Pytest

Aleksandra Stachniak, grupa 1

Faza Red

W pierwszej kolejności należało napisać test, który wywoła algorytm sortowania bąbelkowego.

```
testdata3 = [
    ([1,4,5,2,3,6],[1,2,3,4,5,6]),
    ([9,4,2,6,2,1,3],[1,2,2,3,4,6,9])
    ]

@pytest.mark.parametrize('lst, sorted', testdata3)
def test_bubble_sort(lst, sorted):

    assert bubble_sort(lst) == sorted
```

Jak widać poniżej 4 testy dotyczące wcześniejszych funkcji zostały wykonane, natomiast dla sortowania bąbelkowego 2 testy nie wykonały się z uwagi, że funkcja ta nie została jeszcze stworzona.

```
test\test_app.py:37 (test_bubble_sort[lst0-sorted0])
lst = [1, 4, 5, 2, 3, 6], sorted = [1, 2, 3, 4, 5, 6]

    @pytest.mark.parametrize('lst, sorted', testdata3)
    def test_bubble_sort(lst, sorted):

>       assert bubble_sort(lst) == sorted
E       NameError: name 'bubble_sort' is not defined

test\test_app.py:41: NameError
FAILED                  [100%]
test\test_app.py:37 (test_bubble_sort[lst1-sorted1])
lst = [9, 4, 2, 6, 2, 1, ...], sorted = [1, 2, 2, 3, 4, 6, ...]

    @pytest.mark.parametrize('lst, sorted', testdata3)
    def test_bubble_sort(lst, sorted):

>       assert bubble_sort(lst) == sorted
E       NameError: name 'bubble_sort' is not defined

test\test_app.py:41: NameError
```

Faza Green

Następnym etapem było napisanie funkcji w app.py, tak aby testy zostały wykonane prawidłowo.

```python
def bubble_sort(lst: List[int]):
    l = len(lst)
    for i in range(l):
        change = False
        for j in range(l - i - 1):
            if lst[j] > lst[j + 1]:
                lst[j], lst[j + 1] = lst[j + 1], lst[j]
                change = True
        if not change:
            return lst
    return lst
```

Stworzona funkcja rzeczywiście pozwala na wykonanie się testów.

```
test/test_app.py::test_bubble_sort[lst0-sorted0] PASSED
test/test_app.py::test_bubble_sort[lst1-sorted1] PASSED
```

Faza Refactor

Jak widać wszystkie testy przechodzą. Funkcje są na tyle proste, że nie ma potrzeby ich ulepszania. Natomiast w przypadku bardziej skomplikowanych metod, warto zastanowić się, co zrobić, aby metoda była bardziej czytelna, wydajniejsza itd.

```
============================= test session starts ==============================
collecting ... collected 6 items

test/test_app.py::test_hello PASSED                                      [ 16%]
test/test_app.py::test_extract_sentiment[I think today will be a great day] PASSED [ 33%]
test/test_app.py::test_text_contain_word[There is a duck in this text-duck-True] PASSED [ 50%]
test/test_app.py::test_text_contain_word[There is nothing here-duck-False] PASSED [ 66%]
test/test_app.py::test_bubble_sort[lst0-sorted0] PASSED                  [ 83%]
test/test_app.py::test_bubble_sort[lst1-sorted1] PASSED                  [100%]

============================== 6 passed in 0.93s ===============================
```