PYTHON SNIPPETS

Python's zip_longest Function

Phil Best

2 min read · Jul 22

In last week's snippet post we talked about the incredibly powerful `zip` function, and how we can use it to make our loops more Pythonic. This week we're going to be talking about the less well-known brother of `zip`: `zip_longest`.

`zip_longest` lives in the `itertools` module, which we've spoken about briefly before. `itertools` contains all kinds of useful functions revolving around iterative operations.

So how does `zip_longest` differ from plain old `zip`? Why should we care about it?

Well, when we use `zip`, `zip` will stop combining our iterables as soon as one of them runs out of elements. If the other iterables are longer, we just throw those excess items away. Take a look at this example:

```
l_1 = [1, 2, 3]
l_2 = [1, 2]

combinated = list(zip(l_1, l_2))
```

```
print(combinated)  # [(1, 1), (2, 2)]
```

As you can see, we just lost that `3` in `l_1`. That can sometimes be pretty problematic. We don't generally want to be throwing away data like this.

Luckily we have `zip_longest` here to save us.

Let's look at our example above again. This time using `zip_longest`.

```
from itertools import zip_longest

l_1 = [1, 2, 3]
l_2 = [1, 2]

combinated = list(zip_longest(l_1, l_2, fillvalue="_"))

print(combinated)  # [(1, 1), (2, 2), (3, '_')]
```

There are a few things to note here. For one, we got a third tuple in our zip object, meaning that the longer list was able to provide all of its values. Secondly, we have this keyword argument called `fillvalue`.

If we look at the `print` output for our little code segment, we can get some indication of what this does. As we can see, where there was no value to match to `3`, `zip_longest` matched `3` to our `fillvalue`.

In essence, any time `zip_longest` doesn't have a value to match to one of our iterable's elements, it will place this `fillvalue` there to

plug the gaps.
We can really use any `fillvalue` we want here: numbers, lists, dictionaries, `None`. Whatever you can think of. This makes it incredibly versatile, and it's definitely a good one to know about.

In case you're interested, we can call `zip_longest` without a `fillvalue` argument, in which case it will default to using `None` as a `fillvalue`.

Wrapping up

That's it for this snippet post! I hope you learnt something new, and I'm sure you'll find all sorts of awesome ways to use `zip_longest` in your own code.

As always, if you're serious about improving your Python, I'd recommend you take a look at our Complete Python Course! It has over 35 hours of material, along with quizzes, exercises, and several large projects, so it's an awesome way to develop your Python skills.

You should also think about signing up to our mailing list below, as we post regular coupon codes for our courses, ensuring that you always get the best deal.

Happy coding!

Phil Best
I'm a freelance developer, mostly working in web development. I also create course content for Teclado!

Read More

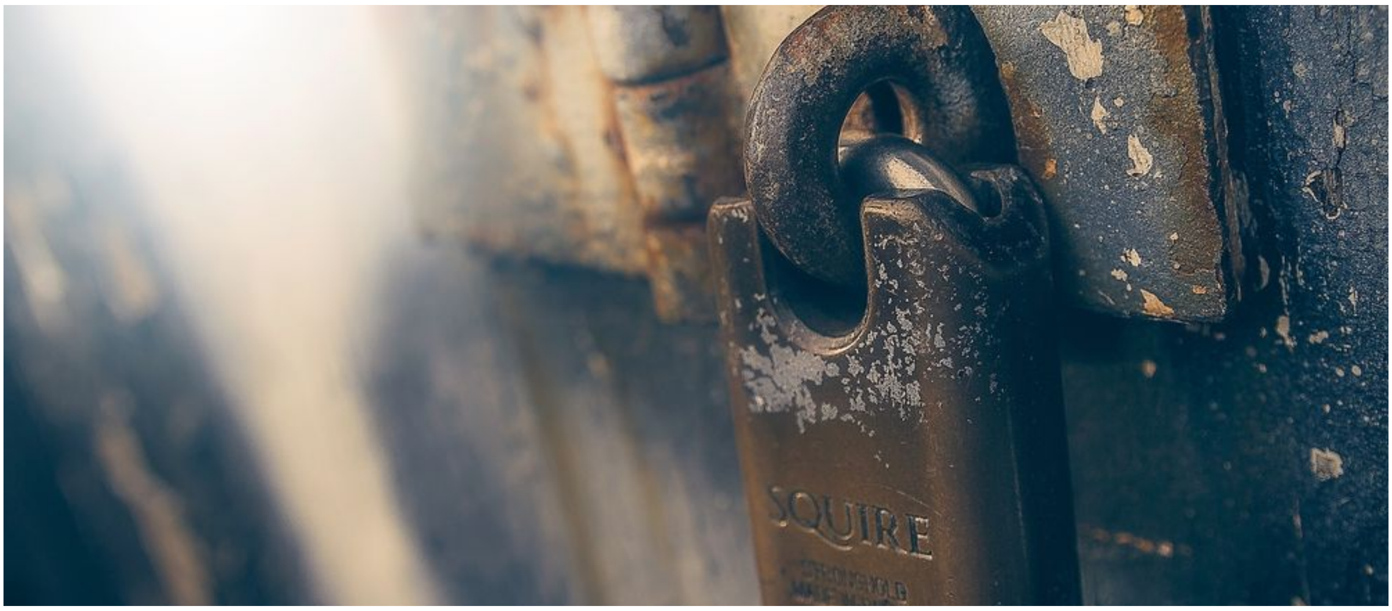
Python Snippets

Assignment expressions in Python →

Python's namedtuples →

Python's divmod Function →

→ See all 26 posts



FLASK

Protecting endpoints in Flask apps by requiring login

In this post we'll learn to prevent unauthorised logins to our Flask endpoints. We'll also look at decorators so that we can secure endpoints with a single line of code for each!

Jose Salvatierra
6 min read · Jul 25

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