TDD Lab: Test-Driven Development and Coverage

Maryam , Nahid

Task 1: Rövarspråket

Screenshot of Test Coverage HTML Report

File ▲	statements	missing	excluded	branches	partial	coverage
Task1_Rover.py	25	0	0	14	0	100%
Total	**25**	**0**	**0**	**14**	**0**	**100%**

Figure 1: Task 1 Coverage Report

What Types of Coverage Are Measured?

The coverage report measures:

- **Statement Coverage**: Measures whether each executable line of code was executed by the tests.
- **Branch Coverage**: Measures whether all decision paths (true and false outcomes of conditional statements) were executed.

What Was the Code Coverage Percentage Achieved?

The implementation file `Task1_Rover.py` achieved:

- **100% Statement Coverage**
- **100% Branch Coverage**

This means all executable lines and all logical paths in the implementation were exercised by the unit tests.

Describe the Bug(s) Found in the Code

The following bugs were identified:

- The lowercase consonant list was missing the letter `g`, so `"g"` was not encoded correctly.
- The uppercase consonant list was missing the letter `D`, so `"D"` was not encoded correctly.
- In the decode function, uppercase decoding used `"O"` instead of lowercase `"o"`, which caused incorrect pattern matching for encoded uppercase consonants (e.g., `"BoB"` was not decoded).

Which Test Case(s) Revealed the Bug(s)?

The following unit tests revealed the defects (they initially failed):

```
def test_enrove_g_should_encode(self):
    self.assertEqual(self.rv.enrove("g"), "gog")

def test_enrove_uppercase_D(self):
    self.assertEqual(self.rv.enrove("D"), "DoD")

def test_derove_uppercase_consonant_B(self):
    self.assertEqual(self.rv.derove("BoB"), "B")
```

How Did You Fix the Bug(s)?

- Added the missing letters `g` and `D` to the lowercase and uppercase consonant lists in `Task1_Rover.py`.
- Corrected the decoding pattern by replacing uppercase `"O"` with lowercase `"o"` in the decode logic.
- After these changes, all tests passed and the coverage reached 100% for the implementation file.

Task 2: Pattern Cipher

Equivalence Class Table

The following table maps each test case to its corresponding equivalence class:

Test Case	Equivalence Class
None	Null Input
""	Empty String
"123"	Non-alphabetic Input
"!@"	Special Characters
"level"	Palindrome (Lowercase)
"Noon"	Palindrome (Case-Insensitive)
"hello"	Repeated Letters
"cat"	Unique Letters
"Python"	Unique Letters with Case Preservation
"a"	Single Letter Boundary Case
"ab"	Two Unique Letters
"aa"	Two Same Letters (Palindrome + Repeated Conflict)

Screenshot of Test Coverage HTML Report

File ▲	statements	missing	excluded	branches	partial	coverage
cipher.py	11	0	0	8	0	100%
Total	11	0	0	8	0	100%

Figure 2: Task 2 Coverage Report

Types of Test Coverage Measured

The following coverage metrics were measured:

- **Statement Coverage**: Ensures every executable line in the code was executed.
- **Branch Coverage**: Ensures all decision paths (true and false outcomes of conditional statements) were executed.

Code Coverage Percentage Achieved

The file `cipher.py` achieved:

- 100% Statement Coverage
- 100% Branch Coverage

This indicates that all logical paths and all executable lines were fully tested.