LINUX RENDSZERFEJLESZTÉS ÉS ÜZEMELTETÉS

6. forduló

ONE IDENTITY by Quest

A kategória támogatója: One Identity - Quest Hungary

Ismertető a feladathoz

Ez a forduló kifejezetten a rendszerfejlesztésre fókuszál!

Felhasznált idő: 30:00/30:00 **Elért pontszám:** 0/20

Indítás utáni csatolmányok

1. feladat 0/5 pont

Kaptál egy könnyű kis feladatot az egyik fejlesztőtől, miszerint kellene írni egy osztályt, aminek 1 példány létrehozása független a példányok azonosítójától. Rögtön eszedbe jut egy megoldás, de kollégád szerint ebben az esetben nem ez lesz a jó, mert szükség lenne shared state -re is a többi példány kezelésére. Kezd gyanús lenni a dolog: lehet, hogy mégsem 1 instance kell kezelni? Vagy a példányoknak ugyanaz a shared state-je van? Végül sikerül implementálni egy skeleton-t.

Melyik pattern végzi el a feladatot?

Válasz

- ○ Singleton
- ○ Facade
- ○ Factory
- ◉ Borg
  Ez a válasz helyes, de nem jelölted meg.
- ○ Cascade

Magyarázat

Az első ötlet valószínűleg a Singleton pattern lenne, amivel garantáljuk, hogy csak 1 instance létezik futási időben. Mivel kellenek shared_state -ek, ahogy a kódban is láthatjuk, ez inkább a monostate, más néven Borg pattern. Ezzel lehetőségünk van az instance-ok együttes kezelésére is. “We are the Borg.”

2. feladat 0/15 pont

Péntek, délután fél5... a főnöködtől kaptál egy scriptet, review-ra. Hát, már nem sok kedved van hozzá, de hát a munka az munka. Ahogy futtatod a kódot, úgy érzed, valami nem kerek, így elkezded nézni is a kódot, és feltűnik az alábbi rész. Gyanakszol a file-ra, de hát abban vannak adatok, ahogy a print is kiírja. Így a normalize függvényt kezded nézni, és írsz egy rövid tesztet is egyből alá:

```
visits = [10, 15, 25, 50]
percentages = normalize(visits)
print("is normalize good: ",percentages)
```

De ez is ad eredményt, sőt többször is sikerül kiíratni. Akkor mégsem ez lenne a baj? **Vajon rájössz, hogy mi történt?**

Válasz

- ○ mégiscsak üres a file, amiben az értékek vannak
- ○ a generátor nincs paraméterezve, nem hívódott meg a StartGenerator makró
- ○ az iterátort nem használtuk
- ○ nem lehet végig iterálni a file-ban lévő értékeken
- ● az iterátor StopIteration kivételt dobott és nem használható mégegyszer

  Ez a válasz helyes, de nem jelölted meg.

Magyarázat

Egy iterátor csak egyszer hozza létre az eredményeit. Ha olyan iterátoron vagy generátoron keresztül iterálunk, amelyik már StopIteration kivételt dobott, akkor másodszorra nem fog eredményt adni. Azt is mondhatjuk, hogy az iterátor kimerült. Ciklusoknál, list konstruktornál, illetve a python standard könyvtárának sok más függvénye esetében a normál müködés az, hogy StopIteration kivétel dobódjon. Ezek a függvények nem tudnak különbséget tenni a kimenet nélküli iterátor és a kimenettel rendelkező iterátor között.

```
def normalize(get_iter):
    total = sum(get_iter())
    result = []
    for value in get_iter():
        percent = 100 * value / total
        result.append(percent)
    return result

def read_visits(data_path):
    with open(data_path) as f:
        for line in f:
            yield int(line)

nums = read_visits('./my_numbers.txt')
print("after READING: ", list(nums))            # miert tudom kiiratni
print("after READING: ", list(nums))            # meg mindig ures lista

percentages = normalize(lambda: read_visits('./my_numbers.txt'))           # ez mar mukodik!!!
print("NORMALIZING: ", list(nums))              # meg mindig ures lista
print("percentage: ", percentages)              # ez is mukodik
```

Egy lehetséges megoldás, ha a normalize függvénynek get_iter -t adunk át, ez kerül a ciklusba, illetve a normalize esetében lambda-val hívjuk meg direkt a read_visits függvényt.

Megjelenés

Világos