# Reading: Demonstrate Conditions, Branching



## Conditional Branching

### 'if' Statement

Branching allows us to run different statements for a different input. It is helpful to think of an “if statement” as a locked room:

- If the statement is true, you can enter the room, and your program can run some predefined task.
- If the statement is false, your program will skip the task within the "if" body.

```
if test expression:
    statement(s)
```

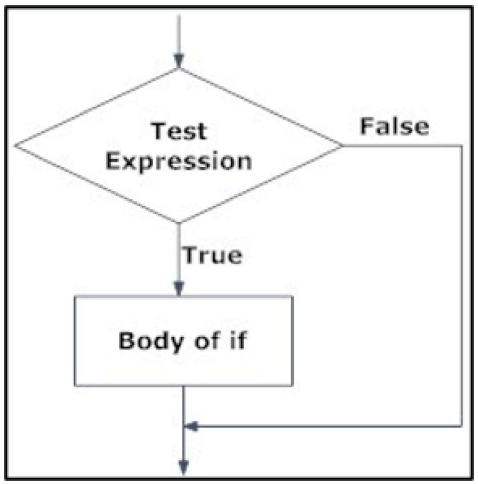
Here, the program evaluates the test expression and **will execute** statement(s) **only if** the test expression is **True**.

If the test expression is **False**, the statement(s) is **not executed**.

In Python, the body of the if statement is indicated by the indentation. The body starts with an indentation and the first un-indented line marks the end.

Python interprets **non-zero** values as True. **None** and **0** are interpreted as False.

The Flowchart of *if statement* is:




Consider the following example:

```
# If the number is greater than 5, we print an appropriate message

num = 7
if num > 5:
    print(num, "is greater than 5.")
print("This is always printed.")

num = 4
if num > 5:
    print(num, "is smaller than 5.")
print("This is also always printed.")
```

```
7 is greater than 5.
This is always printed.
This is also always printed.
```

In the above example, **num > 5** is the test expression. The body of **if** is executed only if this evaluates to True.

When the variable num is equal to 7, test expression is **true** and statements inside the body of if are executed.

If the variable num is equal to 4, test expression is **false** and statements inside the body of if are skipped.

The **print()** statement falls outside of the if block (un-indented). Hence, it is executed regardless of the test expression.

### **'if..else'** Statement

- This is just an extension of if statement. If the condition doesn't match, the else part is executed. At any point in time, only one of the two will be executed. If the statement is true, you can enter the room, and your program can run some predefined task.
- If the statement is false, your program will skip the task within the "if" body and perform whatever is in the "else" part of the body. The program will then continue to execute with rest of the statements if any.

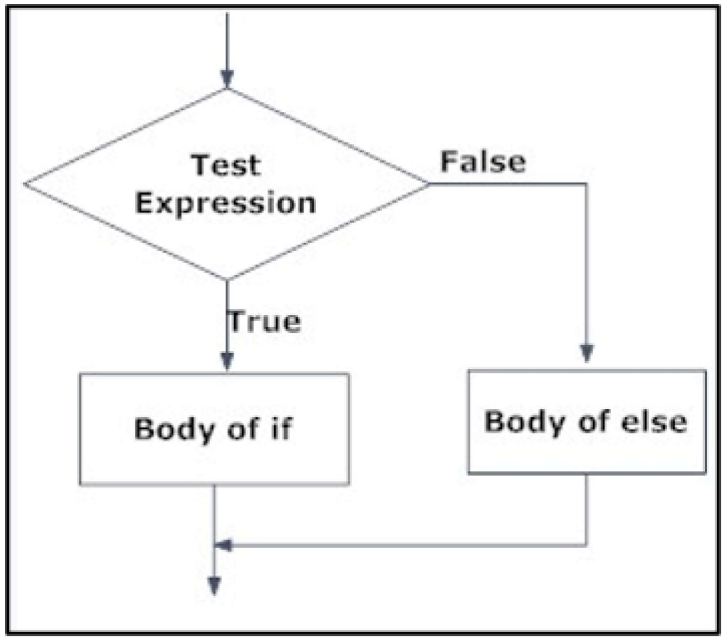
The syntax is:

```
if test expression:
    Body of if
else:
    Body of else
```

Here, the program evaluates the test expression and **will execute** statement(s) **only if** the test expression is **True**.  False, the statement(s) is **not executed**. To evaluate the expression we will use comparison operation and logical operation that we learnt about earlier.
In Python, the body of the "if-statement" is indicated by the indentation. The body starts with an indentation and the first un-indented line marks the end. Great care needs to be taken while indenting the statements as most beginner programmers or programmers used to other languages, tend to make mistakes.

Python interprets **non-zero** values as True. **None** and **0** are interpreted as False.

The Flowchart of *if...else statement* is:




Consider the following example:

```
# Program checks if the number is greater or smaller than 5
num = 1
if num > 5 :
    print("Number greater than 5")
else:
    print("Number smaller than 5")
```

```
Number smaller than 5
```

In the above example, **num > 5** is the test expression. The body of "**if**" is executed only if this evaluates to True.

When the variable num is equal to 7, test expression is **true** and statements inside the body of if are executed.
If the variable num is equal to 4, test expression is **false** and statements inside the body of if are skipped.
The **print()** statement falls outside of the if block (un-indented). Hence, it is executed regardless of the test expression.

### if...elif...else Statement

The elif statement, short for “else if,” allows us to check additional conditions if the proceeding condition is false. If the condition is true, the alternate expressions will be run.

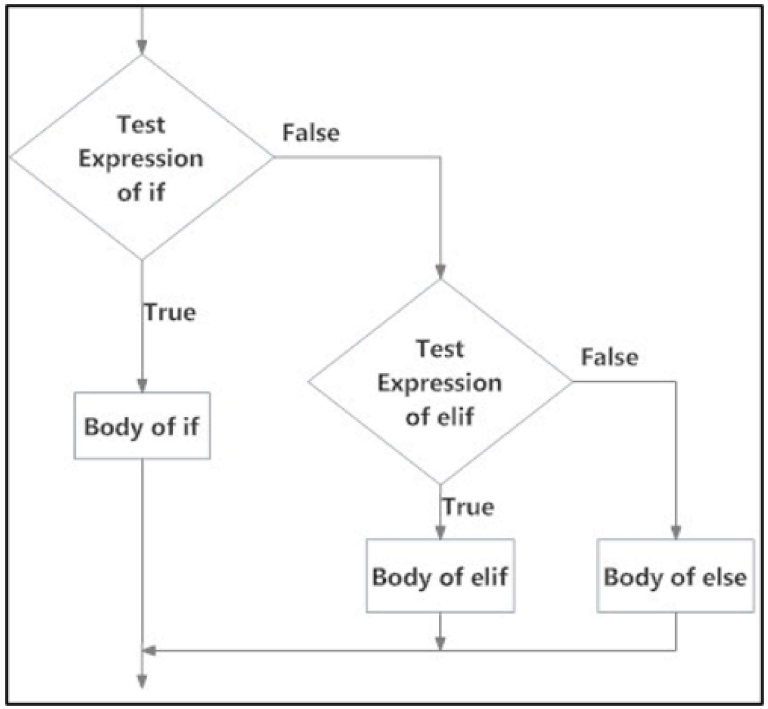
The syntax is:

```
if test expression:
    Body of if
elif test expression:
    Body of elif
else:
    Body of else
```

The elif is short for else if. It allows us to check for multiple expressions.

If the condition for if is **false**, it checks the condition of the next **elif** block and so on.
If **all** the conditions are **False**, the body of **else** is executed.
Only one block among the several if...elif...else blocks is executed according to the condition.
The if block can have **only one** else block. But it can have **multiple** elif blocks.

The Flowchart of *if...elif..else statement* is:




Consider the following example:

```
#In this program, we check
#if the number>5 or
#number<5 or number=0 and
#display an appropriate message

num = 0

if num > 5:
    print("number>5")
elif num == 0:
    print("Zero")
else:
    print("number<5")
```

```
Zero
```

In the above example, when **num = 1**, the test expression is **false** and the body of if is skipped and the **body of else is executed**.
When variable num is **greater than 5**, **number>5** is printed.
If num is **equal to 0**, **Zero** is printed.
If num is **less than 5**, **number<5** is printed.

**Nested if Statement:** We can have a *if...elif...else* statement inside another *if...elif...else* statement. This is known as Nesting. Any number of these statements can be nested inside one another. Indentation is the only way to figure out the level of nesting. Nested if statements must be avoided unless necessary.

You can post your queries or make a small video explaining your queries on **Questionsly.**

In today's modern age of disruption, SkillUp Online is your ideal learning platform that enables you to upskill to the most in-demand technology skills like Data Science, Big Data, Artificial Intelligence, Cloud, Front-End Development, DevOps & many more. In your journey of evolution as a technologist, SkillUp Online helps you work smarter, get to your career goals faster and create an exciting technology led future.

## Corporate

- Home
- About Us
- Enterprise
- Blog
- Press

## Support

- Contact us
- Terms of Service
- Privacy Policy