

Postup rucneho prekladu Matlab -> Python

Predpoklady

Nutne

- nainstalovany Python3, testovana bola verzia 3.6
- nainstalovane python baliky `numpy` a `matplotlib` (oba dostupne cez `pip`)

Odporucane

- nainstalovany Pycharm (staci community edition), idealne s matlab pluginom

Zakladne pojmy

Blok kodu v Matlabe

Zacina klucovymi slovami ako `if`, `while`, `function`.

Konci klucovym slovom `end`.

Mozu byt do seba vnorene, kazdy zaciatok sposobuje vnorenie, koniec vynorenie, blok konci ked sa na urovni vnorenia sposobenu jeho zaciatkom objavi koniec.

Napr:

```
if a>b
    %nejaky kod
    if b>c
        %vnoreny blok
    end
    %dalsi kod
end
%kod za blokom
```

Blok kodu v Pythone

Zacina klucovymi slovami ako `if`, `while`, `def`.

Konci ked sa vyskytne kod rovnako alebo menej odsadeny ako zaciatok prislusneho bloku.

Mozu byt vnorene, uroven vnorenia je vyjadrena dlzkou odsadenia ktora musi byt striktnie vecsia ako vonkajsi blok a konzistentna vramci bloku okrem don vnorených blokov.

Standardne sa pouziva odsadenie 4 medzery od rodicovskeho bloku.

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

Preklad

Vytvorenie pythonoveho suboru

Pre `nazov.m` vytvorime `nazov.py`. V Pycharme sa toto robi cez:

1. pravy klik na priecinok
2. `new`
3. `Python file`
4. vyplnit meno, ponechat zvolene `Python file`
5. Enter

Moze byt vhodne na prvý riadok suboru dat text `#!/bin/env python3` aby ho system bol schopny priamo spustit

Po vynechanom riadku dame `import numpy as np`, kedze je takmer garantovane ze to bude nutne

Preklad samotny

Preklad robime postupne po blokoch, a to tak, ze:

2. podla typu zaciatku `if/else/while/for/function` prepiseme zaciatok ako je uvedene v prislusnych suboroch `task2`, prislusne upravujuc 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

```

3. prelozime obsah bloku, teda

- vnorene bloky rekurzivne
- komentare zacinaju %, to zmenime na pythonovske # pricom podla konvencie ma byt nasledovane medzerou
- aritmeticke vyrazy zostavaju prevazne rovnake, len mocniny sa namiesto .^ pisu **, tiez nemusi byt nevhodne vyraz uzatvorkovat
- prikazy tak ako je uvedene v prislusnych suboroch task2, pricom pre func2str/printf/surfc/fplot skopirujeme definiciu z matlabequiv.py za importy v nasom subore, a bud nechame Pycharm nech ponukne potrebne importy, alebo ich tiez skopirujeme, navyse pouzitie surfc vyžaduje from mpl_toolkits.mplot3d import Axes3D potom teda:

```
fprintf("format",argumenty)
```

```
func2str(funkcia)
```

```
linspace(from,to,count)
```

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

```
pause(cas)
```

```
size(pole,1)
```

```

zeros(pocet)

rand()

na

printf("format",argumenty)

func2str(funkcia)

np.linspace(from,to,count)

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

sleep(cas)

len(pole)

np.ma.zeros(pocet)

random.random()

kde sleep vyzaduje from time import sleep, pouzitie sqrt vyzaduje from
math import sqrt a random potrebuje import random

Narozdiel od Matlabu na konci riadkov netreba ; a odporuca sa ich
odstranit

    • prelozime definicie lambda funcii, nezabudajuc na prelozenie vypoctu
@argument) vypocet

na

lambda argument: vypocet

    • prelozime ostatne grafove funkcie ako je popisane v task2/Plotting,
      je vyzadovane import matplotlib.pyplot as plt, teda

title(nazov)

plot(x,y)

legend(nazov1,nazov2)

contour(X,Y,Z)

figure(nazov);

na

```

```
plt.title(nazov)

plt.plot(x,y)

plt.legend([nazov1,nazov2])

plt.contour(X,Y,Z)

plt.figure(nazov)
```

4. spustit vysledny program, a najst miesta kde sa graf ma zobrazit, resp z grafu ma odstranit predchadzajuci obsah, porovnanim s povodnou matlab verziou, potom zobrazenie sa robi ako `plt.show()` a premazanie `plt.clf()`