And parametrize a test method

```
from my_test_framework.data.shirts import shirts

@pytest.mark.parametrize('shirts_data',  # name of parameter
                         shirts,         # data model
                         ids=[shirts[shirt]['id'] for shirt in shirts])  # ids for tests
def test_add_shirt(self, 'shirts_data'):
    # disambiguate parameter data
    id = shirts_data[‘id’]
    color = shirts_data[‘color’]
    size = shirts_data[‘size’]
    # navigate to the product page for `shirt`
    # verify that you can find `shirt`
    # add `shirt` to cart
    # verify that `shirt` is in cart
```

And generate parametrized test cases, dynamically!

```
$ pytest tests/test_params.py --collect-only
============ test session starts ============
collected 8 items
<Module 'test_params.py'>
  <Class 'SomeTests'>
    <Instance '()'>
      <Function 'test_add_shirt[blue_s]'>
      <Function 'test_add_shirt[blue_m]'>
      <Function 'test_add_shirt[blue_l]'>
      <Function 'test_add_shirt[blue_xl]'>
      <Function 'test_add_shirt[red_s]'>
      <Function 'test_add_shirt[red_m]'>
      <Function 'test_add_shirt[red_l]'>
      <Function 'test_add_shirt[red_xl]'>
```

And parametrize a test method

tests/test_simple.py

```
from my_test_framework.data.shirts import shirts

@pytest.mark.parametrize('shirts_data',  # name of parameter
                         shirts,          # data model
                         ids=[shirts[shirt]['id'] for shirt in shirts])  # ids for tests
def test_add_shirt(self, 'shirts_data'):
    # disambiguate parameter data
    id = shirts_data[‘id’]
    color = shirts_data[‘color’]
    size = shirts_data[‘size’]
    # navigate to the product page for `shirt`
    # verify that you can find `shirt`
    # add `shirt` to cart
    # verify that `shirt` is in cart
```