Branching

Boolean expression

In boolean expression, we compare two values or statements using the comparison operator to yield a boolean value (True or False).

Comparison or Relational operator

a. == (equal): returns True if the first value is equal to the second. [single equal(=) is used for value assignment but double equal (==) is used for value equality]
b. != (not equal): returns True if the first value is not equal to the second.
c. > (greater than): returns True if the first value is greater than the second.
d. < (less than): returns True if the first value is less than the second
e. >= (greater than or equal): returns True if the first value is greater than or equal to the second.
f. <= (less than or equal): returns True if the first value is smaller than or equal to the second.

Example:

Code	Output
`print(5==5)` `print(5==50)`	`True` `False`
`print(5!=5)` `print(5!=50)`	`False` `True`
`print(100>2)` `print(1>20)`	`True` `False`
`print(1<20)` `print(100<2)`	`True` `False`

`print(1>=20)`
`print(100>=2)`
`print(100>=100)`
	`False`
	`True`
	`True`

`print(100<=2)`
`print(2<=2)`
`print(1<=100)`
	`False`
	`True`
	`True`

Logical operator

a. and (logical AND): The logical and returns True if both values are True. Otherwise, returns False.

b. or (logical OR): The logical or returns False if both values are False. Otherwise, returns True.
c. not (Logical NOT): Returns True, if False is given and vise versa.

A	B	A and B	A or B	not A
False	False	False	False	True
False	True	False	True	True
True	False	False	True	False
True	True	True	True	False

Example:

Code	Output	Explanation
variable1 = True print(not variable1))	False	print(not variable1) print(not True) print(not True) print(False) Output: False
variable1 = False print(not variable1)	True	print(not variable1) print(not False) print(not False) print(True) Output: True
variable1 = True variable2 = True print(variable1 and variable2)	True	print(variable1 and variable2) print(True and True) print(True) Output: True
variable1 = True variable2 = False print(variable1 and variable2)	False	print(variable1 and variable2) print(True and False) print(True and False) print(False) Output: False
variable1 = True variable2 = False print(variable1 or	True	print(variable1 or variable2) print(True or False) print(True or False)

variable2)		print(True) Output: True
variable1 = False variable2 = False print(variable1 or variable2)	False	print(variable1 or variable2) print(False or False) print(False or False) print(False) Output: False

Compound Boolean expression (Logical and comparison operator combined) examples:

Code	Output	Explanation
print(not (5>1))	False	Here, (5>1) is True. print(not (5>1)) print(not (True)) print(not True) print(False) Output: False
print(not (5>500))	True	Here, (5>500) is False. print(not (5>500)) print(not (False)) print(not False) print(True) Output: True
print((5>1) and (5<20))	True	Here, (5>1) is True and (5<20) is True. print((5>1) and (5<20)) print((True) and (True)) print(True and True) print(True) Output: True
print((5>1) and (5>500))	False	Here, (5>1) is True and (5>500) is False. print((5>1) and (5>500)) print((True) and (False)) print(True and False) print(False) Output: False

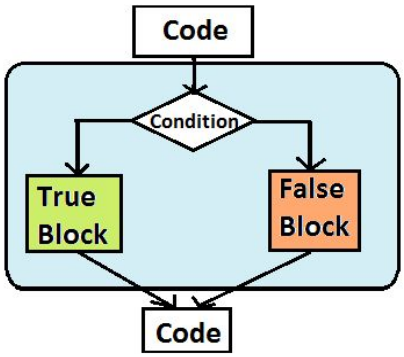
`print((5>1) or (5==50))`	`True`	`Here,(5>1)is True and (5==50) is False.`
		`print((5>1) or (5==50))`
		`print((True) or (False))`
		`print(True or False)`
		`print(True)`
		
		`Output: True`

`print((-6>1) or (5>500))`	`False`	`Here, (-1>1)is False and (5>500) is False.`
		`print((-6>1) or (5>500))`
		`print((False) or (False))`
		`print(False or False)`
		`print(False)`
		
		`Output: False`

Conditional statements

Comparison operators are basically used for writing conditional statements. They have three parts:

1. a condition yielding in either True or False.
2. a block of code is executed if the condition yields True.
3. another non-mandatory block of code is executed if the condition yields False.



Indentation:

Indentation means leading whitespace (spaces and tabs) at the beginning of a particular line of code. It is basically used for grouping a section of code. A particular section or bundle of code is called "Block". For example, in the previous flowchart, "True block" holds a section of code, which will only be executed if the condition yields True and "False block" holds a section of code, which will only be executed if the condition yields False. So, for blocking or in more general terms for paragraphing one or multiple lines, we use indentations.

Note: details about indentation is provided in the link below.
Link: https://docs.python.org/2.0/ref/indentation.html

1. **Basic conditional statements:**

Basic conditional statement code structure

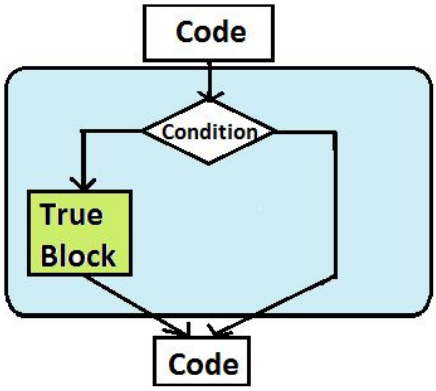
`if(condition):`
`  #codes inside True block`
`  True block`
`  #codes inside True block`
`else:`
`  #codes inside False block`
`  False block`
`  #codes inside False block`
`#codes outside False block`

Example: Code	**Output**

`if( 5 > 10 ):`
`  print("10 is greater")`
`else:`
`  print("5 is greater")`
	`5 is greater`

`if( 30%2==0 and 30%3==0 ):`
`  print("Even and multiple of 3")`
`else:`
`  print("not")`
	`Even and multiple of 3`

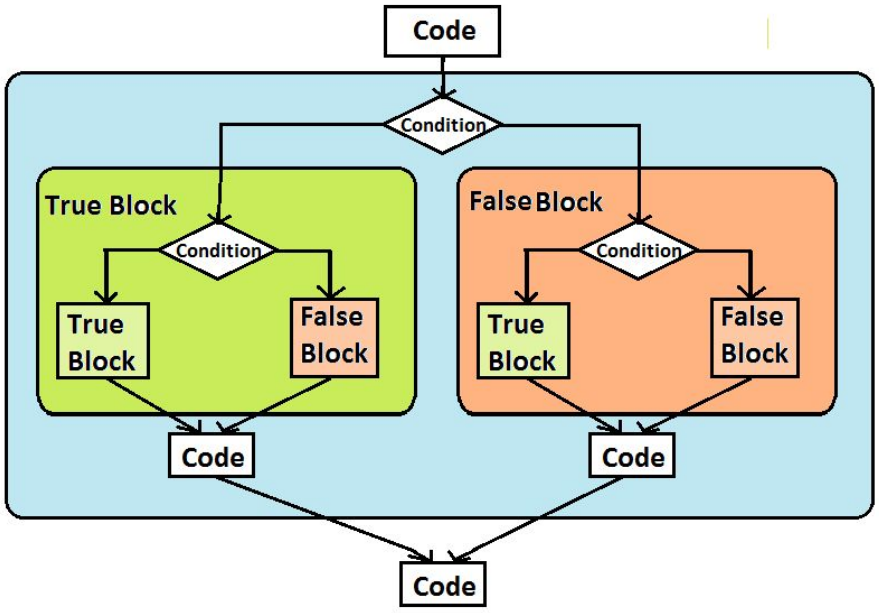
2. **Unary Selection (Omitting the else Clause)**
When the condition evaluates to True, the True block is executed and the else and False block is completely excluded.



Basic conditional statement code structure	
if(condition): **#codes inside True block** **True block** **#codes inside True block** **#codes outside True block**	
Example: Code	**Output**
if(5 > 10): print("10 is greater")	NO OUTPUT IS SHOWN
if(10 == 10): print("equal")	equal

3. **Nested conditionals**

Multiple conditional statements including unary selection statements can be nested inside one another. These are called nested conditional statements. Any number of these statements can be nested inside one another. Indentation is the only way to figure out the level of nesting. Here in the flowchart below, we can see that inside the True block and False block, there is another set of conditions with True block and False block. There True block and False block again can have conditions nested inside them.



Nested conditional statements code structure

```
if(condition):
    #codes inside True block
    if(condition):
        #codes inside True block
        if(condition):
            True block
        else:
            False block
        #codes inside True block
    else:
        False block
    #codes inside True block
else:
    #codes inside False block
    if(condition):
        True block
    else:
        False block
    #codes inside False block
#codes outside False block
```

Example: Code	Output

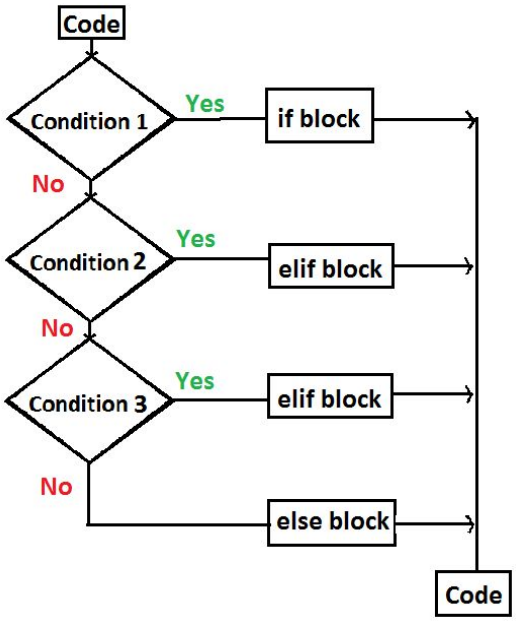
`if 30%2==0 :`
`  if 30%3==0 :`
`  print("Even and multiple of 3")`
`  else:`
`  print("Even and not a multiple of 3")`
`else:`
`  if 30%3==0 :`
`  print("Odd and multiple of 3")`
`  else:`
`  print("Odd and not a multiple of 3")`
	Even and multiple of 3

`number = 2`
`if number > 0 :`
`  if number < 100 :`
`  if number % 2 == 0:`
`  print("Even positive number less than 100")`
`  else:`
`  print("Odd positive number less than 100")`
`  else:`
`  if number % 2 == 0:`
`  print("Even positive number greater than 100")`
`  else:`
`  print("Odd positive number greater than 100")`
	Even positive number less than 100

The level of nesting can be understood only with the help of Indentation. Thus it is less used by the programmers for avoiding confusion.

4. Chained or Ladder conditionals

The same checking of "nested conditionals" can be done using Chained conditionals (if....elif....elif....else). The conditions are checked from top to bottom. First the "if condition" is checked, if it yields False, then the next "elif condition" is checked. One if block, can have multiple "elif blocks" and only one "else block" at the end of the ladder, which is executed if the rest of the conditions yield False. The full form of "elif" is "else if". Among several conditions of if...elif...else blocks, only one block is executed according to the condition.



Chained conditional statements code structure

```
if (condition):
      #codes inside if block
      if code block
      #codes inside if block
elif (condition):
      #codes inside elif block
      elif code block
      #codes inside elif block
elif (condition):
      #codes inside elif block
      elif code block
      #codes inside elif block
.
.
```

```
else:
     #codes inside else block
     else code block
     #codes inside else block
#codes outside the if-elif-else block
```

Example: Code	Output

`time = 4`
 	
`if time == 0:`
`  print("It is midnight")`
`elif time == 1:`
`  print("It is 1 am")`
`elif time == 2:`
`  print("It is 2 am")`
`elif time == 3:`
`  print("It is 3 am")`
`elif time == 4:`
`  print("It is 4 am")`
`else:`
`  print("5am to 12 pm")`
	`It is 4 am`

Style Guide for Python Code

For every programming language, there are few coding conventions followed by the coding community of that language. All those conventions or rules are stored in a collected document manner for the convenience of the coders, and it is called the “Style Guide” of that particular programming language. The provided link gives the style guidance for Python code comprising the standard library in the main Python distribution.

Python style guide link: https://www.python.org/dev/peps/pep-0008/