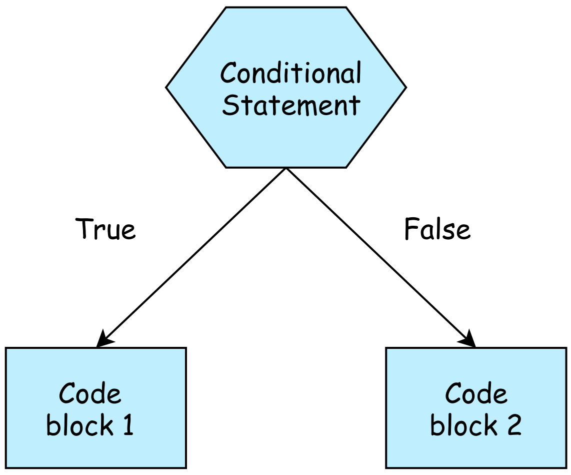
Conditional Statements

This lesson will introduce conditional statements and focus on how to use them.

Conditional statements in Python

A **conditional statement** controls the flow of execution in a program based on a Boolean expression. If the expression evaluates to `True`, then a different piece of code might execute, If it is `False`, then some other piece of code will execute. This is also explained in the following illustration.



These statements give the computer the ability to decide the flow of execution of the program based on the information it receives. There are three kinds of conditional statements:

- `if`
- `if` - `else`

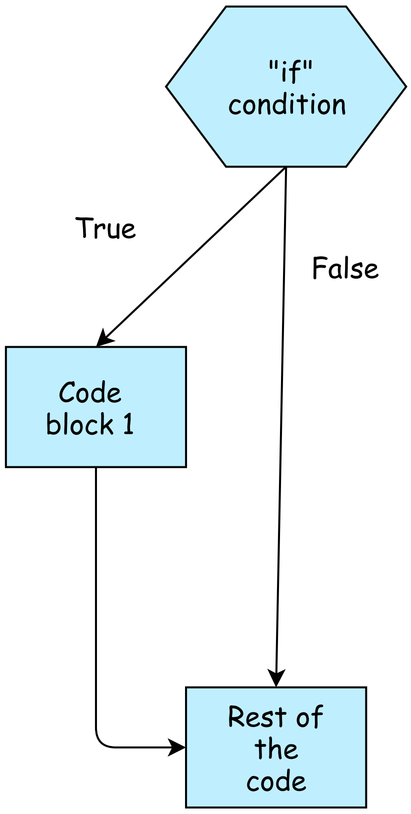
- `if`-`elif`-`else`

`if` statement

It is the simplest conditional statement. The structure of the statement is:

```
if (condition):
  code
```

If the condition provided is satisfied, then the code is executed, otherwise, it is skipped. The colon, `:`, after the condition is necessary as it indicates where the condition ends and the code block begins. Note that the code after the `:` will be indented one tab space to the right. All of the code inside the `if` block after `:` should be at the same indentation level. Otherwise, Python will give an indentation error.



We can see the logic of the `if` statement from the above diagram. Now let's look at some examples.

```
age = 34
if age < 50:
  print("You are still young.")

age = 65
if age < 50:
  print("You are still young.")
```

In the **first line**, we set `age` to `34`. In the next line we write the keyword `if` followed by the condition. The condition here is whether or not `age` is less than `50`. When **line 2** is executed and the condition is satisfied, execution jumps to **line 3**. Here the message, “You are still young,” is printed.

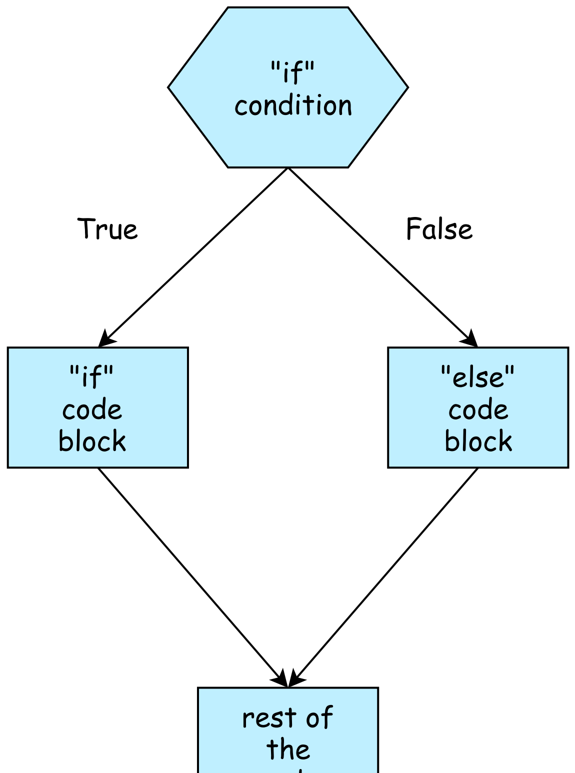
After that, we change the value of `age` to `65`. The same `if` condition is written in **line 6**, but this time the condition is not satisfied. Therefore **line** 7 is skipped.

`if`-`else` statement

The structure is:

```
if (condition):
  code
else:
  code
```

If the condition provided is satisfied, then the code is executed, otherwise the code after `else` is executed. The `else` keyword will be on the same indentation level as the `if` keyword. Its body will be indented one tab to the right just like the `if` statement.



The illustration above explains the logic of the `if-else` statement. Now let's look at some examples in code.

```
age = 34
if age  < 50:
  print("You are still young.")
else:
  print("You are getting old.")

age = 55
if age  < 50:
  print("You are still young.")
else:
  print("You are getting old.")
```

In the **first line**, we set `age` to `34`. In the next line we write the keyword `if` followed by the condition. The condition here is whether or not `age` is less than `50`. When **line 2** is executed and the condition is satisfied, execution jumps to **line 3**. Here the message, "You are still young," is printed.

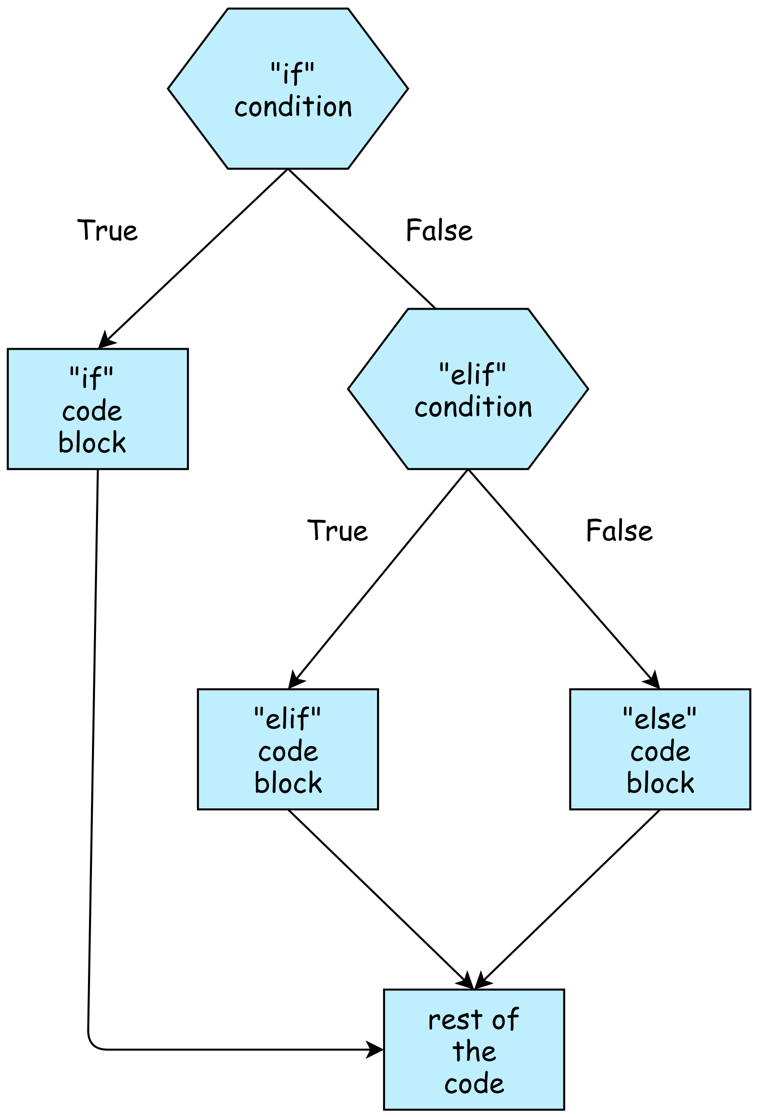
After that, we change the value of `age` to `55`. The same `if` condition is written in **line 8**, but this time the condition is not satisfied. Therefore, **line 9** is skipped and **line 11** is executed.

`if`-`elif`-`else` statement

The structure is:

```
if (condition):
  code
elif (condition):
  code
else:
  code
```

If the condition provided is satisfied, then the code in the `if` block is executed, otherwise the next condition is checked. If that condition is satisfied, the code after `elif` is executed and the `else` condition and code is skipped. But if it is not satisfied, then the code inside the `else` block is executed.



The illustration above explains the logic of the `if-else` statement. Now let's look at some examples in code. Note that we can have multiple `elif` statement blocks between the `if` and `else` statement blocks. However, at any time if an `elif` condition is satisfied, the code inside that block is executed and the rest of the conditions and codes are skipped. Let's look at some examples.

```
color = 'yellow'

if color=='red':
  print("Red is my color")
elif color == 'yellow':
  print("Yellow is my color")
elif color == 'black':
  print("Yellow is my color")
else:
  print("I dont know the color")

color = 'white'
if color=='red':
  print("Red is my color")
```

```
elif color == 'yellow':
  print("Yellow is my color")
elif color == 'black':

  print("Yellow is my color")
else:
  print("I dont know the color")
```

In the **first line**, we set `color` to `yellow`. In **line 3**, we write the keyword `if` followed by the condition. The condition here is whether or not `color` is `red`. When **line 3** is executed and the condition is not satisfied, execution jumps to **line 5** where another condition is checked to see if the `color` is `yellow`. Here the condition is satisfied and **line 6** is executed. After this the execution jumps to **line 12**, where `color` is changed to `white`.

Now all the conditions are checked one by one but no condition is satisfied. Hence, the execution jumps to **line 20** in the `else` block.

Conditional statements can be nested as well. You can try out different sorts of experiments with these. These come in very handy.

In the last example, we had to write the same piece of code twice. This can be quite annoying when we have to write the same piece of code again and again. The solution is described in the next lesson and is known as a *function*.