Python → Control flow statements → Elif statement

Theory: Elif statement

43 minutes 20 / 22 problems solved

Start practicing

11948 users solved this topic. Latest completion was 5 minutes ago.

Since you are familiar with basic conditional statements, such as **if** statement and **if-else** statement, it's time for the more advanced **elif statement**.

Elif statement is used when we want to specify several conditions in our code. How does it differ from **if-else statements**? Well, it's simple. Here we add the **elif** keyword for our additional conditions. It is mostly used with both if and else operators and the basic syntax looks like this:

```
# basic elif syntax
if condition1:
    action1
elif condition2:
    action2
else:
    action3
```

The condition is followed by a colon, just like with the **if-else** statements, the desired action is placed within the **elif** body and don't forget about an **indentation**, i.e., 4 spaces at the beginning of a new line. Here we first check the condition1 in the **if** statement and if it holds (the value of the Boolean expression is **True**), action1 is performed. If it is **False**, we skip action1 and then check the condition2 in the **elif** statement. Again, if condition2 is True, action2 is performed, otherwise, we skip it and go to the **else** part of the code.

Let's take a look at the example below:

```
# elif example
price = 10000 # there should be some int value
if price > 5000:
    print("That's too expensive!")
elif price > 500:
    print("I can afford that!")
else:
    print("That's too cheap!")
```

To buy or not to buy? To answer the question we first check if the price is higher than 5000. If 'price > 5000' is True, we print that it's too expensive and set off, looking for something cheaper. But what if the price was less than 5000? In this case, we check the next condition is 'price > 500', and again, if it is True, we print out that we can afford that, and if it is False, we go to the **else** block and print that it's too cheap. So "I can afford it!" will be printed if the price is less than 5000 but more than 500, and "That's too cheap" if the price is lower than 500.

Elif statement differs from **else** in one key moment: it represents another specific option while **else** fits all cases that don't fall into the condition given in **if** statement. That's why sometimes you may encounter conditional statements without **else**:

```
pet = 'cat'  # or 'dog'?

# cats vs dogs conditional
if pet == 'cat':
    print('Oh, you are a cat person. Meow!')
elif pet == 'dog':
    print('Oh, you are a dog person. Woof!')
```

In this example, it's possible to add an **else** statement to slightly expand a perspective, but it's not necessary if we are only interested in dogs and cats.

Current topic:

✓ Elif statement Stage 1 15★ ···

Topic depends on:

✓ Else statement Stage 1 15★ ···

Topic is required for:

✓ Functional decomposition Stage 1 7★ ···

Decorators in OOP ···

User-defined exceptions ···

§1. Why elif and not if?

The last example probably made you wonder: why did we use `elif` statement instead of just two `if`statements? Wouldn't two `if` statements be easier?

```
if pet == 'cat':
    print('Oh, you are a cat person. Meow!')
if pet == 'dog':
    print('Oh, you are a dog person. Woof!')
```

In this particular case, the result would be the same as with `elif`. But this wouldn't work as needed for the first example of this topic:

```
price = 6000
if price > 5000:
    print("That's too expensive!")
if price > 500:
    print("I can afford that!")
else:
    print("That's too cheap!")
# The output is 'That's too expensive!\nI can afford that!'
```

See? We got two contradicting messages instead of one that we originally intended to output. The difference between the above examples is that in the example with pets, the cases described by conditional statements are ***mutually exclusive***, that is, there's no string that would be equal both to `'dog'` and `'cat'` at the same. In the other example, the cases aren't mutually exclusive, and there are values for `price` that can satisfy both conditions.

So, `elif` statement is a better alternative than two `if` statements when you want to show that only one of the conditions is supposed to be satisfied. A chain of `if` statements implies that the conditions stated in them are totally unrelated and can be satisfied independently of each other, like in the following example:

```
animal = 'unicorn'
if animal in 'crow, dog, frog, pony':
    print('This animal exists')
if animal in 'unicorn, pegasus, pony':
    print('This animal is a horse')
```

With this distinction in mind, you'll be able to make your code more clear and less error-prone. Now, let's get back to studying `elif` functionality.

§2. Multiple elifs and a decision tree

There can be as many **elif** statements as you need, so your conditions can be very detailed. No matter how many **elif** statements you have, the syntax is always the same. The only difference is that we add more **elifs**:

```
# multiple elifs syntax
if condition1:
    action1
elif condition2:
    action2
# here you can add as many elifs as you need
elif conditionN:
    actionN
else:
    actionN1
```

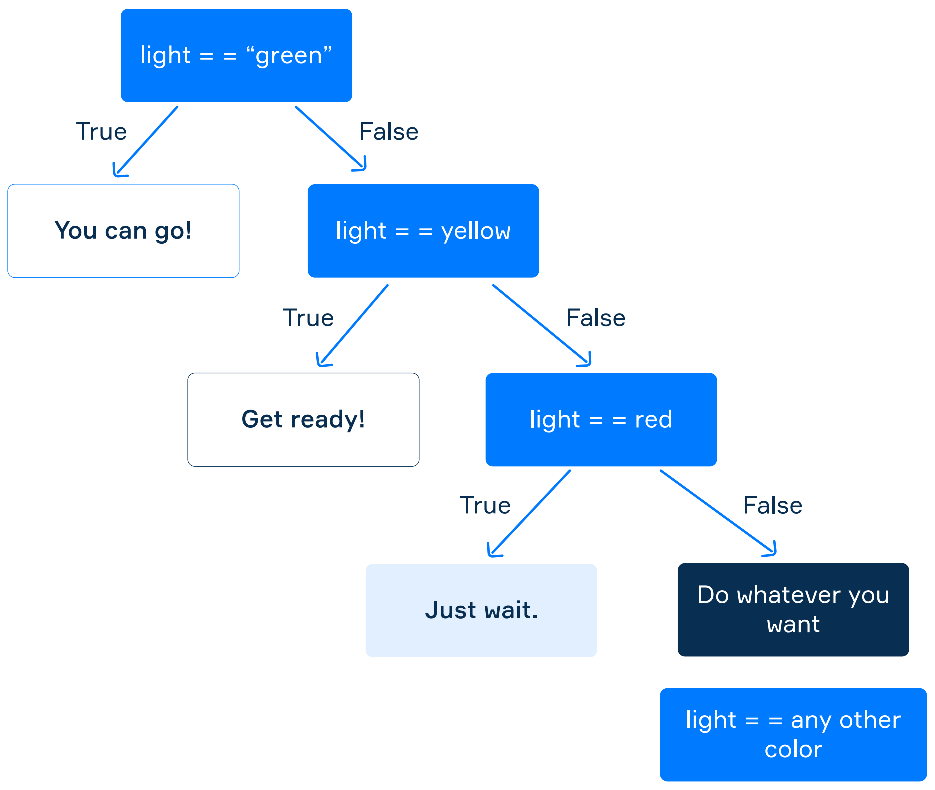
The code inside the **else** block is executed only if all conditions before it are **False**. See the following example:

```
# multiple elifs example
light = "red"  # there can be any other color
if light == "green":
    print("You can go!")
elif light == "yellow":
    print("Get ready!")
elif light == "red":
    print("Just wait.")
else:

    print("No such traffic light color, do whatever you want")
```

In this program, the message from the **else** block is printed for the light of any color except green, yellow and red for which we've written special messages.

Conditionals with multiple branches make a **decision tree**, in which a node is a Boolean expression and branches are marked with either **True** or **False**. The branch marked as True leads to the action that has to be executed, and the False branch leads to the next condition. The picture below shows such a decision tree of our example with traffic lights.



§3. Nested elif statements

Elif statements can also be **nested**, just like **if-else** statements. We can rewrite our example of traffic lights with nested conditionals:

```
# nested elifs example
traffic_lights = "green, yellow, red"
# a string with one color
light = "purple"  # variable for color name
if light in traffic_lights:
    if light == "green":
        print("You can go!")
    elif light == "yellow":
        print("Get ready!")

    else:

        # if the lights are red

        print("Just wait.")

else:

    print("No such traffic light color, do whatever you want")
```

Since you have mastered the topic of conditionals, from now on you can make your program do what you want when you want it!

Report a typo

901 users liked this theory. **10** didn't like it. **What about you?**

Table of contents:

😍 🙂 😐 🙁 😡

Start practicing

Comments (12) Hints (1) Useful links (1) Show discussion