Name: Shreyans Tatiya

Batch: C5_3 Roll No.: 16010123325 (53)

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

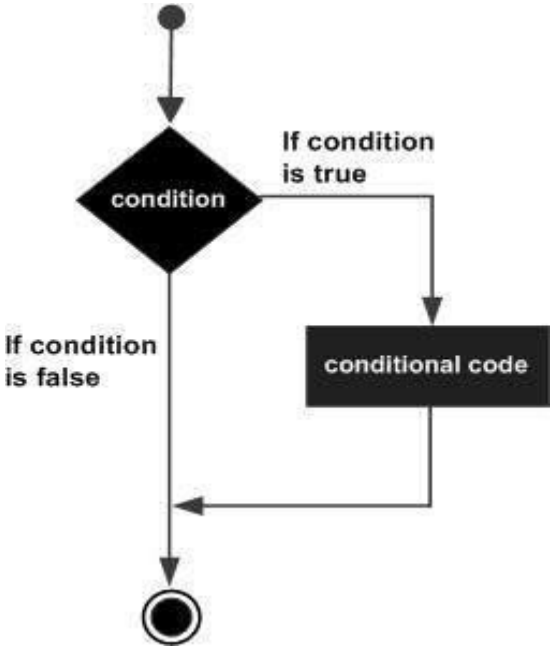
If statement:

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:

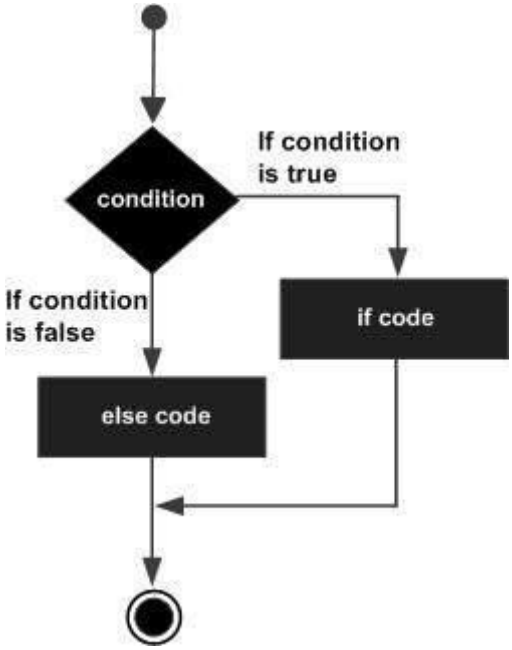
An **else** statement can be combined with an **if** statement. An **else** statement contains the block of code that executes if the conditional expression in the **if** statement resolves to 0 or a FALSE value.

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)
```

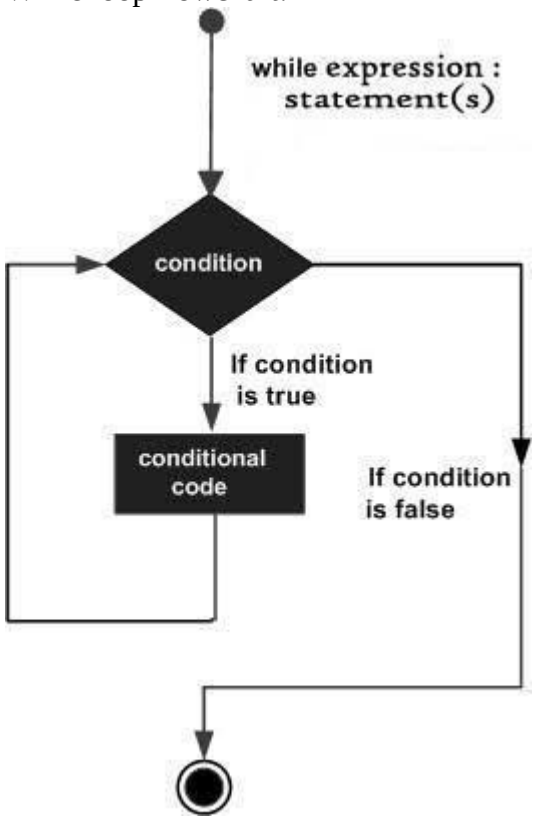
While loop:

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:



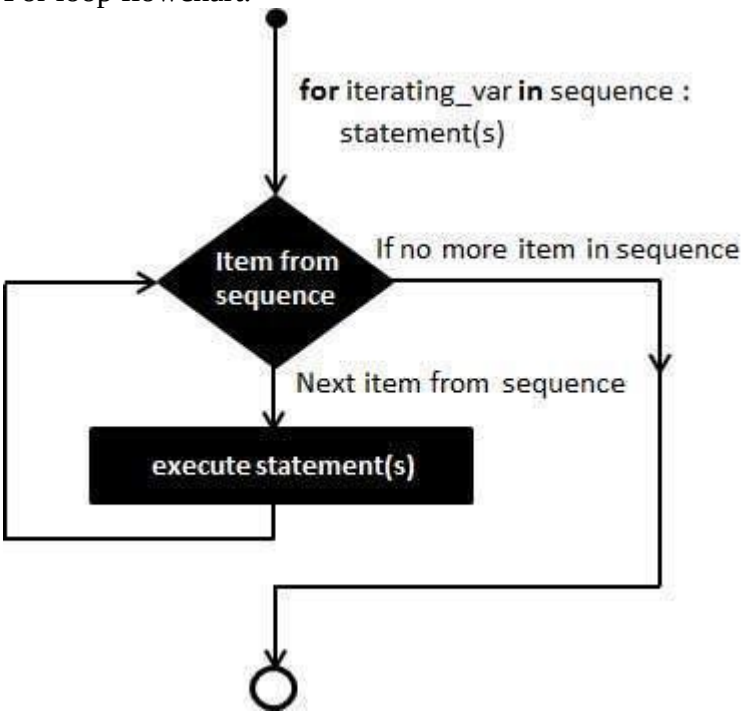
For Loop:

The **for** 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)

```
3
4   def check(n): # Prime Number check function
5       for i in range(2,n): # checks from 2 until n-1
6           if (n%i==0):
7               return False # if n is divisible by i it exits and returns False
8       return True # if n is not divisible by i it is a Prime Number
9
10  n,p,c,k = 0,0,0,0
11  while check(n):
12      n = (int)(input())
13      if(n==-1):
14          break #breaks at n = -1
15      elif (n<0):
16          print("Invalid Input") # negative numbers are invalid for checking
17          break
18      if (n==1):
19          k+=1 #if users inputs n as 1 then counter increases
20      else:
21          if check(n):
22              p+=1 #inceases the prime number counter
23          else:
24              c+=1 # increase the composite number counter
25
26  print("The number of Prime Numbers is",p )
27  print("The number of Composite Numbers is",c )
28  print("The number of time 1 was entered:",k,"times" )
29
```

2)

```
n = (int)(input("Enter a number : "))
sum = 0
num = n
while num!= 0:
    rem =(int)(num % 10)
    rem  = rem ** 3

    sum += rem
    num = num/10

print(sum)
if(sum == n):
    print("It is an armstrong number")
else:
    print("It is not an armstrong number")
```

Output(s):

1)

```
PS C:\Users\Shrey> & C:/Users/Shrey/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/Shrey/OneDrive/D
EM-1/Python Programming/Ongoing/holy molly.py"
2
3
4
The number of Prime Numbers is 2
The number of Composite Numbers is 1
The number of time 1 was entered: 0 times
PS C:\Users\Shrey> & C:/Users/Shrey/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/Shrey/OneDrive/D
EM-1/Python Programming/Ongoing/holy molly.py"
```

```
PS C:\Users\Shrey> & C:/Users/Shrey/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/Shrey/OneDrive/Desktop/KJSCE/S
EM-1/Python Programming/Ongoing/holy molly.py"
2
3
5
7
13
15
The number of Prime Numbers is 5
The number of Composite Numbers is 1
The number of time 1 was entered: 0 times
PS C:\Users\Shrey>
```

2)

Conclusion:

Used nested conditional statements to execute and carry out the give aim of the experiment. Also learnt to utilize multiple conditional statements to check for different possible outcomes.

Post Lab Questions:

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

Ans: We use nested conditional statements under various applications where in multiple checks are needed to carry out under a specific condition.

For example, a program which checks if the number entered is positive or negative and if it's positive then check if its less than 10.

```
n=int(input("Enter a number: "))
if n<0:
    print("The number is Negative")
else:
    if n<10:
        print("The number is Positive and less than 10")
    else:
        print("The number is Positive")
```

So here, the second if and else statement is nested under the first else condition. So once its confirmed that the number is positive the program goes ahead and checks if n is less than 10 and if it's true then executes the print statements. If the condition is false then the else statement is executed.

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

Ans: The **break** statement in python terminates and exits out the current loop and resumes the execution out of the loop statement. For example

```
for i in range(1,10):
    if  (i % 5 == 0):
        break
    print(i)
```

Over here when 'i' becomes 5 the break statement will be executed and the loop ends and so does the program. So, it prints numbers from 1 to 4.

The **continue** statement in python returns the control to the next iteration of the loop. For example.

```
for i in range(1,10):
    if i % 5 == 0:
        continue
    print(i)
```

Here the loops runs normally but when i becomes 5 the continue statements transfers directly to the next iteration without executing the rest of the loop. So it print 1 to 9 except the number 5.

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

Ans:

```
n=input("Enter a string: ")
d,a,s=0,0,0
for i in n:
    if i.isdigit():
        d+=1
    elif i.isalpha():
        a+=1
    else:
        s+=1
print("The number of Digits:",d)
print("The number of Letters:",a)
print("The number of Special Characters:",s)
```

Output:

```
PS C:\Users\Shrey> & C:/Users/Shrey/AppData/Loc
Desktop/Untitled-1.py
Enter a string: Shrey999
The number of Digits: 3
The number of Letters: 5
The number of Special Characters: 0
PS C:\Users\Shrey>
```

```
PS C:\Users\Shrey> & C:/Users/Shrey/AppDa
-1.py
Enter a string: Time 2 go
The number of Digits: 1
The number of Letters: 6
The number of Special Characters: 2
PS C:\Users\Shrey>
```

Date: _____ **Signature of faculty in-charge**