Python or Operator

Summary: in this tutorial, you'll learn about the Python `or` operator and how to use it effectively.

Introduction to the Python or operator

The `or` operator is a logical operator (https://www.pythontutorial.net/python-basics/python-logical-operators/) . Typically, you use the `or` operator to combine two Boolean expressions and return a Boolean value (https://www.pythontutorial.net/python-basics/python-boolean/) .

The `or` operator returns `True` if one of the two operands is `True` . And it returns `False` only if both operands are `False` .

This truth table displays the result of the `or` operator:

x	y	x or y
True	True	True
True	False	True
False	True	True

x	y	x or y
False	False	False

The following example shows how to use the `or` operator:

```
is_admin = False
is_editor = True
can_edit = is_admin or is_editor

print(can_edit)
```

Output:

```
True
```

The Python or operator is short-circuiting

When evaluating an expression that involves the `or` operator, Python can sometimes determine the result without evaluating all the operands. This is called short-circuit evaluation or lazy evaluation.

For example:

```
x or y
```

If `x` is truthy, then the `or` operator returns `x`. Otherwise, it returns `y`.

In other words, if `x` is truthy, then the `or` operator doesn't need to evaluate `y`. It just returns `x` immediately. This is why the evaluation is called lazy or short-circuiting evaluation.

The `or` operator only evaluates `y` and returns the result of the evaluation if `x` is falsy.

In Python, every object associates with a Boolean value (https://www.pythontutorial.net/advanced-python/python-bool/). And the `x` and `y` can be any object.

This opens some useful applications of the `or` operator.

Setting a default value for a variable

The `or` operator allows you to set a default value for a variable, for example:

```
var_name = value or default
```

In this example, if `value` is falsy, the `or` operator return the `default` .

The following example prompts you for input. If you don't enter anything, the `lang` will default to `'Python'` :

```
lang = input('Enter your language:') or 'Python'
print(lang)
```

The following example defines a function `get_data()` that returns a list of numbers. It uses the built-in `min()` function to find the lowest element in the list:

```
def get_data(args=None):
    if args:
        return [1, 2, 3]
    return []


lowest = min(get_data(args=true))
print(lowest)
```

Output:

```
1
```

It returned 1 as expected. However, the `get_data()` may return an empty list like this:

```
lowest = min(get_data())
print(lowest)
```

It returned a `ValueError`.

To fix this, you can use the `or` operator when calling the `min()` function:

```
def get_data(args=None):
    if args:
        return [1, 2, 3]
    return []


lowest = min(get_data() or [0])
print(lowest)
```

Output:

```
0
```

In this example, if the get_data() function returns an empty list, the `or` operator will treat its result as a falsy value.

Since the first operand is falsy, the `or` operator needs to evaluate the second operand `[0]`. In this case, you can specify the default minimum value in the second operand.