**Batch: C4-1 Roll No.: 06**

**Experiment / assignment / tutorial No.**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **TITLE:** Write a program to demonstrate the use of decision-making statements in Python.  **AIM**: 1) Write a program to read the numbers until -1 is encountered. Also, count the number of prime and composite numbers entered by the user.  2) Write a program to check whether a given number is Armstrong.  **Outcome:** Students will be able to  **CO1:** Formulate a problem statement and develop the logic (algorithm/flowchart) for its solution.  **CO3:** Use different decision-making statements and functions in Python. |

Use the input-output function. And different decision-making statements in Python.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Resource Needed: Python IDE**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**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://www.geeksforgeeks.org/python-strings/?ref=lbp

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**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 statements**

a) while loop

b) for loop

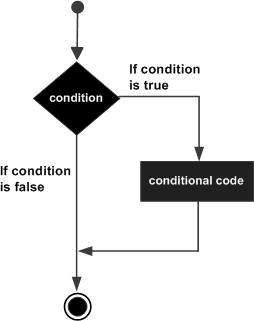
**If statement:**

In Python, the **if** statement is used for decision-making operations. It contains a body of code that 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 the **if**.

### Syntax:

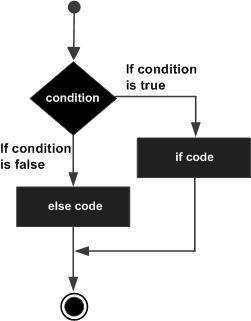
if expression:

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 code block as soon as one of the conditions evaluates to TRUE.

Similar to the else statement, 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:**

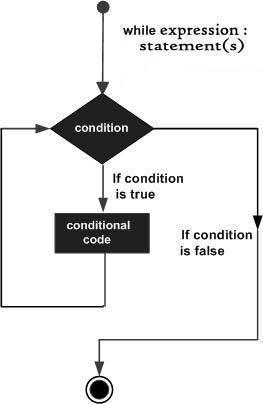
A **while** loop statement in the Python programming language repeatedly executes a target statement as long as a given condition is true.

Syntax:

while expression:

statement(s)

While loop flowchart:



**For Loop:**

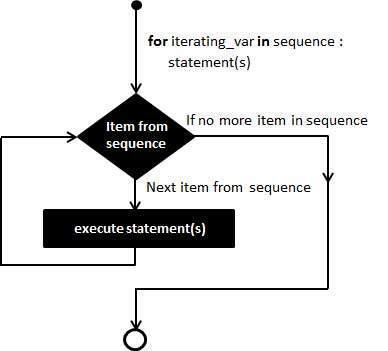
The [**for**](https://docs.python.org/3/reference/compound_stmts.html#for)statement in Python differs a bit from what you may be used to in C. Rather than giving the user the ability to define both the iteration step and halting condition (as C), Python’s **for**statement iterates over the items of any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

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

## (An 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.)

**Implementation details:**

**Exercise 1:**

i = *int*(input("Enter the Number of Elements you want to Enter: "))

j = []

for k in range(i):

  j.append(*int*(input()))

for k in range(len(j)):

  if(j[k]!= -1):

    print(j[k])

  else:

    break

com\_num = 0

prime\_num = 0

for k in range(len(j)):

  for x in range(2, *int*(j[k]+1)):

    if(j[k]%x != 0 or j[k] == 2):

      prime\_num += 1

      break

    else:

      com\_num += 1

      break

print("Count of Prime Numbers is:", prime\_num)

print("Count of Composite Numbers is: ", com\_num)

**Exercise 2:**

i = *str*(input("Enter a Number: "))

j = 0

for k in i:

  k = *int*(k)

  j = j + k \*\* 3

if(*int*(i) == *int*(j)):

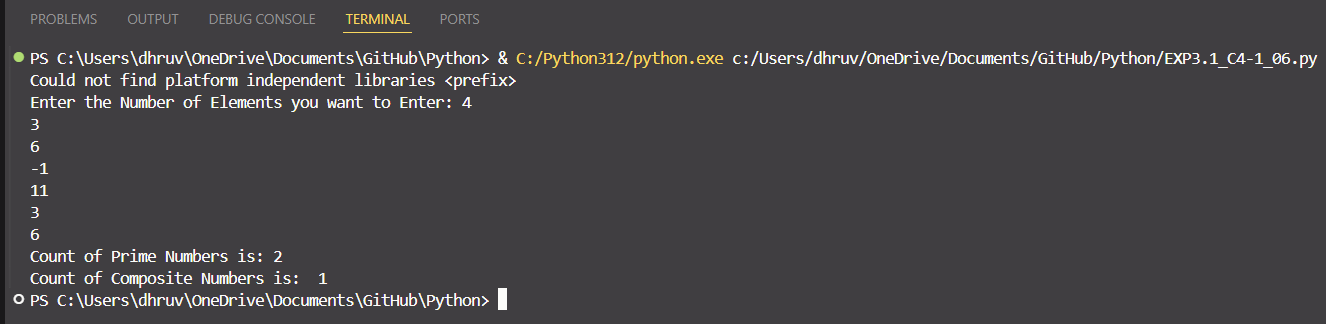
  print("The given Number is an Armstrong Number")

else:

  print("The given Number is not an Armstrong Number")

**Output(s):**

**Exercise 1:**



**Exercise 2:**

A screenshot of a computer

Description automatically generated

**Conclusion:**

We can use the for loops for iterating over a various data and perform various actions on them. We also used the if, if-else and if-elif-else statement which executes code blocks based on various conditions. We also learnt to use nested for loops to execute a code multiple times inside a for loop with multiple iterations. We also learned to use the while loop which keeps iterating a block of code while a specific condition remains true.

**Post Lab Questions:**

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

**Ans.** Nested if statements are useful when we need to check multiple conditions that are dependent on each other. When we need to check multiple criteria where one check depends on the outcome of a previous check. When decisions are hierarchical in nature, i.e. if one condition must be true before we can check another.

**Code:**

score = int(input("Enter the student's score: "))

if score >= 50:

print("The student has passed.")

if score >= 85:

print("Grade: A")

elif score >= 70:

print("Grade: B")

elif score >= 60:

print("Grade: C")

else:

print("Grade: D")

else:

print("The student has failed.")

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

**Ans.** The break and continue statements are control flow statements in Python that alter the behaviour of loops. They are used to control the execution of loops based on certain conditions.

The break statement is used to exit a loop prematurely when a certain condition is met.

The continue statement is used to skip the current iteration of the loop and move to the next iteration.

**Code:**

numbers = [1, -2, 3, 4, -5, 6, 0, 7, 8]

for num in numbers:

if num < 0:

continue

if num == 0:

print("Encountered zero, exiting the loop.")

break

print(f"Odd number: {num}" if num % 2 != 0 else f"Even number: {num}")

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

a = str(input("Enter a String: ")).upper()

b = list(a)

c = list(set(a))

d = {}

i = 0

while(i < len(c)):

  d.update({c[i]:b.count(c[i])})

  i += 1

print(d)