

Batch: P1 – 2 Roll No.: 16014022050

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 -

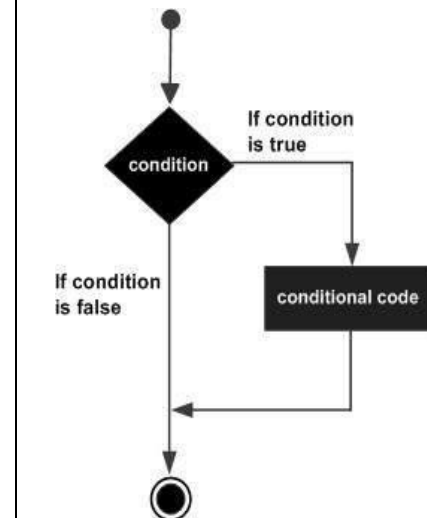
1. **Selection/Conditional branching statements**
 - a. if statement
 - b. if-else statement
 - c. if-elif-else statement
2. **Basic loop Structures/Iterative statement**
 - a. while loop
 - b. for loop

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:



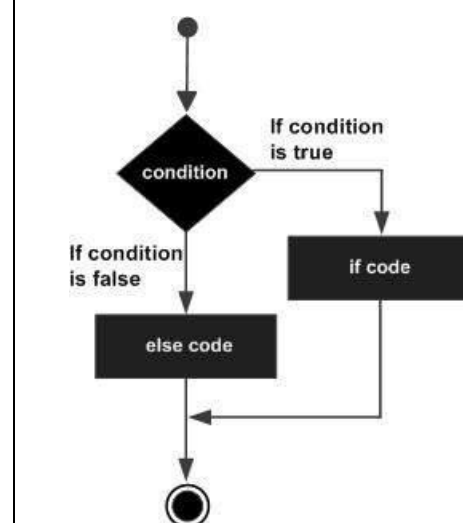
if-else Statement - An **else** statement can be combined with an **if** statement. An **else** statement contains the block of code that executes if the conditional expression in the **if** statement resolves to 0 or a FALSE value.

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:

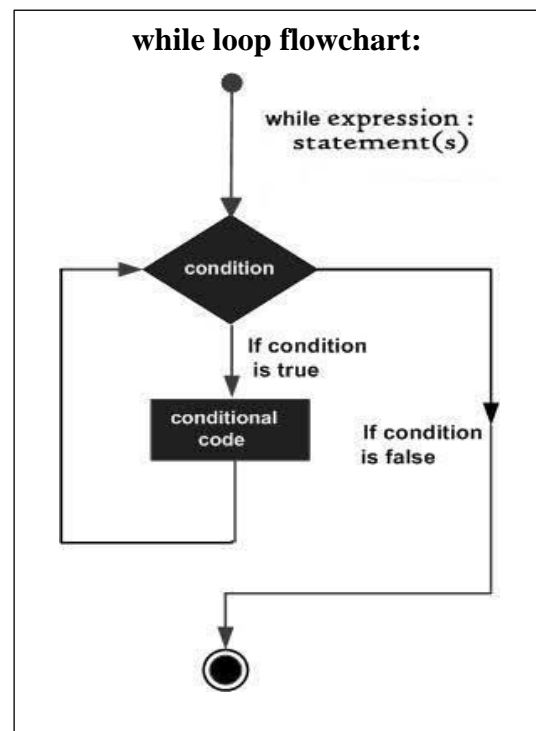
```
if expression1:
    statement(s)
elif expression2:
    statement(s)
elif expression3:
    statement(s)
else:
    statement(s)
```

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)
```

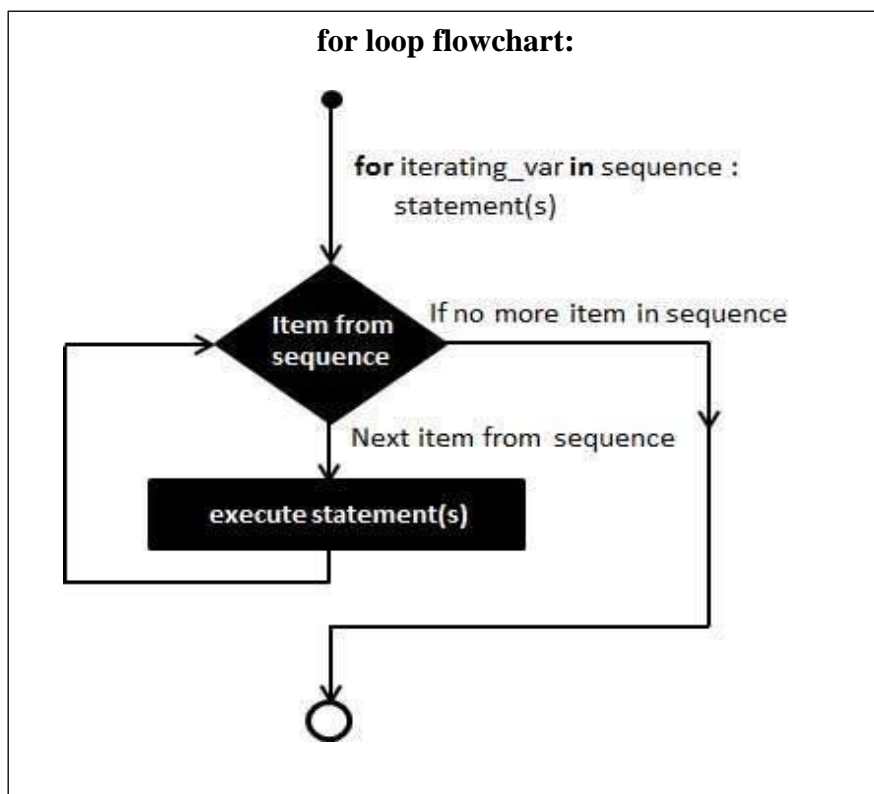
while loop flowchart:



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)
```



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 e.g., $153 = 1^3 + 5^3 + 3^3$)

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.

```
print("\nketaki mahajan / P1-2 / 16014022050")

#initial count for prime and composite (defining variables)
com_count = 0
pri_count = 0

while True:
    num = int(input("\nenter a number: "))

    if num == -1: #condition for -1
        print("break")
        break
    elif num <= 1: #negative number and 1 being exception
        print("neither prime nor composite")
        continue
    else:
        count = 0

        for i in range(2, num):

            if num % i == 0:
                count = count + 1

        if (count<=0): #checking for prime
            print(num, "is PRIME number")
            pri_count = pri_count + 1

        else: #else composite
            print(num, "is COMPOSITE number")
            com_count = com_count + 1

print("\nnumber of composite numbers: ", com_count)
print("number of prime numbers: ", pri_count)
```

2.

```
print("\nketaki mahajan / P1-2 / 16014022050")

num = input("\nenter a number: ")

digits = len(num)
sum = 0

for i in range(digits):
    sum = sum + int(num[i])**3
```

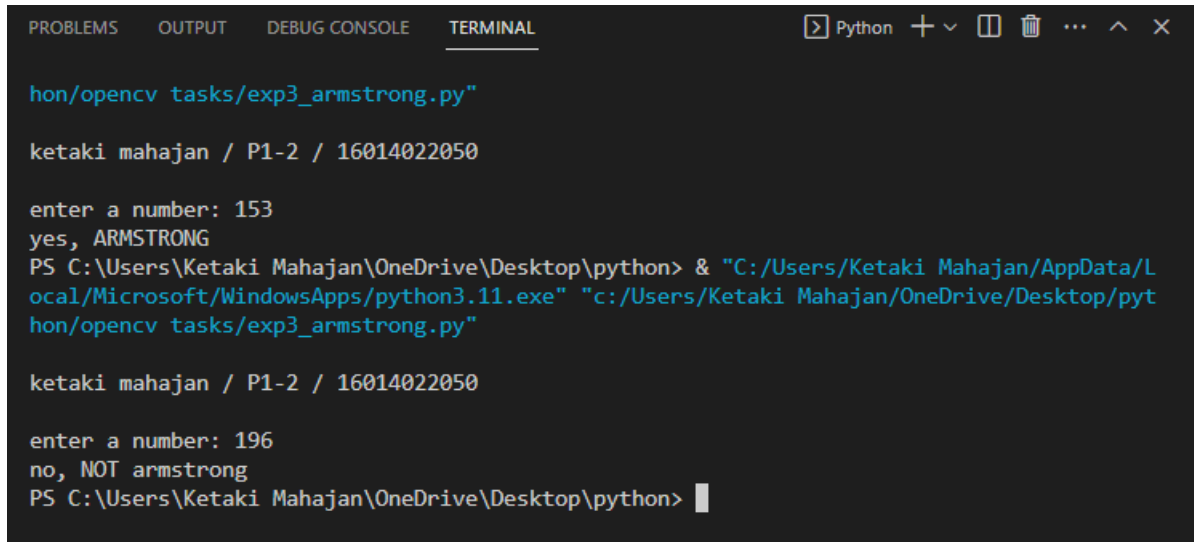
```
if int(num) == sum:  
    print("yes, ARMSTRONG")  
else:  
    print("no, NOT armstrong")
```

Output(s):

1.

```
PS C:\Users\Ketaki Mahajan\OneDrive\Desktop\python> & "C:/Users/Ketaki Mahajan/AppData/Local/Microsoft/WindowsApps/python3.10.0/python.exe" ketaki_mahajan / P1-2 / 16014022050  
ketaki mahajan / P1-2 / 16014022050  
  
enter a number: 2  
2 is PRIME number  
  
enter a number: 3  
3 is PRIME number  
  
enter a number: 4  
4 is COMPOSITE number  
  
enter a number: 5  
5 is PRIME number  
  
enter a number: 6  
6 is COMPOSITE number  
  
enter a number: 7  
7 is PRIME number  
  
enter a number: 8  
8 is COMPOSITE number  
  
enter a number: 9  
9 is COMPOSITE number  
  
enter a number: 10  
10 is COMPOSITE number  
  
enter a number: 12  
12 is COMPOSITE number  
  
enter a number: 13  
13 is PRIME number  
  
enter a number: 14  
14 is COMPOSITE number  
  
enter a number: 15  
15 is COMPOSITE number  
  
enter a number: -1  
break  
  
number of composite numbers: 8  
number of prime numbers: 5  
PS C:\Users\Ketaki Mahajan\OneDrive\Desktop\python>
```

2.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Python + - [] [X] ^ X
hon/opencv tasks/exp3_armstrong.py"
ketaki mahajan / P1-2 / 16014022050
enter a number: 153
yes, ARMSTRONG
PS C:\Users\Ketaki Mahajan\OneDrive\Desktop\python> & "C:/Users/Ketaki Mahajan/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/Ketaki Mahajan/OneDrive/Desktop/python/opencv tasks/exp3_armstrong.py"
ketaki mahajan / P1-2 / 16014022050
enter a number: 196
no, NOT armstrong
PS C:\Users\Ketaki Mahajan\OneDrive\Desktop\python> 
```

Conclusion:

Different decision-making statements and their respective syntax in Python were understood and implemented successfully by solving the defined problems.

Post-Lab Questions:

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

They're situations in real life when we need to make some decisions and based on these decisions, we decide what should we do next. Similar situations arise in programming also where we need to make some decisions and based on these decisions, we will execute the next block of code.

e.g.,

num = 15

if num >= 0:

if num == 0:

print("Zero")

else:

print ("Positive number")

else:

print ("Negative number")

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

Break statement breaks the execution and exits the loop.

e.g.,

#example 1 – output will print p, y and t before stopping at 5 and exiting the loop because of the break statement.

```
for letter in 'Python':  
    if letter == 'h':  
        break  
    print ("Current letter: ", letter)
```

#example 2 – output will print 10, 9, 8 before stopping at 7 and exiting the loop because of break statement.

```
var = 10  
while (var > 0):  
    print ("Current variable value: ", var)  
    var = var -1  
    if var == 7:  
        break
```

Continue statement skips the remaining block of code and jumps to the next iteration.

e.g.,

#example 1 – output will print p, y, t, o, n as when letter == h condition is true, continue statement is run causing it to reject all remaining statements in current iteration and moves control back to top of loop.

```
for letter in 'Python':  
    if letter == 'h':  
        continue  
    print "Current Letter: ", letter)
```

#example 2 – output will print all numbers from 0 to 9 except 5 as the continue statement is run upon the condition being true, causing it to reject all remaining statements in current iteration and moves control back to top of loop.

```
var = 10  
while (var > 0):  
    var = var -1  
    if var == 5:  
        continue  
    print ("Current variable value: ", var)
```


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

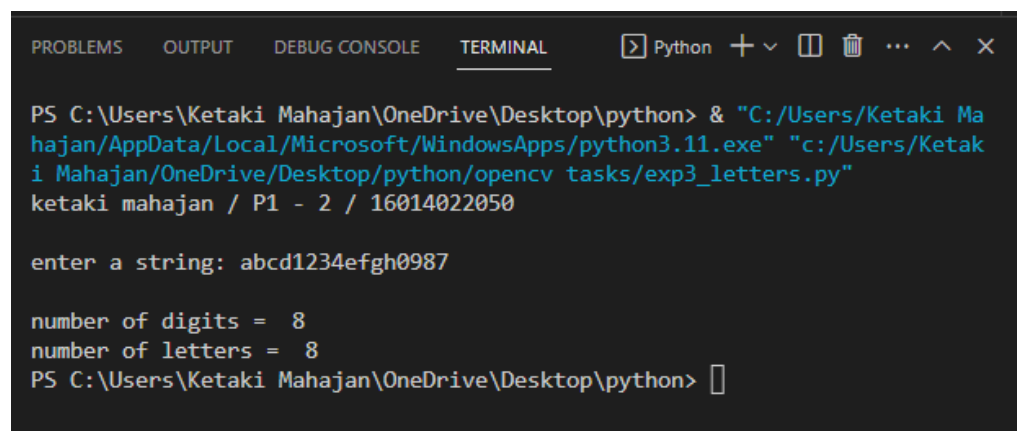
```
print("ketaki mahajan / P1 - 2 / 16014022050")

str = input("\nenter a string: ")

digits = 0 #initializing variables
letters = 0

for char in str:
    if char.isdigit(): #uses isDigit() function to return true
        #when characters are digits
        digits = digits + 1
    elif char.isalpha(): #uses isalphs() function to return true
        #when characters are alphabets
        letters = letters + 1

print("\nnumber of digits = ", digits)
print("number of letters = ", letters)
```



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Python + - [ ] [ ] ... ^ X

PS C:\Users\Ketaki Mahajan\OneDrive\Desktop\python> & "C:/Users/Ketaki Mahajan/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/Ketaki Mahajan/OneDrive/Desktop/python/opencv tasks/exp3_letters.py"
ketaki mahajan / P1 - 2 / 16014022050

enter a string: abcd1234efgh0987

number of digits = 8
number of letters = 8
PS C:\Users\Ketaki Mahajan\OneDrive\Desktop\python> [ ]
```

Date: _____

Signature of faculty in-charge