You should get Packing with Python 3.5

Justus Perlwitz

2015-11-22

Contents


```
def froblog(**kwargs):
    print(kwargs.items())

some_kwargs = {
    'frob': 'lob',
    'frux': 'flib',}

other_kwargs_from_somewhere_else = {
    'floxblum': 'qux',}

more_kwargs = {
    'flipblip': 'foobar',}  # Etc., I think you get the point

some_kwargs.update(other_kwargs_from_somewhere_else)
some_kwargs.update(more_kwargs)

froblog(**some_kwargs)
```

```
dict_items([('flipblip', 'foobar'), ('frob', 'lob'), ('floxblum', 'qux'),
('frux', 'flib')])
```

Now, if we want to reuse `some_kwargs`, they are tainted with the other kwargs. Of course, we could have constructed a new dict and used that to pass all the dict key-value pairs that we want to pass. But either way it feels messy.

1 The solution

Python 3.5 introduced PEP 0448 titled "Additional Unpacking Generalizations". It allows you to do the following instead, assuming we're using the same `froblog` method from above:

```
some_kwargs = {
    'frob': 'lob',
    'frux': 'flib',}

other_kwargs_from_somewhere_else = {
    'floxblum': 'qux',}

more_kwargs = {
    'flipblip': 'foobar',}  # Etc., I think you get the point

froblog(**some_kwargs, **other_kwargs_from_somewhere_else, **more_kwargs)
```

It outputs the same as above:

```
dict_items([('flipblip', 'foobar'), ('frob', 'lob'), ('floxblum', 'qux'),
('frux', 'flib')])
```

This is considerably cleaner. For more awesome unpacking madness, do check out the PEP 0448 document examples.

Happy unpacking!