SOFTWARE TESTING

BEFORE CLASS

1. Zapoznaj się z materiałem dotyczącym testowania oprogramowania „Software testing tutorial for beginner”:
2. https://youtu.be/goaZTAzsLMk
3. Zapoznaj się z materiałem dotyczącym testów jednostkowych „Unit tests in Python”:

   https://youtu.be/1Lfv5tUGsn8
4. Zapoznaj się z materiałem dotyczącym testowania oprogramowania w języku Python „Getting Started With Testing in Python”:

   https://realpython.com/python-testing/
5. Zapoznaj się z materiałem dotyczącym testów jednostkowych w języku Python:

   https://docs.python.org/3/library/unittest.html

DURING CLASS

The __name__ variable

6. Uruchom program greeting.py, wyświetlający pozdrowienia. Zawiera on funkcję say_hi(). Spróbuj teraz wykorzystać tę funkcję w innym programie. Uruchom program greetings_test.py, który używa funkcji say_hi(). Czy dostrzegasz problem, jakie się pojawił?
7. W grupach 2-3 osobowych, odszukaj w Internecie informacje na temat specjalnej zmiennej __name__. Zastanów się, jaką wartość przyjmuje ta specjalna zmienna w zależności od tego, czy plik .py jest uruchamiany, czy importowany.
8. Dokonaj analizy zawartości programu main_greetings.py. Jakie widzisz różnice w kodzie źródłowym programu, w porównaniu z programem greetings.py. Następnie uruchom program main_greetings.py. Czy dostrzegasz jakieś różnice w działaniu tych dwóch programów?
9. Dokonaj analizy programu main_greetings_test.py. Następnie uruchom ten program. W grupie 2-3 osobowej zastanów się, dlaczego w tym przypadku nie jest wyświetlany test „I’m a student”.

Unit tests

Test jednostkowy (ang. unit test) to metoda testowania tworzonego oprogramowania poprzez wykonywanie testów weryfikujących poprawność działania pojedynczych elementów (jednostek) programu, np. metod lub obiektów w programowaniu obiektowym lub procedur w programowaniu proceduralnym. Testowany fragment programu poddawany jest testowi, który wykonuje go i porównuje wynik (np. zwrócone wartości, stan obiektu, zgłoszone wyjątki) z oczekiwanymi wynikami – tak pozytywnymi, jak i negatywnymi (niepowodzenie działania kodu w określonych sytuacjach również może podlegać testowaniu).

10. Program sum_numbers.py zawiera funkcję sum_even(), która sumuje wszystkie liczby naturalne parzyste z przedziału <m,n>. Zapoznaj się z kodem źródłowym programu. Czy rozumiesz wszystkie użyte instrukcje? Następnie uruchom program i sprawdź jego działanie.

11. Twoim zadaniem jest teraz utworzenie testów jednostkowych pozwalających na gruntowne sprawdzenie poprawności działania funkcji sum_even(). Program test_sum_numbers.py zawiera przykład trzech testów:

    a. Test sumy liczb naturalnych parzystych z dowolnego przedziału <m,n>
    b. Test sumy liczb naturalnych parzystych z przedziału <m,n>, gdzie m parzyste
    c. Test sumy liczb naturalnych parzystych z przedziału <m,n>, gdzie n<m

    Dokonaj analizy kodu źródłowego programu. Następnie uruchom program i sprawdź poprawność testów.

12. Dodaj kolejny test sprawdzający, czy funkcja sum_even() wyznacza poprawną sumę liczb naturalnych parzystych z przedziału <m,n>, gdy n jest parzyste. Uruchom testy. Czy wszystkie zakończyły się pomyślnie? Przeanalizuj wyświetlane komunikaty odszukując w nich informacje o błędach.

    Następnie wprowadź zmiany w funkcji sum_even(), aby wyznaczała ona poprawną sumę liczb naturalnych parzystych z przedziału <m,n>, gdy n jest parzyste. Uruchom ponownie testy, aby sprawdzić poprawność działania funkcji.

AFTER CLASS

13. Dodaj kolejny test sprawdzający działanie funkcji sum_even() dla obliczania sumy liczb naturalnych parzystych z przedziału <m,n>, gdy m<0. Następnie uruchom testy. Czy funkcja zwraca poprawne wartości? Jeśli nie, wprowadź stosowne zmiany w funkcji sum_even() i ponownie uruchom testy.

    Dodaj kolejny test sprawdzający działanie funkcji sum_even(). Zweryfikuj działanie funkcji, gdy wartości m i/lub n są liczbami rzeczywistymi. Odpowiedź na pytanie, w jaki sposób skonstruować test sprawdzający typ danych znajdziesz materiale filmowym pod adresem: https://youtu.be/1Lfv5tUGsn8

14. Dodaj kolejny test sprawdzający działanie funkcji sum_even(). Zweryfikuj działanie funkcji, gdy wartości m i/lub n są łańcuchami znakowymi, np. m=’jeden’, a m=’osiem’.

15. Plik programs.pdf zawiera przykładowe zadania. Każde z nich polega na utworzeniu oddzielnego programu. Utwórz te programy. Następnie, dla każdego z nich utwórz przynajmniej dwa testy jednostkowe. Uruchom testy, aby sprawdzić poprawność utworzonych programów.