Aatmaj Mhatre	**Batch: B2 Roll No.: 16010121110**
	Experiment / assignment / tutorial No.
	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

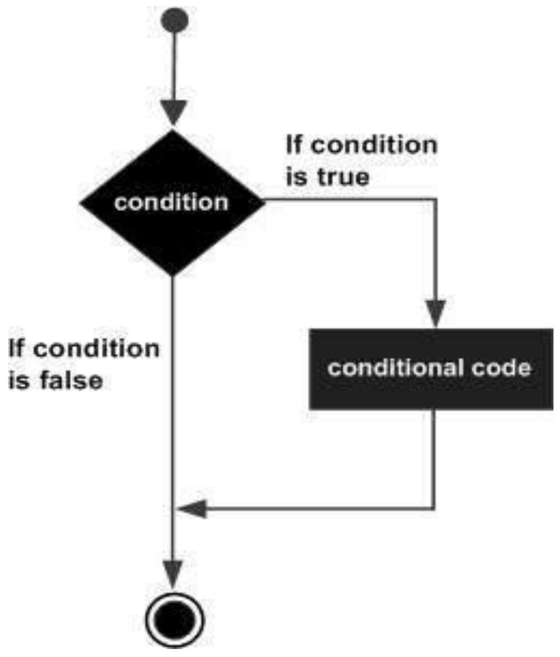
1) Selection/Conditional branching statements
- a) if statement
- b) if-else statement
- c) if-elif-else statement

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)
```

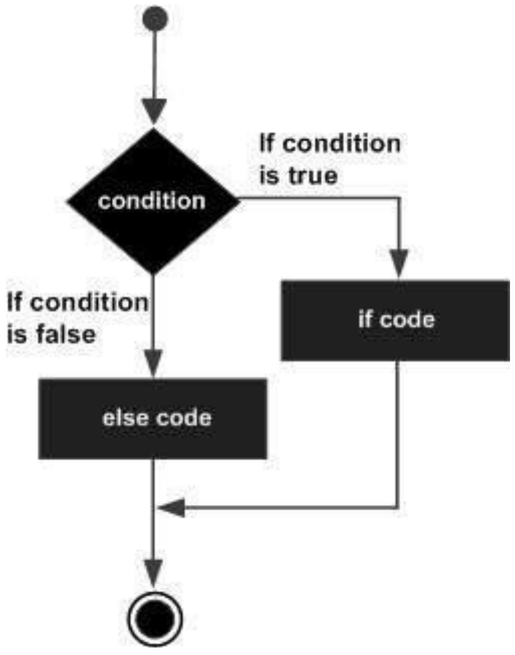
If flowchart:



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:

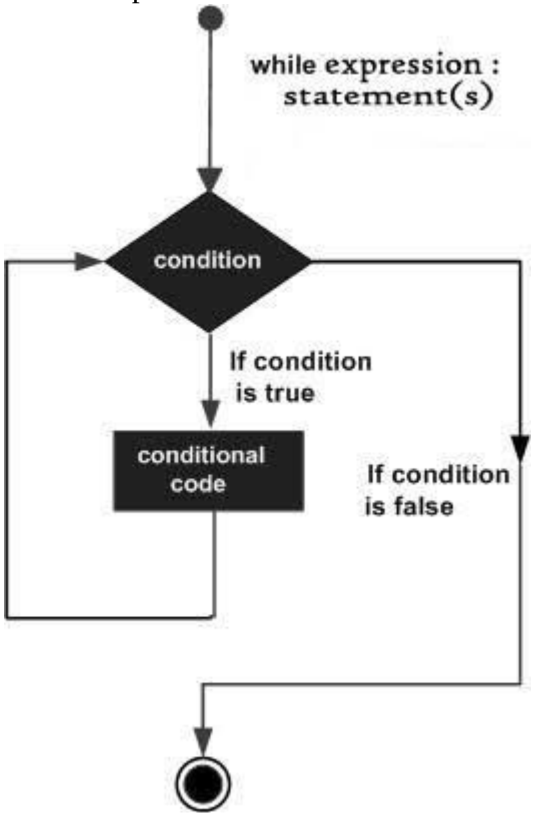
The **elif** statement allows you to check multiple expressions for TRUE and execute a block of code as soon as one of the conditions evaluates to TRUE.
Similar to the else, the **elif** statement is optional. However, unlike **else**, for which there can be at most one statement, there can be an arbitrary 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:

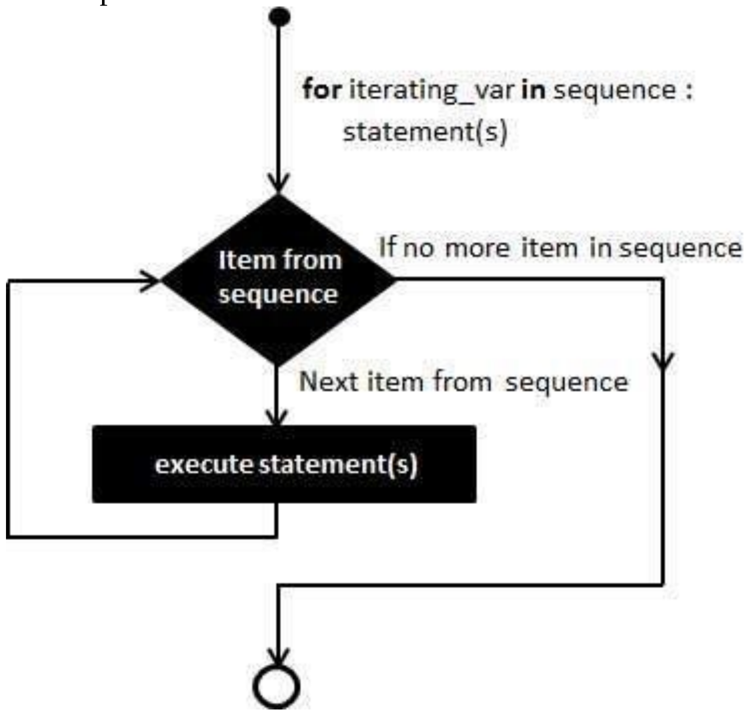


For Loop:
The **for** statement in Python differs a bit from what you may be used to in C. Rather than giving the user the ability to define both the iteration step and halting condition (as C), Python's **for** statement iterates over the items of any sequence (a list or a string), in the order that 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:

```
def IsPrime(a): #function to check if a number is primt or not.
    for i in range(2,a):
        if(a%i==0):
          return False
    return True
Input_List=[]#empty list
while(True):
    a=input("Please enter a valid number ") #take input
    try:
        a=int(a) #a is int

        if(a<-1 or a==0): # number must not be negative
            print("Invalid entry!")
            continue
        if(a==-1):
            break;
        Input_List.append(a) # add to list

    except ValueError:
        print("Invalid entry!")
prime=0;
composite=0;
for i in Input_List:
    if(i==1):
        continue # 1 is neither prime nor composite
    if (IsPrime(i)==True):
         prime+=1 #increase prime
    else:
        composite+=1 #increase composite
print("Number of primes=",prime)
print("Number of composite=",composite)
'''
Test case covering all values
```

```
Please enter a valid number 1
Please enter a valid number 2
Please enter a valid number 3
Please enter a valid number 4
Please enter a valid number 5
Please enter a valid number a
Invalid entry!
Please enter a valid number 0
Invalid entry!
Please enter a valid number -1
Number of primes= 3
Number of composite= 1
'''


a=''
while (True): # until a is valid
    a=input("Please enter a number:") #take number input
    try:
        a=int(a) #a is int
        break
    except ValueError:
        print("Invalid entry!")
        pass
Sum=0 #initiate to 0

for i in str(a):
  Sum=Sum+int(i)**3 #cube and sum
if(Sum==a):
    print("Yes. It is aarmstrong number.")
else:
    print("No. Not a armstrong number.")

'''
Test cases


Please enter a number:112
No. not armstrong number.
```

```
Please enter a number:153
Yes. It is armstrong number.

'''
```

Output(s):

```
Please enter a valid number 1
Please enter a valid number 3
Please enter a valid number 5
Please enter a valid number 7
Please enter a valid number 9
Please enter a valid number 12
Please enter a valid number 14
Please enter a valid number 9
Please enter a valid number abc
Invalid entry!
Please enter a valid number 10
Please enter a valid number -1
Number of primes= 3
Number of composite= 5
```

```
Please enter a number:153
Yes. It is armstrong number.
```

Conclusion:

Post Lab Questions:

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

Nested if statements must be used for conditions where you have to test more than one usage value. Example a program to check if a number is positive or negative and if positive odd or even.

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

Break- used to come out of loop

Continue- used to restart the loop.

Example question 1 both the statements are being used.

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

```
a=input("Please enter a string ")
num=0 # variable to hold numbers count
let=0 # variabl to hold letters count
for i in a:
   if(i==" "):
      continue
   try :
      temp=int(i) # if it is number
      num+=1
   except ValueError: # if it is character
      let+=1
print("Number of letters=",let)
print("Number of Numbers=",num)

'''
Test case

Please enter a string 123bag
Number of letters= 3
Number of Numbers= 3
```

'''

Date: 25 April 2022 **Signature of faculty in-charge**