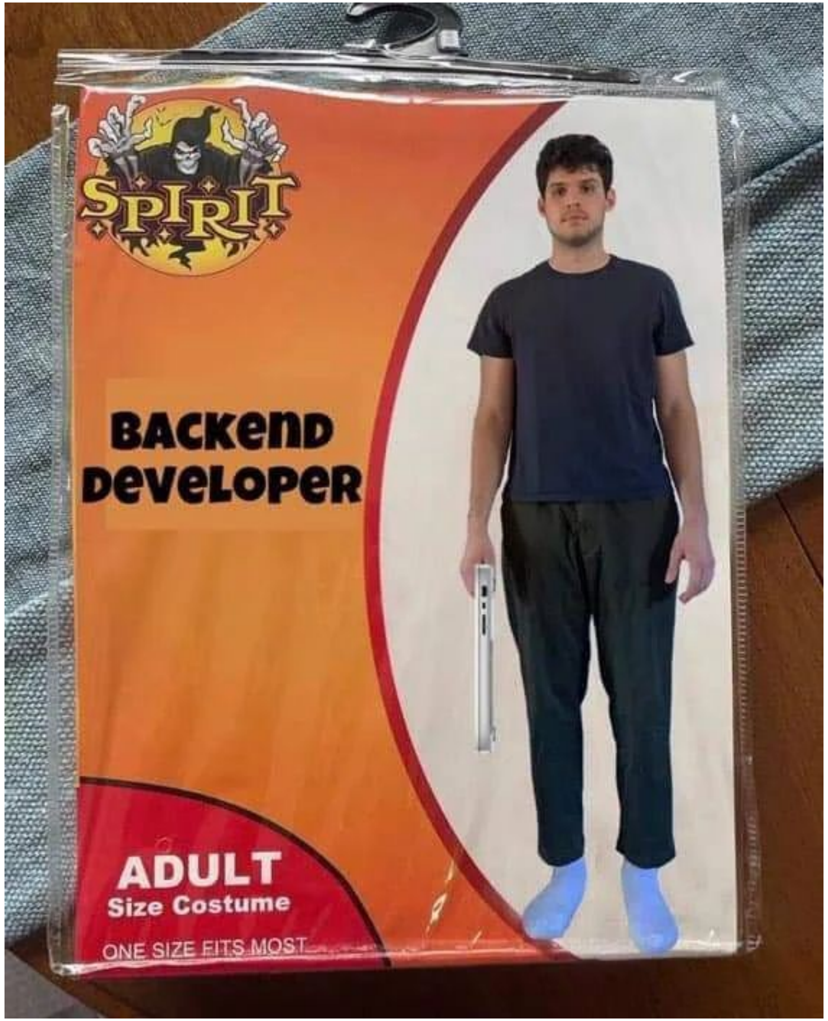
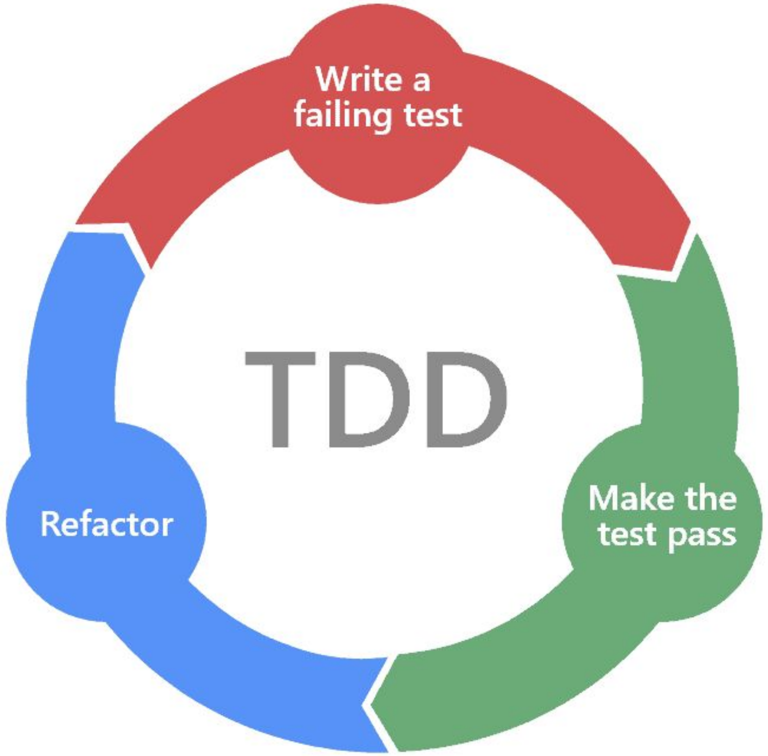
TDD - Cheat sheet for writing good tests on DRF

Reddy Tintaya @reddytocode

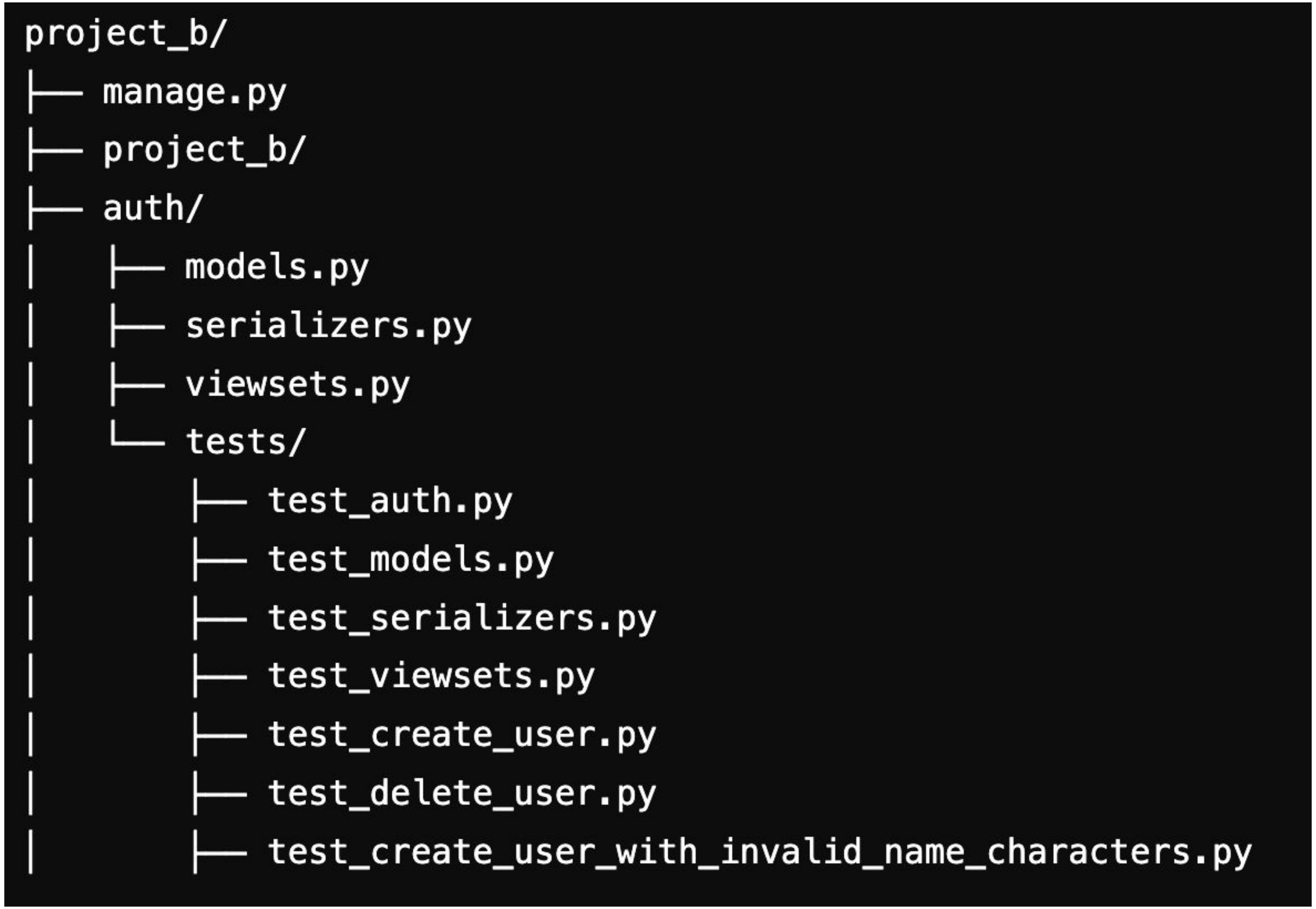
TDD - House of horrors


SPIRIT
BACKEND DEVELOPER
ADULT
Size Costume
ONE SIZE FITS MOST

Follow the mantra

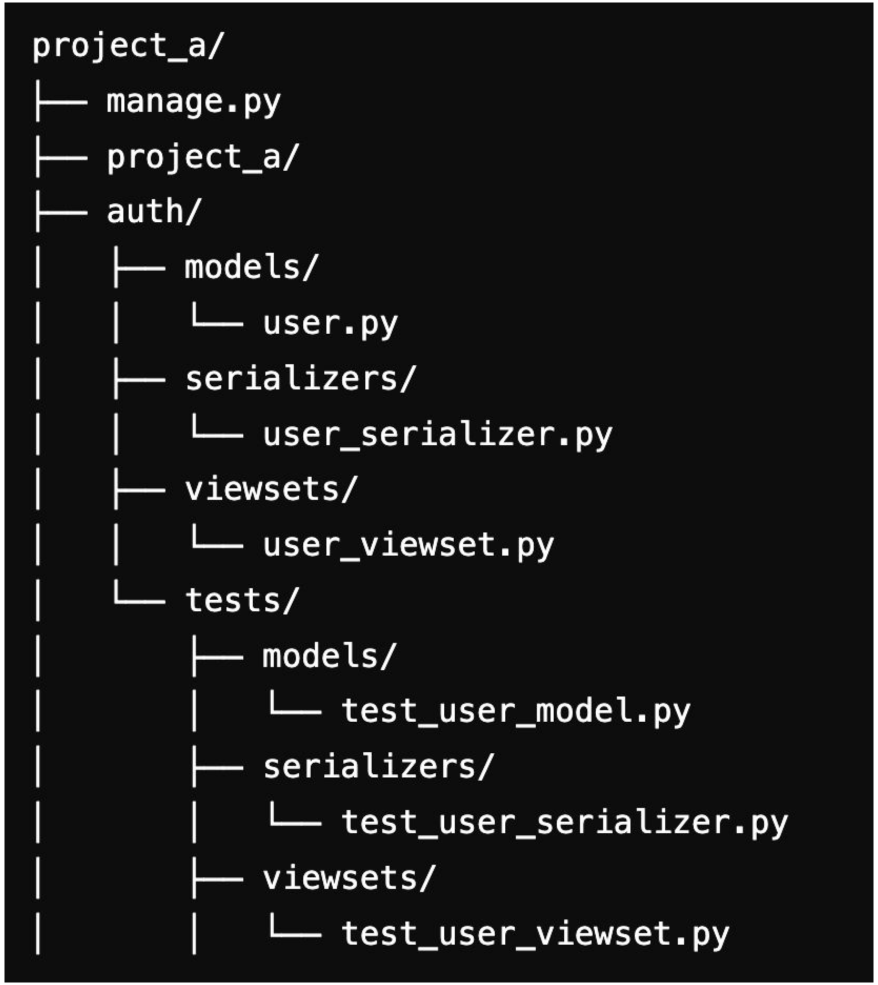


Consistency



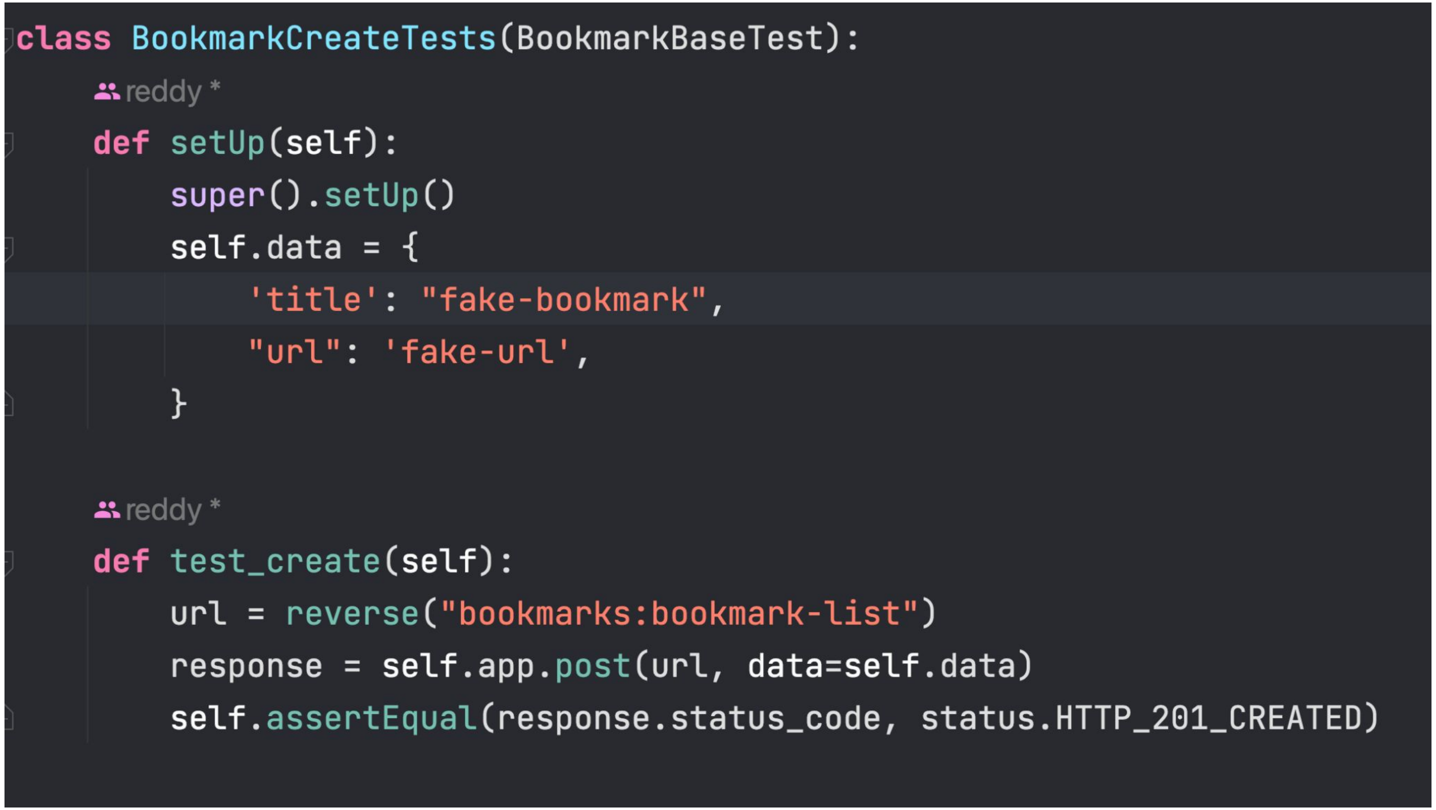
```
project_b/
├── manage.py
├── project_b/
├── auth/
│    ├── models.py
│    ├── serializers.py
│    ├── viewsets.py
│    └── tests/
│         ├── test_auth.py
│         ├── test_models.py
│         ├── test_serializers.py
│         ├── test_viewsets.py
│         ├── test_create_user.py
│         ├── test_delete_user.py
│         ├── test_create_user_with_invalid_name_characters.py
```

Consistency

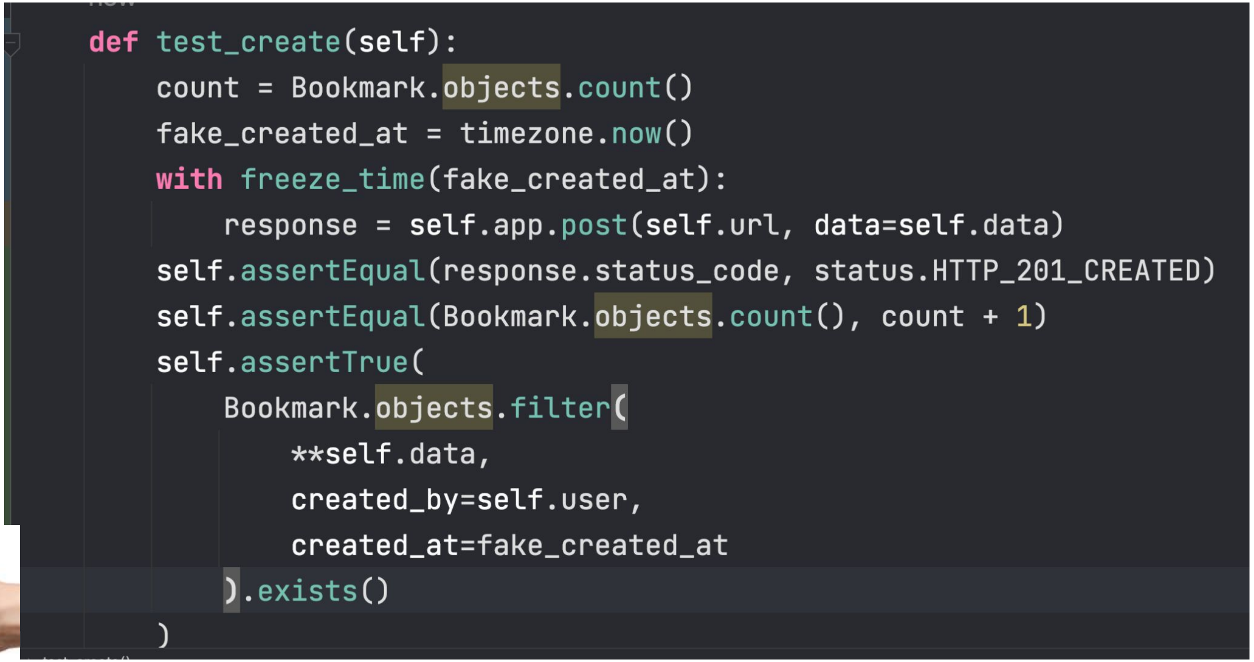


```
project_a/
├── manage.py
├── project_a/
├── auth/
│   ├── models/
│   │   └── user.py
│   ├── serializers/
│   │   └── user_serializer.py
│   ├── viewsets/
│   │   └── user_viewset.py
│   └── tests/
│       ├── models/
│       │   └── test_user_model.py
│       ├── serializers/
│       │   └── test_user_serializer.py
│       ├── viewsets/
│       │   └── test_user_viewset.py
```

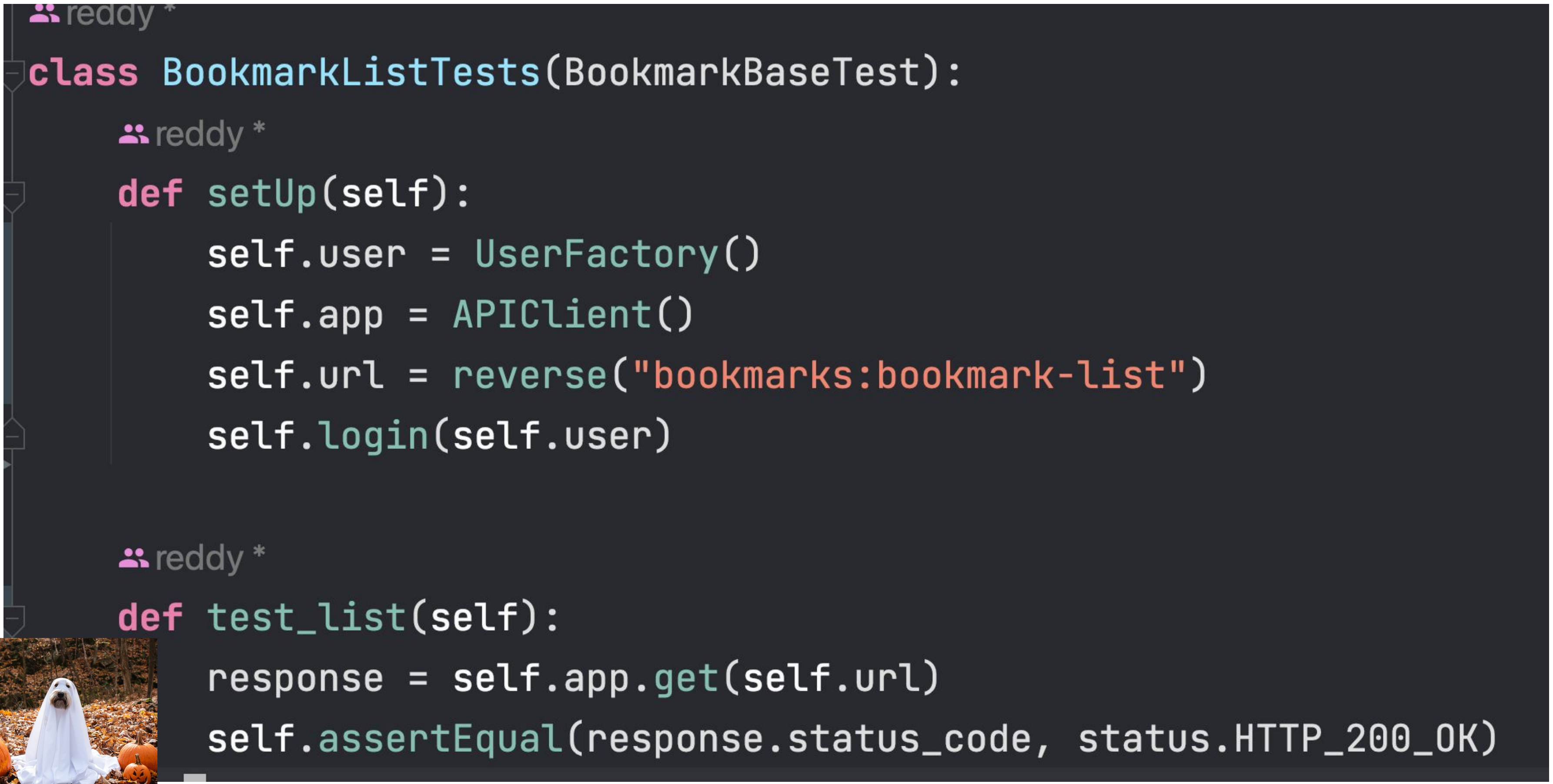
Test more than just status code



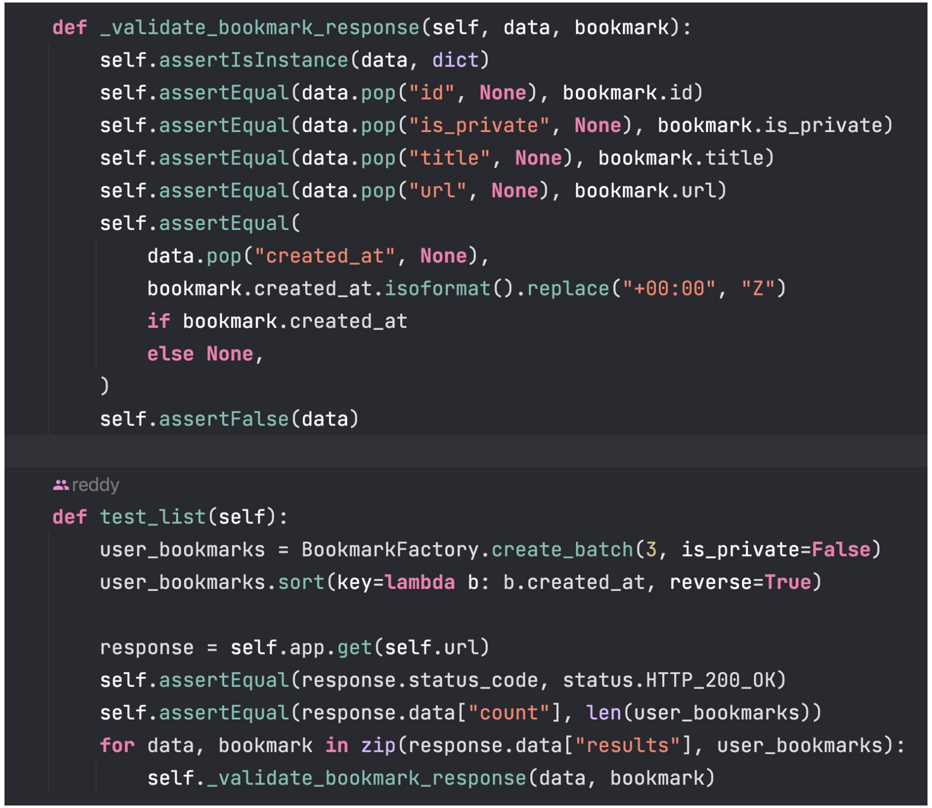
```
class BookmarkCreateTests(BookmarkBaseTest):
    reddy *
    def setUp(self):
        super().setUp()
        self.data = {
            'title': "fake-bookmark",
            "url": 'fake-url',
        }

    reddy *
    def test_create(self):
        url = reverse("bookmarks:bookmark-list")
        response = self.app.post(url, data=self.data)
        self.assertEqual(response.status_code, status.HTTP_201_CREATED)
```



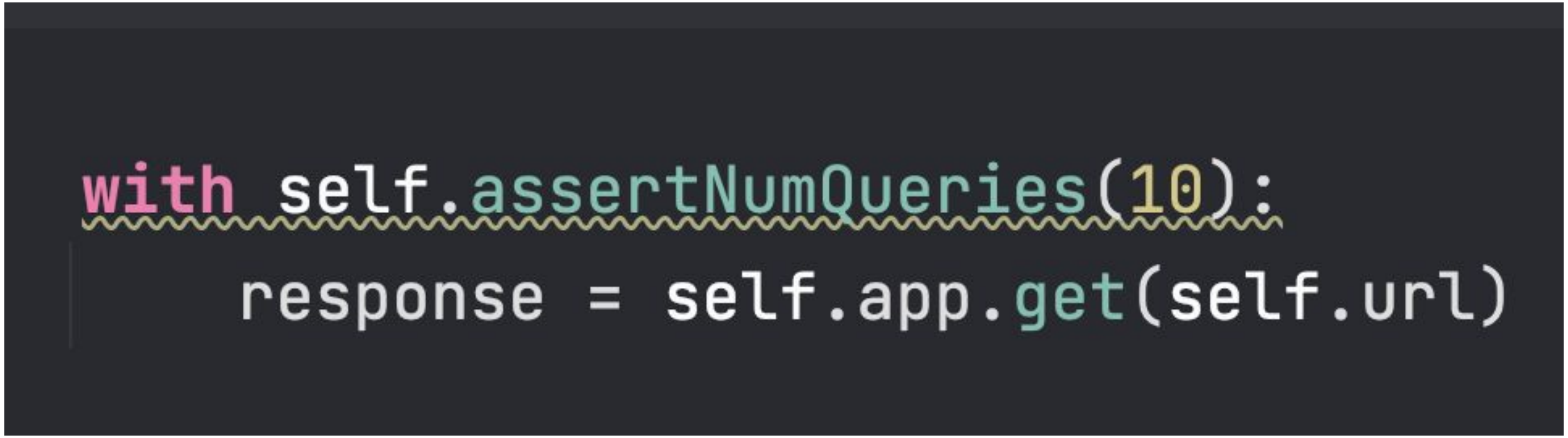
```
def test_create(self):
    count = Bookmark.objects.count()
    fake_created_at = timezone.now()
    with freeze_time(fake_created_at):
        response = self.app.post(self.url, data=self.data)
    self.assertEqual(response.status_code, status.HTTP_201_CREATED)
    self.assertEqual(Bookmark.objects.count(), count + 1)
    self.assertTrue(
        Bookmark.objects.filter(
            **self.data,
            created_by=self.user,
            created_at=fake_created_at
        ).exists()
    )
```



```
reddy *
class BookmarkListTests(BookmarkBaseTest):
    reddy *
    def setUp(self):
        self.user = UserFactory()
        self.app = APIClient()
        self.url = reverse("bookmarks:bookmark-list")
        self.login(self.user)

    reddy *
    def test_list(self):
        response = self.app.get(self.url)
        self.assertEqual(response.status_code, status.HTTP_200_OK)
```

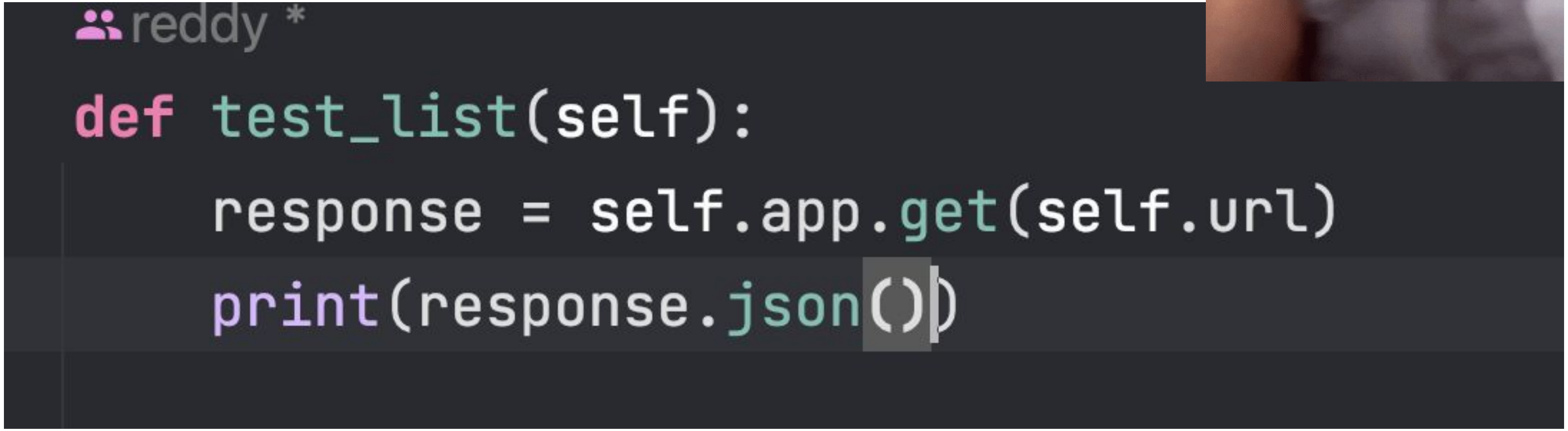


```python
def _validate_bookmark_response(self, data, bookmark):
    self.assertIsInstance(data, dict)
    self.assertEqual(data.pop("id", None), bookmark.id)
    self.assertEqual(data.pop("is_private", None), bookmark.is_private)
    self.assertEqual(data.pop("title", None), bookmark.title)
    self.assertEqual(data.pop("url", None), bookmark.url)
    self.assertEqual(
        data.pop("created_at", None),
        bookmark.created_at.isoformat().replace("+00:00", "Z")
        if bookmark.created_at
        else None,
    )
    self.assertFalse(data)

reddy
def test_list(self):
    user_bookmarks = BookmarkFactory.create_batch(3, is_private=False)
    user_bookmarks.sort(key=lambda b: b.created_at, reverse=True)

    response = self.app.get(self.url)
    self.assertEqual(response.status_code, status.HTTP_200_OK)
    self.assertEqual(response.data["count"], len(user_bookmarks))
    for data, bookmark in zip(response.data["results"], user_bookmarks):
        self._validate_bookmark_response(data, bookmark)
```



```python
with self.assertNumQueries(10):
    response = self.app.get(self.url)
```

Please, don’t forget to clean your prints before pushing your code



```
reddy *
def test_list(self):
    response = self.app.get(self.url)
    print(response.json())
```

Thanks