/* elice */

도전! 디버깅 입문

에러를 만나도 당황하지 않는 법

김건우 선생님

커리큘럼

3 **나의 첫 테스트 코드**

코드가 바뀌어도 올바르게 동작할 수 있도록 도와 주는
테스트 코드를 작성해 봅니다.

4 **실전 디버깅!**

실전 문제를 풀어 보며,
다양한 버그를 찾아내고 안전한 코드를 설계하는 능력을 기릅니다.

목차

1. 왜 테스트 코드가 필요한가요?
2. 유닛 테스트
3. 파이썬의 `unittest`

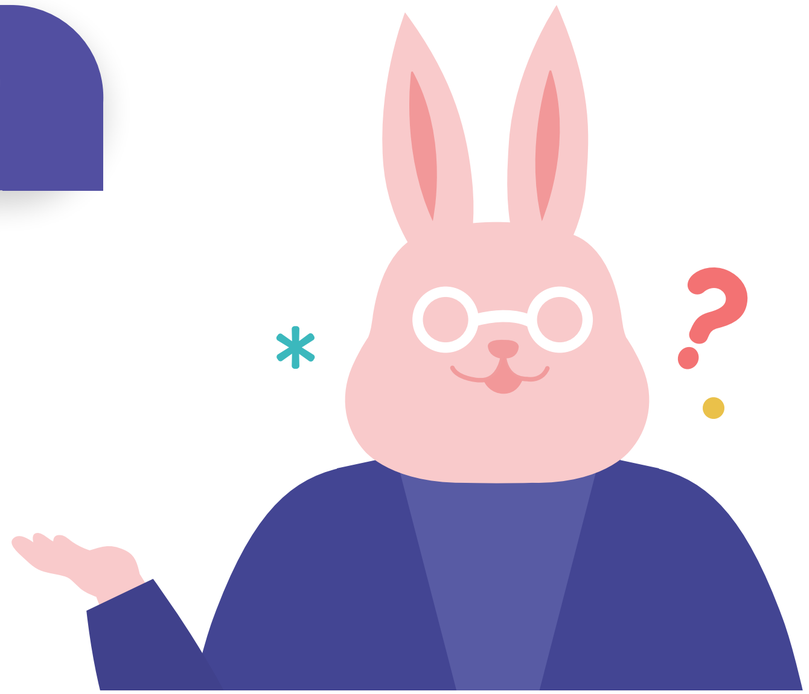
왜 테스트 코드가 필요한가요?

왜 테스트 코드가 필요할까요?

```
def average(numbers):
    return sum(numbers) / len(numbers)
```

여기서 드는 의문

맞게 짠 코드일까요?



왜 테스트 코드가 필요할까요?

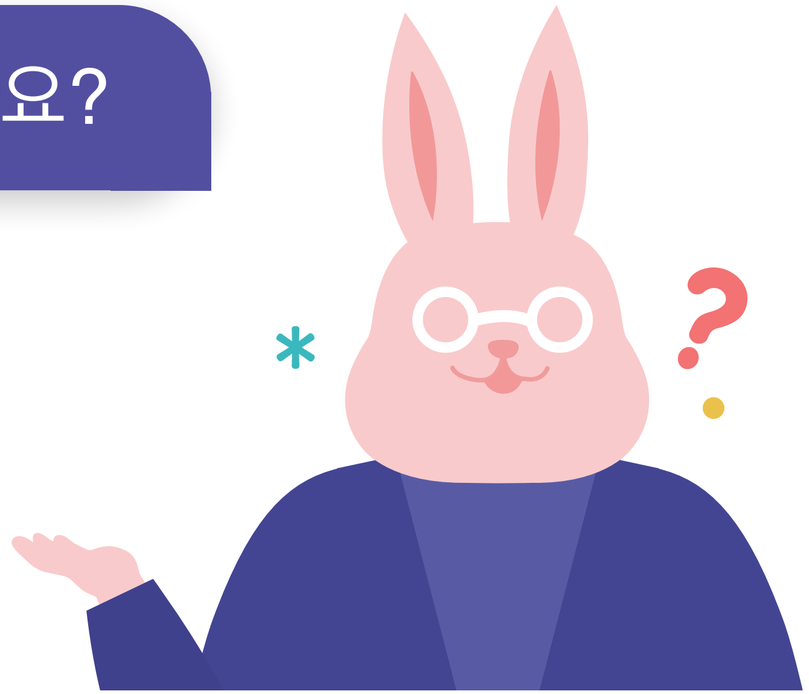
```
def average(numbers):
    if len(numbers) > 0:
        return sum(numbers) / len(numbers)
    print("No numbers!")
```

왜 테스트 코드가 필요할까요?

```
def average(numbers):
    try:
        return sum(numbers) / len(numbers)
    except ZeroDivisionError:
        print("No numbers!")
```

여기서 드는 의문

둘은 정말 같은 코드일까요?

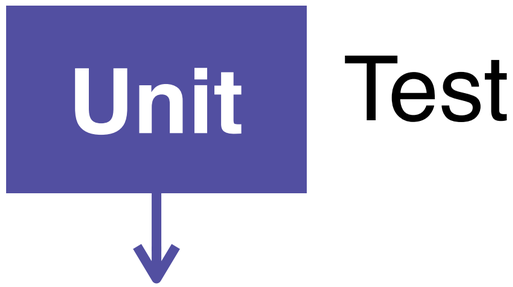


테스트 코드로 알 수 있는 것

- 코드가 **설계된 대로** 작동하는지
- 코드를 수정한 후에도 **동일하게** 작동하는지

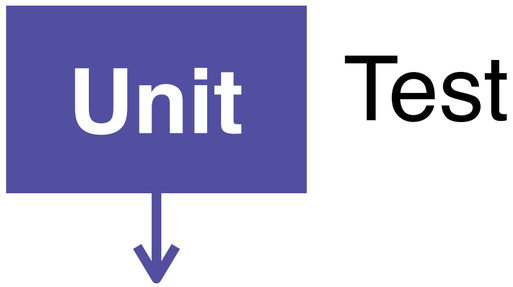
유닛 테스트

유닛 테스트



가장 작은 단위

유닛 테스트



가장 작은 단위

(= 함수 1개)

유닛 테스트

주어진 **입력**(인자)에 대해

예상된 **출력**(리턴 값)을 내놓는가?

유닛 테스트

```
def is_palindrome(text):
    if text[0] == text[-1]:
        ...
```

유닛 테스트

```
def test_is_palindrome_level():
    assert(is_palindrome("level") == True)


def test_is_palindrome_lever():
    assert(is_palindrome("lever") == False)
```

유닛 테스트

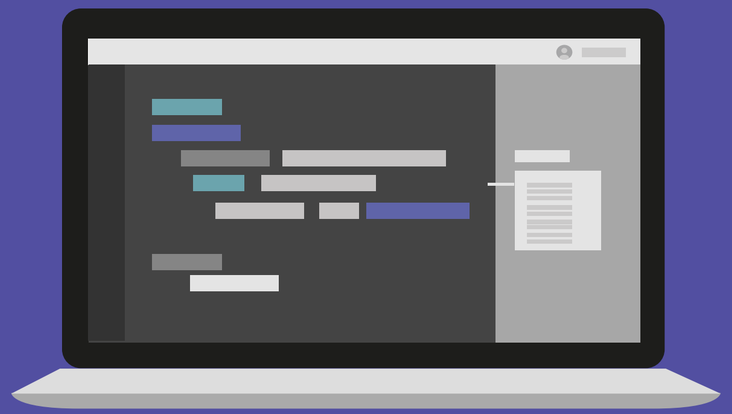
```
def test_is_palindrome_empty():
    assert(is_palindrome("")) # ??


def test_is_palindrome_sentence():
    assert(is_palindrome("Mr. Owl ate my metal
worm")) # ??
```

유닛 테스트

테스트 작성을 어떻게 작성하느냐에 따라
어떤 출력을 의도하는지가 결정됨

즉, **테스트 == 설계**

[실습 1] Palindrome 테스트



유닛 테스트의 조건

1. 읽기 쉽다
2. 독립적이다
3. 충분히 작다
4. 충분히 넓다

읽기 쉽다

```
# Bad
def test1():
    assert(is_palindrome("level") == True)
# Good
def test_is_palindrome_level():
    assert(is_palindrome("level") == True)
```

독립적이다

```
heater = Heater()
def test_control_heater_when_cold():
    heater.current_temperature = -5.0
    heater.preferred_temperature = 18.0
    control_heater(heater)
    assert(heater.is_turned_on == True)
```

독립적이다

```
...
def test_control_heater_when_hot():
    heater.current_temperature = 25.0
    heater.preferred_temperature = 18.0
    control_heater(heater)
    assert(heater.is_turned_on == False)
```

독립적이다

```
# Case 1
test_control_heater_when_cold()  # PASS
test_control_heater_when_hot()   # PASS


# Case 2
test_control_heater_when_hot()   # FAIL
test_control_heater_when_cold()  # PASS
```

⚠ 일정하지 않은 테스트 결과

독립적이다

```
def test_control_heater_when_cold():
    heater = Heater()  # 늘 새로운 히터 생성
    heater.current_temperature = -5.0
    heater.preferred_temperature = 18.0
    control_heater(heater)
    assert(heater.is_turned_on == True)
```

충분히 작다

```
# Bad
def test_heater():
    heater = Heater()
    ...
    assert(heater.is_turned_on == True)
    assert(read_temperature(heater) == "..")
```

충분히 작다

```
# Good
def test_control_heater():
    ...
    assert(heater.is_turned_on == True)
def test_read_temperature():
    ...
```

충분히 넓다

들어올 수 있는 다양한 입력들을 충분히 고려해야 한다

↓

즉, **edge case**를 고려해야 한다

충분히 넓다

```
# 설정 온도와 현재 온도가 같은 경우

def test_control_heater_temperature_same():
    heater.current_temperature = 18.0
    heater.preferred_temperature = 18.0
    control_heater(heater)
    assert(???)
```

충분히 넓다

```
# 현재 온도를 측정할 수 없는 경우
def test_control_heater_temperature_none():
    heater.current_temperature = None
    heater.preferred_temperature = 18.0
    control_heater(heater)
    assert(???)
```

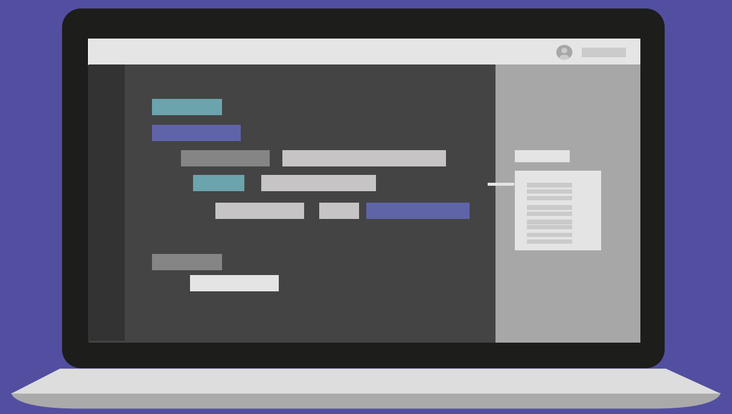
충분히 넓다

```
# 빈 문자열

def test_is_palindrome_empty():

    assert(is_palindrome("") == ???)
```

충분히 넓다

```
# 특수기호로만 이루어진 문자열

def test_is_palindrome_empty():

    assert(is_palindrome("./!@#;?") == ???)
```

[실습 2] 스마트 홈 테스트



파이썬의 `unittest`

unittest

```
import unittest


class IsPalindromeTests(unittest.TestCase):
    def test_level(self):
        self.assertTrue(is_palindrome("level"))
    def test_lever(self):
        self.assertFalse(is_palindrome("lever"))
```

unittest

```
import unittest


class IsPalindromeTests(unittest.TestCase):

    ...


unittest.main()
```

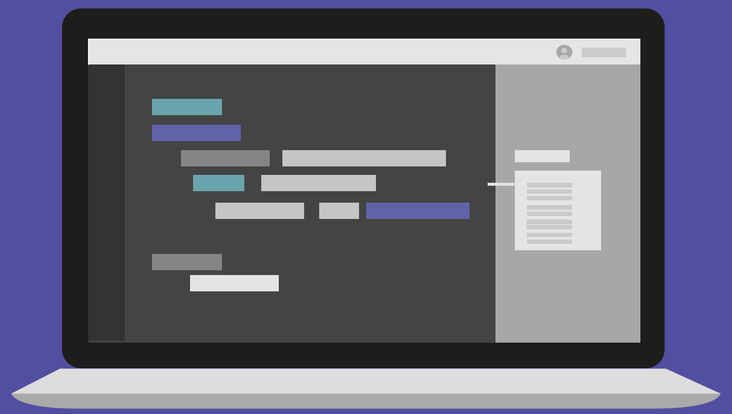
unittest

```
test_level (__main__.IsPalindromeTests) ... ok

test_lever (__main__.IsPalindromeTests) ... ok

----------------------------------------

Ran 2 tests in 0.000s

OK
```

unittest

```
test_level (__main__.IsPalindromeTests) ... FAIL

test_lever (__main__.IsPalindromeTests) ... ok

======================================================================

FAIL: test_level (__main__.IsPalindromeTests)

----------------------------------------------------------------------

Ran 2 tests in 0.000s

FAILED (failures=1)
```

[실습 3] unittest 실습



/* elice */

문의 및 연락처

academy.elice.io

contact@elice.io

facebook.com/elice.io

medium.com/elice