NETWORKZ SYSTEMS

AN ISO 9001:2015 CERTIFIED COMPANY

CHINNAKADA , KOLLAM

PROGRAMMING CONCEPTS

Conditions in Python

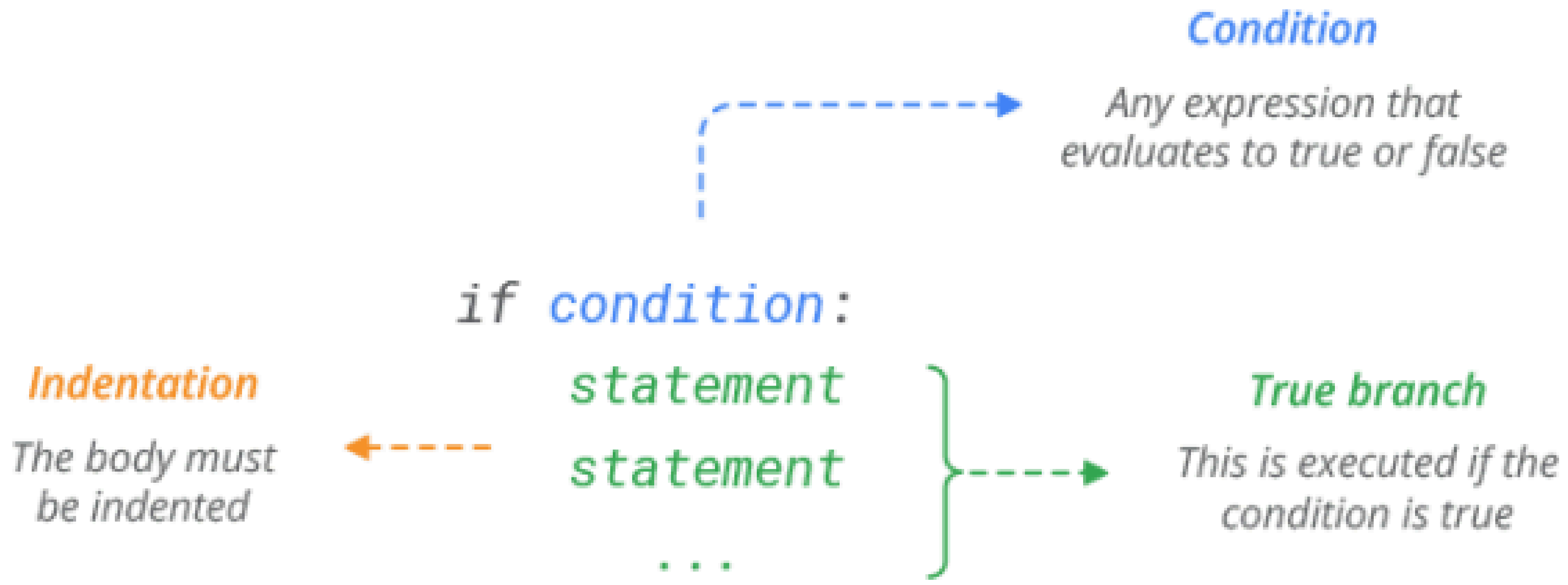
Comparison operations compare some value or operand and based on a condition, produce a Boolean. Python has six comparison operators as below

Less than (<)
Less than or equal to (<=)
Greater than (>)
Greater than or equal to (>=)
Equal to (==)
Not equal to (!=)

The if/elif/else statement is used in Python for decision making.

An else statement can be combined with an if statement.

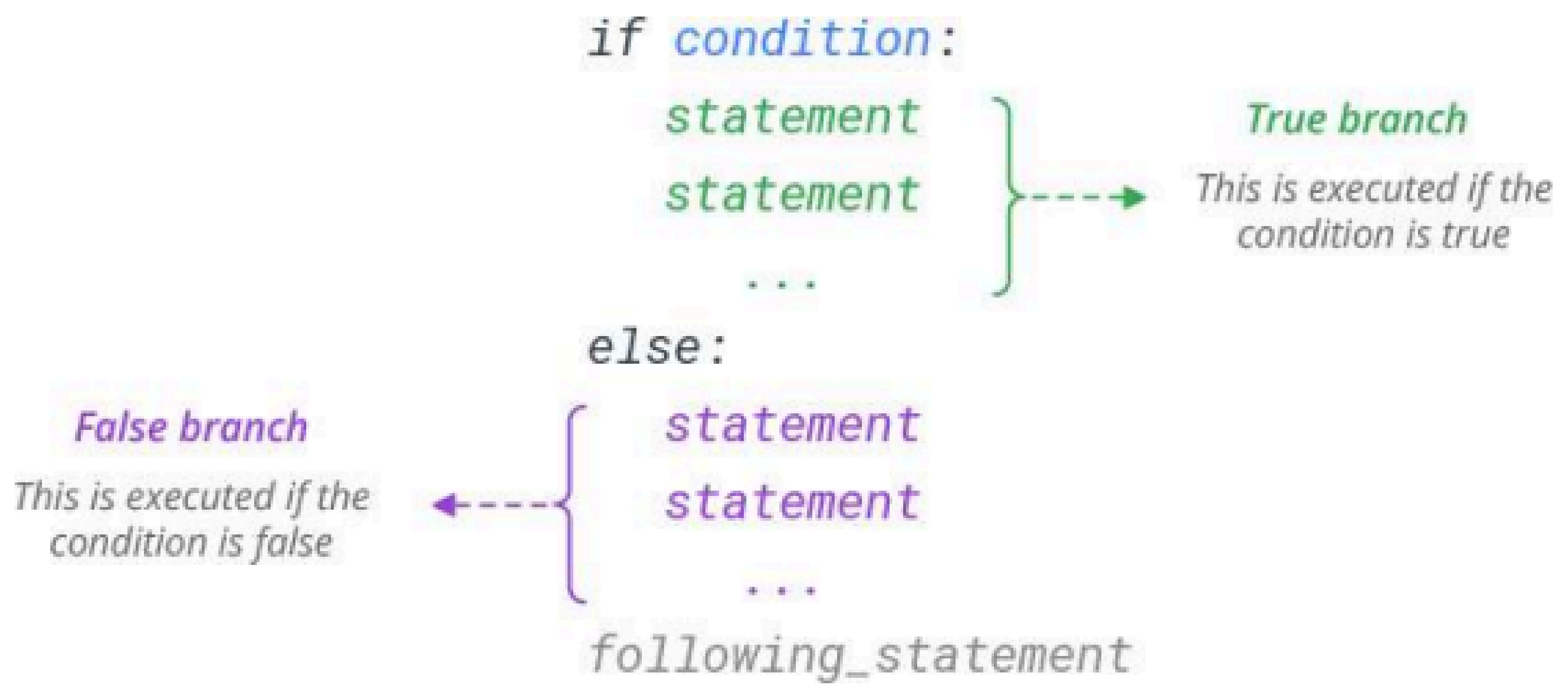
If statement


Condition
Any expression that evaluates to true or false
if condition:
statement
statement
...
Indentation
The body must be indented
True branch
This is executed if the condition is true

example

```
a = 33
b = 200
if b > a:
print("b is greater than a")
```

else statement



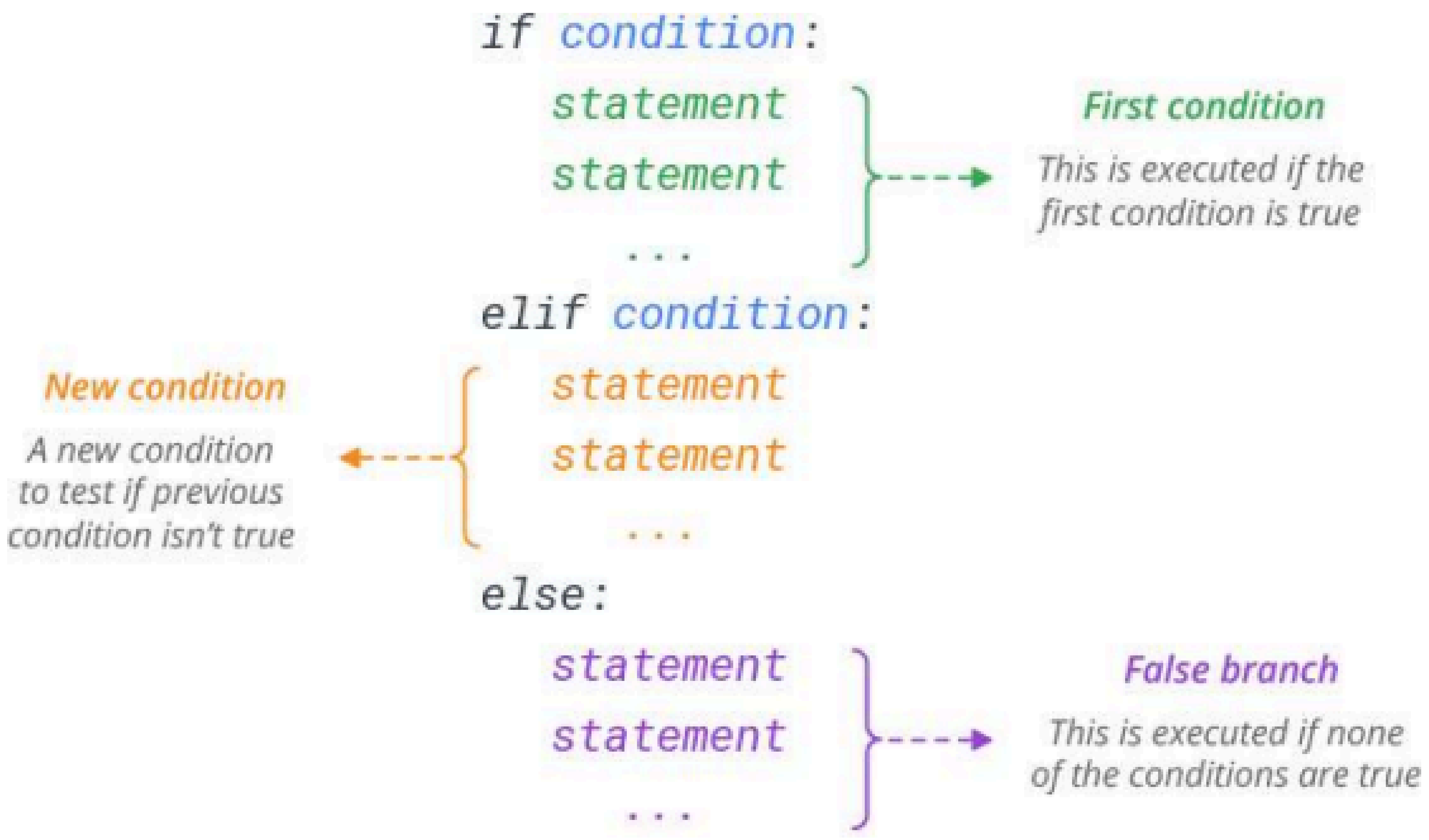
```
if condition:
    statement
    statement
       ...
else:
    statement
    statement
       ...
following_statement
```

True branch
This is executed if the condition is true
False branch
This is executed if the condition is false

example 1
```
num=5
if num > 10:
	print(“This is over 10”)
else:
	print(“This is not over 10”)
```

example 2
```
album_year = 2000
if album_year >= 1995:
	print('Album year is higher than 1995.')
else:
	print('Album year is lower than 1995.')
print('Done!')
```

elif statement



```
if condition:
    statement
    statement
    ...
elif condition:
    statement
    statement
    ...
else:
    statement
    statement
    ...
```

First condition

This is executed if the first condition is true

New condition

A new condition to test if previous condition isn't true

False branch

This is executed if none of the conditions are true

elif statement

```
age = 5
if age > 6:
 print('You can go to primary school.' )
elif age == 5:
 print('You should go to kindergarten.')
else:
 print('You are a baby' )

print('Done!')
```

```
a = 200
b = 33
if b > a:
	print("b is greater than a")
elif a == b:
	print("a and b are equal")
else:
	 print("a is greater than b")
```

NETWORKZ SYSTEMS
AN ISO 9001:2015 CERTIFIED COMPANY
CHINNAKADA , KOLLAM

Ternary Operator in Python

It simply allows to test a condition in a single line replacing the multiline if-else making the code compact.

Example

```
a, b = 10, 20
min = a if a < b else b
print(min)
```

And

The and keyword is a logical operator, and is used to combine conditional statements

Test if a is greater than b, AND if c is greater than a:

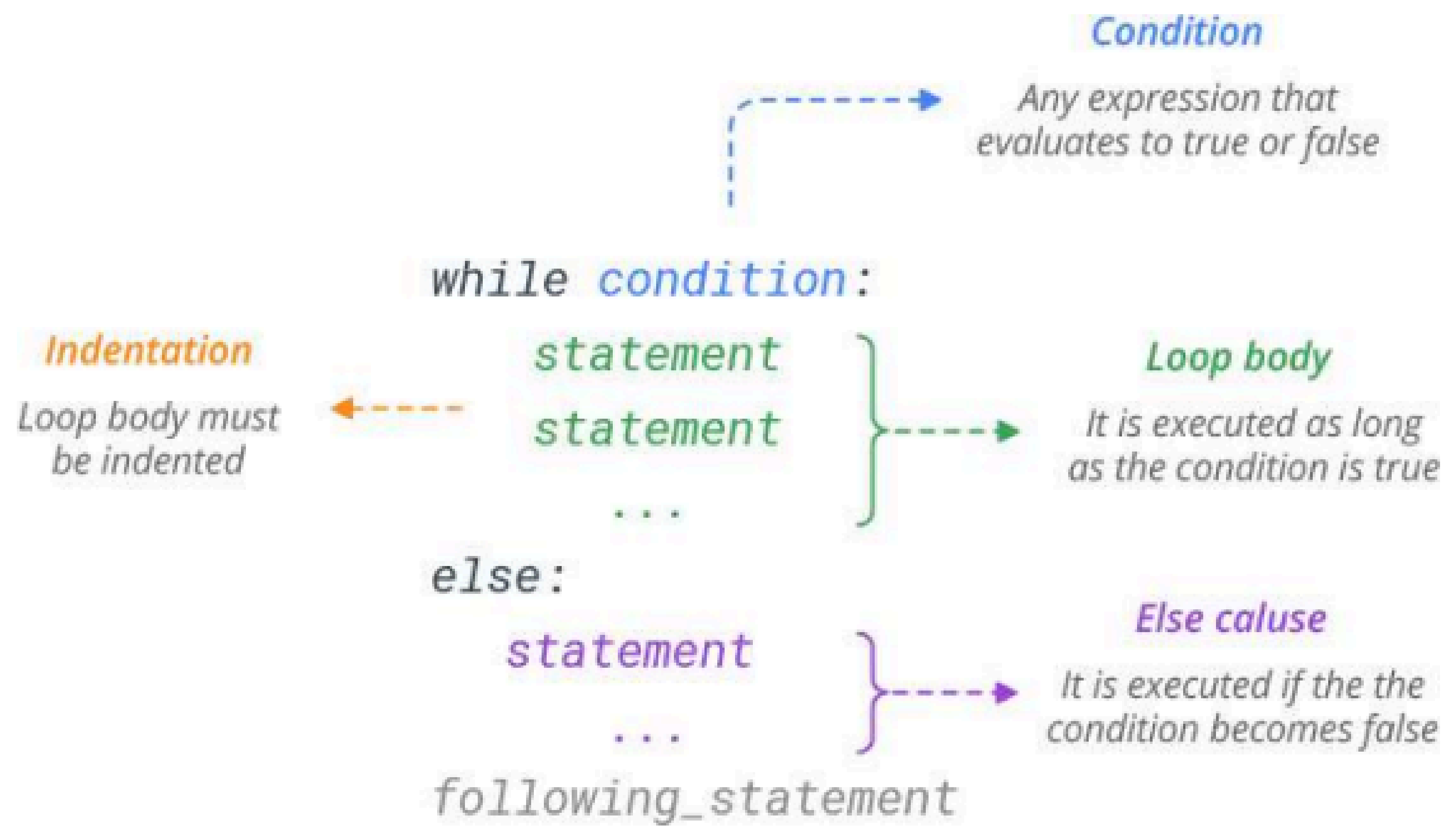
```
if a > b and c > a:
        print("Both conditions are True")
```

Python Loops

Python has two primitive loop commands:

- while loops
- for loops

With the while loop we can execute a set of statements as long as a condition is true.



```
while condition:
    statement
    statement
      ...
else:
    statement
      ...
following_statement
```

Condition
Any expression that evaluates to true or false
Indentation
Loop body must be indented
Loop body
It is executed as long as the condition is true
Else caluse
It is executed if the the condition becomes false

```
i=1
while i<10:
        print("i=",i)
        i+=1


i = 22
while i<27:
       print(i)
       i+=1
```

NETWORKZ SYSTEMS®
AN ISO 9001:2015 CERTIFIED COMPANY
CHINNAKADA , KOLLAM

break in while loop

With the break statement we can stop the loop even if the while condition is true

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

continue in while loop

With the continue statement we can stop the current iteration, and continue with the next.

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

NETWORKZ SYSTEMS
AN ISO 9001:2015 CERTIFIED COMPANY
CHINNAKADA , KOLLAM

for loop

The for loop enables you to execute a code block multiple times.

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

This is less like the for keyword in other programming languages, and works more like an iterator method as found in other object-orientated programming languages.

With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
	print(x)
```

The range() Function

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

```
for x in range(10):
        print(x)

for x in range(2, 6):
        print(x)

for x in range(2, 30, 3):
        print(x)
```

NETWORKZ SYSTEMS®
AN ISO 9001:2015 CERTIFIED COMPANY
CHINNAKADA , KOLLAM

Nested Loops

A nested loop is a loop inside a loop.

```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
    for y in fruits:
      print(x, y)
```

continue in for loop

```
nlis = [1,2,4,5,6,7,8,9,10,11,12,13,14]
for i in nlis:
        if i == 5:
            continue
        print(i)
```

```
For x in range(1,20):
      if x==10:
         continue
      print(“x=",x)
```

NETWORKZ SYSTEMS
AN ISO 9001:2015 CERTIFIED COMPANY
CHINNAKADA , KOLLAM

Loop Through a List

```
thislist = ["apple", "banana", "cherry"]
for x in thislist:
    print(x)
```

Check if Item Exists

Check if "apple" is present in the list

```
thislist = ["apple", "banana", "cherry"]
if "apple" in thislist:
    print("Yes, 'apple' is in the fruits list")
```

Print all values in the dictionary

You can use the values() function to return values of a dictionary

```
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
for x in thisdict.values():
        print(x)
```

NETWORKZ SYSTEMS
AN ISO 9001:2015 CERTIFIED COMPANY
CHINNAKADA , KOLLAM

Loop through both keys and values, by using the items() function:

```
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}

for x, y in thisdict.items():
        print(x, y)
```

NETWORKZ SYSTEMS
AN ISO 9001:2015 CERTIFIED COMPANY
CHINNAKADA , KOLLAM

THANK YOU

NETWORKZ SYSTEMS®

AN ISO 9001:2015 CERTIFIED COMPANY

CHINNAKADA , KOLLAM