is true.



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:

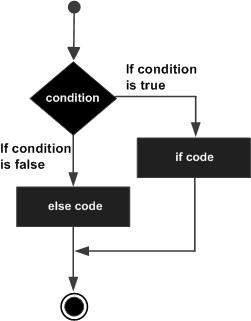
if expression:

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.

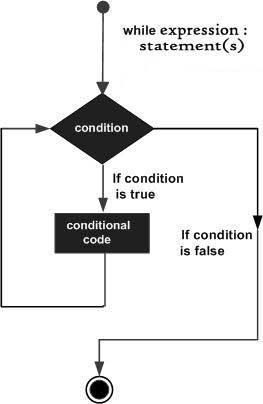


Syntax:

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.

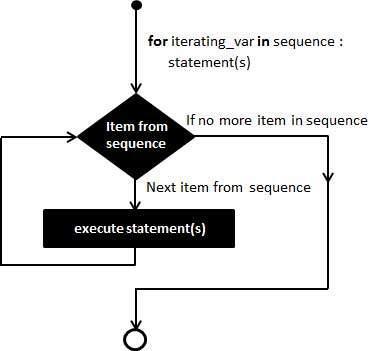


Syntax:

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]

count = 0

composite = 0

prime = 0

while(True):

n = int(input("Enter the number"))

if(n == -1):

break

count += 1

if (n <= 0):

composite+=1

continue

if (n == 1):

composite += 1

continue

else:

for i in range(2, n):

if (n % i == 0):

composite += 1

break

else:

prime += 1

print("Prime: ", prime)

print("Composite ", composite)

2]

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

num\_of\_digits = len(str(num))

sum = 0

temp = num

while temp > 0:

digit = temp % 10

sum += digit \*\* num\_of\_digits

temp //= 10

if num == sum:

print(num, "is an Armstrong number")

else:

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

**Output(s):**

1]

Enter the number(-1 to stop):4

Enter the number(-1 to stop):5

Enter the number(-1 to stop):6

Enter the number(-1 to stop):-1

Prime: 1

Composite: 2

2]

Enter a number: 153

153 is an Armstrong number

2]

Enter a number: 123

123 is not an Armstrong number

**Conclusion:**

Thus, we learned how to use decision making statements in python programming.

**Post Lab Questions:**

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

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

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

1]We use nested if statements when we need to test multiple conditions in a specific order or when we need to check if a condition is true before moving to the next condition. Nested if statements make the code more readable and easier to understand when dealing with multiple levels of conditions.

Example 1:

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

if num % 2 == 0:

if num % 4 == 0:

print("The number is even and divisible by 4.")

else:

print("The number is even but not divisible by 4.")

else:

print("The number is odd.")

Example 2:

age = 17

citizenship = "USA"

if age >= 18:

if citizenship == "USA":

print("You are eligible to vote.")

else:

print("You are not eligible to vote because you are not a US citizen.")

else:

print("You are not eligible to vote because you are under 18 years old.")

2]

break and continue are control flow statements in Python that can be used to modify the behavior of loops.

break statement:

The break statement is used to terminate a loop prematurely. When the break statement is encountered inside a loop, the loop is immediately terminated, and the program execution continues with the next statement after the loop.

Example:

numbers = [3, 7, 9, 2, 5, 8, 10, 12]

for num in numbers:

if num % 2 == 0:

print("The first even number is:", num)

break

Output:

The first even number is: 2

continue statement:

The continue statement is used to skip the current iteration of a loop and move to the next iteration. When the continue statement is encountered inside a loop, the current iteration is immediately skipped, and the program execution continues with the next iteration of the loop.

Example:

numbers = [3, 7, 9, 2, 5, 8, 10, 12]

for num in numbers:

if num % 2 == 0:

continue

print(num)

Output:

3

7

9

5

3]

string = input("Enter a string: ")

n\_count = 0

i\_count = 0

for char in string:

if char.isdigit():

n\_count += 1

elif char.isalpha():

i\_count += 1

print("Number of digits:", n\_count)

print("Number of letters:", i\_count)

Output

Enter a string: hello123

Number of digits: 3

Number of letters: 5

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