7 stycznia 2025

Kurs rozszerzony języka Python

Lista 12.

Poniższe polecenia dotyczą zadań realizowanych w ramach list 11–12.

(A) Korzystając z dedykowanych środowisk `pyunit` czy `nose`, przygotuj zestawy testów jednostkowych sprawdzających poprawność wcześniej zaprogramowanych zadań. Wymagania:

- testy znajdują się w odrębnym katalogu, poza katalogami kodu źródłowego serwera czy interfejsu użytkownika;
- testy są połączone w zestaw testowy tak, aby można było jednym poleceniem uruchomić wszystkie testy;
- przygotuj co najmniej dwie klasy z przypadkami testowymi, przynajmniej po trzy testy w każdej klasie.

(B) Poszukaj informacji o **PEP 8** (Python Enhancement Proposals). Za pomocą pakietu `pep8 checker` czy `pycodestyle`[1] sprawdź zgodność kodu źródłowego zadań (i implementacji testów) z zaleceniami PEP 8. Wysyłając pliki na SKOS umieść pliki z poprawionym formatowaniem kodu źródłowego. W pliku `README.md` zamieść informację, z jakiego narzędzia korzystałeś. Można też zamiennie umieścić plik `Makefile` z poleceniem sprawdzenia kodu.

(C) Poszukaj informacji o automatycznym generowaniu dokumentacji na podstawie kodu źródłowego oraz zawartych w nim komentarzy. Wygeneruj taką dokumentację dla obydwu zadań w jakimś popularnym formacie (`html`, `pdf`, etc). Zamiast umieszczać dokumentację na SKOS'ie, wystarczy zamieścić plik z odpowiednimi poleceniami, np. `Makefile` czy `README.md`.

Do wykonania konieczne jest uzupełnienie komentarzy; wymagane jest skomentowanie wszystkich klas i funkcji (wraz z parametrami). Funkcje lokalne czy nieistotne metody nie muszą być komentowane.

(D) Uzupełnij kod źródłowy o *adnotacje typowe* i sprawdź jego poprawność. Podobnie jak w poprzednich punktach, zamieść informacje o użytym analizatorze w pliku `Makefile` czy `README.md`.

Zadanie jest warte 4 pkt.

Marcin Młotkowski

[1] zgodność z PEP 8 sprawdzają też wtyczki w środowisku VS Code czy serwisy online.