

Test-Driven Development Lab

André Plancha

andre.plancha@hotmail.com

Yana Zlatanova

yana-zlatanova-gold@gmail.com

MALMÖ UNIVERTITY

February 21, 2025

Answer the following questions for Task 1:

- 1. Please run the test coverage module, generate the HTML report, and attach the screenshot of test coverage (see the lecture for reference)**

Coverage for **TDD_lab_student\Task1_Rover.py**: 100%

24 statements

24 run

0 missing

0 excluded

0 partial

« prev ^ index » next coverage.py v7.6.12, created at 2025-02-20 21:44 +0100

```
1 class rovar(object):
2     _LOWER_CONSTANTS = "bcdfghjklmnpqrstvwxyz"
3     _UPPER_CONSTANTS = "BCFGHJKLMNPQRSTUVWXYZ"
4
5     def enrove(self, normal: str) -> str:
6         """Encode the string in rovarspraket.
7         Args:
8             normal (str): Normal string
9         Returns:
10            (str) Encoded String
11         """
12
13         if normal is None:
14             return None
15
16         builder = ""
17         for c in normal:
18             if c in self._LOWER_CONSTANTS:
19                 builder += c + "o" + c
20             elif c in self._UPPER_CONSTANTS:
21                 builder += c + "O" + c
22             else:
23                 builder += c
24
25         return builder
26
27     def derove(self, rov: str) -> str:
28         """
```

- 2. What types of coverage were measured?**

Line Coverage: The percentage of code lines executed.

- 3. What is the code coverage you achieved with the test cases?**

The coverage is 100 %.

4. Please state and describe the defect found in the code. (There is a bug!).

The defect in the code is an inconsistency in the CONSTANTS reference strings where 'g' is missing from *LOWER_CONSTANTS* and 'D' is missing from *UPPER_CONSTANTS*, causing these letters to be treated not as CONSTANTS but as vowels (or other character) and remain unchanged during encoding/decoding which is incorrect.

5. Which test case (s) found the defect (copy and paste the test case to answer the question)?

The defect was found in the test case for encoding and decoding of all CONSTANTS in lower and upper case in a string.

```
# Test encoding
def test_consonants(self):
    test_cases = {}
    # Lowercase
    "b": "bob",
    "c": "coc",
    "d": "dod",
    "f": "fof",
    "g": "gog",
    "h": "hoh",

    "Z": "Z0Z",
    # Multiple consonants
    "br": "bobror",
    "str": "sostotrnr",
    # Mixed case consonants
    "B": "BOB",
    "Kr": "KOKror"
    }

    for input_str, expected in test_cases.items():
        encoded = self.rv.enrove(input_str)
        self.assertEqual(encoded, expected,
            f"Encoding failed for '{input_str}': got '{encoded}', expected '{expected}'")
    # Test decoded
    self.assertEqual(self.rv.derve(encoded), input_str)
```

6. How did you fix the defect(s) found in the code?

The bug was mitigated by adding "g" and "D" to the string with reference constants.

Answer the following questions for Task 2:

1. Please run the test coverage module, generate the HTML report, and attached the screenshot of test coverage (see the lecture for reference)

Coverage for **TDD_lab_student\Task2.py**: 99%

63 statements **63 run** **0 missing** 0 excluded 1 partial

« prev ^ index » next coverage.py v7.6.12, created at 2025-02-21 10:03 +0100

```
1 import string
2 import re
3 from collections import Counter
4
5
6 class TextProcessor:
7     def __init__(self, text: str):
8         # text isn't expecting any other type
9         self.text = text
10
11     # 1
12     def convert_to_lowercase(self) -> str:
13         """Convert all words to lowercase."""
14         return self.text.lower()
15
16     # 2
17     def extract_emails(self) -> list[str]:
18         """Find and extract all email addresses from the document."""
19         # https://emailregex.com/
20         return re.findall(r"[\w_.-]+@[ \w-]+\.\w{2,}", self.text)
21
22     # 3
23     def count_hashtags(self) -> int:
24         """Find and count all unique hashtags (words or phrases starting with #) used in the document."""
25         l = re.findall(r"#[\w]+", self.text.lower())
26         return len(set(l))
27
28     # 4
29     def extract_links(self) -> list[str]:
30         """Identify and list all URLs mentioned in the text."""
31         # https://mathiasbynens.be/demo/url-regex
32         # https://urlregex.com/
```

2. What types of coverage were measured?

Line Coverage: The percentage of code lines executed

3. How many test cases were written to test the user stories mentioned in task 2?

A total of 16 tests we're written, some with a few subtests. Most of them we're one test (with subtests) per story.

4. What is the code coverage you achieved with the test cases?

We achieved 100% code coverage.