Kurs rozszerzony języka Python

Wykład 9.

Marcin Młotkowski

3 grudnia 2024

Plan wykładu

Plan wykładu

Biblioteka NumPy

Obliczenia numeryczne na n–wymiarowych tablicach

Biblioteka

```
import numpy as np
```

Podstawowy typ

`ndarray`: n–dimensional array

Podstawowy typ przypominający listę

Podstawowe cechy tego typu

- przechowują zmienne tylko jednego typu (głównie `np.int32`, `np.float64`);
- mają określony kształt (np. trójwymiarowa macierz rozmiaru 3x4x5);
- *broadcasting*: operacje na wszystkich elementach, np. pomnożenie wszystkich elementów przez liczbę;
- *views*: obiekty które są nie kopią innej tablicy, ale jej rzutem.

Po co ndarray

O wiele szybsze niż listy

Tworzenie tablic

```
import numpy as np
x = np.arange(15)
```

Tworzenie tablic

```
import numpy as np
x = np.arange(15)
x = np.zeros((4,5,6))
```

Tworzenie tablic

```
import numpy as np
x = np.arange(15)
x = np.zeros((4,5,6))
x = np.array([[3.1415, 2.7182, 1.6180],
              [4.135, 1.660, 12.56]])
```

Dostęp do elementów (indexing)

```
x = np.array([[3.1415, 2.7182, 1.6180], [4.135, 1.660, 12.5
```

- prawie tak jak już znamy: x[1,2]
  (Listy pythonowe: x[1][1] lub x[(1,1)]);

Dostęp do elementów (indexing)

```
x = np.array([[3.1415, 2.7182, 1.6180], [4.135, 1.660, 12.5
```

- prawie tak jak już znamy: x[1,2]
  (Listy pythonowe: x[1][1] lub x[(1,1)]);
- slicing:
  x[<*selekcja po wymiarze 0*>, <*selekcja po wymiarze 1*>, ...]

Dostęp do elementów (indexing)

```
x = np.array([[3.1415, 2.7182, 1.6180], [4.135, 1.660, 12.5
```

- prawie tak jak już znamy: x[1,2]
  (Listy pythonowe: x[1][1] lub x[(1,1)]);
- slicing:
  x[<*selekcja po wymiarze 0*>, <*selekcja po wymiarze 1*>, ...]
  `x[2, :]`

Dostęp do elementów (indexing)

```
x = np.array([[3.1415, 2.7182, 1.6180], [4.135, 1.660, 12.5
```

- prawie tak jak już znamy: x[1,2]
  (Listy pythonowe: x[1][1] lub x[(1,1)]);
- slicing:
  x[<*selekcja po wymiarze 0*>, <*selekcja po wymiarze 1*>, ...]

  ```
  x[2, :]
  ```

  ```
  x[:, -1]
  ```

Broadcasting

```
x = np.array([[3.1415, 2.7182, 1.6180], [4.135, 1.660, 12.56]])

y = x + 2.5
```

Broadcasting

```
x = np.array([[3.1415, 2.7182, 1.6180], [4.135, 1.660, 12.56]])

y = x + 2.5

y = x * 2.5
```

Wyszukiwanie

```
x = np.array([np.arange(10), np.arange(10)])
x.shape
x > 5
```

Zbuduj losową tablicę 2×3:

```
r = np.random.randn(2,3)
```

a następnie za wszystkie ujemne wartości wstaw 0:

```
np.where(a > 0, a, 0)
```

Odczyt i zapis

```
dane = np.loadtxt("dane.csv", delimiter=',', usecols=(5,7))
```

Odczyt i zapis

```
dane = np.loadtxt("dane.csv", delimiter=',', usecols=(5,7))
np.save("plik.csv", dane, delimiter='|'})
```

Plan wykładu

Przetwarzanie

W wersji podstawowej `matplotlib.image` przetwarza tylko obrazy w formacie PNG.

Przetwarzanie

W wersji podstawowej `matplotlib.image` przetwarza tylko obrazy w formacie PNG.

Do obsługi innych formatów konieczne jest zainstalowanie dodatkowych pakietów (`pillow`).

Zaczynamy

```
import matplotlib.image as mpimg
import matplotlib.pyplot as plt

img = mpimg.imread("stary_budynek.png")
plt.imshow(img)
plt.show()
```

Czym jest obrazek i jak z tego korzystać

```
print(img)
```

`img`: jest to tablica o kształcie $(M, N, 3)$ dla obrazów RGB.

Redukcja jednej składowej koloru

```
img[:,:,0] = 0
```

Podbicie czerwonego

```
red = np.where(img[:,:,0] > 0.7, 1., img[:,:,0])
img[:,:,0] = red
```

Losowe obrazki

```
img = np.zeros((512, 512, 3))
for i in range(img.shape[0]):
    for j in range(img.shape[1]):
        img[i, j] = np.random.random(3)
imgplot = plt.imshow(img)
plt.show()
```

Plan wykładu

pandas

pandas: *panel data*

```
import pandas as pd
```

Jakie dane

Różne, jakoś uporządkowane: csv, json, etc.

Podstawowe typy danych

Series

Seria danych, być może z etykietami (indeksami):

```
s = pd.Series(np.random.randn(5),
              index=['a', 'b', 'c', 'd', 'e'])
```

Podstawowe typy danych

Series

Seria danych, być może z etykietami (indeksami):

```
s = pd.Series(np.random.randn(5),
              index=['a', 'b', 'c', 'd', 'e'])
```

DataFrame

Dwuwymiarowa tablica, gdzie kolumny mają różne typy:

```
poniedziałek    1.61
wtorek          2.71
środa           3.14
```

Dane meteo

Instytut Meteorologii i Gospodarki Wodnej
`https://danepubliczne.imgw.pl/`

Dane meteo

Instytut Meteorologii i Gospodarki Wodnej
`https://danepubliczne.imgw.pl/`

Dane o opadach i stanie wód z 2010 roku: pliki csv/zip.

Źródło danych

```
import pandas as pd

pd.read_csv(<source>)
```

Źródło:

- nazwa pliku;
- URL;
- *file-like object*

File-like/file objects/streams

Obiekty implementujące metody podobne jak obiekty klasy `File`: `read()`, write() czy `close()`.

Przykład 1:

```
f = open("plik.txt", 'r')
# type(f): File
```

File-like/file objects/streams

Obiekty implementujące metody podobne jak obiekty klasy `File`: `read()`, write() czy `close()`.

Przykład 1:

```
f = open("plik.txt", 'r')
# type(f): File
```

Przykład 2:

```
f = io.StringIO("Jakiś")
f.write("tekst.")
f.close()
```

Rozpakowanie z zip'a

```
import pandas as pd
import zipfile

nagl_opady = ["Kod", "Nazwa", "Rok", "Mies",
        "Suma mies", "Status SUMM", "śnieg",
        "Status LDS", "Opad maks",
        "Status MAXO", "pierwszy", "ostatni",
        "pokrywa", "Status pom"]

with zipfile.ZipFile("2010_m_o.zip", 'r') as zf:
    with zf.open('o_m_2010.csv', 'r') as source:
        opady = pd.read_csv(source, encoding='iso8859-2',
                            header=None,
                            names=nagl_opady)
```

Wczytanie danych

```
nagl_hydro=["Kod", "Nazwa", "Rzeka", "Rok hydr",
       "Mies hydr",  "Ekstremum", "Stan wody",
       "przepływ", "Temp", "Mies kal"]

hydro = pd.read_csv("mies_2010.csv",
            encoding="iso8859-2",
            header=None, names=nagl_hydro)
```

Selekcja danych: Głogów

```
opady_gl = opady.loc[opady['Nazwa'] == 'GŁOGÓW']
hydro_gl = hydro.loc[(hydro['Nazwa'] == 'GŁOGÓW') &
           (hydro['Ekstremum'] == 3)]
hydro_gl = hydro_gl.sort_values(by=["Mies kal"])
```

Konwersja typów

Sprawdzenie typów kolumn:

```
dane.dtypes
```

Często typy kolumn są poprawnie rozpoznawane.
Można wskazać typy kolumn:

```
pd.read_csv("plik.csv", dtype={ "kolumna2" : 'int' })
```

Wyzwania

Kłopoty z konwersją:

- wstępnie sformatowane dane: "19 234";
- bogactwo zapisu daty.

```
strconv = lambda x : str(x.replace(' ', ''))

pd.read_csv("plik.csv",
    converters={ "kolumna2" : strconv }
```

Wykresy

```
import matplotlib.pyplot as plt

fig = plt.figure()
ax1 = fig.add_subplot(211)
opady_gl.plot(x='Mies', y='Opad maks', ax=ax1)
opady_gl.plot(x='Mies', y='Suma mies', ax=ax1)

ax2 = fig.add_subplot(212)
hydro_gl.plot(x='Mies kal', y='przepływ', ax=ax2)
plt.show()
```