
	Politechnika Bydgoska im. J. J. Śniadeckich Wydział Telekomunikacji, Informatyki i Elektrotechniki		
Przedmiot	Skryptowe języki programowania		
Prowadzący	mgr inż. Martyna Tarczewska		
Temat	Numpy		
Student			
Nr ćw.	9	Data wykonania	
Ocena		Data oddania spr.	

1. Cel ćwiczenia

Celem ćwiczenia jest poznanie modułu Numpy, która jest podstawową biblioteką do przetwarzania dużych zbiorów danych oraz obliczeń numerycznych. Wykonane w ramach laboratorium zadania pozwolą na wykorzystanie w praktyce przedstawionych treści.

2. Informacje podstawowe

2.1. Moduł Numpy

Moduł Numpy jest podstawowym zestawem narzędzi dla języka Python umożliwiającym zaawansowane obliczenia matematyczne, w szczególności do zastosowań naukowych (tzw. *obliczenia numeryczne*, jak mnożenie i dodawanie macierzy, całkowanie, rozwiązywanie równań, itd.). Sposób importu modułu został pokazany poniżej (zwykle korzysta się z aliasu *np.*).

```
import numpy as np
```

Najważniejszym obiektem, na którym bazuje pakiet Numpy i szereg pakietów z niego korzystających, jest klasa *ndarray* wprowadzająca obiekty *array*. Obiekty *array* możemy traktować jako uniwersalne pojemniki na dane w postaci macierzy (czyli wektorów lub tablic). W porównaniu ze standardowymi typami sekwencji Pythonowych (lista, krotka) jest kilka różnic w operowaniu tymi obiektami:

1. obiekty przechowywane w tablicy array muszą być wszystkie tego samego typu;
2. obiekty array zachowują swój rozmiar; przy zmianie rozmiaru takiego obiektu powstaje nowy obiekt, a obiekt sprzed zmiany zostaje usunięty;
3. obiekty array wyposażone są w bogaty zestaw funkcji operujących na wszystkich przechowywanych w obiekcie danych, specjalnie zoptymalizowanych do przetwarzania dużych ilości danych.

Sposoby tworzenia i wypełniania wartościami tablic modułu Numpy.

```
a = np.array([1,3,5,7]) # array([1, 3, 5, 7])
b = np.arange(4) # array([0, 1, 2, 3])
np.arange(2,10,1) # array([2, 3, 4, 5, 6, 7, 8, 9])
np.linspace(0,10,6) # array([ 0., 2., 4., 6., 8., 10.] )
```

Moduł Numpy udostępnia również tablice wielowymiarowe.

```
c = np.array([[1,2,3],[4,5,6]])
d = np.zeros((2,3)) # macierz 2x3 wypełniona zerami
d = np.ones((2,3)) # macierz 2x3 wypełniona jedynkami
np.random.random((2,3)) # macierz 2x3 wypełniona losowymi liczbami
e = np.eye(2) # macierz kwadratowa z jedynkami na przekątnej
f = np.full((3,4),2) # macierz 3x4 wypełniona 2
g = np.diag([1,2,3]) # macierz diagonalna
```

Tablice Numpy można do siebie dodawać, odejmować, podnosić ich zawartość do potęgi oraz wykonywać inne proste działania.

```
a + b # suma macierzy
c ** 2 # zawartość podniesiona do potęgi
c.shape # kształt tablicy
c.T # transpozycja macierzy
c > 3 # True/False dla poszczególnych elementów
c.ndim # liczba wymiarów tablicy
c.size # rozmiar tablicy
c.dtype # typ danych w tablicy
a = np.arange(1,10) # a = array([1, 2, 3, 4, 5, 6, 7, 8, 9])
b = a.reshape(3,3) # b = array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
a.resize(3,3) # a = array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
a.ravel() # spłaszczenie tablicy do 1-wymiarowej
a.tolist() # konwersja na listę
a.sum() # suma elementów
a.min() #minimum
a.max() #maksimum
a.average() #średnia z opcją dodania wag
a.mean() #średnia
np.count_nonzero((x == 2) | (x == 7)) #zliczanie elementów
b.max(axis=1) # macierz zredukowana do 1 kolumny z max. wart. W wierszach
np.where(x > 0, 2, -2) # gdzie x > 0 podmień na 2, reszta na -2
if (a > 5).all():
    print("Wszystkie elementy a większe od 5")
if (a > 5).any():
    print("Przynajmniej jeden element a jest większy od 5")
```

Przydatną funkcjonalnością modułu Numpy jest praca z plikami tekstowymi (np. w formacie CSV). Poniżej podany zapis spowoduje wczytanie 10 wierszy z danymi po pominięciu pierwszego wiersza z pliku myData w formacie UTF8, oddzielając kolumny przecinkiem. Opcji wczytywania danych z pliku jest znacznie więcej, a zostały opisane w [dokumentacji](#).

```
data = np.genfromtxt(  
    'myData.csv',  
    dtype=U8,  
    delimiter=',',  
    skip_header=1,  
    max_rows=10)
```

3. Przebieg ćwiczenia

3.1. Zadanie 1.

Stworzyć funkcję *replace_zeros(A, x)*, która przyjmuje macierz dowolnego rozmiaru oraz liczbę x. W wyniku działania funkcji mamy otrzymać listę ze wszystkimi zerami zmienionymi na liczbę x.

3.2. Zadanie 2.

Stworzyć funkcję *medianize(A)*, która od każdego elementu tablicy odejmie średnią wartość tej tablicy.

3.3. Zadanie 3.

Wygeneruj macierz wypełnioną liczbami naturalnymi mniejszymi od 100. Wypisz element największy globalnie oraz największy w każdym wierszu i największy w każdej kolumnie.

3.4. Zadanie 4.

Przetestować funkcję *reshape* na dwa sposoby. Podać -1 jako pierwszy parametr, następnie jako drugi.

3.5. Zadanie 5.

Wczytać plik *oceny.csv*. Wypisać:

- najniższą ocenę z laboratoriów dla każdego studenta,
- średnią ocenę z egzaminu,
- liczbę 2 ze egzaminu,
- czy jest student, który miał same 5 z laboratoriów,
- czy jest student, który miał 2 z LAB2 i LAB3,
- zliczyć ilu studentów dostało wyższą ocenę z egzaminu niż ich średnia ocen z laboratoriów,
- liczbę piątek, którą uzyskał student mający najwięcej 5 w całej grupie.

3.6. Zadanie 6.

Tablice Numpy również dają się bardzo prosto sortować. Wylosować 10 liczb do tablicy Numpy, a następnie przetestować działanie funkcji *sort*. Posortować tablicę rosnąco i malejąco.

3.7. Zadanie 7.

Stworzyć tablicę 5x5. Dla każdego wiersza obliczyć średnią ważoną używając wag 1,2,3,2,1.

3.8. Zadanie 8.

Stworzyć tablicę 10x10. Wypełnić ją losowymi liczbami z zakresu $<0,10>$. Zliczyć i wypisać licznosc wystąpień elementów.

3.9. Zadanie 9.

Przetestować operator `*` oraz operator `@`. Do czego służą?

3.10. Zadanie 10.

Wypróbować funkcję *set_printoptions*. Jakie daje możliwości?

4. Sprawozdanie

Sprawozdanie z laboratorium powinno zawierać:

- wypełnioną tabelę z początku instrukcji,
- kody programów będących rozwiązaniami wszystkich zadań wraz z komentarzami,
- demonstrację działania programu,
- odpowiedzi na pytania zawarte w zadaniach,
- wnioski.