여기서는 assert문에서 테스트가 실패했다. 결과값이 2가 아니라 11로 되어있기 때문이다. 실패 내용은 AssertionError에 요약되어 표시된다. 이 경우에는 2 == 11이 아니기 때문에 실패했다고 알려준다.

pytest가 어떻게 실행되는지 간단히 살펴봤다. 다음으로 pytest의 픽스처fixture를 알아보자.

픽스처를 사용한 반복 제거

픽스처는 재사용할 수 있는 함수로, 테스트 함수에 필요한 데이터를 반환하기 위해 정의된다. pytest.fixture 데코레이터를 사용해 픽스처를 정의할 수 있으며 API 라우트 테스트 시 애플리케이션 인스턴스를 반환하는 경우 등에 사용된다. 테스트 함수가 사용하는 애플리케이션 클라이언트를 픽스처로 정의할 수 있기 때문에 테스트할 때마다 애플리케이션 인스턴스를 다시 정의하지 않아도 된다. 이 방식은 〈8.3 REST API 라우트 테스트 작성〉에서 자세히 다룬다.

픽스처를 어떻게 정의하는지 살펴보자.

```
import pytest
from models.events import EventUpdate

# 픽스처 정의
@pytest.fixture
def event() -> EventUdpate:
    return EventUpdate(
        title="FastAPI Book Launch",
        image="https://packt.com/fastapi.png",
        description="We will be discussing the contents of the FastAPI book in
this event.Ensure to come with your own copy to win gifts!",
        tags=["python", "fastapi", "book", "launch"],
        location="Google Meet"
    )

def test_event_name(event: EventUpdate) -> None:
    assert event.title == "FastAPI Book Launch"
```

이 코드는 EventUpdate pydantic 모델의 인스턴스를 반환하는 픽스처를 정의한다. 이 픽스처는 test_event_name() 함수의 인수로 사용되며 이벤트 속성에 접근할 수 있게 해준다.