SZKOLENIE ZAAWANSOWANE

Python: Zaawansowane programowanie

PYTHON/ADV

Czas trwania: 5 dni

Uczestnicy szkolenia zapoznają się z zaawansowanymi konstrukcjami w języku Python, mechanizmami i praktykami programowania obiektowego oraz przykładami użycia idiomów i wzorców projektowych. Na zajęciach zapoznają się z koncepcjami tworzenia aplikacji wielowątkowych, serializacją danych do różnych formatów i obsługą relacji. Szczególny nacisk kładziemy na praktyczne aspekty programowania oraz osadzenie technik wytwarzania oprogramowania w języku Python we współczesnych realiach. Szkolenie odpowiada potrzebom średnio zaawansowanych i zaawansowanych programistów i nie jest przeznaczone dla osób początkujących lub nieprogramujących

Cele szkolenia

- Kształcenie umiejętności i rozwijanie wiedzy dotyczącej zaawansowanego programowania w języku Python
- Konsolidacja wiedzy i uzupełnienie braków w kompetencjach w zakresie szkolenia
- Pogłębienie znajomości mechanizmów i idiomów języka Python
- Nauka refaktoryzacji i pracy z debuggerem w środowisku IDE

Zalety

- Zajęcia prowadzone są przez doświadczonych praktyków, którzy na co dzień stosują prezentowane techniki i narzędzia
- Na zajęciach stosowane są otwarte rozwiązania
- Szkolenie porusza zagadnienia związane z tworzeniem i rozwijaniem aplikacji, z użyciem biblioteki standardowej oraz szkieletów aplikacyjnych (ang. framework)
- W trakcie ćwiczeń wykorzystywane są przykłady zbliżone do rzeczywistych zastosowań i promowane są praktyki tworzenia łatwego w utrzymaniu kodu

Dla kogo?

- Średnio zaawansowani i zaawansowani programiści, posługujący się językiem Python
- Architekci rozwiązań aplikacyjnych w języku Python

Wymagania

- Umiejętność programowania w języku Python oraz znajomości podstawowych struktur danych
- Znajomość koncepcji programowania obiektowego

- Umiejętność posługiwania się wybranym środowiskiem IDE, dedykowanym dla języka Python
- Ogólna znajomość biblioteki standardowej dla języka Python

Program

1. Type Annotations
    a. Typy proste
    b. Sekwencje
    c. Mapy
    d. Funkcje
    e. Obiekty i metody
2. Rozpakowywanie obiektów
    a. Unpacking Assignment
    b. Rozpakowywanie parametrów (*args, **kwargs)
    c. Rozpakowywanie argumentów (*args, **kwargs)
    d. Assignment Expression
3. Zaawansowane użycie funkcji i elementy paradygmatu programowania funkcyjnego
    a. Składnia parametrów do definiowania API
    b. Generatory
    c. Przestrzenie nazw
    d. Paradygmat programowania funkcyjnego
    e. Callable
    f. Domknięcia (closure)
    g. Moduł Functools
4. Dekoratory
    a. Rodzaje dekoratorów i przykłady zastosowania
    b. Dekoratory funkcji, klas, metod
    c. Dekoratory z wrapperami funkcyjnymi i klasowymi
    d. Dekoratory z argumentami i bez
    e. Dekoratory w bibliotece standardowej
5. Paradygmat obiektowy
    a. Mutowalne argumenty
    b. Dataclasses
    c. Pola i metody statyczne
    d. Modyfikatory dostępu
    e. Przeciążanie operatorów
    f. Dziedziczenie i kompozycja, klasy domieszkowe (mixin)
    g. Obiekty i relacje
    h. Klasy abstrakcyjne
    i. Tożsamość obiektów, haszowalność, string interning
    j. Konstruktor i fabryki obiektów
    k. S.O.L.I.D. i dobre praktyki OOP
6. Protokoły
    a. Iterator
    b. Context Manager
    c. Staticmethod
    d. Classmethod

e. Property
f. Refleksja
g. Deskryptory

7. Zagadnienia wydajnościowe i optymalizacja
   a. Profiling aplikacji
   b. Microbenchmarking
   c. Wydajność wbudowanych struktury i typów danych
   d. Alternatywne kompilatory i interpretery
8. Współbieżność
   a. Modele współbieżności
   b. Kolejki
   c. Komunikacja międzyprocesowa i międzywątkowa
   d. Mechanizmy blokujące
   e. Wprowadzenie do programowania wielowątkowego
   f. Wprowadzenie do programowania wieloprocesowego
   g. Wprowadzenie do programowania asynchronicznego
9. Dobre praktyki i jakość oprogramowania
   a. Wykorzystanie debuggera w IDE
   b. Refactoring
   c. Techniki pracy z legacy code
   d. CI/CD