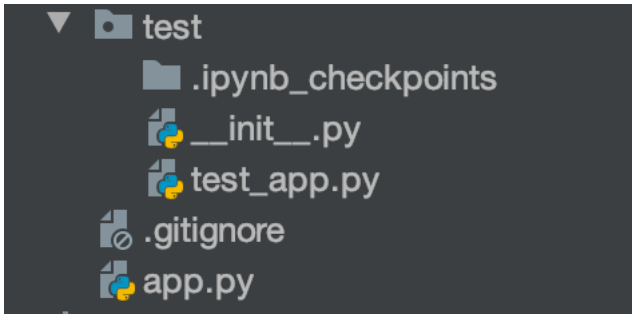
AIBD sprawozdanie laboratorium 14

Wojciech Żyła, grupa piątek 10:45

W trakcie zajęć w pierwszej kolejności utworzyłem następującą hierarchię katalogów i plików:



Rys. 1. Katalogi i pliki projektu.

W pierwszej kolejności do pliku *test_app.py* dodałem test funkcjonalności sortowania bąbelkowego. Funkcję sparametryzowałem w taki sposób, żeby testowała trzy przypadki:

- Otrzymana tablica jest posortowana
- Otrzymana tablica jest w odwrotnej kolejności
- Otrzymana tablica posiada elementy w losowej kolejności

Ostatecznie plik *test_app.py* wygląda następująco:

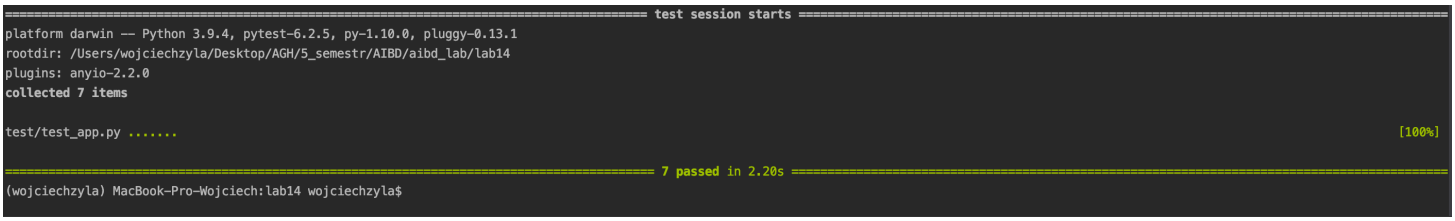
```
from app import hello
from app import extract_sentiment
from app import text_contain_word
from app import bubble_sort
import pytest


def test_hello():
    got = hello("Aleksandra")
    want = "Hello Aleksandra"

    assert got == want

testdata1 = ["I think today will be a great day"]

@pytest.mark.parametrize('sample', testdata1)
def test_extract_sentiment(sample):

    sentiment = extract_sentiment(sample)

    assert sentiment > 0
```

```
testdata2 = [
    ('There is a duck in this text', 'duck', True),
    ('There is nothing here', 'duck', False)
    ]

@pytest.mark.parametrize('sample, word, expected_output', testdata2)
def test_text_contain_word(sample, word, expected_output):

    assert text_contain_word(word, sample) == expected_output


testdata3 = [
    ([1,23,26,37,50,100,340,900],[1,23,26,37,50,100,340,900]),
    ([900,340,100,50,37,26,23,1],[1,23,26,37,50,100,340,900]),
    ([37,100,1,900,23,340,26,50],[1,23,26,37,50,100,340,900])
    ]

@pytest.mark.parametrize('sample, expected', testdata3)
def test_bubble_sort(sample, expected):

    assert bubble_sort(sample) == expected
```

Następnie w pliku *app.py* do istniejącego kodu dopisałem funkcję sortowania bąbelkowego. Całość pliku wygląda następująco:

```
from textblob import TextBlob
from typing import List

def hello(name):
    output = f'Hello {name}'
    return output


def extract_sentiment(text):
    text = TextBlob(text)

    return text.sentiment.polarity

def text_contain_word(word: str, text: str):
    return word in text

def bubble_sort(numbers: List):
    for _ in range(len(numbers)):
        changes = False
        for i in range(len(numbers)-1):
            if numbers[i]>numbers[i+1]:
                numbers[i+1], numbers[i] = numbers[i], numbers[i+1]
                changes = True
        if not changes:
            break
    return numbers
```

Wyniki działania testów przeprowadzonych z wykorzystaniem frameworku pytest



Rys. 2. Wyniki testów.

Jak widać wszystkie testy przyszły z wynikiem pozytywnym.

Wnioski

Zadanie było proste do wykonania ze względu na obszerną i wyczerpującą instrukcję do laboratorium. Dotychczas potrafiłem pisać testy we frameworku unittest lecz w trakcie tych zajęć poszerzyłem swoją wiedzę o znajomość kolejnego frameworka. Dowiedziałem się również dlaczego pytest jest lepszym framewrokiem w porównaniu do unittesta.