14.159 - Format Strings

July 23, 2018

0.1 Format strings

Sometimes you want to put some values in a string. For example, you want to create the output, "I have 5 dogs". Up to now this was always quite unwieldy, with the str() function:

```
In [1]: n = 5
        print("I got " + str(n) + " dogs")
```

```
I have 5 dogs
```

In this lesson you will learn a new, practical method to create such an output!

0.1.1 Let's look at an example

Imagine you want to translate your application. That's when you'll run into problems:

```
In [2]: n = 5
        print("Yo tengo " + str(n) + " perros")
        print("I got " + str(n) + " dogs")
```

```
Yo tengo 5 perros
I got 5 dogs
```

How do you get it now that the language module is interchangeable and not "interwoven" with the `+ str(n) +` command?

Idea: You could define the language modules in a dictionary and use a placeholder via `.replace()`:

```
In [3]: translations = {
            "number_of_dogs": "I got XXX dogs"
        }
        print(translations["number_of_dogs"].replace("XXX", str(n)))
```

```
I got 5 dogs
```

Problem: This will quickly become confusing.

Solution: Fortunately Python provides us here with a `.format()` method, which makes it a lot easier. We use `{0}` for the position where we want to use the `n` parameter of the `.format(n)` call:

```
In [4]: print("I got {0} dogs".format(n))

I got 5 dogs
```

Result: Our translation code is much more pleasant:

```
In [5]: translations = {
            "number_of_dogs": "I got {0} dogs"
        }
        print(translations["number_of_dogs"].format(n))

I got 5 dogs
```

This `.format()` method also works with multiple parameters. Here `{0}` defines the position for the first parameter, `{1}` the position where the 2nd parameter should be set.

In the following case `{1}` is replaced by `"cats"` and `{0}` by the number 5:

```
In [6]: print("I got {1} {0}x".format(5, "cats"))

I got cats 5x
```

Decimal and round numbers The `format()` function allows you to comfortably round comma numbers. A `f` is appended to a formatting command (e.g. `{0}` or `{1}`) within the curly brackets.

This tells Python that this number should be considered a comma number. Accordingly, the number 5 is used here now, but the `.000000` are added, because it is displayed as a comma number:

```
In [7]: print("I got so many cats: {0:f}".format(5))

I got so many cats: 5.000000
```

If we want to limit the number of digits, we can do this by writing `.2` after the colon to limit the number of decimal places to 2, for example:

```
In [8]: print("I got so many cats: {0:.2f}".format(5))

I got so many cats: 5.00
```

We can now use this for real comma numbers to round them! :-)

```
In [9]: print("Pi has the value: {0:.3f}".format(3.141529))

Pi has the value: 3.142
```

0.1.2 Name parameters

If you have a string with many placeholders, the notation `{0}`, `{1}`, `{2}`, `{3}`, `{4}`, `{5}`, ... confusing at some point.

Fortunately, we can also name the parameters. It is important that if a parameter on the left side is called `{animal}`, the parameter `animal` is passed to the `.format()` function accordingly.

In this case, the value of the `.format()` call `animal = "dogs"` (specifically the word "dogs") is used instead of `{animal}`:

```
In [11]: print("I got {number:.3f} {animal}s".format(number = 5, animal = "dog"))
```

```
I got 5.000 dogs
```