


Informatyka Stosowana		
Laboratorium 1	<i>Wprowadzenie do NumPy – podstawowe operacje i porównanie z listami Pythonowymi</i>	 POLITECHNIKA BYDGOSKA Wydział Telekomunikacji, Informatyki i Elektrotechniki
Przedmiot	Matematyczne Podstawy Sztucznej Inteligencji - laboratorium	
Prowadzący	mgr inż. Gracjan Kątek	

1. Wprowadzenie

NumPy (Numerical Python) to biblioteka do obliczeń numerycznych w Pythonie. Umożliwia efektywne wykonywanie operacji na tablicach wielowymiarowych (ang. arrays), które mogą reprezentować wektory, macierze lub nawet tensory. NumPy oferuje zoptymalizowane operacje matematyczne, które są znacznie szybsze od analogicznych operacji na standardowych listach.

Główne cechy NumPy:

- ☑ Tablice wielowymiarowe: NumPy wprowadza własny typ danych, znany jako `numpy.ndarray`, który pozwala na tworzenie tablic wielowymiarowych. Te tablice są bardziej wydajne niż listy Pythona i umożliwiają wygodne przeprowadzanie operacji na danych numerycznych.
- ☑ Szybkie operacje matematyczne: NumPy oferuje bogaty zestaw funkcji do wykonywania operacji matematycznych na tablicach, takie jak dodawanie, odejmowanie, mnożenie, dzielenie i wiele innych. Te operacje są zoptymalizowane pod kątem wydajności.
- ☑ Broadcasting: NumPy obsługuje broadcasting, co oznacza, że może wykonywać operacje między tablicami różnych kształtów, automatycznie dostosowując ich rozmiar w sposób logiczny.
- ☑ Obsługa losowości: NumPy zawiera narzędzia do generowania liczb losowych i rozkładów prawdopodobieństwa, co jest przydatne w analizie statystycznej i symulacjach.
- ☑ Integracja z innymi bibliotekami: NumPy jest często używane w połączeniu z innymi bibliotekami do analizy danych, takimi jak pandas, matplotlib czy scikit-learn.
- ☑ Obsługa operacji na danych tablicowych, takich jak transpozycja, indeksowanie, wycinanie i inne.

NumPy jest niezwykle przydatne w dziedzinach takich jak nauki przyrodnicze, inżynieria, analiza danych, uczenie maszynowe i wiele innych, gdzie wydajne i efektywne obliczenia numeryczne są niezbędne.



2. Zadania do wykonania

Zadanie 1. Tworzenie list i tablic NumPy

Twoim zadaniem jest utworzenie listy Pythonowej zawierającej 1 000 elementów. Następnie stwórz tablicę NumPy o tym samym rozmiarze, czyli również 1 000 elementów. Kolejnym krokiem będzie zsumowanie wszystkich elementów zarówno w liście, jak i w tablicy. Po zakończeniu sumowania, przekształć każdy element listy i tablicy poprzez pomnożenie go przez 2.

Zadanie 2. Porównanie czasu wykonywania operacji

Twoim zadaniem jest porównanie czasu wykonania operacji matematycznych na liście Pythonowej oraz tablicy NumPy. Najpierw zmierz czas potrzebny na zsumowanie wszystkich elementów w liście Pythonowej, a następnie dokonaj takiego samego pomiaru dla tablicy NumPy. Po zsumowaniu elementów, zmierz czas wykonania operacji polegającej na pomnożeniu każdego elementu listy oraz tablicy przez 2. Porównaj uzyskane czasy, aby zobaczyć różnice w wydajności obu struktur danych.

Zadanie 3. Praca z tablicami NumPy – operacje na macierzach

Twoim zadaniem jest stworzenie dwuwymiarowej macierzy o wymiarach 10x10 za pomocą biblioteki NumPy. Następnie oblicz sumę wszystkich elementów tej macierzy. Po obliczeniu sumy, wykonaj operację mnożenia tej macierzy przez inną macierz o takich samych wymiarach, czyli 10x10.

Zadanie 4. Analiza wyników

Twoim zadaniem jest zbadanie różnic w czasie działania operacji matematycznych na tablicach NumPy w porównaniu do list Pythonowych. W pierwszej kolejności zidentyfikuj operacje, w których NumPy było szybsze od list. Następnie przeanalizuj potencjalne przyczyny zauważalnej różnicy w wydajności. Zadanie obejmuje również odpowiedzi na kilka kluczowych pytań, które powinny zostać uwzględnione w sprawozdaniu:

- ☒ Czy różnice w czasie działania są zauważalne przy małych rozmiarach danych?
- ☒ Jak zmienia się czas działania w miarę zwiększania rozmiaru danych?
- ☒ Jakie inne zalety NumPy można wskazać, oprócz wydajności?

Dodatkowo, wygeneruj tablicę NumPy o wielkości 1 000 000 elementów i porównaj czas wykonania operacji matematycznych na tej tablicy z czasem dla analogicznej listy Pythonowej. Na koniec zapisz wyniki swoich pomiarów i wyciągnij wnioski na temat wydajności obu struktur

danych oraz ich zastosowań. Pamiętaj, aby odpowiedzi na powyższe pytania zamieścić w końcowej części sprawozdania.

Punktacja zadań:

Zadanie	Zadanie 1	Zadanie 2	Zadanie 3	Zadanie 4	Suma
Punktacja	2	2	2	2	8

3. Sprawozdanie

Sprawozdanie powinno zawierać:

- ☒ Treść zadania
- ☒ Kod napisanego programu
- ☒ Wynik działania napisanego programu
- ☒ Opis działania programu
- ☒ Wnioski końcowe

Sprawozdanie musi być przesłane w formacie .pdf i zgodne z formatką. Sprawozdanie należy dostarczyć najpóźniej do północy dnia poprzedzającego dzień kolejnych laboratoriów. W przypadku spóźnienia przysługują 2 terminy poprawkowe wskazane przez prowadzącego.