

201818716

컴퓨터공학부

김용현

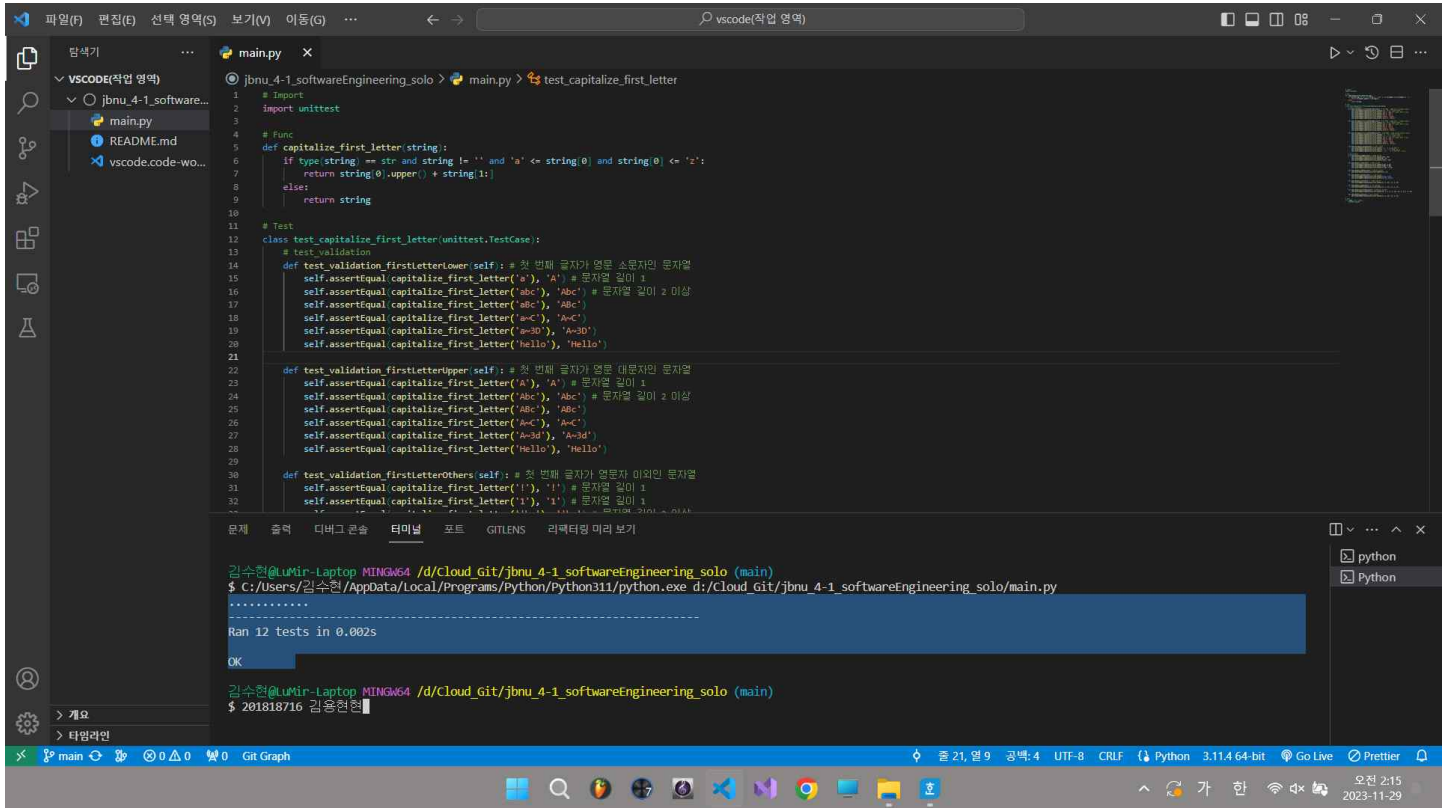
1. 코드: 작성된 코드 전체(테스트코드 포함).

```
1  # Import
2  import unittest
3
4  # Func
5  def capitalize_first_letter(string):
6      if type(string) == str and string != '' and 'a' <= string[0] and string[0] <= 'z':
7          return string[0].upper() + string[1:]
8      else:
9          return string
10
11 # Test
12 class test_capitalize_first_letter(unittest.TestCase):
13     # test_validation
14     def test_validation_firstLetterLower(self): # 첫 번째 글자가 영문 소문자인 문자열
15         self.assertEqual(capitalize_first_letter('a'), 'A') # 문자열 길이 1
16         self.assertEqual(capitalize_first_letter('abc'), 'Abc') # 문자열 길이 2 이상
17         self.assertEqual(capitalize_first_letter('aBc'), 'ABc')
18         self.assertEqual(capitalize_first_letter('a~C'), 'A~C')
19         self.assertEqual(capitalize_first_letter('a~3D'), 'A~3D')
20         self.assertEqual(capitalize_first_letter('hello'), 'Hello')
21
22     def test_validation_firstLetterUpper(self): # 첫 번째 글자가 영문 대문자인 문자열
23         self.assertEqual(capitalize_first_letter('A'), 'A') # 문자열 길이 1
24         self.assertEqual(capitalize_first_letter('Abc'), 'Abc') # 문자열 길이 2 이상
25         self.assertEqual(capitalize_first_letter('ABc'), 'ABc')
26         self.assertEqual(capitalize_first_letter('A~C'), 'A~C')
27         self.assertEqual(capitalize_first_letter('A~3d'), 'A~3d')
28         self.assertEqual(capitalize_first_letter('Hello'), 'Hello')
29
30     def test_validation_firstLetterOthers(self): # 첫 번째 글자가 영문자 이외인 문자열
31         self.assertEqual(capitalize_first_letter('!'), '!') # 문자열 길이 1
32         self.assertEqual(capitalize_first_letter('1'), '1') # 문자열 길이 1
33         self.assertEqual(capitalize_first_letter('!bc'), '!bc') # 문자열 길이 2 이상
34         self.assertEqual(capitalize_first_letter('!Bc'), '!Bc')
35         self.assertEqual(capitalize_first_letter('!~3'), '!~3')
36         self.assertEqual(capitalize_first_letter('~B3d'), '~B3d')
37         self.assertEqual(capitalize_first_letter('#ello'), '#ello')
38
39     def test_validation_blank(self): # 공백 문자열
40         self.assertEqual(capitalize_first_letter(''), '') # 공백 길이 0
41         self.assertEqual(capitalize_first_letter(' '), ' ') # 공백 길이 1
42         self.assertEqual(capitalize_first_letter('  '), '  ') # 공백 길이 2 이상
43
44     # test_defect
45     def test_defect_integer(self): # 정수 자료형
46         self.assertEqual(capitalize_first_letter(123), 123)
47         self.assertEqual(capitalize_first_letter(0), 0)
48         self.assertEqual(capitalize_first_letter(-123), -123)
49
50     def test_defect_float(self): # 실수 자료형
51         self.assertEqual(capitalize_first_letter(3.24), 3.24)
52         self.assertEqual(capitalize_first_letter(0.00), 0.00)
53         self.assertEqual(capitalize_first_letter(-3.24), -3.24)
54
55     def test_defect_complex(self): # 복소수 자료형
56         self.assertEqual(capitalize_first_letter(1j), 1j)
57
58     def test_defect_boolean(self): # 불린 자료형
59         self.assertEqual(capitalize_first_letter(True), True)
60         self.assertEqual(capitalize_first_letter(False), False)
61
62     def test_defect_list(self): # 리스트 자료형
63         self.assertEqual(capitalize_first_letter([1, 2, 3]), [1, 2, 3])
64
65     def test_defect_tuple(self): # 튜플 자료형
66         self.assertEqual(capitalize_first_letter((1, 2, 3)), (1, 2, 3))
67
68     def test_defect_dictionary(self): # 딕셔너리 자료형
69         self.assertEqual(capitalize_first_letter({'1: 1, 2: 2, 3: 3'}), {'1: 1, 2: 2, 3: 3'})
70
71     def test_defect_set(self): # 집합 자료형
72         self.assertEqual(capitalize_first_letter({1, 2, 3}), {1, 2, 3})
73
74 # Main
75 if __name__ == '__main__':
76     unittest.main()
```

2. 해당 코드를 위한 테스트 코드.

```
1  # Test
2  class test_capitalize_first_letter(unittest.TestCase):
3      # test_validation
4      def test_validation_firstLetterLower(self): # 첫 번째 글자가 영문 소문자인 문자열
5          self.assertEqual(capitalize_first_letter('a'), 'A') # 문자열 길이 1
6          self.assertEqual(capitalize_first_letter('abc'), 'Abc') # 문자열 길이 2 이상
7          self.assertEqual(capitalize_first_letter('aBc'), 'ABc')
8          self.assertEqual(capitalize_first_letter('a~C'), 'A~C')
9          self.assertEqual(capitalize_first_letter('a~3D'), 'A~3D')
10         self.assertEqual(capitalize_first_letter('hello'), 'Hello')
11
12     def test_validation_firstLetterUpper(self): # 첫 번째 글자가 영문 대문자인 문자열
13         self.assertEqual(capitalize_first_letter('A'), 'A') # 문자열 길이 1
14         self.assertEqual(capitalize_first_letter('Abc'), 'Abc') # 문자열 길이 2 이상
15         self.assertEqual(capitalize_first_letter('ABc'), 'ABc')
16         self.assertEqual(capitalize_first_letter('A~C'), 'A~C')
17         self.assertEqual(capitalize_first_letter('A~3d'), 'A~3d')
18         self.assertEqual(capitalize_first_letter('Hello'), 'Hello')
19
20     def test_validation_firstLetterOthers(self): # 첫 번째 글자가 영문자 이외인 문자열
21         self.assertEqual(capitalize_first_letter('!'), '!') # 문자열 길이 1
22         self.assertEqual(capitalize_first_letter('1'), '1') # 문자열 길이 1
23         self.assertEqual(capitalize_first_letter('!bc'), '!bc') # 문자열 길이 2 이상
24         self.assertEqual(capitalize_first_letter('1Bc'), '1Bc')
25         self.assertEqual(capitalize_first_letter('!~3'), '!~3')
26         self.assertEqual(capitalize_first_letter('~B3d'), '~B3d')
27         self.assertEqual(capitalize_first_letter('#ello'), '#ello')
28
29     def test_validation_blank(self): # 공백 문자열
30         self.assertEqual(capitalize_first_letter(''), '') # 공백 길이 0
31         self.assertEqual(capitalize_first_letter(' '), ' ') # 공백 길이 1
32         self.assertEqual(capitalize_first_letter(' '), ' ') # 공백 길이 2 이상
33
34     # test_defect
35     def test_defect_integer(self): # 정수 자료형
36         self.assertEqual(capitalize_first_letter(123), 123)
37         self.assertEqual(capitalize_first_letter(0), 0)
38         self.assertEqual(capitalize_first_letter(-123), -123)
39
40     def test_defect_float(self): # 실수 자료형
41         self.assertEqual(capitalize_first_letter(3.24), 3.24)
42         self.assertEqual(capitalize_first_letter(0.00), 0.00)
43         self.assertEqual(capitalize_first_letter(-3.24), -3.24)
44
45     def test_defect_complex(self): # 복소수 자료형
46         self.assertEqual(capitalize_first_letter(1j), 1j)
47
48     def test_defect_boolean(self): # 불린 자료형
49         self.assertEqual(capitalize_first_letter(True), True)
50         self.assertEqual(capitalize_first_letter(False), False)
51
52     def test_defect_list(self): # 리스트 자료형
53         self.assertEqual(capitalize_first_letter([1, 2, 3]), [1, 2, 3])
54
55     def test_defect_tuple(self): # 튜플 자료형
56         self.assertEqual(capitalize_first_letter((1, 2, 3)), (1, 2, 3))
57
58     def test_defect_dictionary(self): # 딕셔너리 자료형
59         self.assertEqual(capitalize_first_letter({1: 1, 2: 2, 3: 3}), {1: 1, 2: 2, 3: 3})
60
61     def test_defect_set(self): # 집합 자료형
62         self.assertEqual(capitalize_first_letter({1, 2, 3}), {1, 2, 3})
```


3. 테스트 결과 및 설명.



[작성한 함수를 테스트 하기 위해 만든 테스트 코드와 테스트 결과를 간략하게 설명하세요.] & [설계한 테스트 케이스를 왜 만든 것인지 설명하고 이를 정당화 하세요.]

파이썬 3.11.4 64-bit을 통해 개발되었습니다.

테스트 코드는 크게 2종류로 partitioning 하였습니다.

첫 번째는, validation testing 진행을 위한 테스트 코드들. 두 번째는, defect testing 진행을 위한 테스트 코드들입니다.

validation testing과 관련된 테스트 코드들은 총 4개로 partitioning 하였습니다.

defect testing과 관련된 테스트 코드들은 총 8개로 partitioning 하였습니다.

테스트는 모두 정상적으로 실행됩니다. (OK, 스크린샷 참조)

[validation testing]

1. test_validation_firstLetterLower(self)는 첫 번째 글자가 영문 소문자인 문자열을 테스트하기 위한 것으로, 문자열의 첫 번째 글자가 영문 대문자로 바뀌면 정상 실행입니다. 각각 문자열 길이가 1일 때와 2이상일 때를 구분하여 테스트 하였습니다. 또한, 첫 문자를 제외한 나머지 문자열에 영문 소·대문자, 특수문자, 숫자 등을 혼용한 경우도 테스트를 진행하였습니다. 이 테스트는, 문자열의 첫 문자를 기준으로 test case를 partitioning 하였는데, 첫 문자가 영문 소문자일 경우를 테스트하기 위해 추가하였습니다. 해당 기능은 명세(specification)에 주어진 조건이므로, validation testing이라 할 수 있습니다. 또한, 앞서 설명한 다양한 범주의 테스트 케이스의 추가를 통해 함수의 정상 작동 여부를 다방면에서 확인합니다.

2. test_validation_firstLetterUpper(self)는 첫 번째 글자가 영문 대문자인 문자열을 테스트하기 위한 것으로, 이미 조건을 만족하였으므로 입력을 그대로 되돌려 주면 정상 실행입니다. 각각 문자열 길이가 1일 때와 2이상일 때를 구분하여 테스트 하였습니다. 또한, 첫 문자를 제외한 나머지 문자열에 영문 소·대문자, 특수문자, 숫자 등을 혼용한 경우도 테스트를 진행하였습니다. 이 테스트는, 문자열의 첫 문자를 기준으로 test case를 partitioning 하였는데, 첫 문자가 영문 대문자일 경우를 테스트하기 위해 추가하였습니다. 해당 기능은 명세(specification)에 주어진 조건이므로, validation testing이라 할 수 있습니다. 또한, 앞서 설명한 다양한 범주의 테스트 케이스의 추가를 통해 함수의 정상 작동 여부를 다방면에서 확인합니다.

3. `def test_validation_firstLetterOthers(self)`는 첫 번째 글자가 영문자 이외인 문자열을 테스트하기 위한 것으로, 별도의 처리를 하지 않고 그대로 되돌려 주면 정상 실행입니다. 각각 문자열 길이가 1일 때와 2이상일 때를 구분하여 테스트 하였습니다. 또한, 첫 문자를 제외한 나머지 문자열에 영문 소·대문자, 특수문자, 숫자 등을 혼용한 경우도 테스트를 진행하였습니다. 이 테스트는, 문자열의 첫 문자를 기준으로 test case를 partitioning 하였는데, 첫 문자가 영문자 이외의 문자열인 경우를 테스트하기 위해 추가하였습니다. 해당 기능은 명세(specification)에 주어진 조건이므로, validation testing이라 할 수 있습니다. 또한, 앞서 설명한 다양한 범주의 테스트 케이스의 추가를 통해 함수의 정상 작동 여부를 다방면에서 확인합니다.

4. `def test_validation_blank(self)`는 공백 문자열을 테스트하기 위한 것으로, empty string이 입력될 경우, 그대로 empty string이 출력되면 정상 실행입니다. 공백문자 길이가 0일 때, 1일 때, 2이상일 때를 모두 테스트 하였습니다. 이 테스트는, 문자열의 첫 문자를 기준으로 test case를 partitioning 하였는데, 문자열 전체가 empty string 경우를 테스트하기 위해 추가하였습니다. 해당 기능은 명세(specification)에 주어진 조건이므로, validation testing이라 할 수 있습니다. 또한, 앞서 설명한 다양한 범주의 테스트 케이스의 추가를 통해 함수의 정상 작동 여부를 다방면에서 확인합니다.

[defect testing]

각각 `capitalize_first_letter()` 함수의 파라미터에 문자열이 아닌, 정수, 실수, 복소수, 불린, 리스트, 튜플, 딕셔너리, 집합 자료형이 왔을 때의 예외 처리를 확인하기 위해 테스트 코드를 추가하였습니다. (정수·실수의 경우 양수·0·음수, 불린의 경우 true·false 등으로 추가적으로 partitioning 하였습니다.) 문자열이 아닌 자료형이 파라미터로 전달될 경우, 파라미터로 전달된 값이 그대로 출력되게 설계하였습니다. 즉, 입력값과 출력값이 동일한 경우가 정상 실행입니다. 이 테스트는, 사용자가 문자열 이외의 값을 파라미터로 넘겼을 경우, 오작동을 방지하고, 해당 값을 그대로 출력해주기 위해 만들었습니다. 해당 기능은 정상적이지 않은 동작을 확인하기 위해 추가한 테스트이므로, defect testing이라 할 수 있습니다. 또한, 앞서 설명한 다양한 범주의 테스트 케이스를 통해 함수의 오작동 여부를 다방면에서 확인합니다.