Walrus Operator

Table of contents

01

The What

What is the Walrus Operator?

What is the Walrus Operator?

Walrus Operator allows you to both **assign and return a value** in a single expression.

That means after evaluation of expression the result is assigned to variable and it can be reused without another evaluation.

The Walrus Operator is written as **:=**.

Syntax:

```
variable := expression
```

Here, **expression** is **evaluated** and the **result is assigned** to **variable**. The entire expression returns the **value of expression**.

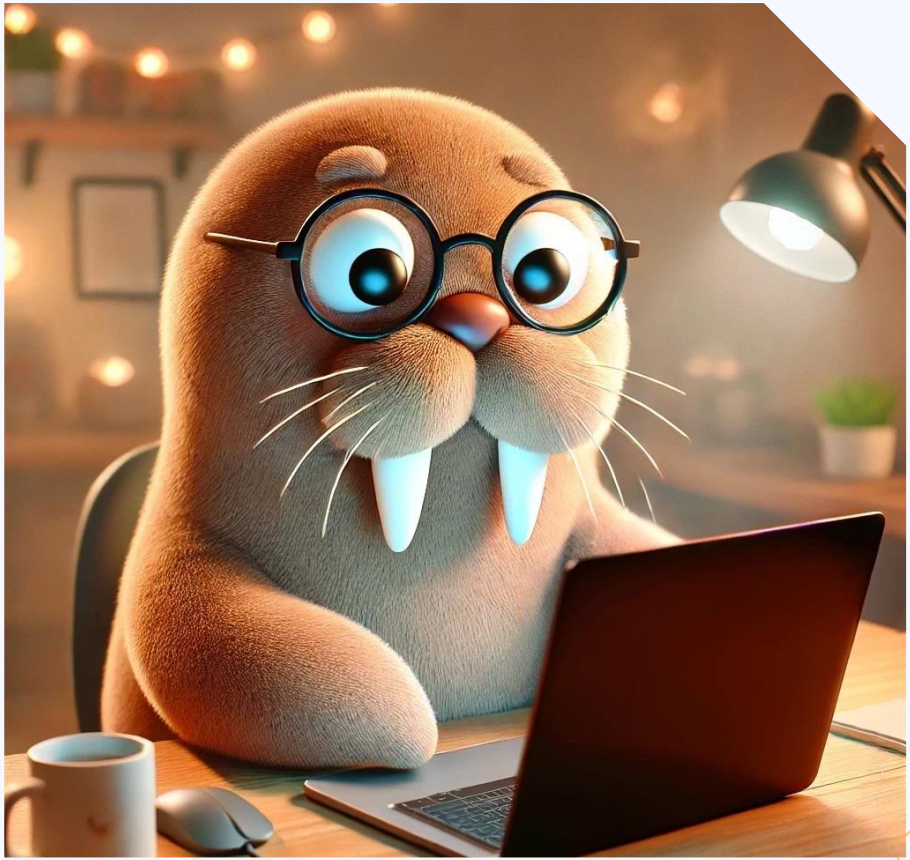
02

The Why

Why is it called the Walrus Operator?

Why is it called the Walrus Operator?

The operator looks like a **walrus eyes and tusks: :=.**

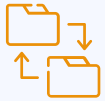


03

The Where

Where Can You Use the Walrus Operator?

Where Should You Use the Walrus Operator?



Loops

where an assignment is part of the loop condition.

Comprehensions

where the same expression needs to be evaluated multiple times.

Conditions

where you want to assign and test a variable in a single step.

In a while loop

The Walrus Operator shines in situations like loops where you might otherwise need to evaluate an expression twice—once to assign it and once to check it.

Without Walrus

Python

```python
line = input("Enter a line: ")
while line != "quit":
    print(f"You entered: {line}")
    line = input("Enter a line: ")
```

With Walrus

Python

```python
while (line := input("Enter a line: ")) != "quit":
    print(f"You entered: {line}")
```

Here, the **line** variable is assigned the input, and the condition is checked in the same line.

In list comprehensions

The Walrus Operator can be used to perform both the filtering and the assignment in list comprehensions, which makes them more efficient and readable.

Without Walrus

Python

```python
numbers = [1, 2, 3, 4, 5]
results = [n**2 for n in numbers if n**2 > 10]
```

With Walrus

Python

```python
numbers = [1, 2, 3, 4, 5]
results = [y for n in numbers if (y := n**2) > 10]
```

The calculation **n**2** is only performed once per iteration using the Walrus Operator, making the code slightly more efficient.

In if statements

You can use the Walrus Operator in an **if** statement to assign a value and then use it within the same conditional block.

Without Walrus

Python

```
text = input("Enter something: ")
if len(text) > 5:
    print(f"Input is longer than 5 characters: {text}")
```

With Walrus

Python

```
if (length := len(text := input("Enter something: "))) > 5:
    print(f"Input is longer than 5 characters: {text}")
```

04

Limitations

Limitations and Considerations

Limitations and Considerations

Complexity

Overuse of the Walrus Operator can make the code harder to read

Python version

The Walrus Operator is only available in Python 3.8 and later versions. Code that uses it will not run in earlier versions.

Knowledge Reinforcement

1

Question

What is the primary purpose of the Walrus Operator in Python?

Answer

a) To write multi-line expressions

b) To assign and return a value in a single expression

c) To define a lambda function

d) To declare global variables

1

Question

What is the primary purpose of the Walrus Operator in Python?

Answer

a) To write multi-line expressions

b) To assign and return a value in a single expression

c) To define a lambda function

d) To declare global variables

2

Question

Which version of Python introduced the Walrus Operator?

Answer

a) Python 3.6

b) Python 3.7

c) Python 3.8

d) Python 3.9

2

Question

Which version of Python introduced the Walrus Operator?

Answer

a) Python 3.6

b) Python 3.7

c) Python 3.8

d) Python 3.9

3

Question

The Walrus Operator can only be used inside loops.

Answer

a) True

b) False

3

Question

The Walrus Operator can only be used inside loops.

Answer

a) True

b) False

4

Question

What’s the benefit of using the Walrus Operator in list comprehensions?

Answer

a) To create nested loops

b) To reduce the number of calculations by reusing expressions

c) To check if elements are None

d) To handle exceptions in a single line

4

Question

What’s the benefit of using the Walrus Operator in list comprehensions?

Answer

a) To create nested loops

b) To reduce the number of calculations by reusing expressions

c) To check if elements are None

d) To handle exceptions in a single line

WATCH

Level up your coding with each episode in this focused Python series.

Next Video!

User Input