**Analiza i bazy danych**

**Laboratorium 14: Test Driven Development**

**Artur Mzyk, 400658, gr. 2b**

1. **Wstęp**

Test Driven Development to podejście do tworzenia oprogramowania, którego myślą przewodnią jest to, że przed napisaniem właściwej funkcjonalności tworzy się test, który ma sprawdzać jej działanie.

Cały proces pisania kodu składa się z cykli, które powtarza się jeden po drugim.

Każdy cykl składa się z 3 faz:

* Faza Red:

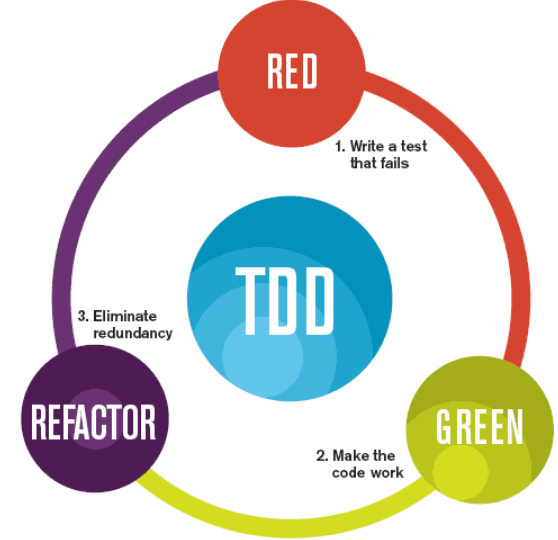
Na samym początku należy napisać test. Nie jest on w stanie wykonać się z sukcesem, gdyż testowana funkcjonalność nie została jeszcze zaimplementowana.

* Faza Green:

Następnym krokiem jest implementacja właściwej funkcjonalności. Wystarczy tylko tyle, aby test przeszedł.

* Faza Refactor:

Ostatnia faza cyklu to refaktoryzacja, czyli ulepszanie istniejącego kodu, aby był bardziej czytelny i mniej złożony.



Rysunek 1. Cykl TDD

1. **Framework**

Najpopularniejszym frameworkiem do tworzenia testów w Pythonie jest Pytest. Umożliwia korzystanie z wbudowanych funkcji do porównywania czy korzystanie z pluginów.

Obraz zawierający tekst

Opis wygenerowany automatycznie

Rysunek 2. Potwierdzenie instalacji Pytest

1. **Struktura katalogów**

Dobrą praktyką jest zdefiniowanie struktury katalogów już na samym początku.

Obraz zawierający tekst

Opis wygenerowany automatycznie

Rysunek 3. Zastosowana struktura katalogów

Kod tworzony będzie w katalogu *myapp* w module *app.py*. Testy będą tworzone w katalogu *test* w module *test\_app.py*. Nazewnictwo w przypadku testów jest ważne.

1. **Pierwsza funkcja**

Celem jest napisanie funkcji, która jako argument przyjmuje imię i zwraca przywitanie składające się ze słowa „Hello” i tegoż imienia.

* 1. **Faza Red**



Rysunek 4. Pierwszy test

Stworzono i wykonano test. Naturalnie pojawił się komunikat o błędzie, gdyż funkcja nie została jeszcze zaimplementowana.

Obraz zawierający tekst, zrzut ekranu, monitor, ekran

Opis wygenerowany automatycznie

Rysunek 5. Niepomyślne wykonanie pierwszego testu

* 1. **Faza Green**

Zaimplementowano funkcję *hello()*.



Rysunek 6. Implementacja pierwszej funkcji

Funkcja pomyślnie przeszła test.

Obraz zawierający tekst, zrzut ekranu, monitor, ekran

Opis wygenerowany automatycznie

Rysunek 7. Pomyślne wykonanie pierwszego testu

* 1. **Faza Refactor**

Funkcja jest na tyle prosta, że nie ma potrzeby ulepszania.

1. **Druga funkcja**

Sentyment to wydźwięk emocjonalny wypowiedzi. Analiza sentymentu to mechanizm analizy tekstów online, gdzie oceniany jest ton emocjonalny.

W tym celu można wykorzystać bibliotekę TextBlob, która służy do przetwarzania danych tekstowych.

Celem jest napisanie funkcji, która zwróci wartość większą od 0, jeżeli zdanie ma wydźwięk pozytywny, i mniejszą od 0, jeżeli zdanie ma wydźwięk negatywny.

* 1. **Faza Red**

Stworzono i wykonano test. Naturalnie pojawia się komunikat o błędzie, gdyż funkcja nie została jeszcze zaimplementowana. Wykorzystano dekorator *pytest.mark.parametrize()*, który pozwala na przetestowanie wielu danych wejściowych.

Obraz zawierający tekst

Opis wygenerowany automatycznie

Rysunek 8. Drugi test

Obraz zawierający tekst

Opis wygenerowany automatycznie

Rysunek 9. Niepomyślne wykonanie drugiego testu

Ostrzeżenie pojawia się ze względu na bibliotekę TextBlob.

* 1. **Faza Green**

Zaimplementowano funkcję *extract\_sentiment()*.

**

Rysunek 10. Implementacja drugiej funkcji

Funkcja dla pierwszych danych wejściowych przeszła test pomyślnie, dla drugich już nie – zgodnie z założeniem.

Obraz zawierający tekst

Opis wygenerowany automatycznie

Rysunek 11. Pomyślne i niepomyślne wykonanie drugiego testu

* 1. **Faza Refactor**

Funkcja jest na tyle prosta, że nie ma potrzeby ulepszania.

1. **Trzecia funkcja**

Celem jest napisanie funkcji, która zwraca informację, czy zdanie zawiera wybrany wyraz.

* 1. **Faza Red**

Obraz zawierający tekst

Opis wygenerowany automatycznie

Rysunek 12. Trzeci test

Stworzono i wykonano test. Naturalnie pojawia się komunikat o błędzie, gdyż funkcja nie została jeszcze zaimplementowana. Wykorzystano dekorator *pytest.mark.parametrize()*, który pozwala na przetestowanie wielu danych wejściowych.

Obraz zawierający tekst

Opis wygenerowany automatycznie

Rysunek 13. Niepomyślne wykonanie trzeciego testu

* 1. **Faza Green**

Zaimplementowano funkcję *text\_contains\_word()*.



Rysunek 14. Implementacja trzeciej funkcji

Funkcja pomyślnie przeszła test.

Obraz zawierający tekst, zrzut ekranu, monitor, ekran

Opis wygenerowany automatycznie

Rysunek 15. Pomyślne wykonania trzeciego testu

* 1. **Faza Refactor**

Funkcja jest na tyle prosta, że nie ma potrzeby ulepszania.

1. **Czwarta funkcja**

Celem jest napisanie funkcji realizującej sortowanie bąbelkowe.

* 1. **Faza Red**

Obraz zawierający tekst

Opis wygenerowany automatycznie

Rysunek 16. Czwarty test

Stworzono i wykonano test. Naturalnie pojawia się komunikat o błędzie, gdyż funkcja nie została jeszcze zaimplementowana. Wykorzystano dekorator *pytest.mark.parametrize()*, który pozwala na przetestowanie wielu danych wejściowych.

Obraz zawierający tekst

Opis wygenerowany automatycznie

Rysunek 17. Niepomyślne wykonanie czwartego testu

* 1. **Faza Green**

Zaimplementowano funkcję *bubble\_sort()*.

Obraz zawierający tekst

Opis wygenerowany automatycznie

Rysunek 18. Implementacja czwartej funkcji

Funkcja pomyślnie przeszła test.

Obraz zawierający tekst, zrzut ekranu, monitor, wewnątrz

Opis wygenerowany automatycznie

Rysunek 19. Pomyślne wykonania czwartego testu

* 1. **Faza Refactor**

Funkcja jest na tyle prosta, że nie ma potrzeby ulepszania.

1. **Podsumowanie**

Implementowane funkcje były na tyle proste, że wystarczający był 1 cykl.  
W ogólnym przypadku może ich być znacznie więcej.

Pytest jest bardzo przydatnym frameworkiem, który pozwala na dogłębne i przejrzyste przetestowanie implementowanych funkcjonalności.

Test Driven Development to na pierwszy rzut oka nietypowe podejście, gdyż zaczyna się od napisania testu, a dopiero potem implementuje się właściwą funkcjonalność. Jednak taka zasada pozwala na uporządkowanie programu  
i zapewnienie, że na każdym etapie działa on poprawnie.