Języki programowania – Python
Przetwarzanie tablic
NumPy

Marzena Nowakowska

Wydział Zarządzania i Modelowania Komputerowego
Politechnika Świętokrzyska

Budynek C, p. 3.21

spimn@tu.kielce.pl

Biblioteka NumPy

NumPy to biblioteka dla języka programowania Python, dodająca obsługę dużych, wielowymiarowych tablic i macierzowych struktur danych. Należy bibliotekę zaimportować, aby móc korzystać z jej zasobów:

```
import numpy          # rzadsze, odwołanie: numpy.funkcja()
```

albo

```
import numpy as np   # częstsze, odwołanie: np.funkcja()
```

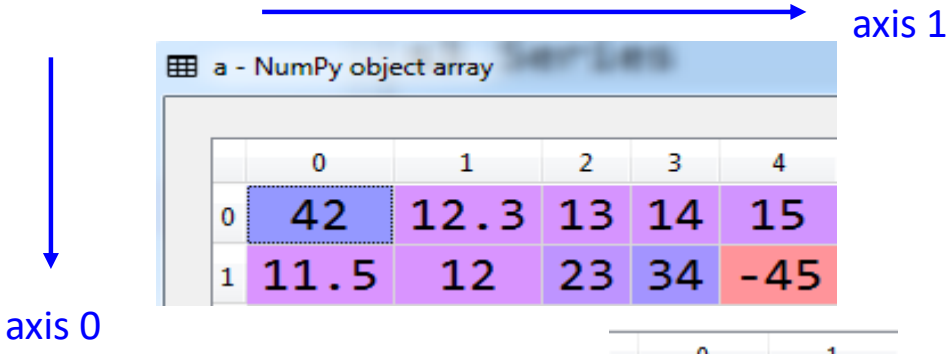
NumPy zapewnia *N*-wymiarowy typ tablicy, ***ndarray***, który opisuje zbiór „elementów" tego samego typu. Pozycje mogą być indeksowane za pomocą np. liczb całkowitych. Odpowiednikiem tablic może być lista list w Pythonie. Lista może zawierać różne typy danych w ramach jednej listy, natomiast wszystkie elementy tablicy *NumPy* są jednorodne; każdy element tablicy zajmuje ten sam rozmiar bloku pamięci.

Uwaga: może istnieć potrzeba instalacji biblioteki, co można zrealizować z wykorzystaniem programu pip w oknie konsoli Spydera:

```
pip install numpy
```

Tablica – główna struktura danych NumPy

Jest to tablica elementów (zwykle liczb), wszystkich tego samego typu, indeksowanych krotką nieujemnych liczb całkowitych



macierz transponowana: a.T

	0	1
0	42	11.5
1	12.3	12
2	13	23
3	14	34
4	15	-45

Inne funkcje tworzące tablicę

- *numpy.**eye***
- *numpy.**empty***
- *numpy.**ones***
- *numpy.**zeros***
- *numpy.**full***
- *numpy.**linspace***
- *numpy.**diag***

Tworzenie tablicy

*numpy.**array**(object, dtype=None, *, copy=True, order='K', subok=False, ndmin=0, like=None)*

Najczęściej wywoływana dla pierwszego (obowiązkowego) parametru. Parametr *object* musi mieć strukturę kompatybilną ze strukturą tablicy (*array-like*). W szczególności, może być zagnieżdżoną sekwencją.

Przykład

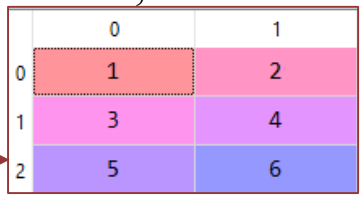
```
a = numpy.array([(42, 12.3, 13, 14, 15), (11.5, 12, 23, 34, -45)])
b1 = numpy.array([1,  2,  3]) ; b2 =  numpy.array([[1], [2], [3]])      # kolumna
c = numpy.array([[1,2,3]])                                              # wiersz
print(a[0], "\n", a[1] , "\n", a[0, 2] , "\n", a[1][2])
```

NumPy – dostęp do tablicy

Kiedy wyświetla się tablicę, *NumPy* wyświetla ją w sposób podobny do list zagnieżdżonych. Tablice jednowymiarowe są wyświetlane jako wiersze, dwuwymiarowe jako macierze, a trójwymiarowe jako listy macierzy.

a = numpy.array([1,2,3,4])

b = numpy.array([[1,2], [3,4], [5,6]])



c = numpy. array([[[2,17], [45, 78]], [[88, 92], [60, 76]],[[76,33],[20,18]]])

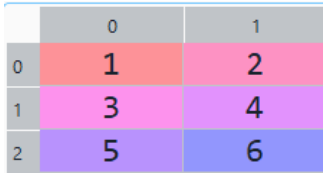
Por. wyniki instrukcji: *print(a); print(b); print(c)*

Dostęp do elementów tablicy uzyskuje się poprzez podanie ich indeksów lub zakresów indeksów (wycinanie). Wykorzystując wycinanie i indeksowanie można wartościować elementy tablicy.

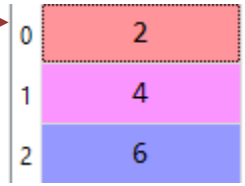
Wycinanie. Operator wycinania umieszcza się w miejscu indeksowania elementów tablicy. Ma składnię: *start : stop : step*, przy czym obszar tablicy o indeksie *stop* jest wykluczany.

Brak wartości w operatorze wycinania oznacza wartości domyslne definiujące zakres od wartości na pietrwszej pozycji do wartości na ostatniej pozycji,z krokiem 1

b[:, :]



b[:,1]



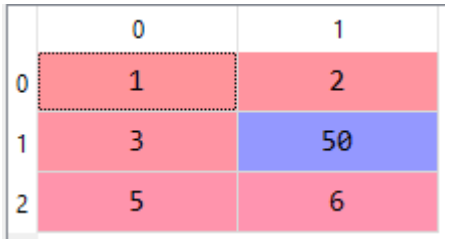
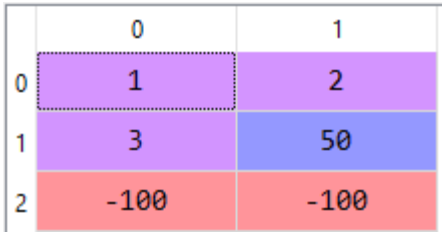
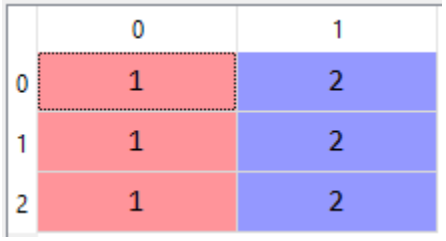
b[0]



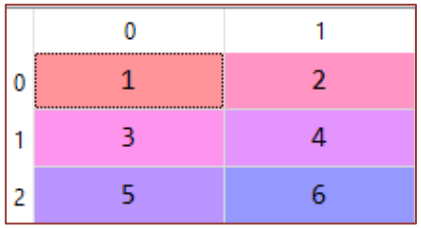
b[2][1] albo *b[2,1]*



NumPy – dostęp do tablicy; ilustracja

b[1,1] = 50	b[2] = -100	b[1:3, 0:2] = b [0:1, 0:2]
wartościowanie elementu	wartościowanie wiersza (drugiego) takimi samym	wartościowanie wycinka tablicy
		

b = numpy.array([[1,2], [3,4], [5,6]])




```
a  = np.array((1,2,3,4,5))
print("Macierz a elementami”)
# macierz jednowymiarowa
for el in a:
    print(el)
print("Macierz a całosciowo")
print(a)
```

```
print("Macierz bb wierszami")
# macierz dwuwymiarowa
bb = np.array([[1,2,3,4], [9,8,7,6]])
for el in bb:
    print(el)
print("Macierz b elementami")
rozmiar=bb.shape
for el1 in range(0, rozmiar[0]):
    for el2 in range(0,rozmiar[1]):
        print(bb[el1,el2], end=" ")
    print()
print("Macierz b całosciowo")
print(bb)
```

Pozycyjne operacje arytmetyczne na tablicach

To są operacje dodawania, odejmowania, mnożenia i dzielenia, w których w których argumentami:

- tablice o takich samych kształtach (liczbie kolumn i wierszy),
- tablica i skalar (liczba)

Gdy argumentami są tablice to muszą być o takich samych kształtach

```
import numpy
a=numpy.array([1,2]); b=numpy.array([3,4])
c1=a+b; c2=a+2
d1=a-b; d2=a-c2
e1=a*b; e2=a*2
f1=a/b; f2=a/2
g1=a**b; g2=a**2
```

Obliczenia na tablicach

https://numpy.org/doc/stable/reference/index.html. M.in.:

- **Universal functions**; działają na kolejnych elementach tablicy zwracając wynik typu *ndarray*:
  - operacje matematyczne,
  - funkcje trygonometryczne,
  - funkcje bitowe,
  - obsługa operacji porównania (relacyjnych) oraz logicznych,
  - grupa funkcje różnorodnych (*floating functions*)

  Funkcja uniwersalna jest „zwektoryzowanym" opakowaniem dla funkcji, która pobiera stałą liczbę określonych danych wejściowych i generuje stałą liczbę określonych danych wyjściowych.
- **Routines** to funkcje blokowe działające na całej tablicy
  - funkcje matematyczne (*Mathematical functions with automatic domain*)
  - dyskretne transformaty Fouriera (*Discrete Fourier Transform*)
  - **algebra liniowa (*Linear algebra*)**
  - funkcje wielomianowe (*Polynomials*)
  - generatory liczb pseudolosowych (*Random sampling*); wg zadanego rozkładu
  - sortowanie, wyszukiwanie, liczenie (*Sorting, searching, and counting*)
  - wyznaczanie statystyk, korelacji i szeregów rozdzielczych (w dokumentacji jest podane, że to histogramy, więc ta informacją jest myląca ponieważ histogram jest wykresem) wg wskazanych osi tablicy (*Statistics*).

Algebra liniowa w NumPy

Pakiet *NumPy* zawiera klasę ***linalg***, który zapewnia wszystkie funkcje wymagane do algebry liniowej. Niektóre z wybranych funkcji biblioteki *numpy* oraz metody klasy *linalg* z ważnych narzędzi tego modułu są opisane poniżej.

dot(A, B). Iloczyn punktowy (*dot*) dwóch macierzy (liczba kolumn pierwszej macierzy musi być taka sama jak liczba wierszy drugiej macierzy). Jeżeli macierze są dwuwymiarowe, to jest iloczyn tych macierzy (wynikiem jest macierz). Jeżeli macierze są jednowymiarowe (wektory), funkcja wyznacza iloczyn skalarny (skalar).

outer(A, B). Iloczyn zewnętrzny dwóch wektorów. Wynikiem jest macierz o wymiarach zdefiniowanych przez długości wektorów.

vdot(A, B). Iloczyn skalarny dwóch wektorów. Jeżeli argumentami są macierze dwuwymiarowe, wcześniej zostaną przekształcone do wektorów (wiersz za wierszem)

linalg.det(A). Do obliczenia wyznacznika macierzy kwadratowej wykorzystywana jest klasa *linalg* biblioteki *numpy.*

linalg.matrix_rank(A). Wyznaczenie rzędu macierzy (wykorzystuje metodę SVD; dla tej metody rząd macierzy to liczba jej wartości własnych większych niż pewna tolerancja).

linalg.inv(A). Wyznaczenie macierzy odwrotnej do macierzy zadanej. Macierz pierwotna musi być kwadratowa. Nie można obliczyć macierzy odwrotnej z macierzy osobliwej, tzn. takiej której wyznacznik jest równy zero. Przy odwracaniu macierzy należy więc sprawdzić, czy jej wyznacznik jest niezerowy.

linalg.solve(A, B). Wyznacza rozwiązanie równania macierzowego AX = B, gdzie A jest macierzą kwadratową współczynników, B jest wektorem wyrazów wolnych.

Zad_06-numpy_ArgLin.py

Algebra liniowa w NumPy; ilustracja

```
import numpy
# Iloczyn punktowy
a=numpy.array([[-1,-2], [0,2],  [-1, 3]]); b=numpy.array([[1, 5, 1], [2,1,2]]); c=numpy.dot(a,b)

# Iloczyn wektorów dostarczający macierz [1 2 3 4]T *[5 6 7]
# Nie ma operacji transponowania wektora
aa=numpy.array([1, 2, 3, 4]); cc=numpy.array([5,6,7]) ; dd=numpy.outer(aa, cc)

# Wyznacznik macierzy
a = numpy.array([[1, 2], [3, 4]]); b = numpy.array([[1,  5,  1], [2, 1, 2], [-4, -5, -8]])
dt1=numpy.linalg.det(a) ; dt2=numpy.linalg.det(b)

# Wyznaczenie rzędu macierzy; rozmiar największej kwadratowej podmacierzy macierzy
# danej macierzy o niezerowym wyznaczniku
a = numpy.array([[1,2,3], [2,4,1], [1,2,2]]) ; rz = numpy.linalg.matrix_rank(a)

# Wyznaczenia macierzy odwrotnej
a =  numpy.array([[1, 2, 8], [3, 5,-5], [6, 4, 10]])
b = numpy.round(numpy.linalg.inv(a), 2)
```