Python. Instrukcja Assert. Podstawy testowania aplikacji.

Pytest jest biblioteką, która rozszerza funkcjonalności unittest. Pozwala również na tworzenie testów w nieco prostszej strukturze oraz daje możliwość używania wbudowanego wyrażenia assert do sprawdzenia poprawności testów.

Aby pracować z tą kartą pracy zainstaluj w terminalu PyCharm moduł pytest za pomocą tego polecenia: **pip install pytest**

Dokumentacja : https://docs.pytest.org/en/stable/

Asercja to formuła logiczna używana do kontrolowania czy np. wartość zmiennej czy ma odpowiedni typ lub mieści się w pożądanym zakresie.

Przykłady Asercji

```
x = 0

assert x > 0, 'Tylko wartosci dodatnie'

print('x jest dodatnie.')
```

Przykład Asercji z użyciem funkcji utwórz w pliku o nazwie test_example.py

```
def test_sum():
    assert sum([1, 2, 3]) == 6, "Should be 6"

def test_sum_tuple():
    assert sum((1, 2, 2)) == 6, "Should be 6"

def test_add_two(x):
    return x + 2
```

Uruchom w terminalu komendę pytest.

Zwróć uwagę, że nasz plik posiada nazwę zaczynającą się od “test“. Zabieg ten pozwala rozpoznać pytestowi, że w tym pliku znajdują się testy. Podobna zasada dotyczy nazw funkcji.

Pytest uruchamiając testy wywoła te funkcje, których nazwy zaczynają się od “test_”. Tak przygotowane testy możemy uruchomić, będąc w katalogu z plikiem komendą: **pytest**

Ogólna konwencja proponuje, żeby testy przechowywać w osobnych plikach w katalogu tests w naszym projekcie. Wtedy wywołanie pytesta w terminalu będzie wyglądało następująco **pytest tests**

Kolejny przykład

Utwórz plik o nazwie test_secondary.py w folderze tests w swoim projekcie.

```
import pytest

def div_two_by(x):
    return 2/x
```

```
def test_div_two_by_zero():
    with pytest.raises(ZeroDivisionError):
        div_two_by(0)
def test_add_two_and_two():
    assert add_two(3) == 5
def test_add_two_and_two_fail():
    assert add_two(3) == 3
```

Lista zadań.

1. Napisz program, który będzie odwracał kolejność liter w wyrazach w celu uzyskania palindromu. Dodaj warunek w którym sprawdzamy, czy wczytany z klawiatury string jest palindromem. Przetestuj program za pomocą pytest i asercji.
2. Napisz program, który będzie kalkulatorem BMR - Podstawowa Przemiana Materii (PPM). Wykorzystaj wzór wzór Harrisa i Benedicta oraz przetestuj program za pomocą pytest i asercji.