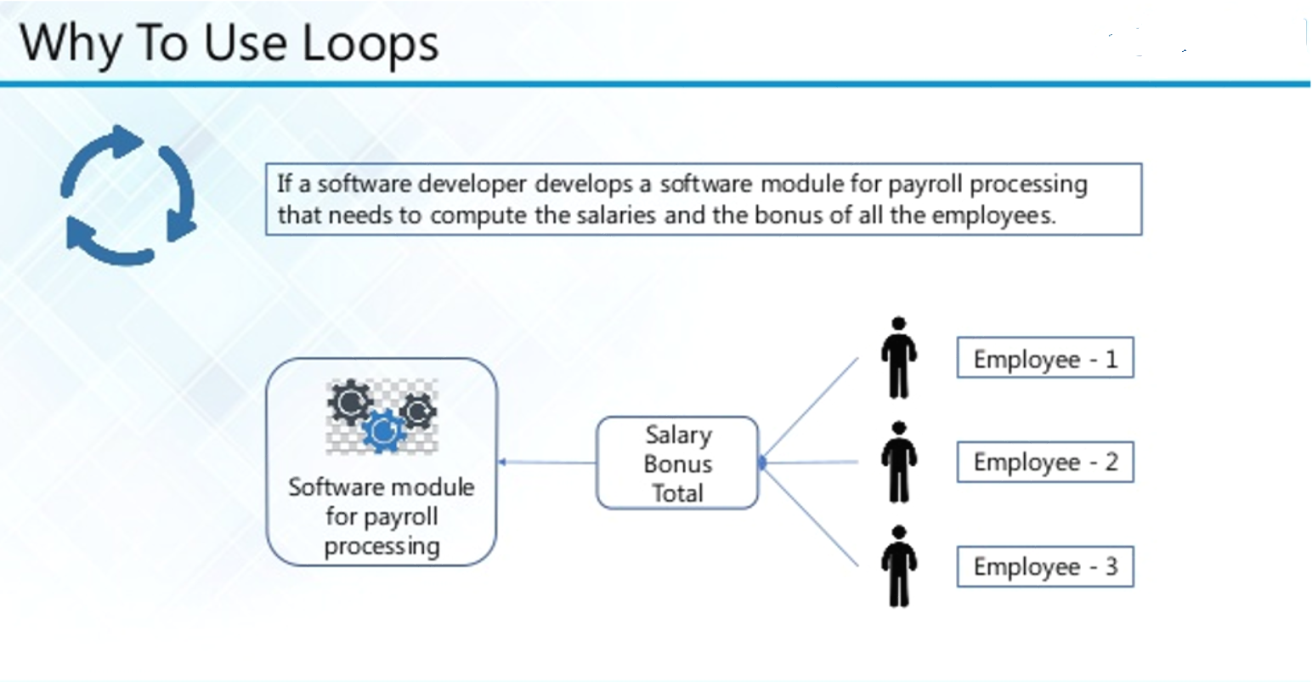
**7. Loops,Control Statements**

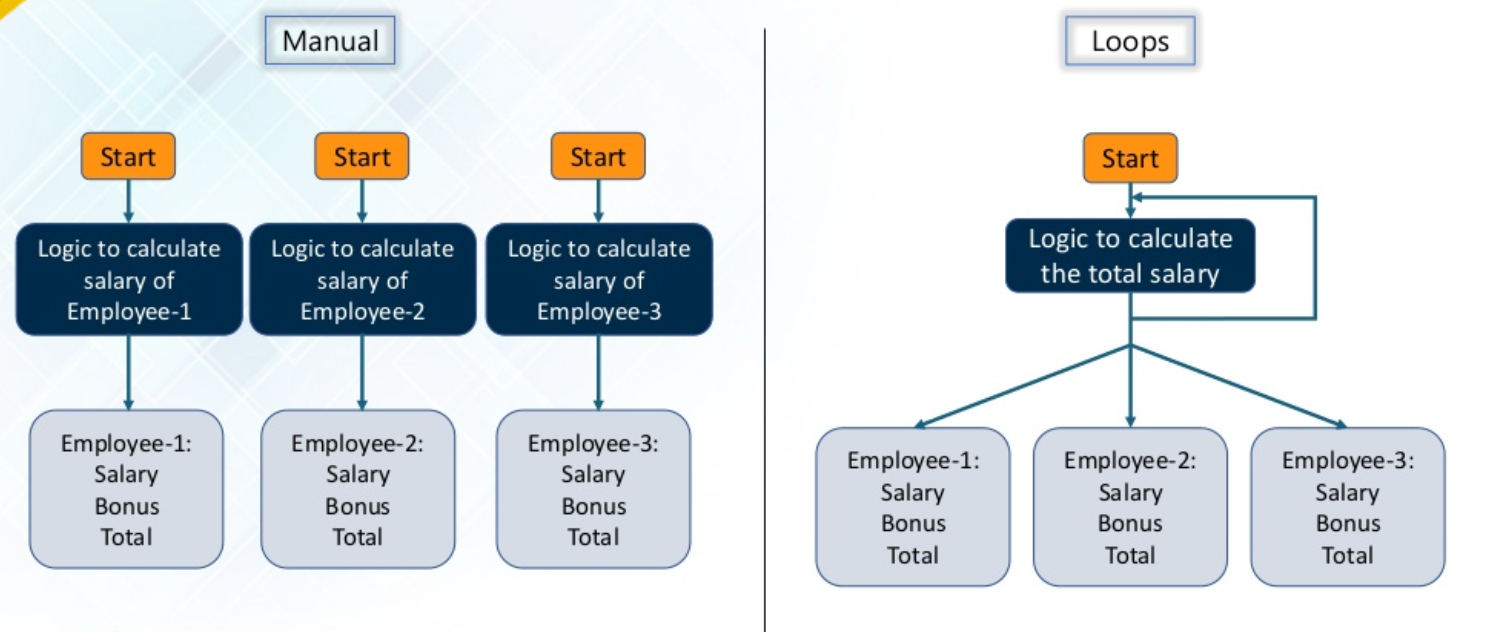
Types of loops in python,

1. **while**
2. **for**
3. **nested**



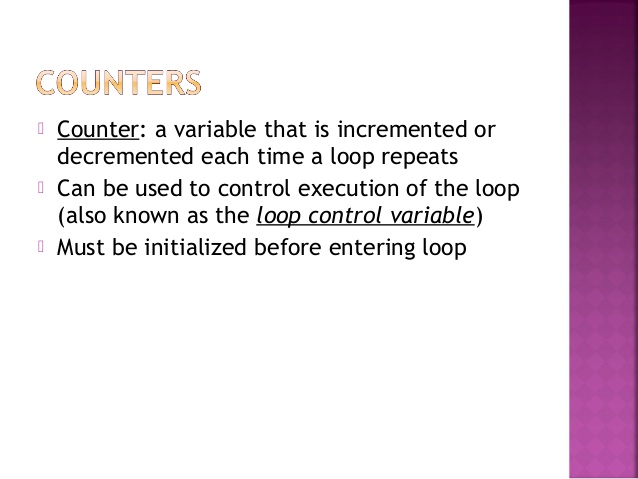
**Why to use loops :**



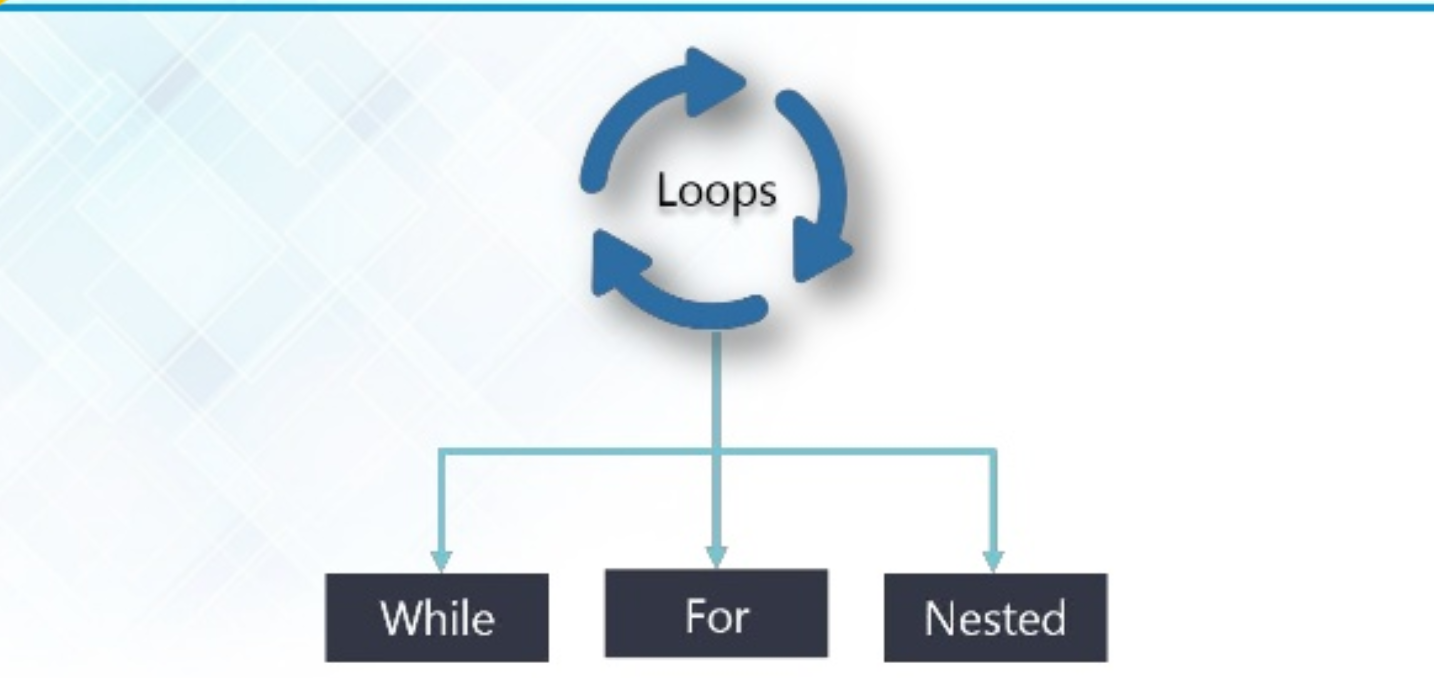


â¢ When a program repeatedly runs a set of
statements it is referred to as a loop, iteration
or repetition structure.
â¢ The...

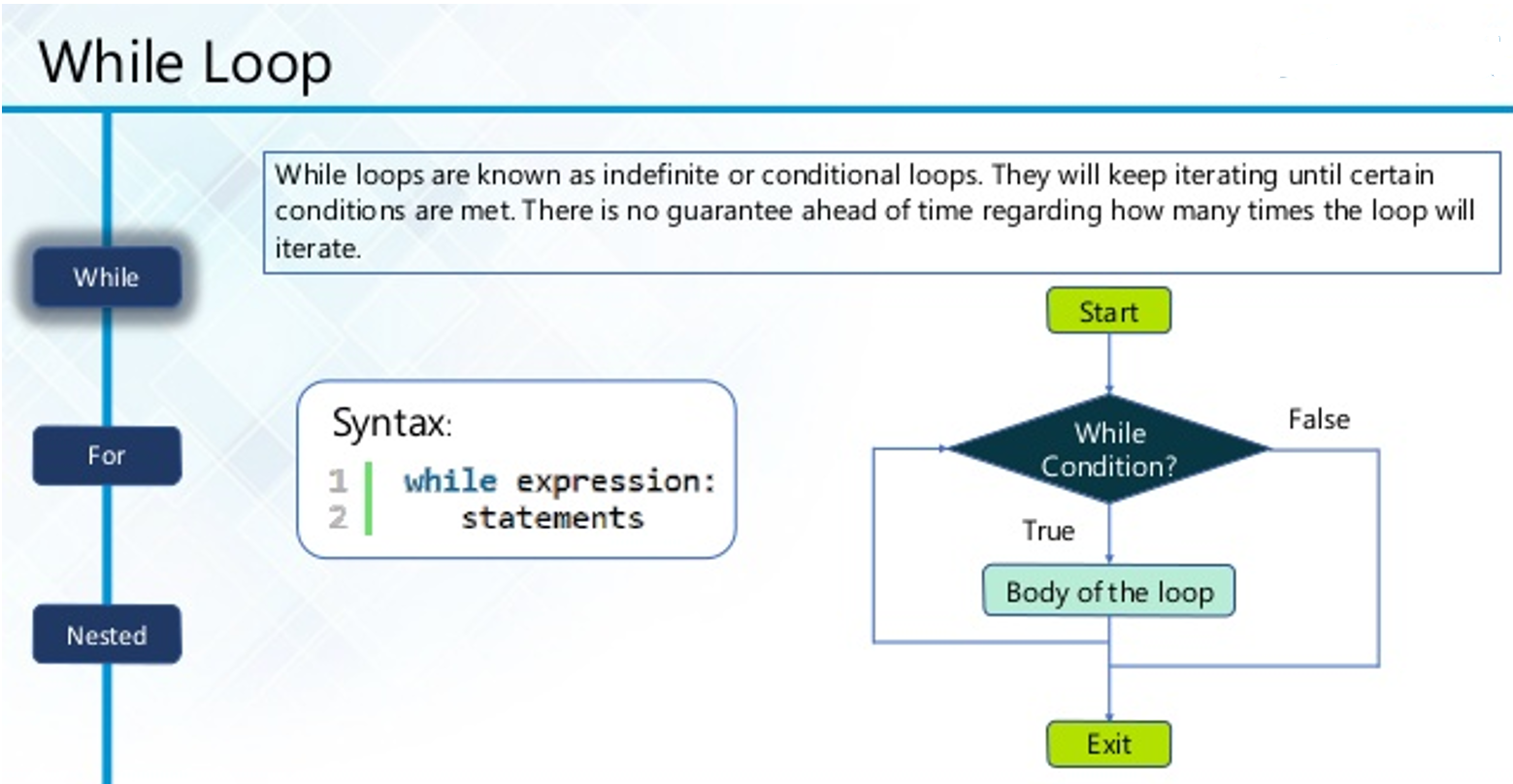
ï Question: How many statements are needed
for the following?
ï¡ Ask the same question 100 times
ï¡ Store the answer in a va...â¢ Loops are similar to conditionals because they run
on a true/false value set. The loop continuously
runs while the condi...

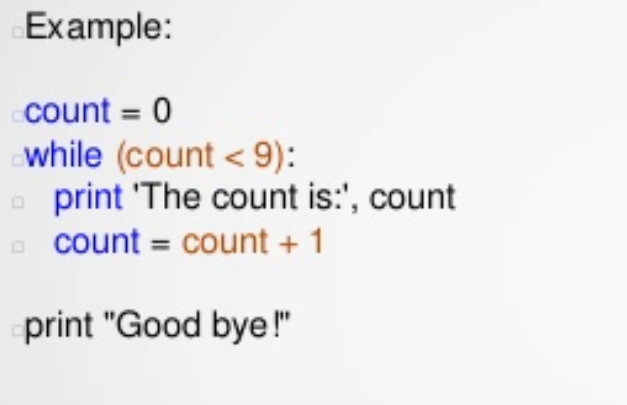


**LOOP TYPES :**

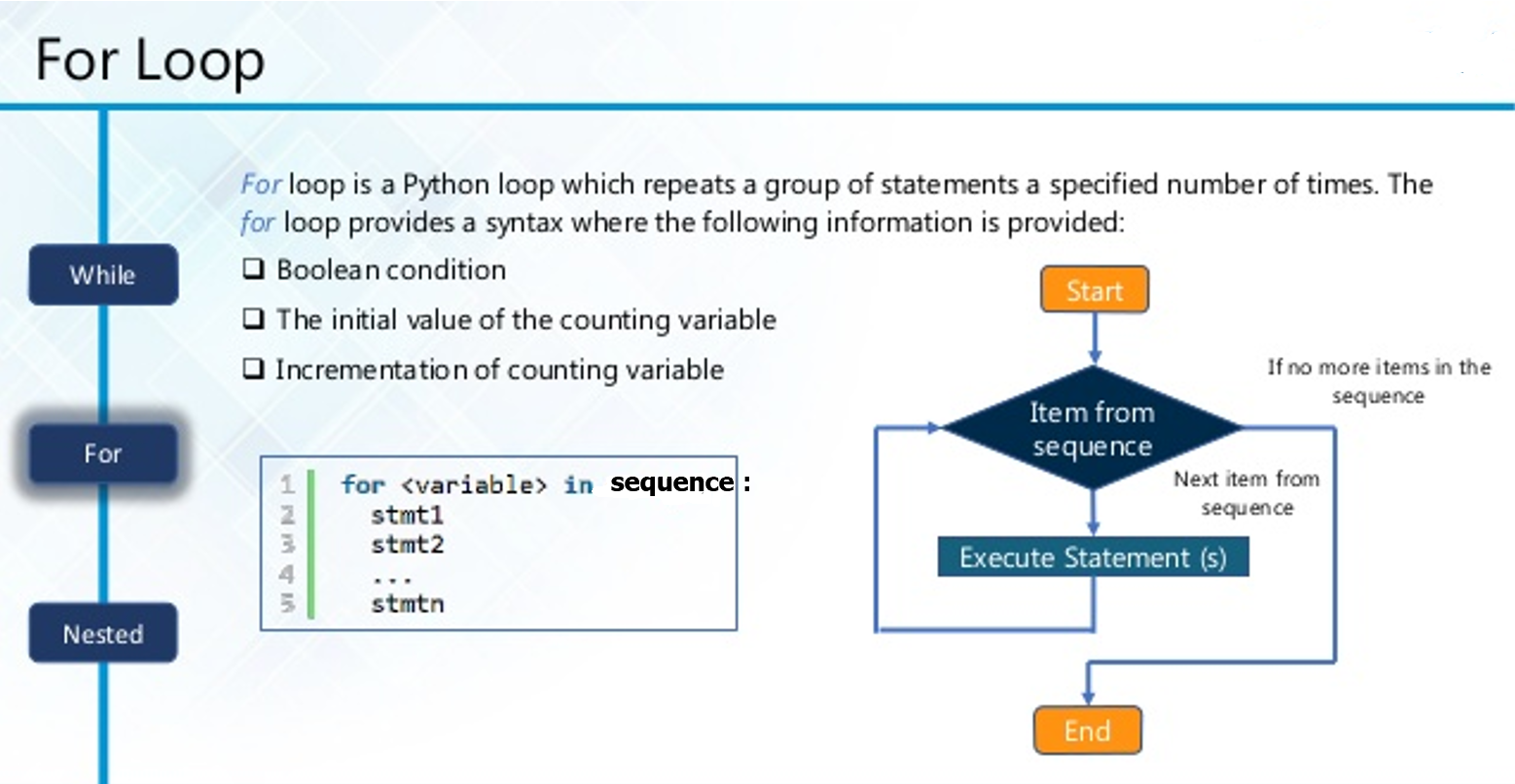


**1.WHILE :**





**2. FOR :**



# First Example

for letter in **'Python':**

print'Current Letter :', letter

# Second Example

Fruits = ['banana','apple','mango']

for fruit in **fruits**:

print'Current fruit :', fruit

print"Good bye!"

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!

## 2.1 Iterating by Sequence Index:

fruits=['banana','apple','mango']

for index in range(len(fruits)):

print'Current fruit :', fruits[index]

print"Good bye!"

Current fruit : banana

Current fruit : apple

Current fruit : mango

Good bye!

## 2.2 Using else Statement with Loops:

Python supports to have an else statement associated with a loop statement

* If the **else** statement is used with a **for** loop, the **else** statement is executed when the loop has exhausted iterating the list.
* If the **else** statement is used with a **while** loop, the **else** statement is executed when the condition becomes false.

The following example illustrates the combination of an else statement with a for statement that searches for prime numbers from 10 through 20.

lower=10

upper=20

print("Prime numbers between",lower,"and",upper,"are:")

fornumin range(lower,upper+1):

foriin range(2,num):

if(num%i)==0:

break

else:

print'%d is a prime number \n'%(num),

('Prime numbers between', 10, 'and', 20, 'are:')

11 is a prime number

13 is a prime number

17 is a prime number

19 is a prime number

1. **NESTED :**

**4.1 nested while loop:**

while expression:

while expression:

statement(s)

statement(s)

i=2

while(i<100):

j =2

while(j <=(i/j)):

ifnot(i%j):break

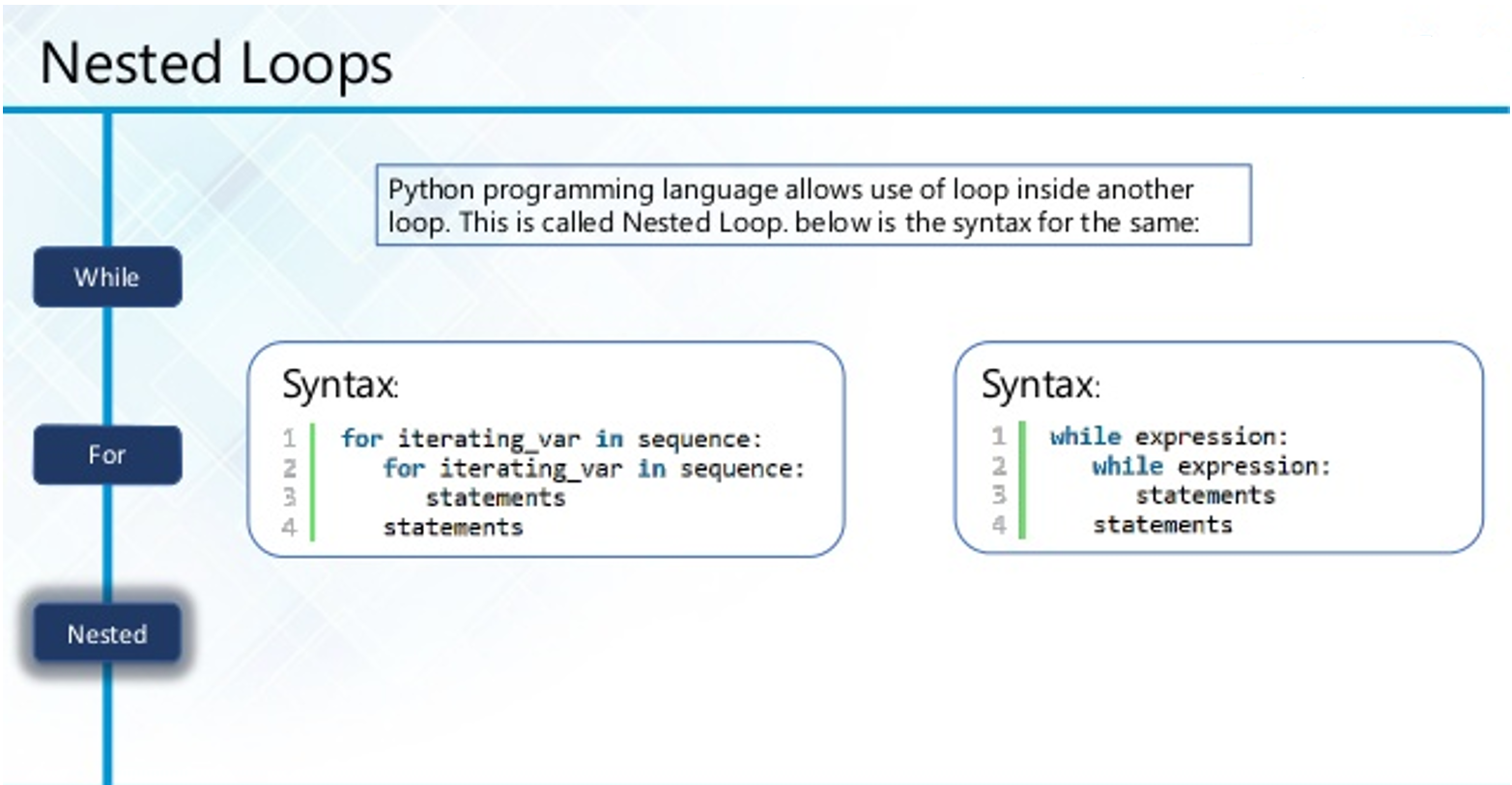
j = j +1

if(j >i/j):printi," is prime"

i=i+1

print"Good bye!"

**4.2 nested for loop:**



**(OR)**

**for** [first iterating variable] **in** [outer loop]: # Outer loop

[do something] # Optional

**for** [second iterating variable] **in** [nested loop]: # Nested loop

[do something]

**Example1:**

num\_list = [1, 2, 3]

alpha\_list = ['a', 'b', 'c']

**for** number **in**num\_list:

print(number)

**for** letter **in**alpha\_list:

print(letter)

**Example2:**

list\_of\_lists = [['hammerhead', 'great white', 'dogfish'],[0, 1, 2],[9.9, 8.8, 7.7]]

**for** list **in**list\_of\_lists:

**for** item **in** list:

print(item)

## Control Statements :

Loop control statements change execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed.

Python supports the following control statements

|  |  |
| --- | --- |
| **S.No.** | **Control Statement & Description** |
| 1 | [**break statement**](https://www.tutorialspoint.com/python/python_break_statement.htm) :  Terminates the loop statement and transfers execution to the statement immediately following the loop.  Break statement is a jump statement which is used to transfer execution control. It breaks the current execution and in case of inner loop, inner loop terminates immediately.  When break statement is applied the control points to the line following the body of the loop, hence applying break statement makes the loop to terminate and controls goes to next line pointing after loop body.   1. **for** i **in** [1,2,3,4,5]: 2. **if** i==4: 3. **Print**("Element found") 4. **break** 5. **print** i, 6. **---------------------------** 7. **for** letter **in** 'Python3': 8. **if** letter == 'o': 9. **break** 10. **print** (letter) |
| 2 | [**continue statement**](https://www.tutorialspoint.com/python/python_continue_statement.htm) :  Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.  Python Continue Statement is a jump statement which is used to skip execution of current iteration. After skipping, loop continue with next iteration.  We can use continue statement with for as well as while loop in Python.  Continue Statement   1. a=0 2. **while** a<=5: 3. a=a+1 4. **if** a%2==0: 5. **continue** 6. **print** a 7. **print** "End of Loop" |
| 3 | [**pass statement**](https://www.tutorialspoint.com/python/python_pass_statement.htm) :  The pass statement in Python is used when a statement is required syntactically but you do not want any command or code to execute.  In Python, pass keyword is used to execute nothing; it means, when we don't want to execute code, the pass can be used to execute empty. It is same as the name refers to. It just makes the control to pass by without executing any code. If we want to bypass any code pass statement can be used.   1. **for** i **in** [1,2,3,4,5]: 2. **if** i==3: 3. **pass** 4. **print** "Pass when value is",i 5. **print** i, |