Cvičenie na Numpy a Matplotlib

Jednoduché cvičenia

1

Vyskúšajte si nasledujúce príkazy a vysvetlite ich chovanie:

```
import numpy as np

x = np.linspace(0, 1, 10)
y = x**2 + x
z = y * x
w = np.pi * y[x > 0.3]
z = y[list(range(len(w)))]
z = np.sin(z)
z @ w.T
```

2

Napíšte program, ktorý vykreslí funkciu:

$$y(x) = \pi \sin(x^2) + \mathrm{e}^{-x^2} \quad \textbf{if} \quad x > 0 \quad \textbf{else} \quad \cos\left(\frac{x}{|x^2 - 3|}\right),$$

v intervale $x \in (-2, 2)$.

Skript, v ktorého môžete vychádzať (vykreslí jednoduchšiu funkciu):

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

x = np.linspace(-32, 32, 10000)
y = np.sin(np.pi * x)

plt.plot(x, y)
plt.show(block=True)
```

Obrázky

Obrázok je možné zapísať do 2D matice – každý prvok je intenzita pixela (v prípade farebného obrázku je 1 prvok RGB trojica). Zobraziť obrázok je možné pomocou funkcie `matplotlib.pyplot.imshow`.

Napr.:

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

# 100x100 pixelov
img = np.zeros((100, 100))

R = 40
for i in range(img.shape[0]):
    for j in range(img.shape[1]):
        r = i**2 + j**2
        if r < R**2:
            img[i, j] = r

plt.imshow(img)
plt.show()
```

Modifikujte program, tak aby vykreslil štvorec v strede obrázku.

Komplexné čísla a fraktály

Komplexné čísla sú v Pythone štandardný dátový typ. Komplexné číslo je možné vytvoriť rôznymi spôsobmi:

```
c = 3j + 1
c = complex(real=1, imag=3)
c = complex(1, 3)
```

Komplexné číslo má dve časti:

```
c = 3j + 1
c.imag == 3
c.real == 1
```

a funkcia `abs` na komplexné číslo funguje presne tak, ako v matematike:

```
abs(3j + 4) == 5
```

Táto úloha je zo stránky scipython.com. Využijeme komplexné čísla pre tvorbu fraktálu – konkrétne Juliovej množiny. Algoritmus pre Juliovu množinu je nasledujúci:

- Začnite s bodom z_0 na komplexnej rovine:

$$\mathrm{Re}(z_0) \in (-1.5, 1.5) \qquad \mathrm{Imag}(z_0) \in (-1.5, 1.5)$$

- Iterujte podľa vzťahu $z_{n+1} = z_n^2 + c$, kde c je komplexná konštanta.
- Ukončite iteráciu pokiaľ počet iterácií prevýši $n_{\max}$ alebo $|z_n| > z_{\max}$.
- Na obrázku, ktorý reprezentuje komplexnú rovinu, zafarbite pixel prislúchajúci z_0 farbou podľa počtu iterácií.

Začnite so vstupnými parametrami:

- $c = -0.1 + 0.65\mathrm{i}$
- $z_{\max} = 10$
- $n_{\max} = 500$
- Veľkosť obrázku 500×500 pixelov.

Vašou odmenou by mal byť nádherný obrázok. Výpočet by mal trvať pár sekúnd.

Fitovanie

V štúdijných materiáloch nájdete datový súbor s dvoma stĺpcami. Data som získal spojením dvoch datasetov z owid-datasets. (Majú na tie datasety veľmi peknú stránku.)

Prvý stĺpec je priemerný index šťastia – číslo od 0 do 10, druhý stĺpec je priemerná spotreba energie na osobu v jednotkách kcal/osoba/deň.

- Načítajte data zo súboru pomocou `numpy.genfromtxt` (alebo inak, ale `genfromtxt` je najjednoduchší spôsob.).
- Predpokladajte lineárnu závislosť medzi indexom šťastia a zjedeným jedlom a nafitujte priamku na data.
- Vykreslite graf s datami a priamkou. Koľko kalórií by ste museli zjesť, aby ste boli úplne šťastnou osobou?

K fitovaniu použite funkciu `scipy.optimize.curve_fit`. Jej použitie je ukázané v nasledujúcom príklade:

```
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt
import numpy as np

def model(x, a, b):
    return a * x + b

x = np.array([1, 2, 3, 4, 5])
y = np.array([2, 6, 4, 8, 10])

param_opt, param_covariance = curve_fit(model, x, y)
param_std_dev = np.sqrt(np.diag(param_covariance))

a = round(param_opt[0], 3)
b = round(param_opt[1], 3)

print("Fitted param 'a': {} +/- {}".format(
    a, param_std_dev[0]))
print("Fitted param 'b': {} +/- {}".format(
    b, param_std_dev[1]))

plt.scatter(x, y, label='data')
plt.plot(x, model(x, a, b),
    color='black',
    label=r'${} \cdot x + {}$'.format(a, b)
)
plt.legend()
plt.show()
```