Module - 1

Python Fundamentals

Control Flow

Presentation By

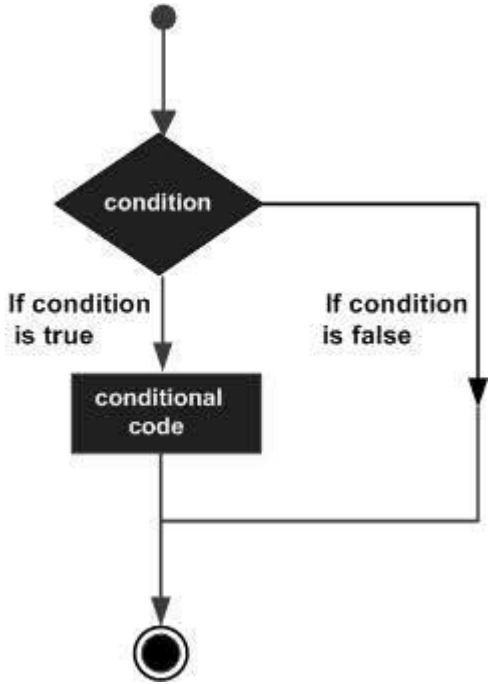
Mohammed Tahir Mirji

MTech CS

CONTENTS

Introduction to Control Flows

- A program's control flow is the order in which the program's code executes.
- The control flow of a Python program is regulated by conditional statements, loops, and function calls.



Python has three types of control structures:

1. **Sequential** - default mode
2. **Selection** - used for decisions and branching
3. **Repetition** - used for looping, i.e., repeating a piece of code multiple times.

Introduction to Control Flows(contd.)

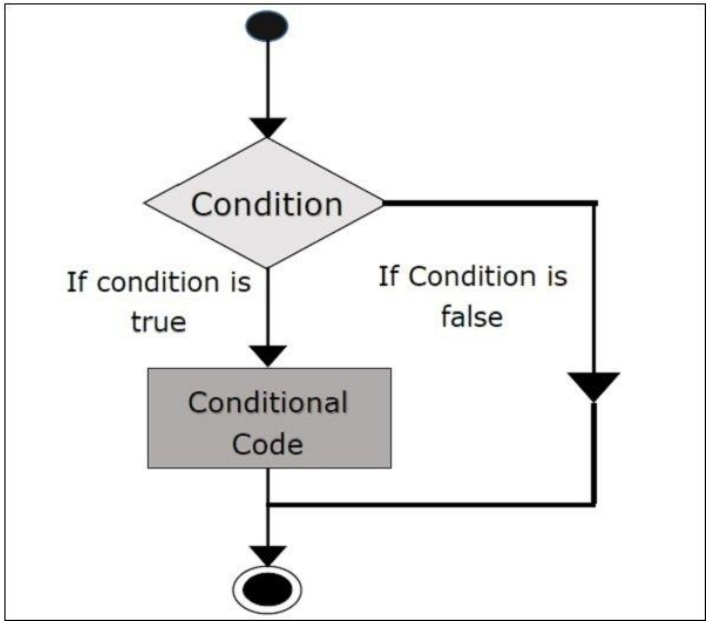
- **Sequential:** Sequential statements are a set of statements whose execution process happens in a sequence.

- **Selection/Decision control statements:** The selection statement allows a program to test several conditions and execute instructions based on which condition is true.(Simple if, if-else, nested if, if-elif-else)

- **Repetition:** A repetition statement is used to repeat a group(block) of programming instructions.

Introduction to simple if

- **Simple if:** If statements are control flow statements that help us to run a particular code, but only when a certain condition is met or satisfied.
- A simple if only has one condition to check.




```
n = 10
if n % 2 == 0:
   print("n is an even number")
```

What are if else

- **if-else**: The if-else statement evaluates the condition and will execute the body of **if,** if the test condition is True, but if the condition is False, then the body of **else** is executed.

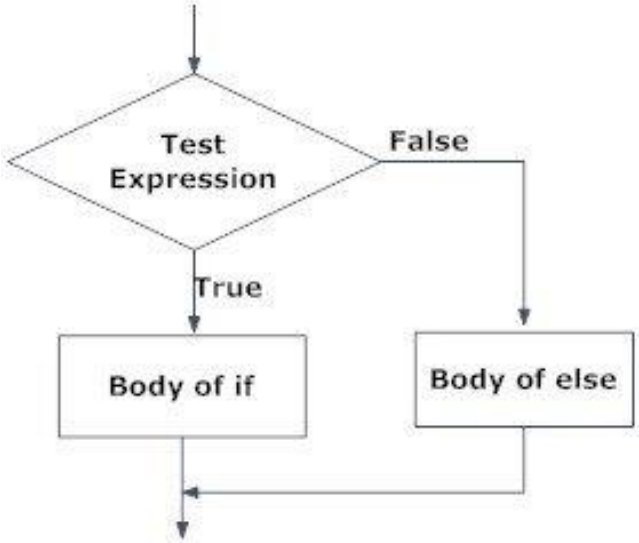


Fig: Operation of if...else statement

```
n = 5
if n % 2 == 0:
    print("n is even")
else:
    print("n is odd")
```

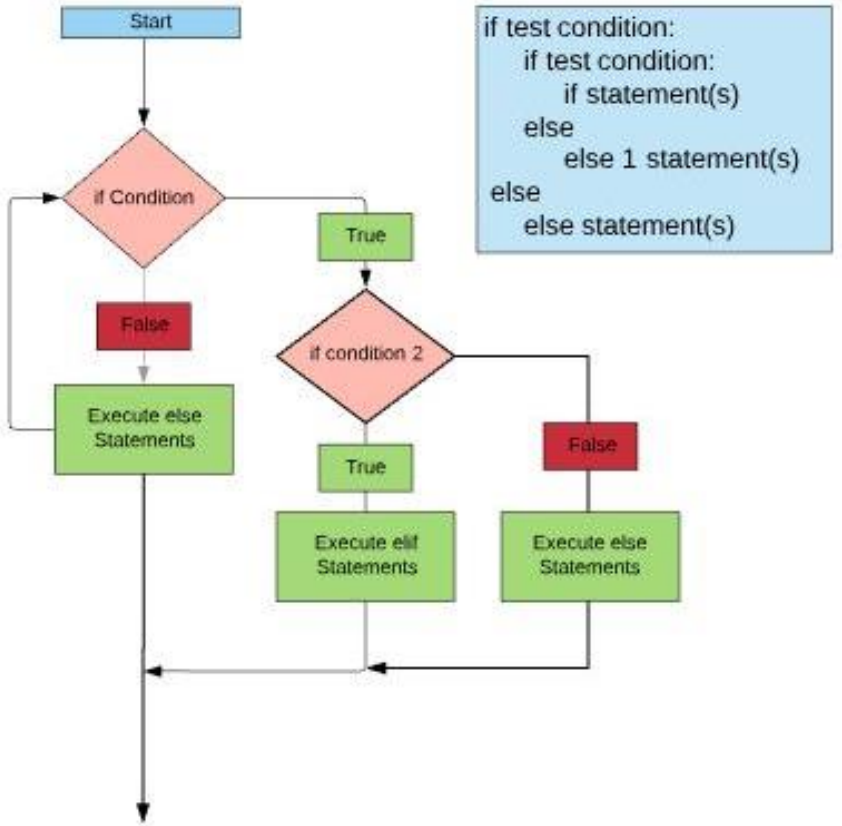
Shorthand if-else

```
i = 10
if i < 15 print(True) else print(False)
```

What are Nested if

- **nested if**: Nested if statements are an if statement inside another if statement.

```
a = 5;  b = 10; c = 15
if a > b:
   if a > c:
      print("a value is big")
   else:
       print("c value is big")
elif b > c:
    print("b value is big")
else:
     print("c is big")
```



What is If-elif-else

- **if-elif-else:** The if-elif-else statement is used to conditionally execute a statement or a block of statements.

```
x = 15
y = 12
if x == y:
   print("Both are Equal")
elif x > y:
    print("x is greater than y")
else:
    print("x is smaller than y")
```

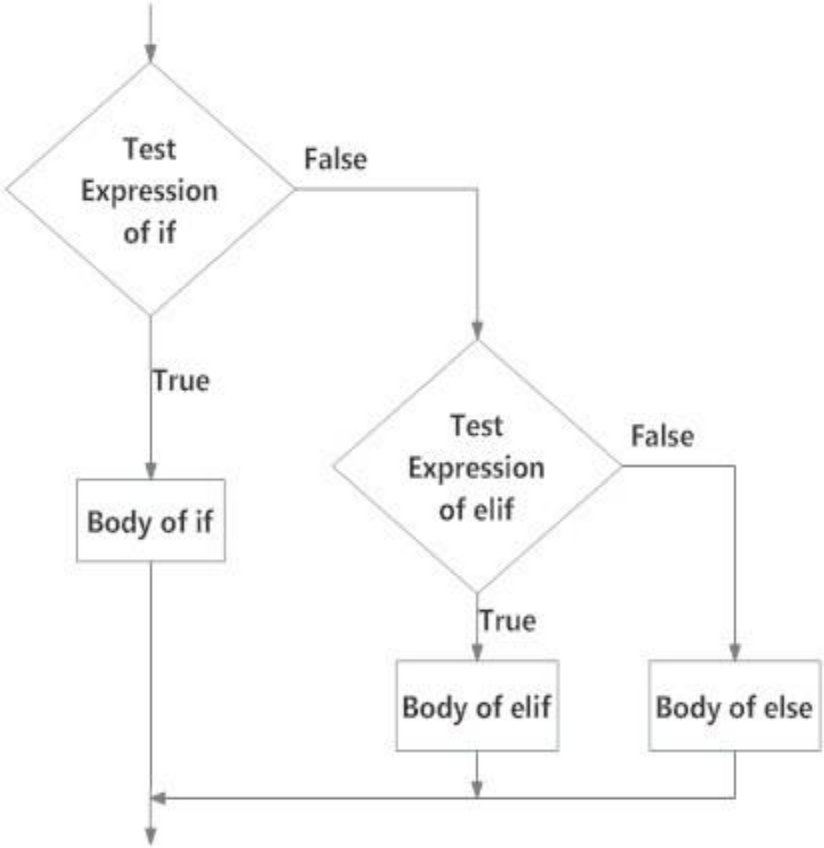
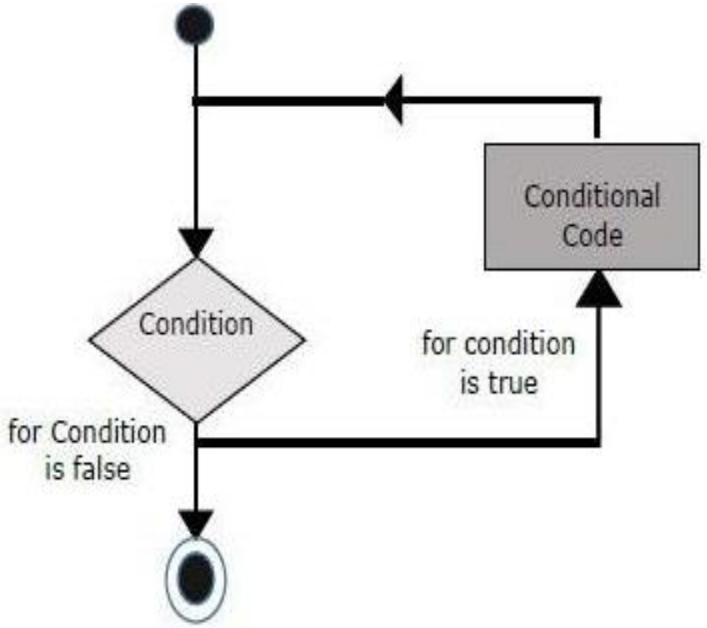


Fig: Operation of if...elif...else statement

Repeat code with for loops

- **for loop:** A *for loop* is used to iterate over a sequence that is either a list, tuple, dictionary, or a set.
- We can execute a set of statements once for each item in a list, tuple, or dictionary.

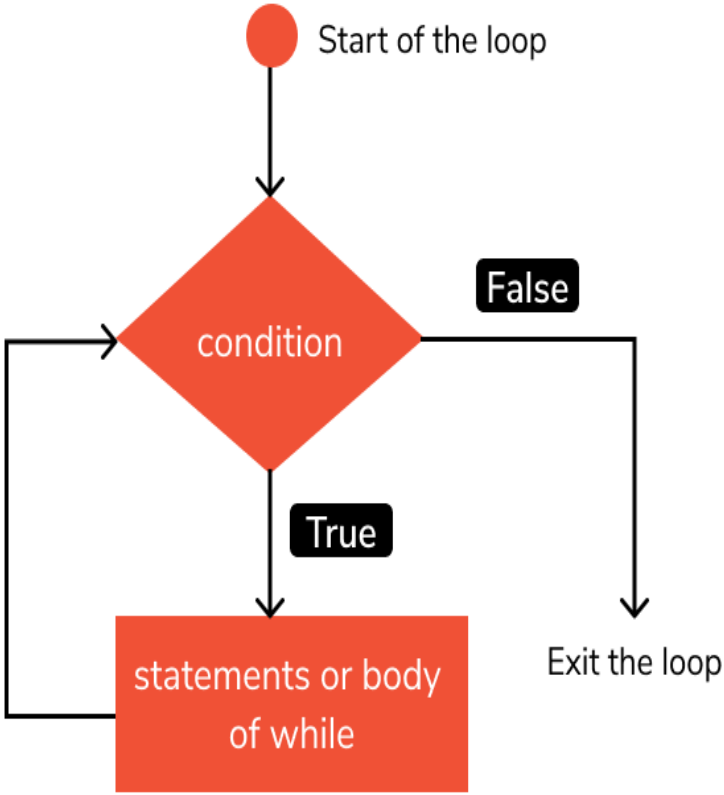



```
lst = [1, 2, 3, 4, 5]
for i in range(len(lst)):
    print(lst[i], end = " ")
for j in range(0,10):
    print(j, end = " ")
```

Repeat code with while loops

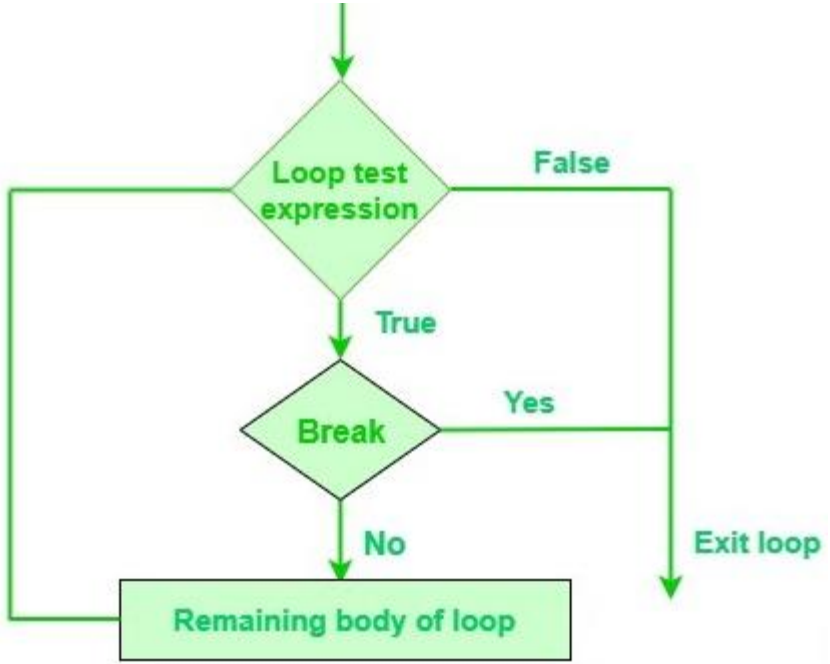
- **while loop:** In Python, ***while*** *loops* are used to execute a block of statements repeatedly until a given condition is satisfied.
- Then, the expression is checked again and, if it is still true, the body is executed again. This continues until the expression becomes false.

```
m = 5
i = 0
while i < m:
    print(i, end = " ")
    i = i + 1
print("End")
```



Exit a loop with break

- Python **break** is used to terminate the execution of the loop.
- **break** statement in Python is used to ***bring the control out of the loop*** when some external condition is triggered. break statement is put inside the loop body (generally after if condition).
- It terminates the current loop, i.e., the loop in which it appears, and resumes execution at the next statement immediately after the end of that loop.



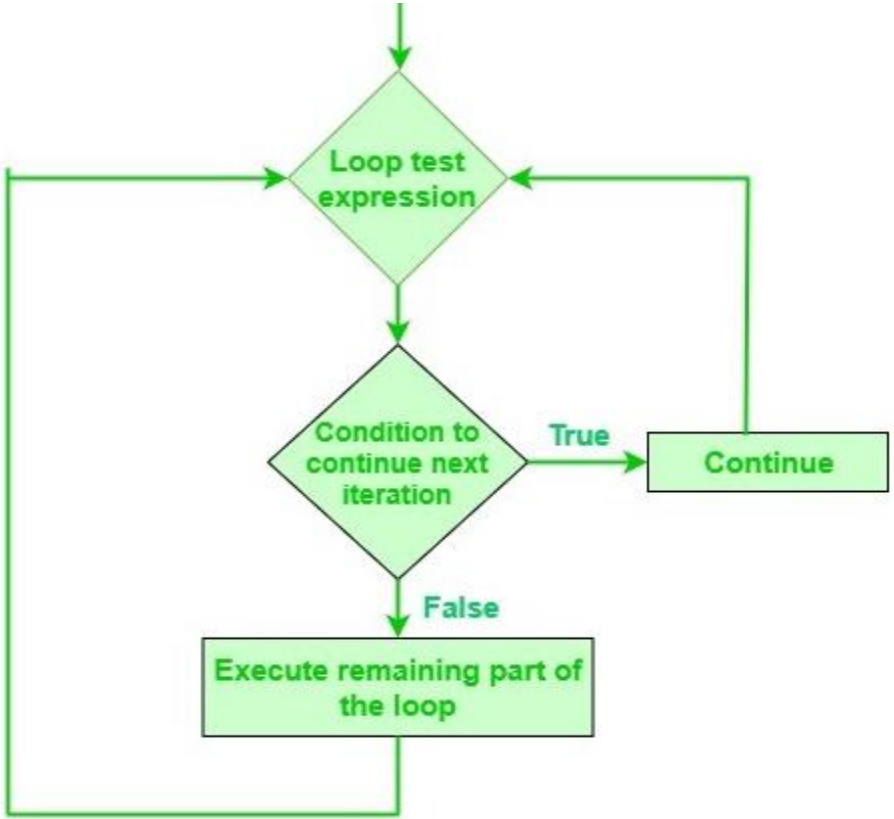

```
for i in range(10):
    print(i)
    if i == 2:
        break
print("Outside loop")
```

What is Continue

Continue Statement ends the iteration from loops and start to continue the next iteration.

- **Python Continue statement** is a loop control statement that forces to execute the next iteration of the loop while skipping the rest of the code inside the loop for the current iteration only.

- when the continue statement is executed in the loop, the code inside the loop following the continue statement will be skipped for the current iteration and the next iteration of the loop will begin.




```
for var in “Python world":
    if var == "e":
        continue
    print(var)
```

What are functions

- **Functions** are just bundled sets of instructions that we can repeatedly use without having to write them again. They make the code more concise and readable in any language, including Python.
- You can pass data, known as parameters, into a function.
- A function can return data as a result.

Types of functions:

1. Built-in (A built-in function is a function that is already available in a programming language)
2. Recursion (it is a process in which a function calls itself directly or indirectly)
3. Lambda (Python Lambda Functions are anonymous function means that the function is without a name)
4. user-defined (User-defined functions are functions that you use to organize your code)

Useful functions

len() : It takes one input and it outputs a whole number representing the length of the input provided.

String1=“Hello world”

Print(**len**(String1))

Str() : The **str**() function converts the specified value into a string.

```
 pi = 3.14
# text = 'The value of pi is ' + pi
text = 'The value of pi is '  + str(pi)  # yes
```

Useful functions(contd.)

zip() : zip() method takes iterable or containers and returns a single iterator object, having mapped values from all the containers.

```
name = [ “Sagar", “Hemanth", “Yusuf", “Pushpa" ]
roll_no = [ 4, 1, 3, 2 ]
 # using zip() to map values
mapped = zip(name, roll_no)
 print(set(mapped))
```

Useful functions(contd.)

Enumerate() : Often, when dealing with iterators, we also get need to keep a count of iterations. Python eases the programmers' task by providing a built-in function enumerate() for this task. Enumerate() method adds a counter to an iterable and returns it in a form of enumerating object. This enumerated object can then be used directly for loops or converted into a list of tuples using the list() method.

```
# Python program to illustrate
# enumerate function
L1 = ["eat", "sleep", "repeat"]
S1 = "Python"

# creating enumerate objects
obj1 = enumerate(l1)
obj2 = enumerate(s1)

print ("Return type:", type(obj1))
print (list(enumerate(l1)))

# changing start index to 2 from 0
print (list(enumerate(s1, 2)))
```

Assignments Control Flows Module are uploaded with below link

Refer Github repository ai_with_python_keonics for list of assignments:

https://github.com/tahirmirji/ai_with_python_keonics

Thank You