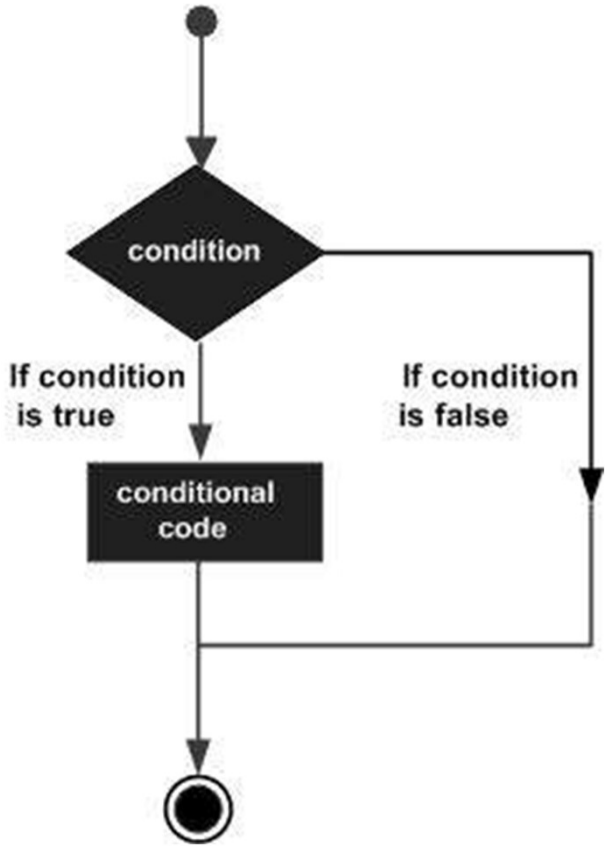
Programação Orientada a Objetos II

Tiago Bacciotti Moreira

Introdução

Fonte: https://www.tutorialspoint.com/python3/python_overview.htm


condition
If condition
is true
If condition
is false
conditional
code

```
#!/usr/bin/python3

var = 100
if ( var  == 100 ) : print ("Value of expression is 100")
print ("Good bye!")
```

Output

When the above code is executed, it produces the following result −

```
Value of expression is 100
Good bye!
```

```
#!/usr/bin/python3

var1 = 100
if var1:
   print ("1 - Got a true expression value")
   print (var1)

var2 = 0
if var2:
   print ("2 - Got a true expression value")
   print (var2)
print ("Good bye!")
```

Output

When the above code is executed, it produces the following result −

```
1 - Got a true expression value
100
Good bye!
```

Example

```
#!/usr/bin/python3

amount = int(input("Enter amount: "))

if amount<1000:
   discount = amount*0.05
   print ("Discount",discount)
else:
   discount = amount*0.10
   print ("Discount",discount)

print ("Net payable:",amount-discount)
```

```
Enter amount: 600
Discount 30.0
Net payable: 570.0

Enter amount: 1200
Discount 120.0
Net payable: 1080.0
```

The elif 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 the 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)
```

Example

```
#!/usr/bin/python3

amount = int(input("Enter amount: "))

if amount<1000:
   discount = amount*0.05
   print ("Discount",discount)
elif amount<5000:
   discount = amount*0.10
   print ("Discount",discount)
else:
   discount = amount*0.15
   print ("Discount",discount)

print ("Net payable:",amount-discount)
```

When the above code is executed, it produces

```
Enter amount: 600
Discount 30.0
Net payable: 570.0

Enter amount: 3000
Discount 300.0
Net payable: 2700.0

Enter amount: 6000
Discount 900.0
Net payable: 5100.0
```

```
if expression1:
   statement(s)
   if expression2:
      statement(s)
   elif expression3:
      statement(s)
   else
      statement(s)
elif expression4:
   statement(s)
else:
   statement(s)
```

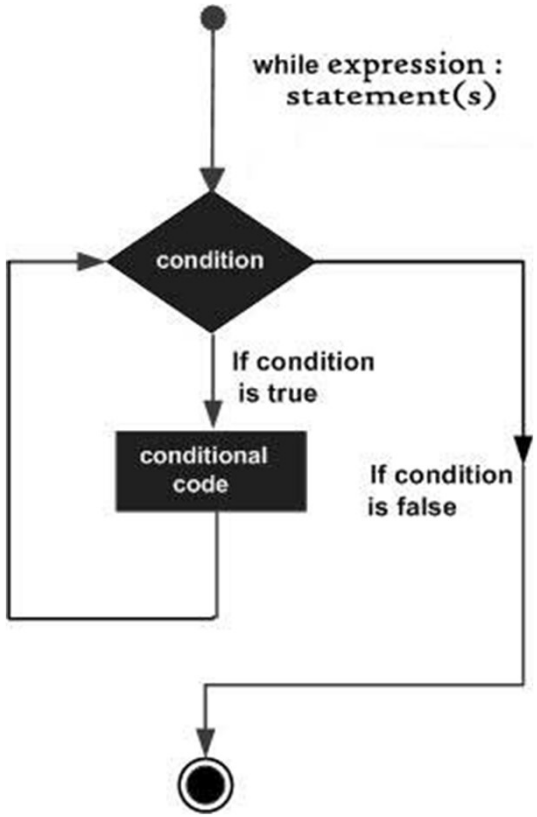
```
num = int(input("enter number"))
if num%2 == 0:
   if num%3 == 0:
      print ("Divisible by 3 and 2")
   else:
      print ("divisible by 2 not divisible by 3")
else:
   if num%3 == 0:
      print ("divisible by 3 not divisible by 2")
   else:
      print  ("not Divisible by 2 not divisible by 3")
```

```
enter number8
divisible by 2 not divisible by 3

enter number15
divisible by 3 not divisible by 2

enter number12
Divisible by 3 and 2

enter number5
not Divisible by 2 not divisible by 3
```


while expression :
statement(s)
condition
If condition
is true
conditional
code
If condition
is false

```
#!/usr/bin/python3

count = 0
while (count < 9):
   print ('The count is:', count)
   count = count + 1

print ("Good bye!")
```

```
The count is: 0
The count is: 1
The count is: 2
The count is: 3
The count is: 4
The count is: 5
The count is: 6
The count is: 7
The count is: 8
Good bye!
```

The Infinite Loop

A loop becomes infinite loop if a condition never becomes FALSE. You must be cautious when using while loops because of the possibility that this condition never resolves to a FALSE value. This results in a loop that never ends. Such a loop is called an infinite loop.

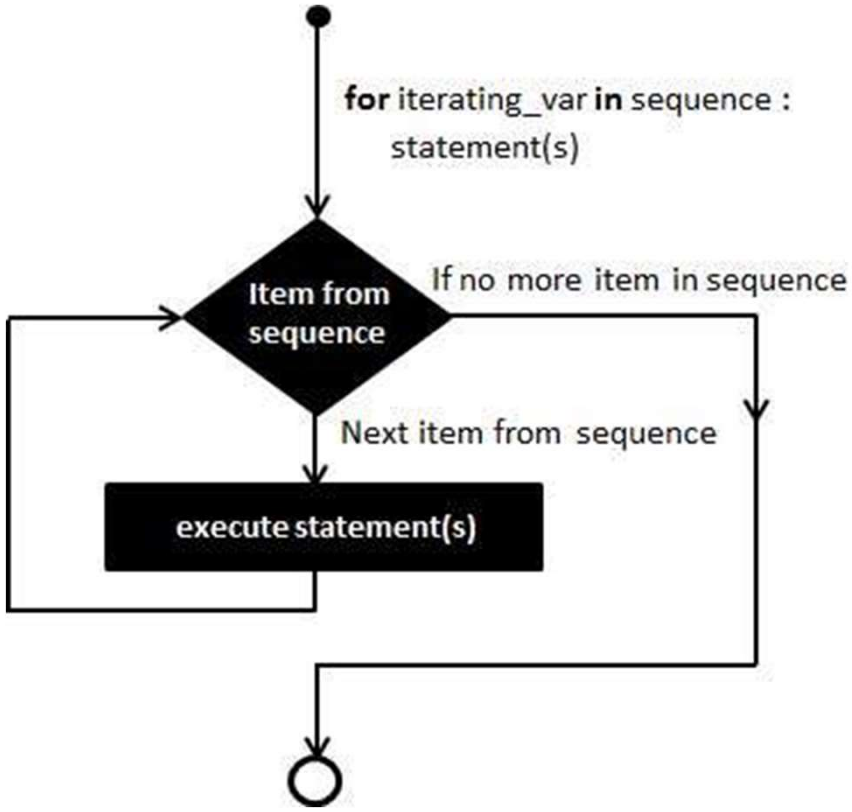
An infinite loop might be useful in client/server programming where the server needs to run continuously so that client programs can communicate with it as and when required.

Example

```
#!/usr/bin/python3

var = 1
while var == 1 :  # This constructs an infinite loop
   num = int(input("Enter a number  :"))
   print ("You entered: ", num)

print ("Good bye!")
```


for iterating_var in sequence :
statement(s)
Item from sequence
If no more item in sequence
Next item from sequence
execute statement(s)

The range() function

The built-in function range() is the right function to iterate over a sequence of numbers. It generates an iterator of arithmetic progressions.

Example

```
>>> range(5)
range(0, 5)
>>> list(range(5))
[0, 1, 2, 3, 4]
```

Example

range() generates an iterator to progress integers starting with 0 upto n-1. To obtain a list object of the sequence, it is typecasted to list(). Now this list can be iterated using the for statement.

```
>>> for var in list(range(5)):
   print (var)
```

Output

This will produce the following output.

```
0
1
2
3
4
```

```
#!/usr/bin/python3

for letter in 'Python':     # traversal of a string sequence
   print ('Current Letter :', letter)
print()
fruits = ['banana', 'apple',  'mango']

for fruit in fruits:        # traversal of List sequence
   print ('Current fruit :', fruit)

print ("Good bye!")
```

Output

When the above code is executed, it produces the following result −

```
Current Letter : P
Current Letter : y
Current Letter : t
Current Letter : h
Current Letter : o
Current Letter : n

Current fruit : banana
Current fruit : apple
Current fruit : mango
Good bye!
```

```
#!/usr/bin/python3

fruits = ['banana', 'apple',  'mango']
for index in range(len(fruits)):
   print ('Current fruit :', fruits[index])

print ("Good bye!")
```

Output

When the above code is executed, it produces the following result −

```
Current fruit : banana
Current fruit : apple
Current fruit : mango
Good bye!
```

```
#!/usr/bin/python3

numbers = [11,33,55,39,55,75,37,21,23,41,13]

for num in numbers:
   if num%2 == 0:
      print ('the list contains an even number')
      break
else:
   print ('the list doesnot contain even number')
```

Output

When the above code is executed, it produces the following result −

```
the list does not contain even number
```