

TDD		
Wydział <i>EAIiB</i>	Kierunek <i>Automatyka i Robotyka</i>	Rok <i>III</i>
Grupa 1, czwartek 8:30		Data <i>16 stycznia 2023 r.</i>
Zrealizował: Marcin Ryznar		

FAZA RED

Służy do wykonywania zawsze na początku. Test nie jest w stanie wykonać się poprawnie, ponieważ funkcje jeszcze nie zostały w pełni zadeklarowane.

```

test > test_app.py > ...
2 from app import extract_sentiment
3 from app import text_contain_word
4 import pytest
5
6
7 def test_hello():
8     got = hello("Marcin")
9     want = "Hello Marcin"
10
11     assert got == want
12
13

```

```

platform win32 -- Python 3.11.1, pytest-7.2.1, pluggy-1.0.0
rootdir: C:\Marcin\STUDIA\Automatyka i Robotyka\Semestr 5\Analiza i bazy danych\L13. Marcin Ryznar
plugins: anyio-3.6.2
collected 1 item

test\test_app.py F

===== FAILURES =====
test_hello

>
> def test_hello():
>     got = hello("Marcin")
E       NameError: name 'hello' is not defined

test\test_app.py:8: NameError
FAILED test\test_app.py::test_hello - NameError: name 'hello' is not defined
===== short test summary info =====
1 failed in 0.11s
PS C:\Marcin\STUDIA\Automatyka i Robotyka\Semestr 5\Analiza i bazy danych\L13. Marcin Ryznar>

```

```

test > test_app.py > ...
2 from app import extract_sentiment
3 from app import text_contain_word
4 from app import hello
5 import pytest
6
7
8
9
10 def test_hello():
11     got = hello("Marcin")
12     want = "Hello Marcin"
13
14     assert got == want
15

```

```

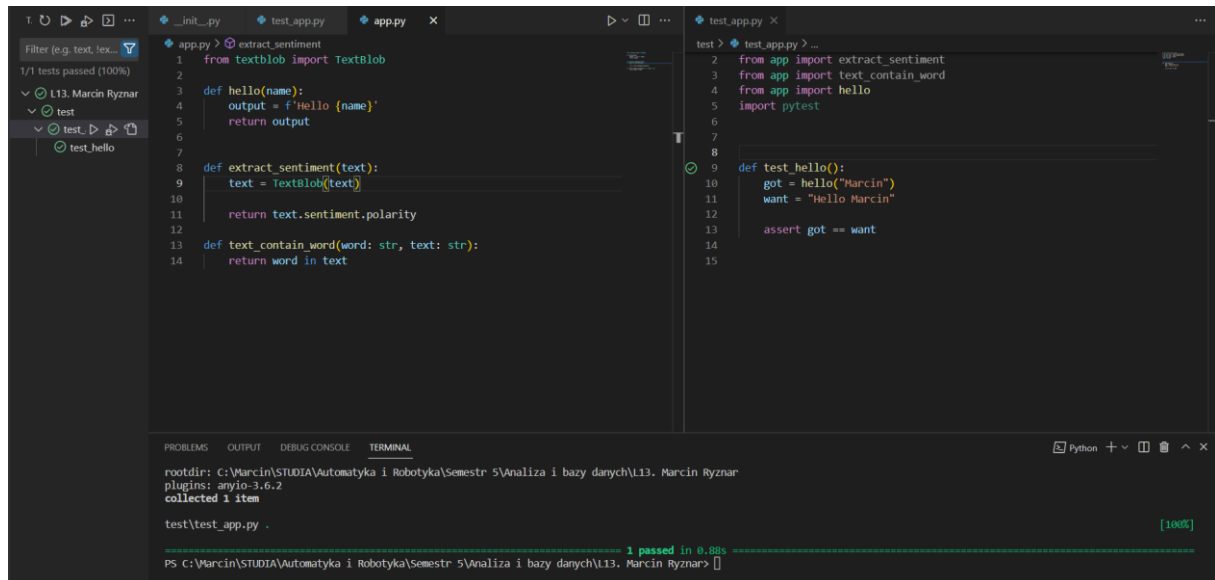
rootdir: C:\Marcin\STUDIA\Automatyka i Robotyka\Semestr 5\Analiza i bazy danych\L13. Marcin Ryznar
plugins: anyio-3.6.2
collected 0 items / 1 error

===== ERRORS =====
ERROR collecting test\test_app.py
ImportError while importing test module 'C:\Marcin\STUDIA\Automatyka i Robotyka\Semestr 5\Analiza i bazy danych\L13. Marcin Ryznar\test\test_app.py'.
Hint: make sure your test modules/packages have valid python names.
Traceback:
C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.11_3.11.496.0_x64-gb25n2kfra8p0\Lib\importlib\__init__.py:126: in import_module
    return _bootstrap._gcd_import(name[level:], package, level)
test\test_app.py:2: in <module>
    from app import extract_sentiment
E   ImportError: cannot import name 'extract_sentiment' from 'app' (C:\Marcin\STUDIA\Automatyka i Robotyka\Semestr 5\Analiza i bazy danych\L13. Marcin Ryznar\app.py)
===== short test summary info =====
1 error during collection
1 error in 1.36s
PS C:\Marcin\STUDIA\Automatyka i Robotyka\Semestr 5\Analiza i bazy danych\L13. Marcin Ryznar>

```

FAZA GREEN

W tej fazie brakujące funkcje zostają zaimplementowane. Nie dodajemy jednak wszystkich funkcji do naszego programu, robimy tylko tak aby zniwelować błędy. Wtedy mamy pewność o poprawności działania.



The screenshot shows a code editor with three files: `_init_.py`, `test_app.py`, and `app.py`. The `app.py` file contains the following code:

```
1 from textblob import TextBlob
2
3 def hello(name):
4     output = f'Hello {name}'
5     return output
6
7
8 def extract_sentiment(text):
9     text = TextBlob(text)
10
11     return text.sentiment.polarity
12
13 def text_contain_word(word: str, text: str):
14     return word in text
```

The `test_app.py` file contains the following code:

```
1
2 from app import extract_sentiment
3 from app import text_contain_word
4 from app import hello
5 import pytest
6
7
8
9 def test_hello():
10     got = hello("Marcin")
11     want = "Hello Marcin"
12
13     assert got == want
14
15
```

The left sidebar shows a test runner with the following output:

```
Filter (e.g. text, test...)
1/1 tests passed (100%)
✓ L13. Marcin Ryznar
✓ test
✓ test_
  ✓ test_hello
```

The bottom panel shows the terminal output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
rootdir: C:\Marcin\STUDIA\Automatyka i Robotyka\Semestr 5\Analiza i bazy danych\L13. Marcin Ryznar
plugins: anyio-3.6.2
collected 1 item

test\test_app.py . [100%]

===== 1 passed in 0.08s =====
PS c:\Marcin\STUDIA\Automatyka i Robotyka\Semestr 5\Analiza i bazy danych\L13. Marcin Ryznar>
```

FAZA REFACTOR

Jest ona ostatnią fazą i służy ona do rozbudowania naszego kodu. Zalety polegają na lepszej przejrzystości oraz stopnia skomplikowania pisanego programu. Nasz zaimplementowany poprzez swoją prostotę nie potrzebuje stosowania powyżej opisanej fazy.

ZADANIE

1. Implementacja funkcji oraz podmienienie wyrażenia

`assert sentiment > 0`

na

`assert sentiment < 0`

Jak widać poniżej testy się nie wykonują. Wykonałem tą zmianę w celach zobaczenia poniższych oczekiwanych rezultatów.

```
1 from textblob import TextBlob
2
3 def hello(name):
4     output = f'Hello {name}'
5     return output
6
7
8 def extract_sentiment(text):
9     text = TextBlob(text)
10
11     return text.sentiment.polarity
12
13 def text_contain_word(word: str, text: str):
14     return word in text
15
```

```
17 def test_extract_sentiment():
18     text = "I think today will be a great day"
19     sentiment = extract_sentiment(text)
20     assert sentiment < 0
21
```

```
test/test_app.py:23: AssertionError
24
25 text = "I think today will be a great day"
26 sentiment = extract_sentiment(text)
27
```

```
===== short test summary info =====
FAILED test/test_app.py::test_extract_sentiment - assert 0.8 < 0
1 failed, 1 passed in 1.14s
```

2. Testowanie wiele możliwych danych wejściowych i wyjściowych.

```
1 from textblob import TextBlob
2
3 def hello(name):
4     output = f'Hello {name}'
5     return output
6
7
8 def extract_sentiment(text):
9     text = TextBlob(text)
10
11     return text.sentiment.polarity
12
13 def text_contain_word(word: str, text: str):
14     return word in text
15
```

```
17 def test_text_contain_word():
18     testdata1 = ["I think today will be a great day"]
19     @pytest.mark.parametrize('sample', testdata1)
20     def test_extract_sentiment(sample):
21         sentiment = extract_sentiment(sample)
22         assert sentiment > 0
23
24     testdata2 = [
25         ('There is a duck in this text', 'duck', True),
26         ('There is nothing here', 'duck', False)
27     ]
28
29     @pytest.mark.parametrize('sample, word, expected_output', testdata2)
30     def test_text_contain_word(sample, word, expected_output):
31         assert text_contain_word(word, sample) == expected_output
32
```

```
===== test session starts =====
platform win32 -- Python 3.11.1, pytest-7.2.1, pluggy-1.0.0
rootdir: C:\Marcin\STUDIA\Automatyka i Robotyka\Semestr 5\Analiza i bazy danych\L13. Marcin Ryznar
plugins: anyio-3.6.2
collected 4 items

test/test_app.py .... [100%]

4 passed in 0.09s
```

Można zauważyć, że funkcja przechodzi testy oraz zostaje wykonana dwa razy, zgodnie z ilością danych we/wy wcześniej podanych.

ZADANIE DO SAMODZIELNEGO WYKONANIA

Implementacja algorytmu quick sort.

FAZA RED

The screenshot shows the PyCharm IDE interface. The left sidebar displays a test runner with a list of tests: `test_text_contain_word`, `test_hello`, `test_quick_sort`, and `test_text_contain_word`. The `test_quick_sort` test is highlighted in red, indicating a failure. The main editor shows the code for `test_app.py`, which includes a `test_quick_sort` function. The terminal window at the bottom displays the following output:

```
plugins: anyio-3.6.2
collected 4 items

test/test_app.py ....

===== 4 passed in 105.48s (0:01:45) =====

PS C:\Marcin\STUDIA\Automatyka i Robotyka\Semestr 5\Analiza i bazy danych\L13. Marcin Ryznar> & C:/Users/marci/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Marcin/STUDIA/Automatyka i Robotyka/Semestr 5/Analiza i bazy danych/L13. Marcin Ryznar/app.py"
PS C:\Marcin\STUDIA\Automatyka i Robotyka\Semestr 5\Analiza i bazy danych\L13. Marcin Ryznar>
```

FAZA GREEN

The screenshot shows the PyCharm IDE interface. The left sidebar displays a test runner with a list of tests: `test_text_contain_word`, `test_hello`, `test_quick_sort`, and `test_text_contain_word`. All tests are highlighted in green, indicating success. The main editor shows the code for `test_app.py`, which includes a `test_quick_sort` function. The terminal window at the bottom displays the following output:

```
===== test session starts =====
platform win32 -- Python 3.11.1, pytest-7.2.1, pluggy-1.0.0
rootdir: C:\Marcin\STUDIA\Automatyka i Robotyka\Semestr 5\Analiza i bazy danych\L13. Marcin Ryznar
plugins: anyio-3.6.2
collected 5 items

test/test_app.py ....

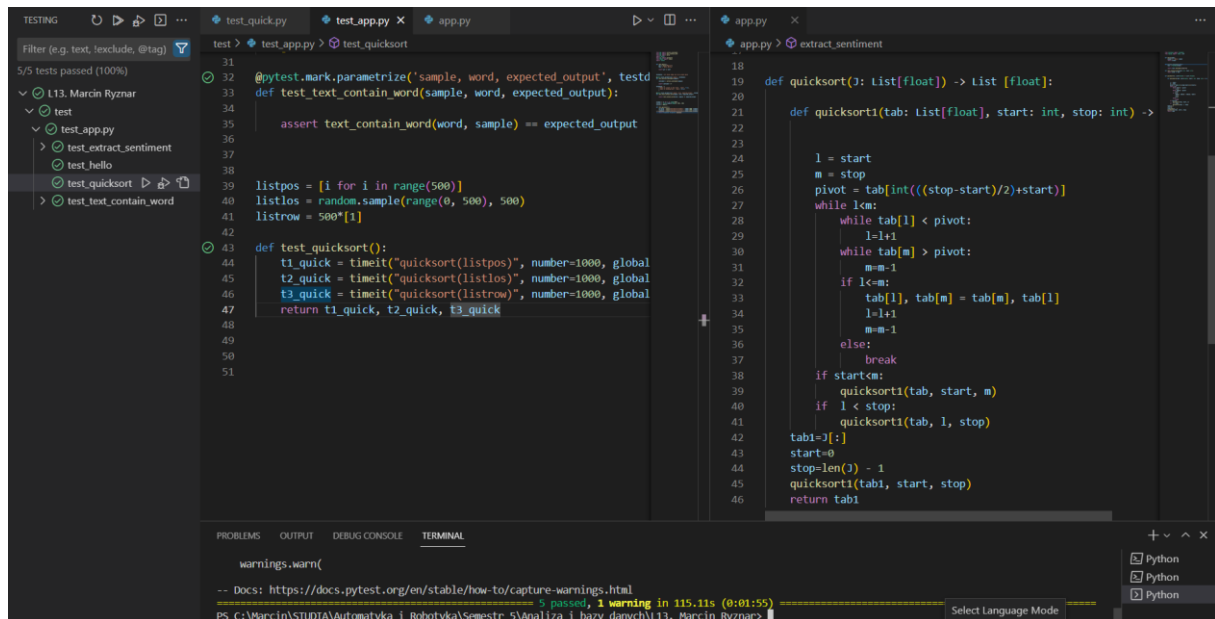
===== warnings summary =====
test/test_app.py::test_quick_sort
  C:\Users\marci\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\pytest\python.py:199: PytestReturnNotNoneWarning: Expected None, but test/test_app.py::test_quick_sort returned 0.00013859790000014983, which will be an error in a future version of pytest. Did you mean to use "assert" instead of "return"?
    warnings.warn(

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
===== 5 passed, 1 warning in 140.58s (0:02:20) =====

PS C:\Marcin\STUDIA\Automatyka i Robotyka\Semestr 5\Analiza i bazy danych\L13. Marcin Ryznar>
```

FAZA REFACTOR

Dodanie dodatkowych możliwości odnoszących się do sprawdzenia czasu szybkiego sortowania.



The screenshot shows an IDE with three main panels. The left panel displays test results for a test suite named 'L13. Marcin Ryznar'. It shows 5/5 tests passed (100%). The tests listed are 'test', 'test_app.py', 'test_extract_sentiment', 'test_hello', 'test_quicksort', and 'test_text_contain_word'. The 'test_quicksort' test is currently selected. The middle panel shows the code for 'test_app.py' with the 'test_quicksort' function. The function uses 'pytest.mark.parametrize' to test 'quicksort' with two different inputs: 'listpos' and 'listlos'. It uses 'timeit' to measure the execution time of 'quicksort' for each input. The right panel shows the code for 'app.py' with the 'quicksort' function. The function is a recursive implementation of quicksort. The bottom panel shows the terminal output, which includes a warning message: 'warnings.warn()' and a link to the pytest documentation: 'https://docs.pytest.org/en/stable/how-to/capture-warnings.html'. The terminal also shows the test results: '5 passed, 1 warning in 115.11s (0:01:55)'.

```
31
32 @pytest.mark.parametrize('sample, word, expected_output', testd
33 def test_text_contain_word(sample, word, expected_output):
34
35     assert text_contain_word(word, sample) == expected_output
36
37
38
39 listpos = [i for i in range(500)]
40 listlos = random.sample(range(0, 500), 500)
41 listrow = 500*[1]
42
43 def test_quicksort():
44     t1_quick = timeit("quicksort(listpos)", number=1000, global
45     t2_quick = timeit("quicksort(listlos)", number=1000, global
46     t3_quick = timeit("quicksort(listrow)", number=1000, global
47     return t1_quick, t2_quick, t3_quick
48
49
50
51
```

```
18
19 def quicksort(j: List[float]) -> List [Float]:
20
21     def quicksort1(tab: List[float], start: int, stop: int) ->
22
23         l = start
24         m = stop
25         pivot = tab[int((stop-start)/2)+start])
26         while l<m:
27             while tab[l] < pivot:
28                 l=l+1
29             while tab[m] > pivot:
30                 m=m-1
31             if l<m:
32                 tab[l], tab[m] = tab[m], tab[l]
33                 l=l+1
34                 m=m-1
35             else:
36                 break
37         if start<m:
38             quicksort1(tab, start, m)
39         if l < stop:
40             quicksort1(tab, l, stop)
41     tab1=tab[:]
42     start=0
43     stop=len(tab) - 1
44     quicksort1(tab1, start, stop)
45     return tab1
46
```

```
warnings.warn(
-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
5 passed, 1 warning in 115.11s (0:01:55)
PS C:\Marcin\STUDIA\Automatka i Robotyka\Semestr 5\Analiza i bazy danych\L13. Marcin Ryznar>
```