Python f-String Tutorial – String Formatting in Python Explained with Code Examples

When you're formatting strings in Python, you're probably used to using the `format()` method.

But in Python 3.6 and later, you can use *f-Strings* instead. f-Strings, also called *formatted string literals,* have a more succinct syntax and can be super helpful in string formatting.

In this tutorial, you'll learn about f-strings in Python, and a few different ways you can use them to format strings.

What are f-Strings in Python?

Strings in Python are usually enclosed within double quotes (`""`) or single quotes (`''`). To create f-strings, you only need to add an `f` or an `F` before the opening quotes of your string.

> For example, `"This"` is a string whereas `f"This"` is an f-String.

How to Print Variables using Python f-Strings

When using f-Strings to display variables, you only need to specify the names of the variables inside a set of curly braces `{}`. And at runtime, all variable names will be replaced with their respective values.

> If you have multiple variables in the string, you need to enclose each of the variable names inside a set of curly braces.

The syntax is shown below:

```
f"This is an f-string {var_name} and {var_name}."
```

▶ Here's an example.

You have two variables, `language` and `school`, enclosed in curly braces inside the f-String.

```
language = "Python"
school = "freeCodeCamp"
print(f"I'm learning {language} from {school}.")
```

Let's take a look at the output:

```
#Output
I'm learning Python from freeCodeCamp.
```

Notice how the variables `language` and `school` have been replaced with `Python` and `freeCodeCamp`, respectively.

How to Evaluate Expressions with Python f-Strings

As f-Strings are evaluated at runtime, you might as well evaluate valid Python expressions on the fly.

▶ In the example below, `num1` and `num2` are two variables. To calculate their product, you may insert the expression `num1 * num2` inside a set of curly braces.

```
num1 = 83
num2 = 9
print(f"The product of {num1} and {num2} is {num1 *
num2}.")
```

Notice how `num1 * num2` is replaced by the product of `num1` and `num2` in the output.

```
#Output
The product of 83 and 9 is 747.
```

I hope you're now able to see the pattern.

In any f-String, `{var_name}`, `{expression}` serve as placeholders for variables and expressions, and are replaced with the corresponding values at runtime.

Head over to the next section to learn more about f-Strings.

How to Use Conditionals in Python f-Strings

Let's start by reviewing Python's `if..else` statements. The general syntax is shown below:

```
if condition:
  # do this if condition is True <true_block>
else:
  # do this if condition is False <false_block>
```

Here, `condition` is the expression whose truth value is checked.

- If the `condition` evaluates to `True`, the statements in the if block (`<true_block>`) are executed.
- If the `condition` evaluates to `False`, the statements in the else block (`<false_block>`) are executed.

There's a more succinct one-line equivalent to the above `if..else` blocks. The syntax is given below:

```
<true_block> if <condition> else <false_block>
```

> In the above syntax,`<true block>` is what's done when the `condition` is `True`, and `<false_block>` is the statement to be executed when the condition is `False`.

This syntax may seem a bit different if you haven't seen it before. If it makes things any simpler, you may read it as, "*Do this* `if` `condition` is `True`; `else`, *do this*".

This is often called the *ternary* operator in Python as it takes 3 operands in some sense – the *true block*, the *condition* under test, and the *false block*.

▶ Let's take a simple example using the ternary operator.

Given a number `num`, you'd like to check if it's even. You know that a number is even if it's evenly divisible by 2. Let's use this to write our expression, as shown below:

```
num = 87;
print(f"Is num even? {True if num%2==0 else False}")
```

In the above code snippet,

- `num%2==0` is the condition.
- If the condition is `True`, you just return `True` indicating that `num` is indeed even, and `False` otherwise.

```
#Output
Is num even? False
```

In the above example, `num` is 87, which is odd. Hence the conditional statement in the f-String is replaced with `False`.

How to Call Methods with Python f-Strings

So far, you've only seen how to print values of variables, evaluate expressions, and use conditionals inside f-Strings. And it's time to level up.

▶ Let's take the following example:

```
author = "jane smith"
print(f"This is a book by {author}.")
```

The above code prints out `This is a book by jane smith.`

Wouldn't it be better if it prints out `This is a book by Jane Smith.` instead? Yes, and in Python, string methods return modified strings with the requisite changes.

> The `title()` method in Python returns a new string that's formatted in the title case - the way names are usually formatted (`First_name Last_name`).

To print out the author's name formatted in title case, you can do the following:

- use the `title()` method on the string `author`,
- store the returned string in another variable, and
- print it using an f-String, as shown below:

```
author = "jane smith"
a_name = author.title()
print(f"This is a book by {a_name}.")

#Output
This is a book by Jane Smith.
```

However, you can do this in just one step with f-Strings. You only need to call the `title()` method on the string `author` inside the curly braces within the f-String.

```
author = "jane smith"
print(f"This is a book by {author.title()}.")
```

When the f-String is parsed at runtime,

- the `title()` method is called on the string `author`, and
- the returned string that's formatted in title case is printed out.

You can verify that in the output shown below.

```
#Output
This is a book by Jane Smith.
```

You can place method calls on any valid Python object inside the curly braces, and they'll work just fine.

How to Call Functions Inside Python f-Strings

In addition to calling methods on Python objects, you can also call functions inside f-Strings. And it works very similarly to what you've seen before.

Just the way variable names are replaced by values, and expressions are replaced with the result of evaluation, function calls are replaced with the return value from the function.

▶ Let's take the function `choice()` shown below:

```
def choice(c):
  if c%2 ==0:
    return "Learn Python!"
  else:
    return "Learn JavaScript!"
```

The above function returns `"Learn Python!"` if it's called with an even number as the argument. And it returns `"Learn JavaScript!"` when the argument in the function call is an odd number.

▶ In the example shown below, you have an f-String that has a call to the choice function inside the curly braces.

```
print(f"Hello Python, tell me what I should learn.
{choice(3)}")
```

As the argument was an odd number (`3`), Python suggests that you learn JavaScript, as indicated below:

```
#Output
Hello Python, tell me what I should learn. Learn
JavaScript!
```

If you call the function `choice()` with an even number, you see that Python tells you to learn Python instead. 🙂

```
print(f"Hello Python, tell me what I should learn.
{choice(10)}")
```

```
#Output
Hello Python, tell me what I should learn. Learn
Python!
```

And that ends our tutorial on a happy note!

Conclusion

In this tutorial, you've learned how you can use f-Strings to:

- print values of variables,
- evaluate expressions,
- call methods on other Python objects, and
- make calls to Python functions.

Related Posts

Here's a post by Jessica that explains string formatting using the `format()` method.