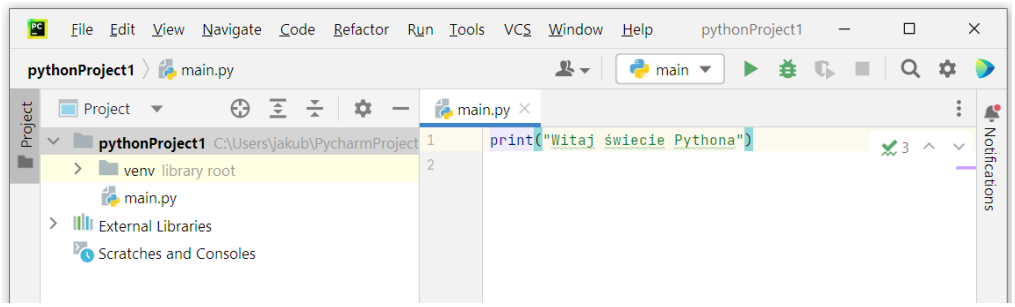
Wstęp do programowania



Lista 6

opracował dr inż. Jakub Długosz

Tematyka:

Klasy i obiekty.
Dziedziczenie. Mechanizm name mangling.
Parsowanie, serializacja, deserializacja.
Zmienne i metody klasy oraz statyczne.
Asercje i debugowanie. Testowanie.

Lista nie na ocenę.

Zadanie 1 – klasa `LogExp`

Napisz klasę `LogExp`, która będzie zawierać definicje funkcji: $\log_a x$ oraz a^x, gdzie `a` może być określone wyłącznie przez konstruktor.

Zadanie 2 – klasy `Prostokat` oraz `Kwadrat`

Napisz program, w którym zdefiniujesz klasę `Prostokat` oraz dziedziczącą z niej klasę `Kwadrat`. Program powinien obliczać pola obu rodzajów figur, korzystając z wcześniej podanej długości boków.

Zadanie 3 – oceny studentów

Napisz program, który pozwoli zapisać oceny studentów. Każdy student jest identyfikowany za pomocą imienia, nazwiska oraz numeru indeksu (6 cyfr). Przyjmijmy, że w danym semestrze studenci mają trzy przedmioty - dla każdego musi być możliwość wpisania, edycji i wyświetlania oceny. `Mała podpowiedź`: Lista może zawierać obiekty różnego typu, również obiekty, które zdefiniuje się w programie. Klasa reprezentująca pojedynczego studenta może zawierać słownik, w którym nazwy przedmiotów będą kluczami.

Zadanie 4 – klasa `Employee` oraz testowanie

Napisz klasę o nazwie `Employee`, która reprezentuje pracownika. Metoda `__init__()` powinna pobierać i zapisywać atrybuty: `first_name` (imię), `last_name` (nazwisko) i `annual_salary` (roczna pensja). Napisz metodę `give_raise()`, która dodaje 2000PLN podwyżki do rocznej pensji domyślnie, czyli jako default, ale również dopuszcza dodanie innej kwoty. Napisz testy dla klasy `Employee`. Za pomocą `unittest` napisz dwie metody testowe: `test_give_default_raise()` i `test_give_custom_raise()`. Użyj metody `setUp()`, by móc stworzyć tylko jedną instancję pracownika `employee` dla obu metod testujących. Uruchom testy i upewnij się, czy przechodzą pomyślnie.

Możesz również wykonać testy za pomocą `pytest`.