Tartalomjegyzék

- 1 Megjegyzések
- 2 8. Gyakorlat
  - 2.1 Összefoglalás
  - 2.2 Feladatok
  - 2.3 Megjegyzés
- 3 9. Gyakorlat
  - 3.1 Összefoglalás
  - 3.2 Feladatok
  - 3.3 Megjegyzés
- 4 10. Gyakorlat
  - 4.1 Összefoglalás
  - 4.2 Feladatok
- 5 11. Gyakorlat
  - 5.1 Összefoglalás
  - 5.2 Feladat

Megjegyzések

Az összefoglalást a gyakorlatokból írom csak, az elõadáson tanult elmélet nem igazán fog megjelenni benne.

Minden feladathoz legalább 3 letölthetõ file van: egy komment nélküli, egy kommentezett, és egy kevésbé szájbarágósan kommentezett.

8. Gyakorlat

Összefoglalás

A gyakorlat célja az emlékeztetõ volt az info1-ben tanult sage és python használatra. Az ismétlés részben összefoglaltam ezeket, ezt most nem írnám le itt mégegyszer.

Feladatok

A feladatok célja az volt, hogy felfrissítse az emlékezeteteket a listák, ciklusok, stringek és szótárakról. A feladatok tényleg nagyon egyszerûek, jó ujjgyakorlat, ha megnézitek és megértitek a megoldásokat, ezután átnézitek a 9. gyakot, majd megpróbáljátok megoldani magatoktól a megnézett feladatokat (hogy ne az legyen, hogy csak a rövidtávú memóriátokban maradtak meg a megoldások ötletei). Az utolsó két feladatot nem oldottuk meg.

Kiegészítés 1: alap, komment, minkomment

Lista 1: alap, komment, minkomment

Lista 2: alap, komment, minkomment

Kiegészítés 2: alap, komment, minkomment

String 1: alap, komment, minkomment

String 2: alap, komment, minkomment

Megjegyzés

Nagyon fontos volt az elõadásról, hogy vannak objektumok melyeket nem lehet csak úgy =-vel lemásolni, erre egy példa:

```
l = [1, 8, 5]
l2 = l
l2[1] = 7
print l # [1, 7, 5]
```

Amint látjátok az l lista is megváltozott, pedig csak az l2 elemét változtattuk meg, ez azért van, mert az = jellel igazából azt mondtuk, hogy l2-vel érhessük el az l listát is.

A következõképpen lehet pl helyesen másolni egy listát (egy szintig).

```
l = [1, 8, 5]
l2 = l[:]
l2[1] = 7
print l # [1, 8, 5]
```

Az l[:]-al lekérem az l azon részlistáját, amely a 0. indexû elemtõl kezd, és az utolsóig tart, tehát a teljes listát. DE ez egy másolat, nem maga az l lista, tehát nem lesz vele gond.

9. Gyakorlat

Összefoglalás

A gyakorlat mindenféle input / output dolgokról szólt. Ezeket szerintem elég jól leírtam a gyakorlaton, így megint nem ismételném meg: link

Az osztályokról nem igazán volt idõm beszélni a 9. gyakorlatokon, a következõ résznél kezdem.

Feladatok

A feladatok elég közvetlenek voltak:

Hatványok kiírása : alap, komment, minkomment

Telefonkönyv : alap, komment, minkomment

Telefonkönyv file: alap, komment, minkomment

Telefonkönyvbõl olvasás : alap, komment, minkomment, bemenet

Megjegyzés

Egy tanulsága az volt a gyakorlatnak, hogy stringeket raw_input-al olvasunk be, míg minden mást input-al. Mivel az input python szintaktikának megfelelõ bemenetet vár, szóval ha string-et akarnánk adni neki, akkor "http://wiki.math.bme.hu-k közé kellene tenni.

Továbbá azt is megtanultuk, hogy nem csak listákat és szótárakat lehet bejárni for ciklussal, hanem akár file-okat is.

10. Gyakorlat

Összefoglalás

Ezen a gyakorlaton kezdtünk el osztályokkal foglalkozni. Az osztályokról van itt egy leírás, valamint az elõadásokon is meg lehet találni. Csak pár szóban foglalnám össze õket:

- Osztályokat tekinthetjük úgy, mintha egy új típust hoznánk létre, mondjuk a komplex számok típust.
- Amikor létrehozunk egy adott osztályba tartozó (adott osztály típusú) változót, azt objektumnak hívjuk.
- Az objektumoknak lehetnek adattagjaik, ezek változók, melyek az objektumhoz tartoznak. Valamint lehetnek metódusai, melyek függvények és az osztályában vannak definiálva.
- Léteznek beépített metódusok, melyek adott parancsokra futnak le, ilyen például az __init__ mely akkor fut le amikor létrehozunk egy objektumot.
- Az osztályok metódusainak elsõ paramétere mindig a **self**, ami azt az objektumot tárolja amin meghívtuk a metódust.

Feladatok

A gyakorlat elsõ feladata azt mutatta be, hogy mi a különbség az osztály szintû és az objektum szintû adattagok között. Amikor **A.v1**-nek adunk értéket, azzal létrehozunk egy osztály szintû adattagot **v1** néven. Mikor egy **A** típusú objektumnak kérjük le a v1 adattagját, de neki nincs ilyenje, akkor megnézi hogy létezik-e ilyen néven osztály szintû adattag, ezt írja ki ilyenkor. Viszont ha adunk az objektumnak objektum szintû adattagot, akkor már ezt fogja használni és nem az osztály szintût.

A második feladatról a 11. gyakorlat összefoglalásában írok.

11. Gyakorlat

Itt írok a 10. gyakorlat második feladatáról is.

Összefoglalás

A gyakorlat célja egy olyan Matrix osztály írása volt, amely az elvárásoknak megfelelõen mûködik, össze lehet adni mátrixokat, szorozni stb.

A 10. gyakorlaton megírtuk a Matrix alapjait:

- minden Matrix objetumnak van egy **A** adattagja mely egy listákat tartalmazó lista, úgy kell rá tekinteni, hogy ez a lista tartalmazza a mátrix sorait, melyek szintén listák és tartalmazzák a sorok elemeit.
- egy **n** adattagot is tartalmaznak, mely a mátrix dimenzióját tárolja (kényelmi okokból, hisz ezt len(self.A)-val bármikor lekérhetnénk)
- amikor létrehozunk egy mátrixot csupa nullára állítja az elemeit, létrehozáshoz meg kell adnunk a dimenzióját (négyzetes mátrixokat csinálunk csak), ha nem adtunk dimenziót akkor alapértelmezetten 5x5-ös lesz.
- a különbözõ mûveletek metódusai (ez minden osztályra igaz, nem csak speciálisan a mátrixra):
  - __add__: összeadás
  - __sub__: kivonás
  - __mul__: szorzás
  - __div__: osztás
- az osztást természetesen nem, de a többit megírtuk.

Továbbá ezen a két gyakorlaton ismertük meg a hibakezelést, melyre egy egyszerû példa:

```
l = [1, 8, 5]
try:
    print l[1]
except:
    print "hiba"
```

A fenti kód, kiírja a 8-ast és nem írja, hogy hiba, mert a hibakezelés úgy mûködik, hogy ha egy try blokkban hiba történt, akkor ugrik a legközelebbi megfelelõ except blokkra és lefuttatja ami benne van. A következõben megtörténik a hiba:

```
l = [1, 8, 5]
try:
    print l[5]
except:
    print "hiba"
```

Túlindexeltük a listát így hiba történt és azt írja ki szimplán, hogy hiba. Egy utolsó példa erre:

```
l = [1, 8, 5]
try:
    print l[5]
    print "Ezt nem irja ki"
except:
    print "hiba"
```

Ez arra jó példa, hogy a try blokkban a hiba utáni sorokat nem futtatja le, így az "http://wiki.math.bme.huEzt nem irja ki"http://wiki.math.bme.hu sort tényleg nem írja ki, csak azt hogy "http://wiki.math.bme.huhiba"http://wiki.math.bme.hu.

- Hibakezelést nem minden metódusában alkalmaztunk a Matrix osztálynak, többnyire csak a 10. gyakorlatba tartozókban, a gyakorlatok véges ideje miatt.

- A **det**, **subMatrix** és **random** metódusokat nem feltétlen kell érteni, ezeket a gyakorlatokon se beszéltük meg, de ha valaki megérti õket, azzal nagyon jó gyakorlást végez a ZH-ra.

Feladat

Matrix: alap, komment minkomment