```
def add(a: int , b: int) -> int:
    return a + b

def subtract(a: int, b: int) -> int:
    return b - a

def multiply(a: int, b: int) -> int:
    return a * b

def divide(a: int, b: int) -> int:
    return b // a
```

테스트 대상 함수를 만들었다. 이제 이 함수들을 테스트할 함수를 만들어야 한다. 테스트 함수는 계산 결과가 맞는지 검증하는 역할을 한다. assert 키워드는 식의 왼쪽에 있는 값이 오른쪽에 있는 처리 결과와 일치하는지 검증할 때 사용된다. 이 경우에는 사칙연산의 결과가 실제 계산 결과와 일치하는지 확인한다.

다음 테스트 함수를 테스트 파일에 추가하자.

```
def test_add() -> None:
    assert add(1, 1) == 2

def test_subtract() -> None:
    assert subtract(2, 5) == 3

def test_multiply() -> None:
    assert multiply(10, 10) == 100

def test_divide() -> None:
    assert divide(25, 100) == 4
```

일반적으로 테스트 파일이 아닌 별도의 파일에 테스트 대상 함수(add(), subtract(), multiply(), divide())를 정의한다. 그런 다음 이 파일을 테스트 파일에 임포트하여 테스트를 수행한다.

테스트 함수가 준비됐으면 테스트를 실행해보자. 테스트는 pytest 명령을 사용해 실행한다. 단, 이 명령은 명령을 실행하는 위치에 있는 모든 테스트 파일을 실행한다. 테스트를 하나만 실행하려면 파일명을 인수로 지정해야 한다. 다음 명령을 사용해 우리가 작성한 테스트 파일만 실행해보자.