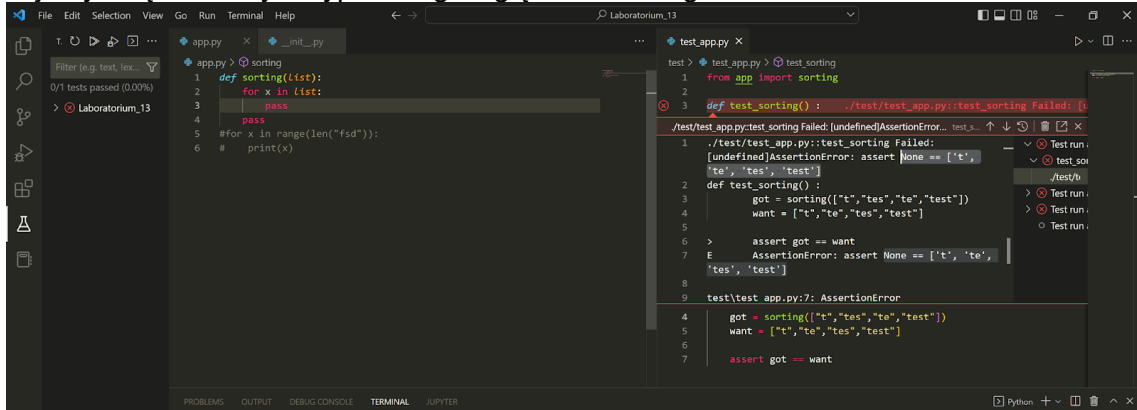


Sprawozdanie z Laboratorium nr 13 z AiBD

Tomasz Jamro Grupa nr 1 (czw. 8:30)

Faza Red

Podczas tego etapu stworzony został test do sprawdzenia funkcjonalności. Do przetestowania jest metoda sortowania bąbelkowego, z małym utrudnieniem. Zamiast liczb sortujemy listę zmiennych typu string względem ich długości.



The screenshot shows the VS Code interface with two files open: `app.py` and `test_app.py`. The `app.py` file contains a simple `sorting` function that currently does nothing. The `test_app.py` file contains a `test_sorting` function that calls `sorting` with a list of strings and asserts that the result is sorted by length. The test is failing with an `AssertionError` because the `sorting` function is not implemented. The terminal shows the test output.

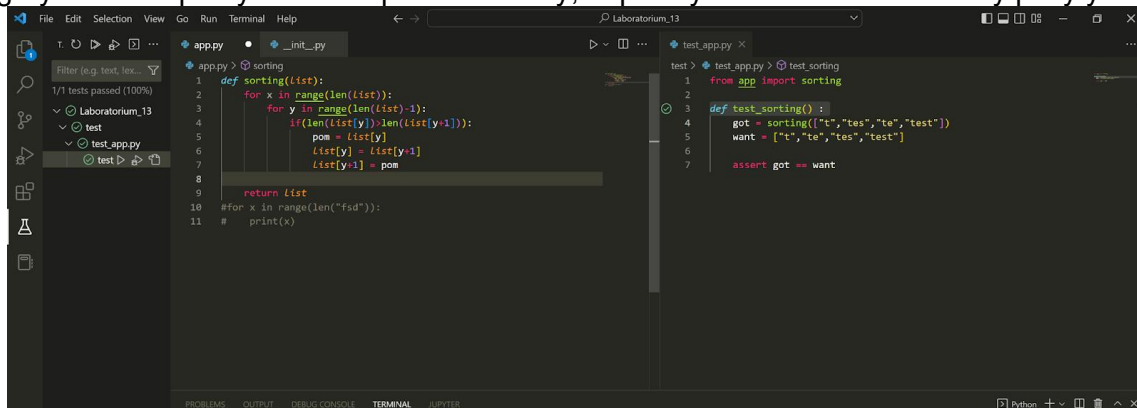
```
def sorting(list):
    for x in list:
        pass
    for x in range(len("fsd")):
        # print(x)
```

```
def test_sorting():
    got = sorting(["t", "tes", "te", "test"])
    want = ["t", "te", "tes", "test"]
    assert got == want
```

```
test_app.py:7: AssertionError
got = sorting(["t", "tes", "te", "test"])
want = ["t", "te", "tes", "test"]
assert got == want
```

Faza Green

Algorytm został pomyślnie zaimplementowany, napisany test został zakończony pozytywnie.



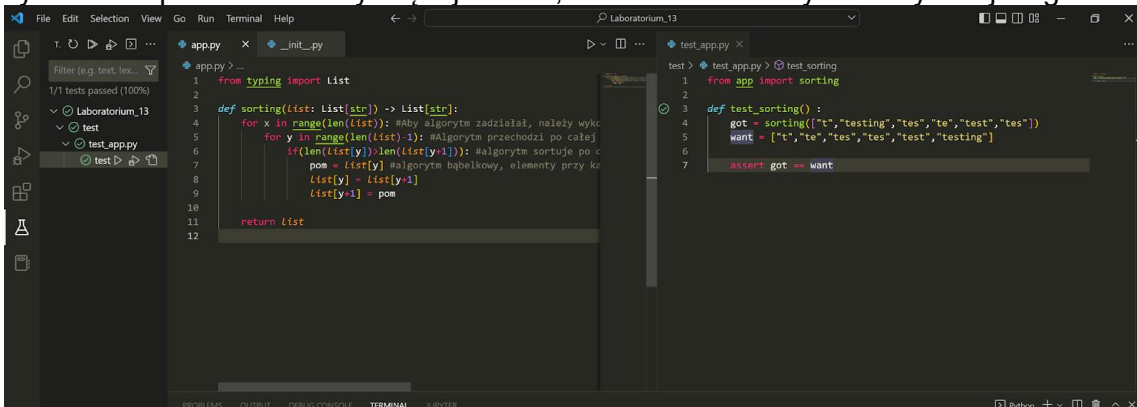
The screenshot shows the VS Code interface with the same files as before. The `app.py` file now contains a complete implementation of the bubble sort algorithm. The `test_app.py` file remains the same. The test is now passing, and the terminal shows the test output.

```
def sorting(list):
    for x in range(len(list)):
        for y in range(len(list)-1):
            if(len(list[y])>len(list[y+1])):
                pom = list[y]
                list[y] = list[y+1]
                list[y+1] = pom
        return list
    for x in range(len("fsd")):
        # print(x)
```

```
def test_sorting():
    got = sorting(["t", "tes", "te", "test"])
    want = ["t", "te", "tes", "test"]
    assert got == want
```

Faza Refactor

Aby kod był czytelny i schludny dla przyszłych użytkowników, dodano podpowiedzi typu danych. Skomentowano działanie algorytmu. Testowa lista została rozszerzona o element który zostanie przemieszczony więcej niż raz, oraz dwa elementy o identycznej długości.



The screenshot shows the VS Code interface with the same files. The `app.py` file has been refactored to include type hints and comments explaining the bubble sort algorithm. The `test_app.py` file has been updated with a more complex test list. The test is still passing.

```
from typing import List

def sorting(list: List[str]) -> List[str]:
    for x in range(len(list)):
        for y in range(len(list)-1):
            if(len(list[y])>len(list[y+1])):
                pom = list[y]
                list[y] = list[y+1]
                list[y+1] = pom
        return list
```

```
def test_sorting():
    got = sorting(["t", "testing", "tes", "te", "test", "test"])
    want = ["t", "te", "tes", "test", "testing", "test"]
    assert got == want
```