Kurs rozszerzony języka Python

Numpy, SciPy

Marcin Młotkowski

20 listopada 2019

Plan wykładu

1. Wprowadzenie
   - NumPy
2. matplotlib
   - Wprowadzenie
   - Funkcje parametryczne
   - Wykresy animowane

Plan wykładu

Wstęp

Analiza, przetwarzanie i wizualizacja danych

Pakiety

- NumPy
- SciPy
- matplotlib
- Pandas

Narzędzia

IPython

Jupyter Notebook

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, 12.56]])

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

Dostęp do elementów (indexing)

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

- prawie tak jak już znamy: x[1,2] (Python: 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, 12.56]])

- prawie tak jak już znamy: x[1,2] (Python: 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, 12.56]])

- prawie tak jak już znamy: x[1,2] (Python: 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

$y > 5$
albo x.where(x $>$ 5)

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', dane, delimiter='|')
```

Plan wykładu

Co to jest

Narzędzie do rysowania wykresów. Bardzo różnych.

Prosty przykład

```
import numpy as np
import matplotlib/pyplot as plt

x = np.arange(-4*np.pi, 4*np.pi, 0.1)
y = np.sin(x)

plt.plot(x,y)
plt.show()
```

Inny przykład: histogram

```
x = np.random.randint(1,10, size=1000)
plt.hist(10)
plt.show()
```

Wykres temperatur w Jarocinie w styczniu

Jeszcze inny przykład: rzut ukośny

Wykresy funkcji parametrycznych

Przykład: figury Lissajous

$$x(t) = sin(a * t + \pi/2)$$
$$y(t) = sin(b * t)$$

gdzie a i b są pewnymi stałymi.

Jak animować wykresy

- wykres początkowy dla pewnych danych (wektory x i y) początkowych;
- aktualizacja: zmodyfikować x i y, narysować;
- wykorzystać obiekt klasy `matplotlib.animation.FuncAnim`

Początek

```
fig = plt.figure()
ax = plt.axes(xlim = (-2,2), ylim = (-2, 2))
```

Początek

```
fig = plt.figure()
ax = plt.axes(xlim = (-2,2), ylim = (-2, 2))

xdata, ydata = [], []
line, = ax.plot([], [])

def init():
    line.set_data([].[])
    return line,
```

Aktualizacja wykresu

```
def animate(i):
    t = 0.01*i
    x = np.sin(a * t + np.pi / 2.0)
    y = np.sin(b*t)
    xdata.append(x)
    ydata.append(y)
    line.set_data(xdata, ydata)
    return line,
```

I na koniec:

```
ani = animation.FuncAnimation(fig, animate, init_func=init,
                              frames=500, interval=50, blit=True)
plt.show()
```