Biblioteka NumPy

1 Instalacja Biblioteki NumPy

Aby móc skorzystać z biblioteki Numpy (i każdej innej zewnętrznej biblioteki) w swoim projekcie należy ją najpierw zainstalować w środowisku wirtualnym danego projektu. W Instalację możemy wykonać na kilka sposobów.

1.1 Instalacja z wykorzystaniem środowiska PyCharm

1. z menu File wybieramy opcję Settings...
2. z kategorii po lewej stronie wybieramy Project: nazwa_projektu a następnie Project Interpreter
3. po prawej stronie wybieramy i w polu wyszukiwania wpisujemy numpy. Następnie z listy wybieramy pozycje numpy i na dole przycisk Install Package. Po pomyślnym zakończeniu procesu instalacji możemy korzystać poprzez import z zainstalowanej biblioteki.

1.2 Instalacja z wiersza poleceń

Aby dokonać instalacji z wiersza poleceń należy najpierw uruchomić terminal (Windows lub Linux) a następnie aktywować wybrane środowisko wirtualne Pythona. Aktywacja środowiska polega na uruchomieniu w terminalu pliku ścieżka_do_środowiska_wirtualnego/Scripts/activate(.bat lub .sh w zależności od systemu operacyjnego). Jeżeli środowisko zostało aktywowane powinniśmy widzieć jego nazwę przed ścieżką i znakiem zachęty wiersza poleceń. Łatwiejszy sposób polega na wybraniu zakładki Terminal na dole okna programu Pycharm – tam środowisko wirtualne jest już aktywowane. Narzędzie, które służy do instalacji pakietów Pythona to PIP. Bibliotekę Numpy zainstalujemy po wpisaniu polecenia:

```
python -m pip install numpy
```

1.3 Instalacja przy pomocy pliku requirements.txt

W środowisku PyCharm możemy w projekcie utworzyć plik requirements.txt w którym możemy wskazać wymagane w projekcie biblioteki w konkretnych wersjach. Zawartość tego pliku powinna wyglądać w następujący sposób:

```
nazwa_biblioteki==wersja
```

Przykładowo:

```
contourpy==1.0.7
cycler==0.11.0
et-xmlfile==1.1.0
fonttools==4.38.0
kiwisolver==1.4.4
matplotlib==3.6.3
numpy==1.24.1
openpyxl==3.1.2
packaging==23.0
pandas==1.5.3
Pillow==9.4.0
pyparsing==3.0.9
```

```
python-dateutil==2.8.2
pytz==2022.7.1
seaborn==0.12.2
six==1.16.0
```

2 Tworzenie tablic NumPy

Tablice biblioteki Numpy to kolekcje, które mogą przechowywać dane jednorodne, czyli dane tego samego typu. Taki stan rzeczy powoduje, że w kwestii przechowywania danych nie są tak uniwersalne jak listy, ale z racji tego, że znając typ danych, który będzie przechowywany można łatwo obliczyć jaki będzie rozmiar tablicy w pamięci. Dzięki temu Numpy może wykonywać operacje na całych wektorach wartości a nie na pojedynczych elementach jak w przypadku list. Biblioteka Numpy w znakomitej części jest napisana w języku C co zapewnia bardzo wysoką wydajność większości operacji.
Deklaracja tablicy korzystającej z podobnego mechanizmu działania jak funkcja range():

```
import numpy as np
a = np.arange(2)
```

Po wypisaniu zmiennej otrzymamy informacje postaci `array([0, 1])` lub `[0 1]` w zależności od tego czy kod zostanie uruchomiony w konsoli czy ze skryptu.

Faktyczną nazwą klasy dla tablic Numpy jest `ndarray` co stanowi skrót od n dimensional array czyli tablica n wymiarowa.

Przykład 1.

```
import numpy as np
# inicjalizacja tablicy
a = np.arange(2)
print(a)
# wypisanie typu zmiennej tablicy (nie jej elementow) - ndarray
print(type(a))
# sprawdzenie typu danych tablicy
print(a.dtype)
# inicjalizacja tablicy z konkretnym typem danych
a = np.arange(2, dtype='int64')
print(a.dtype)
# zapisanie kopii tablicy jako tablicy z innym typem
b = a.astype('float')
print(b)
# wypisanie rozmiarow tablicy
print(b.shape)
# mozna tez sprawdzic ilosc wymiarow tablicy
print(a.ndim)
# stworzenie tablicy wielowymiarowej moze wygladac tak
# parametrem przekazywanym do funkcji array jest obiekt, ktory
#zostanie skonwertowany na tablice
# moze to byc Pythonowa lista
m = np.array([np.arange(2), np.arange(2)])
print(m)
# ponownie typem jest ndarray
print(type(m))
```

Pełna lista typów danych, które możemy umieścić w tablicach Numpy znajduje się pod adresem: https://docs.scipy.org/doc/numpy-1.14.0/user/basics.types.html

Zadanie 1

Za pomocą funkcji arange stwórz tablicę Numpy składającą się 20 kolejnych wielokrotności liczby 2.

Zadanie 2 DO DOMU
Stwórz listę składającą się z wartości zmiennoprzecinkowych a następnie zapisz do innej zmiennej jej kopię przekonwertowaną na typ int64
Zadanie 3
Napisz funkcję, która będzie:

- przyjmowała jeden parametr 'n' w postaci liczby całkowitej
- zwracała tablicę Numpy o wymiarach n*n kolejnych liczb całkowitych poczynając od 1

Przykład 2.
Istnieją sposoby na szybkie stworzenie bardziej rozbudowanych tablic/macierzy.

```
import numpy as np
# mozemy w latwy sposob stworzyc macierz danego rozmiaru wypelniona
#zerami lub jedynkami
zera = np.zeros((5,5))
jedynki = np.ones((4,4))
# warto sprawdzic jaki jest domyslny typ danych takich tablic
# mozna rowniez stworzyc "pusta" macierz o podanych wymiarach, ktora
# wcale pusta nie jest
# wartosci umieszczane sa losowe
pusta = np.empty((2, 2))
# do elementow tablic mozemy odwolac sie tak jak do
# elementow np. listy czyli podajac indeksy
poz_1 = pusta[1, 1]
poz_2 = pusta[0, 1]
# funkcja arange potrafi rowniez tworzyc
# ciagi liczb zmiennoprzecinkowych
liczby = np.arange(1, 2, 0.1)
# podobnie dziala funkcja linspace, ktorej dzialanie jest
# rownowazne tej samej funkcji w MATLAB-ie
liczby_lin = np.linspace(1, 2, 5)
# sprawdz rowniez dzialanie funkcji logspace
# a teraz mozemy utworzyc dwie macierze, najpierw wartosci
# iterowane sa w kolumnie a nastepnie w wierszu
z = np.indices((5, 3))
# wielowymiarowe macierze mozemy rowniez generowac funkcja mgrid
x, y = np.mgrid[0:5, 0:5]
# podobnie jak w MATLAB-ie mozemy tworzyc macierze diagonalne
mat_diag = np.diag([a for a in range(5)])
# w powyzszym przykladzie stworzony wektor wartosci zostanie
# umieszczony na glownej przekatnej macierzy
# mozemy podac drugi parametr funkcji diag, ktory okresla
# indeks przekatnej wzgledem glownej przekatnej,
# ktora zostanie wypelniona wartosciami podanego wektora
mat_diag_k = np.diag([a for a in range(5)], -2)
# Numpy jest w stanie stworzyc tablice jednowymiarowa z dowolnego
# obiektu iterowalnego (iterable)
z = np.fromiter(range(5), dtype='int32')
# ciekawa funkcja Numpy jest funkcja frombuffer, dzieki ktorej
# mozemy stworzyc np. tablice znakow
marcin = b'Marcin'
mar = np.frombuffer(marcin, dtype='S1')
mar_2 = np.frombuffer(marcin, dtype='S2')
# powyzsza funkcja ma jednak pewna wade dla Pythona 3.x,
# ktora powoduje, ze trzeba jawnie okreslac
# iz ciag znakow przekazujemy jako ciag bajtow co osiagamy
# poprzez podanie litery 'b' przed wartoscia
# zmiennej tekstowej. Mozna podobny efekt osiagnac inaczej,
```

```
# sprawdz ponizsze przyklady
marcin = 'Marcin'
mar_3 = np.array(list(marcin))
print(mar_3)
mar_3 = np.array(list(marcin), dtype='S1')
print(mar_3)
mar_3 = np.array(list(b'Marcin'))
print(mar_3)
mar_3 = np.fromiter(marcin, dtype='S1')
print(mar_3)
mar_3 = np.fromiter(marcin, dtype='U1')
print(mar_3)
# tablice numpy mozemy w prosty sposob do siebie dodawac
mat = np.ones((2, 2))
mat2 = np.ones((2, 2))
mat = mat + mat2
print(mat)
# odejmowac
print(mat - mat2)
# mnozyc
print(mat * mat)
# dzielic
print(mat / mat)
```

Zadanie 4
Napisz funkcję, która będzie przyjmowała 2 parametry: liczbę, która będzie podstawą operacji potęgowania oraz ilość kolejnych potęg do wygenerowania. Korzystając z funkcji logspace generuj tablicę jednowymiarową kolejnych potęg podanej liczby, np. generuj(2,4) -> [2 4 8 16]
Zadanie 5
Napisz funkcję, która:

- na wejściu przyjmuje jeden parametr określający długość wektora,
- na podstawie parametru generuje wektor, ale w kolejności odwróconej (czyli np. dla n=3 =>[3 2 1])
- generuje macierz diagonalną z w/w wektorem jako przekątną

Zadanie 6 DO DOMU
Stwórz skrypt który na wyjściu wyświetli macierz numpy (tablica wielowymiarowa) w postaci wykreślanki, gdzie jedno słowo będzie wypisane w kolumnie, jedno w wierszu i jedno po ukosie. Jedno z tych słów powinno być wypisane od prawej do lewej.
Zadanie 7
Napisz funkcję, która wygeneruje macierz wielowymiarową postaci:
[[2, 4, 6]
[4, 2, 4]
[6, 4, 2]]
Przy założeniach:

- funkcja przyjmuje parametr n, który określa wymiary macierzy jako n*n i umieszcza wielokrotność liczby 2 na kolejnych jej przekątnych rozchodzących się od głównej przekątnej.

Więcej przykładów tworzenia tablic Numpy można znaleźć pod adresem: https://docs.scipy.org/doc/numpy1.14.0/re
creation.html

3 Indeksowanie i cięcie tablic

Cięcie i indeksowanie danych w tablicach Numpy jest możliwe do wykonania na bardzo wiele sposobów.
Poniżej przykłady niektórych z nich.

Przykład 3

```
# ciecie (slicing) tablic numpy mozna wykonac za pomoca wartosci z
# funkcji slice lub range
a = np.arange(10)
s = slice(2, 7, 2)
print(a[s])
s = range(2, 7, 2)
print(a[s])
# # mozemy ciac tablice rowniez w sposob znany z ciecia list
# (efekt jak wyzej)
print(a[2:7:2])
# # lub tak
print(a[1:])
print(a[2:5])
# w podobny sposob postepujemy w przypadku tablic wielowymiarowych
mat = np.arange(25)
# teraz zmienimy ksztalt tablicy jednowymiarowej na macierz 5x5
mat = mat.reshape((5,5))
print(mat)
print(mat[1:]) # od drugiego wiersza
print(mat[:,1]) # druga kolumna jako wektor
print(mat[...,1]) # to samo z wykorzystaniem ellipsis (...)
print(mat[:,1:2]) # druga kolumna jako ndarray
print(mat[:,-1:]) # ostatnia kolumna
print(mat[1:3, 2:4]) # 2 i 3 kolumna dla 3 i 5 wierszu
print(mat[:, range(2,6,2)]) # 3 i 5 kolumna
# bardziej zaawansowane, lecz trudniejsze do zrozumienia ciecie
# mozna osianac wg. ponizszego przykladu
# y bedzie tablica zawierajaca wierzcholki macierzy x
x = np.array([[0, 1, 2], [3, 4, 5], [6, 7, 8], [9, 10, 11]])
rows = np.array([[0, 0],[3,3]])
cols = np.array([[0, 2], [0,2]])
y = x[rows, cols]
```

Zadanie 8
Napisz funkcję, która:

- jako parametr wejściowy będzie przyjmowała tablicę wielowymiarową numpy oraz parametr 'kierunek'
- parametr kierunek określa czy tablica wejściowa będzie dzielona w pionie czy poziomie
- funkcja dzieli tablicę wejściową na pół (napisz warunek, który wyświetli komunikat, że ilość wierszy lub kolumn, w zależności od kierunku podziału, nie pozwala na operację)

Zadanie 9 DO DOMU
Wykorzystaj poznane na zajęciach funkcje biblioteki Numpy i stwórz macierz 5x5, która będzie zawierała kolejne wartości ciągu Fibonacciego.