**Batch: B2 Roll No.: 16010121121**

**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

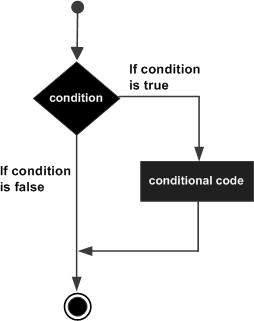
**If statement:**

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:

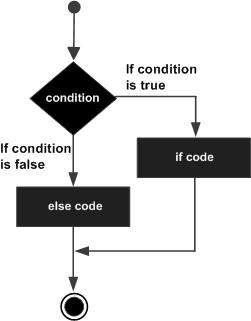
if expression:

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:**

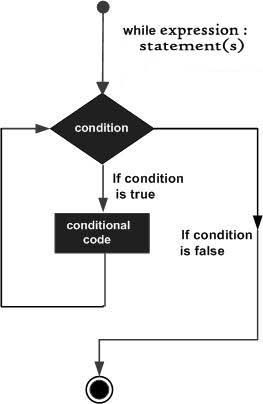
A **while** loop statement in Python programming language repeatedly executes a target statement as long as a given condition is true.

Syntax:

while expression:

statement(s)

While loop flowchart:



**For Loop:**

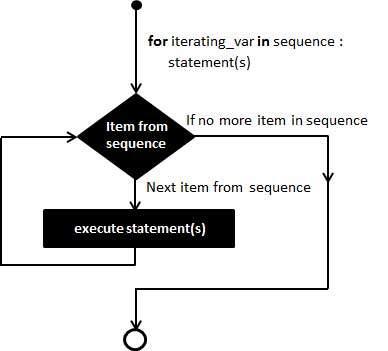
The [**for**](https://docs.python.org/3/reference/compound_stmts.html#for)statement in Python differs a bit from what you may be used to in C. Rather than giving the user the ability to define both the iteration step and halting condition (as C), Python’s **for**statement iterates over the items of any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

Code 1:

i = 1

j = 1

c=0

cp=0

cc=0

#this for loop can go for 20,000-1 iterations

#which means a total of 20,000-1 inputs can

#be accepted at maximum

for i in range(1, 20000):

num = int(input("Enter an integer: "))

if num==-1:

break

elif num<=0:

print("An incorrect input has been entered. So, the input is ignored")

continue

elif num==1:

print("The number ", num, " is composite.")

cc+=1

else:

c = 0 #for the next number, counter has to be reset

for j in range(1, num):

if num%j==0:

c+=1

if c<=1:

print("The number ", num, " is prime.")

cp+=1

else:

print("The number ", num, " is composite.")

cc+=1

print("The number of prime numbers entered are ", cp)

print("The number of composite numbers entered are ", cc)

Code 2:

armnum = 0

num = int(input("Enter a number to check if it is Armstrong or not: "))

orignum = num

if(num<0):

print("An invalid input has been entered.")

else:

while(num!=0):

num2 = int(num%10)

armnum += pow(num2, 3)

num = int((num-num2)/10)

if armnum == orignum:

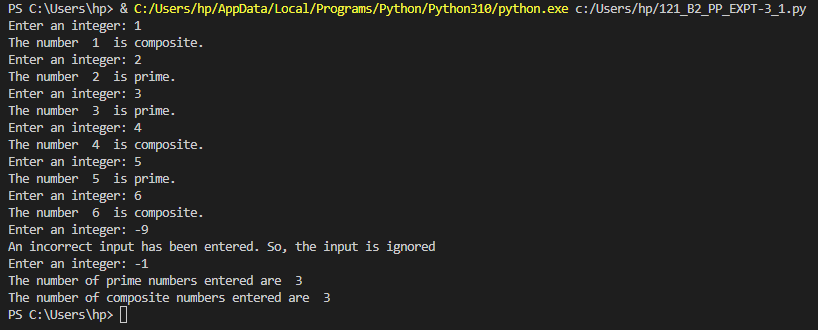
print("The number ", orignum, "is an Armstrong number.")

else:

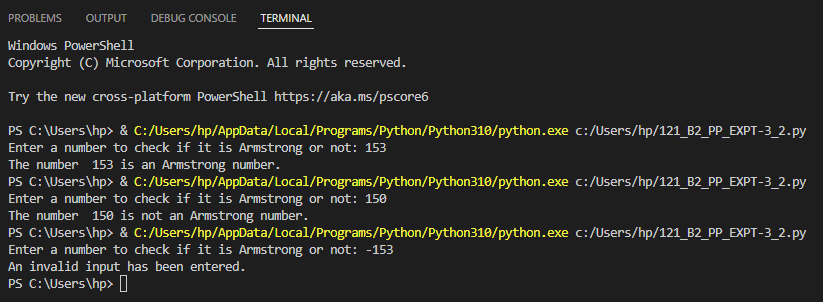
print("The number ", orignum, "is not an Armstrong number.")

**Output(s):**

Code 1:



Code 2:



**Conclusion:**

Thus, in this experiment, I have learnt to code a program which counts the number of prime numbers and composite numbers entered by the user. I have also learnt to code a program which accepts a number and checks if it is Armstrong or not. In these programs, I have applied the newly-taught (as of Python classes) concept of decision making and looping statements. I have learnt especially in the case of looping statements, how Python for loop code is different from that of C. While the Python for loop code requires a variable and iterable only (which can be a list, tuple, dictionary, range, etc.) whereas in C language three things are required – variable, qualifying condition and increment/decrement/variable-value-updating-operation. Also, for the programs, the different test cases have been included in the outputs. I have also noted that there is some alteration required in the code in Python language to find whether a number is Armstrong or not, as compared to C language,

**Post Lab Questions:**

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

Ans. Nested if statements should be used when there are higher degrees of branching. It allows the programmer to check more than one condition at once and return different values depending on the results of those checks in an efficient manner.

For example, there is a college which has three types of members – faculty, students and staff. So for logging in to the website of that college, first the user may be asked to specify the type of member he is. So that involves an if-elif-else structure for member type. Further, if say, the students’ category is selected, then there are more possibilities – the student may be in Computer Engineering branch or Mechanical Engineering branch or some other branch. So for the student to enter his branch, there is one more if-elif-else structure within the if-branch for student. Further, for each branch, the student can be asked to enter the year one is studying in, say First Year. So that necessitates another if-elif-else structure within say, Computer Engineering branch of the if-elif-else structure. So in this way, there is nested if statement usage.

if(member==”student”):

if(branch==”Computer Engineering”):

if(year==”FY”):

#some statements

elif(year==”SY”):

#some statements

elif(year==”TY”):

#some statements

elif(“year==”LY”):

#some statements

else:

print(“Incorrect input.”)

elif(branch==”Mechanical Engineering”):

#and so on the program code goes

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

Ans. The break and continue statements are the jump statements that are used to skip some statements inside the loop or terminate the loop immediately without checking the test condition. These statements can be used inside any loops like for and while loop.

The break statement is used to terminate from the loop immediately. When a break statement is encountered inside a loop, the loop execution stops there, and the control returns from the loop innediately to the first statement after the loop.

Ex: for x in range(0, 10):

if x==5:

break

print(“Number is: “, x)

Output:

Number is: 0

Number is: 1

Number is: 2

Number is: 3

Number is: 4

The continue statement is used to skip the current iteration of a loop. When the continue statement is encountered, then the statements following continue for that iteration of the iterating variable will be skipped, but the statements during the next iteration of the iterating variable will be executed.

Ex: for x in range(0, 10):

if x==5:

continue

print(“Number is: “, x)

Output:

Number is: 0

Number is: 1

Number is: 2

Number is: 3

Number is: 4

Number is: 6

Number is: 7

Number is: 8

Number is: 9

Number is: 10

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

Ans.

st = input("Enter a string: ")

cstr=0

cdig=0

for var in st:

if var.isdigit()==True:

cdig+=1

elif var.isspace()==True:

continue

else:

cstr+=1

print("The number of digits are: ", cdig)

print("The number of strings are: ", cstr)

**Date: \_\_29-04-2022\_\_ Signature of faculty in-charge**