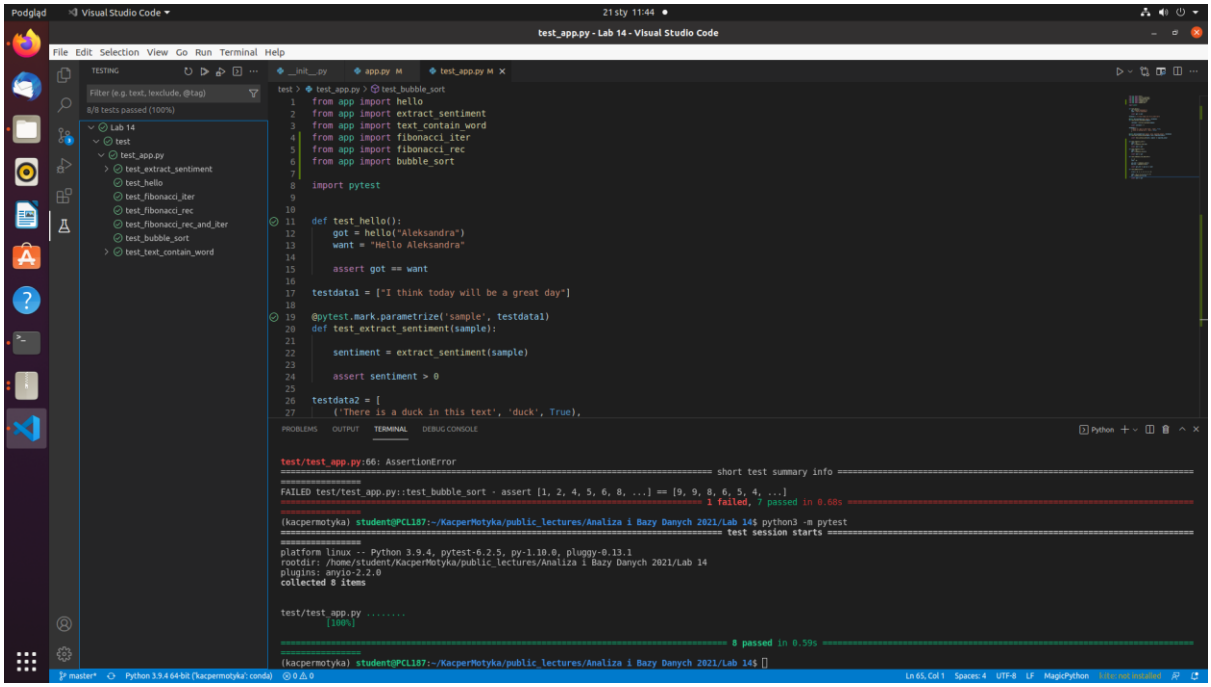
Analiza i Bazy Danych

Kacper Motyka

Sprawozdanie z laboratorium 14

PyTest

1. Analiza przykładowych zadań



Rys. 1. Urządzenie do testowania

Wszystkie testy przygotowane przez prowadzącą zadziałały poprawnie. Aby to było możliwe, należało zainstalować pakiet służący do przeprowadzania testów jednostkowych (PyTest) a także pakiet TextBlob, który służy do analizy tekstu (na przykład pod kątem nacechowania emocjonalnego). Następnie należało włączyć testy, jednak nie można przy tym korzystać z normalnego przycisku run w prawym górnym rogu wklejonego powyżej zrzutu ekranu a należy wpisać w terminal komendę **python3 -m pytest** bądź skorzystać z narzędzia do przeprowadzania testów, które dostarcza Visual Studio Code. W tym celu należy kliknąć ikonę Testing (fiolka chemiczna po lewej stronie) a następnie zamiast domyślnego modułu **unittest**

wybrać moduł **Pytest**. Jest to narzędzie dużo wygodniejsze niż zwykły terminal ponieważ wygodniej można podglądać wyniki poszczególnych testów, są one wylistowane jeden po drugim więc możemy się upewnić, że wszystkie testy zostały przeprowadzone a także możemy za jego pomocą przeprowadzać pojedyncze testy. Jest to szczególnie przydatne, gdy mamy łącznie na przykład 100 testów a chcemy przeprowadzić jeden test. Dzięki temu zamiast czekać na przykład minutę, testy mogą zakończyć się w sekundę dzięki czemu zaoszczędzimy trochę czasu. Dodatkowo warto wspomnieć, że istnieje możliwość przeprowadzania danych grup testów, które dotyczą jednej funkcjonalności.

W zadaniach przygotowanych przez prowadzącą testowaliśmy działanie systemu sprawdzania nacechowania emocjonalnego zdania, obecności słowa w zdaniu a także funkcji, która powinna przywitać użytkownika gdy poda on swoje imię. Wszystkie funkcje działały poprawnie co potwierdza Rysunek 1.

```python
def fibonacci_rec(n: int):
    if n == 0:
        return 0

    if n == 1:
        return 1

    if n > 1:
        return fibonacci_rec(n-1) + fibonacci_rec(n-2)


def fibonacci_iter(n: int):
    if n == 0:
        return [0]

    if n == 1:
        return [0, 1]

    result = [0, 1]

    for i in range(2, n+1):
        result.append(result[i-2] + result[i-1])

    return result[-1]

def bubble_sort(array: List):

    n = len(array)

    for i in range(n-1):
        for j in range(n-i-1):

            if array[j] > array[j+1]:
                array[j], array[j+1] = array[j+1], array[j]

    return array
```

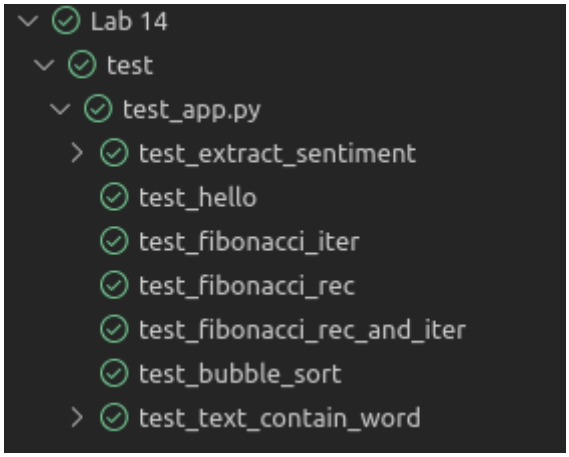
Rys. 2. Funkcje przygotowane przeze mnie

Zdecydowałem się przetestować podejście TDD dla funkcji Fibonacciego oraz Bubble Sort. Funkcję Fibonacciego napisałem rekurencyjnie a także iteracyjnie. Sprawdziłem działanie obu funkcji z osobna, następnie sprawdziłem czy dla takich samych danych wejściowych zwracają one dokładnie takie same wyniki a następnie przygotowałem test funkcji bubble sort.

Następnie zaimplementowałem powyższe funkcje i sprawdziłem poprawność działania za pomocą testów. Na początku jeden z testów Fibonacci nie działał, ponieważ zwracałem w niej listę elementów zamiast elementu ostatniego. Po małych poprawkach wszystkie funkcje działały a testy przeszły poprawnie.

```python
def test_fibonacci_iter():
    want = 21
    got = fibonacci_iter(n=8)

    assert want == got

def test_fibonacci_rec():
    want = 13
    got = fibonacci_rec(n=7)

    assert want == got

def test_fibonacci_rec_and_iter():

    n = 9
    want = 34

    got_iter = fibonacci_iter(n)
    got_rec = fibonacci_rec(n)

    assert got_iter == got_rec == want

def test_bubble_sort():

    array = [6, 2, 1, 8, 9, 9, 5, 4]

    want = [1, 2, 4, 5, 6, 8, 9, 9]
    got = bubble_sort(array)

    assert want == got
```

Rys. 3. Testy mojego autorstwa



Rys. 4. Wyniki testów

Wnioski:

- PyTest to bardzo użyteczne narzędzie, ponieważ pozwala na sprawne testowanie oprogramowania (dużo lepsze podejście niż wstawianie wielu funkcji print() i sprawdzanie gdzie jest błąd)
- Podejście TDD na pierwszą myśl wydaje się dziwne, jednak często robimy tak, że na początku implementacji funkcji pod nią piszemy print(result) i sprawdzamy czy nasza funkcja działa poprawnie - TDD jest tylko bardziej sformalizowanym podejściem.