KALYANI GOVERNMENT ENGINEERING COLLEGE
Department of Computer Application

CONTROL STATEMENT AND ITERATION

NAME - MADHUSUDAN CHAND
ROLL NO. - 10271023013
REG. NO. - 231020510013
STREAM - MCA
SEMESTER - 1ST

Paper - Programming Concept through Python
Paper Code - MCAN101

Table of CONTENTS

INTRODUCTION

Programs are written for the solution to the real world problems. A language should have the ability to control the flow of the execution so that at different intervals different statements can be executed. Structured programming is a paradigm aims at controlling the flow of the execution of the statements on a program by using control structures. A language which supports the control structures is called as structured programming language.

Control Statement

Selection

A selection statement is a control statement causes the program control to be transferred to a specific flow based upon whether a certain condition is true or not.

Conditional Construct - if-else statement

Conditional Construct (also known as if statements) provide a way to execute a chosen block of code based on the run-time evaluation of one or more Boolean expressions.

Each condition is a Boolean expression , and each body contains one or more commands that are to be executed conditionally.
If the first condition succeeds , the first body will executed; no other conditions or bodies are evaluated in that case .

```
if first condition:
    first body
elif second condition:
    second body
else:
    third body
```

Control Statement

Conditional Construct - if-else statement

If the first condition fails , then the process continues in similar manner with the evaluation of the second condition. The execution of this overall construct will cause precisely one of the bodies to be executed.

There may be any number of elif clauses (including zero), and the final else is optional.

Code:

```
def if_else_example():
    age = 27
    if age >= 60:
        print("Senior Discount")
    elif 18 <= age < 60:
        print("No Discount")
    else:
        print("Junior Discount")

if_else_example()
```

Output:

```
No Discount
```

ITERATION

What is loop or iteration?

Loops can execute a block of code number of times until a certain condition met. The iteration statement allows instructions to be executed until a certain condition is to be fulfilled.

The iteration statements are also called as loops or looping statements.
Python provides two kinds of loops and the are -

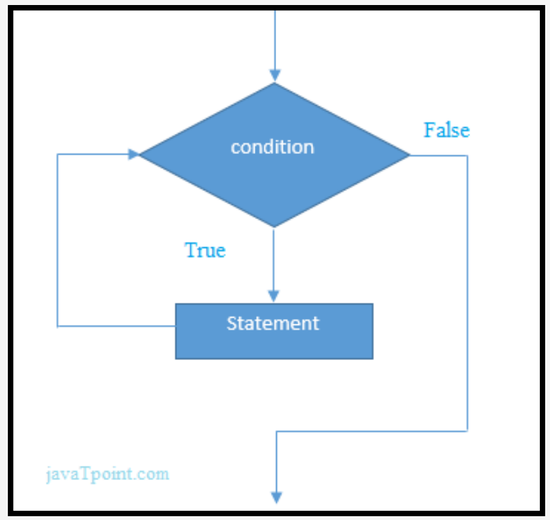
- while loop
- for loop

ITERATION

- **WHILE LOOP**

A while loop allows general repetition based upon the repeated testing of a Boolean condition.
The syntax for a while loop in python is as follows

```
while condition :
    body
```



Where loop body contain the single statement or set of statement or an empty statement.

The loop iterates while the expression evaluates to true , when expression becomes false the loop halts.

ITERATION

- WHILE LOOP

```python
#sum of n natural number
try:
    n=int(input("Enter the range\n"))
    count=1
    sum=0
    while count<n:
        sum += count
        count += 1
    print("Sum of",count,"natural number",sum)
except:
    print("Invalid input")
```

```
Enter the range
10
Sum of 10 natural number 45
```

ITERATION

- for loop

Python’s for-loop syntax is a more convenient alternative to a while loop when iterating through a series of elements. The for-loop syntax can be used on any type of iterable structure, such as a list , tuple str,set,dict or file. Syntax or general format of for loop is

```
for element in iterable:
        body
```

- for loop - range keyword

The range() function returns a sequence of numbers , strating from 0 by default and increments by 1 (by default) and ends at a specified number.

range (start , stop , step)

Another function is used i.e len() into range keyword. Example in next slide

ITERATION

```
#printing string char by char
def for_loop_example():
    name=input("Enter string: ")
    for i in range(0,len(name)):
        print(name[i])
for_loop_example()
```

```
Enter string: Sainik
S
a
i
n
i
k
```

BRANCHING OR JUMPING STATEMENTS

Python has an unconditional branching statements and they are

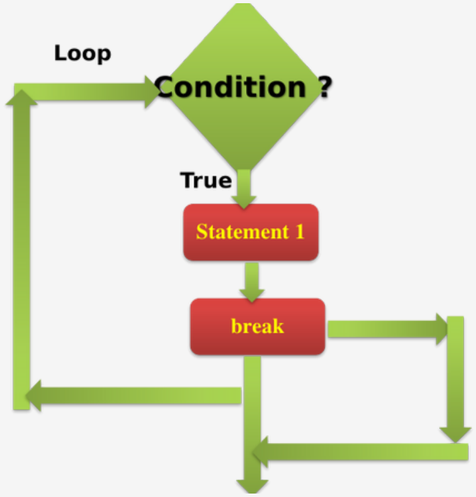
- break statement
- continue statement

Break can be used to unconditionally jump out of the loop. It terminates the execution of the loop. Break can be used in while or for loop break is mostly required , when because of some external condition, we need to exit from a loop.

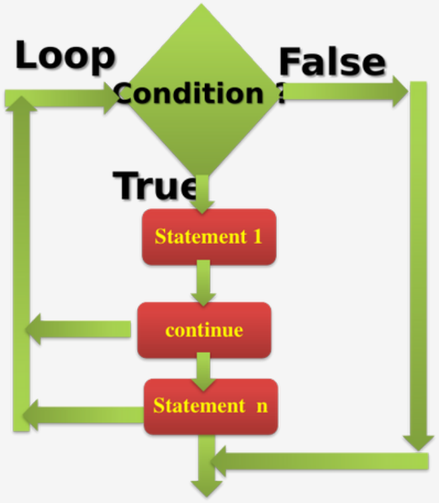
The continue statement in Python returns the control to the beginning of the while loop. The continue statement rejects all the remaining statements in the current iteration of the loop and moves the control back to the top of the loop. The continue statement can be used in both while and for loops.

The pass statement in Python is used when a statement is required syntactically but you don't want any command or code to execute. It is a null operation; nothing happens when it executes.

Break flow-chart

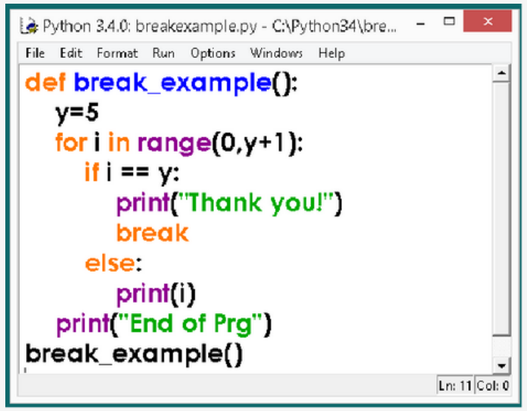


Continue flow-chart



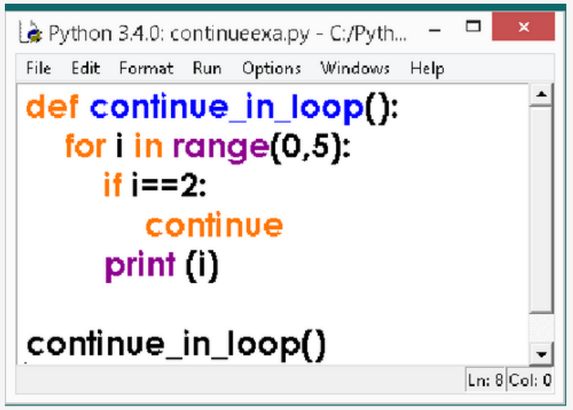
BRANCHING OR JUMPING STATEMENTS

using break statement




```
0
1
2
3
4
Thank you!
End of Prg
```

using continue statement




```
>>>
0
1
3
4
>>>
```

CONCLUSION

In Python, Loops are used to iterate repeatedly over a block of code. To change the way a loop is executed from its usual behavior, we use control statements in python. Control statements are used to control the flow of the execution of the loop based on a condition.

A loop repeats a sequence of instructions until a specific condition is met. Loops allow you to repeat a process over & over without having to write the same instructions each time you want your program to perform a task.

THANK YOU