Using Data Classes to Self Document External APIs

Jeffrey Whitehead

serenitycode.co.uk

Our Code Example

```
from typing import Dict
import requests

def create_blog_post() -> Dict:
    request_data = {"title": "My New Post",
                    "body": "Lorem Ipsum",
                    "userId": 1,}
    response = requests.post("https://jsonplaceholder.typicode.com/posts", data=request_data)
    created_post = response.json()
    return created_post

print(create_blog_post())
```

The Issues

```python
from typing import Dict
import requests

def create_blog_post() -> Dict:
    request_data = {"title": "My New Post",
                    "body": "Lorem Ipsum",
                    "userId": 1,}
    response = requests.post("https://jsonplaceholder.typicode.com/posts", data=request_data)
    created_post = response.json()
    return created_post

print(create_blog_post())
```

- The supported request parameters are unknown
- The parameter types are unknown
- The response is unknown

Why Do We Care?

```python
from typing import Dict
import requests

def create_blog_post() -> Dict:
    request_data = {"title": "My New Post",
                    "body": "Lorem Ipsum",
                    "userId": 1,}
    response = requests.post("https://jsonplaceholder.typicode.com/posts", data=request_data)
    created_post = response.json()
    return created_post

print(create_blog_post())
```

- Readability
- Access to documentation

The Solution

```python
import dataclasses
from typing import Optional

@dataclasses.dataclass
class RequestData:
    title: str
    body: str
    userId: int
    author: Optional[str] = None


@dataclasses.dataclass
class ResponseData:
    id: int
    title: str
    body: str
    userId: int


def create_blog_post() -> ResponseData:
    request_data = RequestData(title="My New Post",
                               body="Lorem Ipsum",
                               userId=1,)
    response = requests.post("https://jsonplaceholder.typicode.com/posts", data=request_data.__dict__)
    return ResponseData(**response.json())
```

Side Benefits

- Unit Tests
- Code Completion
- Type Checking

```python
import dataclasses
from typing import Optional

@dataclasses.dataclass
class RequestData:
    title: str
    body: str
    userId: int
    author: Optional[str] = None


@dataclasses.dataclass
class ResponseData:
    id: int
    title: str
    body: str
    userId: int


def create_blog_post() -> ResponseData:
    request_data = RequestData(title="My New Post",
                               body="Lorem Ipsum",
                               userId=1,)
    response = requests.post("https://jsonplaceholder.typicode.com/posts", data=request_data.__dict__)
    return ResponseData(**response.json())
```

What About Docstrings?

```
import requests
from typing import Dict

def create_blog_post() -> Dict:
    """
    Return the blog post created by calling the placeholder API

    API request parameters
    ----------------------
      - title: str
        required: true
      - body: str
        required: true
      - userId: int
        required: true
      - author: str
        required: false

    Returns
    -------
        created_post: dict
            - id: int
            - title: str
            - body: str
            - userId: int

    """
    request_data = {"title": "My New Post",
                    "body": "Lorem Ipsum",
                    "userId": 1,}
    created_post = response.json()
    return created_post
```

Thanks for listening!

https://github.com/SerenityCode/djc-eu-data-classes-lightning

https://github.com/SerenityCode/djc-eu-data-classes-lightning