

Postup ručného prekladu Matlab -> Python

Predpoklady

Nutné

- nainštalovaný Python3, testovaná bola verzia 3.6
- nainštalované Python balíky `numpy` a `matplotlib` (oba dostupné cez `pip`, `numpy` je často tiež dostupný cez systémový package manager)

Odporúčané

- nainštalovaný Pycharm (stačí Community edition), ideálne s Matlab pluginom

Základné pojmy

Blok kódu v Matlabe

Začína kľúčovými slovami ako `if`, `while`, `function`.

Končí kľúčovým slovom `end`.

Bloky môžu byť do seba vnorené, každý začiatok spôsobuje vnorenie, koniec vynorenie, blok končí, keď sa na úrovni vnorenia spôsobenej jeho začiatkom objaví koniec.

Napr:

```
if a>b
    % nejaký kód
    if b>c
        % vnoreny blok
    end
    % ďalší kód
end
% kód za blokom
```

Blok kódu v Pythone

Začína kľúčovými slovami ako `if`, `while`, `def`.

Končí, keď sa vyskytne kód rovnako alebo menej odsadený ako začiatok príslušného bloku.

Bloky môžu byť vnorené, úroveň vnorenia je vyjadrená dĺžkou odsadenia, ktorá musí byť striktné väčšia ako vonkajší blok, a konzistentná v rámci bloku, okrem doň vnorených blokov.

Štandardne sa používa odsadenie 4 medzery od rodičovského bloku.

```
if a > b:
    # nejaky kod
    if b > c:
        # vnoreny blok
    # dalsi kod
# kod za blokom
```

Preklad

Vytvorenie Pythonového súboru

Najprv potrebujeme vytvoriť súbor, do ktorého budeme preklad písať.

Pre súbor `nazov.m` vytvoríme `nazov.py`. V Pycharme sa toto robí cez:

1. Pravý klik na priečinok
2. **new**
3. **Python file**
4. Vyplniť meno, ponechať zvolené **Python file**
5. Enter

Môže byť vhodné na prvý riadok súboru dať text `#!/bin/env python3` aby ho systém bol schopný priamo spustiť.

Po vynechanom riadku dáme `import numpy as np`, čo je knižnica potrebná pre veľa matematickej práce v Pythone.

Následne môžeme vložiť obsah pôvodného súboru, ktorý budeme postupne prerábať.

Preklad samotný

Preklad robíme postupne po blokoch, a to tak, že:

2. Ak sú na konci riadka tri bodky, ďalší riadok sa považuje za pokračovanie tohto, nahradíme ich teda medzerou.
3. Podľa typu začiatku `if/else/while/for/function` prepíšeme začiatok ako je uvedené v príslušných súboroch `task2`, príslušne upravujúc koniec bloku, teda:

```
if podmienka
    % obsah1
```

```

else
    % obsah2
end

while podmienka
    % obsah3
end

for premmenna = 1:limit
    % obsah4
end

function vystup = nazov(vstupy)
    % obsah5
end

na
if podmienka:
    # obsah1
else:
    # obsah2

while podmienka:
    # obsah3

for premenna in range(1,limit+1):
    # obsah4

def nazov(vstupy):
    # obsah5
    return vystup

```

4. Preložíme obsah bloku, teda

- vnorené bloky preložíme rekurzívne
- komentáre začínajú %, to zmeníme na Pythonovské #, ktoré má byť nasledované medzerou
- aritmetické výrazy zostávajú prevažne rovnaké, len mocniny sa namiesto ^ píšú **, operátor ' sa nahradí .conj().T a namiesto operátorov začínajúcich ., ako napríklad .*, sa použijú obyčajné *, tiež môže byť vhodné výraz uzátvorkovať
- príkazy tak, ako je uvedené v príslušných súboroch task2, pričom pre func2str, printf, surfc, plot, fplot a size skopírujeme definíciu z matlaubequiv.py za importy v našom súbore, a buď necháme Pycharm, nech ponúkne potrebné importy, alebo ich tiež skopírujeme. Pre

fplot treba nakopírovať aj definíciu plot

- navyše použitie surfc vyžaduje `from mpl_toolkits.mplot3d import Axes3D`

```
fprintf("format",argumenty)

func2str(funkcia)

linspace(from,to,count)

[X,Y] = meshgrid(a,b)

pause(cas)

zeros(pocet)

rand()

na

printf("format",argumenty)

func2str(funkcia)

np.linspace(from,to,count)

X,Y = np.meshgrid(a,b)

plt.pause(cas)

np.ma.zeros(pocet)

random.random()

kde pause vyžaduje import matplotlib.pyplot as plt, použitie sqrt
vyžaduje from math import sqrt a random potrebuje import random

Narozdiel od Matlabu na konci riadkov netreba ; a odporúča sa ich
odstrániť.
```

- preložíme definície lambda funkcií, nezabúdajúc na preloženie výpočtu

```
@(argument) vypocet

na

lambda argument: vypocet
```

- preložíme ostatné grafové funkcie ako je popísane v task2/Plotting, je vyžadované `import matplotlib.pyplot as plt`, teda:

```

title(nazov)

legend(nazov1,nazov2)

contour(X,Y,Z)

figure(nazov);
na
plt.title(nazov)

plt.legend([nazov1,nazov2])

plt.contour(X,Y,Z)

plt.figure(nazov)

```

5. spustíme výsledný program, a nájdeme miesta, kde sa graf má zobraziť, resp. z grafu má odstrániť predchádzajúci obsah porovnaním s pôvodnou Matlab verziou, potom zobrazenie sa robí príkazom `plt.show()` a premazanie `plt.clf()`
6. Môžeme nechať Pycharm preformátovať súbor, aby zodpovedal štylistickým štandardom pre Python, na toto sa používa klávesová skratka `Ctrl + Alt + L`