

Szkolenie: Python Academy High-Performance Computation with Python



DOSTĘPNE TERMINY

2025-05-12 | 5 dni | Virtual Classroom
2025-05-12 | 5 dni | Warszawa / Wirtualna sala

Cel szkolenia:

Potrzeba skomplikowanych i wydajnych obliczeń jest dużo starsza niż przemysł komputerowy. Dzisiaj oprogramowanie jest codziennym rozwiązaniem problemów obliczeniowych - musi być ciągle od nowa projektowane lub adaptowane w niekończącej się ilości miejsc, żeby osiągnąć cele użytkowników i przeprowadzić wydajne i poprawne obliczenia. Nic tak nie może zaspokoić tych wymagań, jak język programowania, który jest łatwy do nauczenia się przez użytkowników i umożliwia im aktywne realizowanie rozwiązań swoich problemów.

Python jest właśnie tym językiem - jest łatwy do nauki, do użycia i do czytania. Składnia języka umożliwia relatywnie proste pisanie łatwego w utrzymaniu kodu, jak również użycie go jako swoistej 'formy komunikacji' z innymi ludźmi. Co więcej, jest łatwy do optymalizacji i specjalizacji kodu, celem użycia go dla wymagających, ograniczonych zasobowo obliczeń. Będzie to głównym przedmiotem tego kursu

Plan szkolenia:

- Optymalizacja programów w Pythonie
 - Wprowadzenie do optymalizacji.
 - Strategie optymalizacyjne - Benchmarking z Pystone, profilowanie CPU z cProfile, pomiar zużycia pamięci z Guppy_PE Framework. Uczestnicy są zachęceni do przyniesienia swoich własnych programów celem profilowania ich na kursie.
 - Algorytmy i anty-wzorce projektowe - przykłady algorytmów, które działają wyjątkowo wolno lub szybko w Pythonie.
 - Odpowiednie struktury danych - porównanie wbudowanych struktur danych: list, zbiorów, podwójnie zakończonych kolejek, standardowych słowników.
 - Caching - deterministyczne i niedeterministyczne spojrzenie na caching i proces tworzenia dekoratorów.
 - Przykład - znajdziemy numerycznie i obliczeniowo wymagający problem i zaimplementujemy go w czystym Pythonie. Później popatrzymy na możliwości algorytmicznej poprawy szybkości obliczeń.
 - Testowanie prędkości - znajdowanie rozwiązań do prawidłowego pomiaru czasu

- wykonania programu.
- Psyco - 'just-in-time-compiler' (JIT), pozwalający na tłumaczenie części kodu bajtowego na kod maszynowy. Przykłady celem pokazania możliwości użycia Psyco.
 - Obliczenia numeryczne z NumPy - podstawy użycia biblioteki.
 - Używanie wielu jednostek obliczeniowych z PyProcessing/Multiprocessing.
 - Kombinacja strategii optymalizacyjnych.
 - Przegląd rozszerzeń do Pythona z innych języków.
 - Rozszerzenie Pythona z innych języków
 - Wprowadzenie do przykładu, który będzie użyty w dalszej części tego modułu.
 - Użycie C-API Pythona - standardowy Python jest zaimplementowany w C i oferuje solidne API do pisania rozszerzeń.
 - Użycie plików DLL z ctypes - dostęp do bibliotek DLL lub bibliotek współdzielonych z poziomu Pythona.
 - Rozszerzenia Pythona z Pyrex/Cython.
 - Automatyczna generacja rozszerzeń ze SWIG - "Simplified Wrapper and Interface Generator", pozwalający tworzyć biblioteki C/C++ z 13 języków programowania - jednym z nich jest Python. Przykłady zarówno w C, jak i w C++.
 - Jython - podstawy implementacji Pythona w Javie. Przykłady użycia zarówno już istniejących klas Javy, jak i napisanych własnoręcznie.
 - IronPython - implementacja Pythona w .NET, pozwalająca na dostęp do wszystkich funkcjonalności .NET i stawiająca IronPythona zaraz obok C# i VisualBasica, jako pełnoprawny język platformy .NET.
 - Użycie subrutyn FORTRANa w Pythonie - przykłady użycia F2PY do połączenia FORTRAN77, FORTRAN90/95 programów z Pythonem. Projektowanie interfejsów zorientowanych obiektowo do tych bibliotek.
 - Szybki kod z kompilatorem Cython
 - Używanie pyximport do szybkiego budowania/aktualizowania modułów rozszerzeń.
 - Użycie cython.inline() do kompilacji kodu w trakcie wykonania programu.
 - Budowanie rozszerzeń z distutils.
 - Szybki dostęp do typów wbudowanych w Pythonie.
 - Szybkie iterowanie po typach Pythonowych i C.
 - Przetwarzanie łańcuchów napisów.
 - Szybka arytmetyka.
 - Inkrementacyjne optymalizowanie kodu Cythona.
 - Wielowątkowość poza GIL(Global Interpreter Lock).
 - Wywołania zewnętrznych bibliotek C.
 - Pisanie 'opakowań' API Pythona.
 - Wywołania funkcji z C przez moduł rozszerzeń.
 - Numeryczne obliczenia z NumPy

- Standardowe obliczenia algebry liniowej i operacje na tablicach.
- Konstrukcja macierzy i ich właściwości w przykładach.
- Porównanie prędkości pomiędzy dynamicznie określanymi typami danych Pythona i definiowanymi `explicit` tablicami NumPy.
- Odniesienie do zależności pomiędzy typami danych w NumPy i C.
- Cięcie i rzutowanie macierzy - czytanie i pisanie do niezależnych części tablic, zastosowanie rzutowania do operacji na macierzach o różnych wymiarach.
- Funkcje uniwersalne - wykonywanie wielu operacji na całych macierzach, niezależnie od ich wymiaru, przykłady użycia.
- Algebra numeryczna.
- Praca z brakującymi wartościami - maskowane i NA-maskowane macierze, użycie do obliczeń na tablicach z brakującymi lub nieprawidłowymi danymi.
- Dostosowywanie obsługi błędów - NumPy oferuje wysokopoziomowe podejście do obsługi błędów bez strat na wydajności działania.
- Wsparcie testowania - NumPy oferuje funkcjonalności pomagające w pisaniu testów - kurs pokrywa podstawy korzystania z nich.
- Szybkie przetwarzanie NumPy z Cythonem
 - Użycie interfejsu buforowania Pythona z Cythonem.
 - Bezpośredni dostęp do buforów danych innych rozszerzeń Pythona.
 - Odzyskiwanie metadanych o warstwie buforowej.
 - Ustawianie wydajnych widoków pamięci na zewnętrznych buforach.
 - Implementacja szybkich pętli Cythona nad macierzami NumPy.
 - Iteracja nad buforami wyeksportowanymi z NumPy
 - Implementacja prostego algorytmu przetwarzania obrazów
 - Użycie "typów skondensowanych" do implementacji algorytmu i wydajnego przeprowadzenia jego wykonania z użyciem różnych typów danych z C.
 - Użycie równoległe wykonujących się pętli celem skorzystania z przetwarzania wieloprocessorowego.
 - Budowanie modułów z OpenMP.
 - Równoległe przetwarzanie danych.
 - Przyspieszenie działania pętli używając wątków OpenMP.

Wymagania:

- Doświadczenie w programowaniu w Pythonie jest wymagane.
- Podstawowa znajomość języka C jest przydatna - ale nie wymagana.

Poziom trudności



Certyfikaty:

Uczestnicy otrzymują po zakończeniu szkolenia zaświadczenie o ukończeniu autoryzowanego kursu Python Academy.

Prowadzący:

Autoryzowany wykładowca Python Academy.