PYTHON SNIPPETS

Python's "Ternary Operator"

Phil Best
2 min read · Aug 26

Welcome to another Python snippet post. This week we're going to take a brief look at conditional expressions in Python, sometimes referred to as Python's ternary operator.

Conditional expressions are a slightly obscure bit of syntax in Python, but they essentially allow us to assign values to variables based on some condition.

Let's take a look at a quick example:

```
x = 6
value = x if x < 10 else "Invalid value"  # 6
```

In this case we have some value bound to the variable `x`, and we check if the value of `x` is less than `10`. If it is, we assign the number to `value`; otherwise, we assign the string, `"Invalid value"`.

We can see a case where the value of `x` is not less than `10` below:

```
value = x if x < 10 else "Invalid value"  # Invalid value
```

Let's break down the syntax.

First we start with the value to return if the condition is `True`. In our case, this is `x`. We then have the `if` keyword followed by some condition. In our case this is a comparison using the less than operator, but any expression which can evaluated to a Boolean value is fine. After the condition, we use the `else` keyword, followed by the value to return if the condition evaluates to `False`.

```
<value if condition True> if <condition> else <value if condition Fa
```

One thing to keep in mind when it comes to conditional expressions is that we actually need all of the parts. We can't simply do away with the `else` clause if we don't care about it. We have to explicitly define a value for if the condition evaluates to `False`.

Failing to add an `else` clause results in a `SyntaxError`.

Conditional expressions can be chained by appending to the `else` clause, but the syntax is already confusing enough that I wouldn't ever recommend doing this:

```
x = 16
value = x if x < 10 else "Invalid value" if x < 15 else "Super inval
```

So, should you be using conditionally expressions all over your own code? Probably not.

Personally, I think the order of the conditions and the return values is pretty unintuitive, and it can be hard to follow the logic of these conditional expressions. Often it's much clearer to just use an if statement, even if it's a little longer:

```
x = 10

if x < 10:
    value = x
else:
    value = "Invalid value"
```

There are, however, plenty of examples of this structure being used in the wild, so it's important to be able to recognise it and understand it when it's used in other people's code.

Wrapping up

That's it for this snippet post. Hopefully you learnt something new, and I hope you find some use cases for conditional expressions in your own code. Sometimes they can be very useful.

If you want to upgrade your Python skills even further, I'd recommend checking out our Complete Python Course over on Udemy! There's over 35 hours of material, as well as dozens of exercises and projects to get you really comfortable working with Python.

I'd also recommend signing up to your mailing list below, as we post regular discount codes for our courses, ensuring you get the best deal.

Phil Best

Python Snippets

Assignment expressions in Python →

Python's namedtuples →

Python's divmod Function →

→ See all 26 posts

LEARN PYTHON PROGRAMMING

How to write decorators in Python

post we'll explain them from the ground up, including how they work, their syntax, and much more!

Jose Salvatierra
8 min read · Aug 29

Like what you see? Enter your e-mail to hear when new posts come out!

E-mail*

Name*

Receive our newsletter and get notified about new posts we publish on the site, as well as occasional special offers.

Sign Up