el?z? fel következ?

Tartalomjegyzék

Feladatok

Bevásárlás

A `hazi` rendszerben az **`shopping`** feladat.

Enciklopédia

A `hazi` rendszerben az **`enciclopedia`** feladat.

Osztályzás

A vizsga kijavítása után azt szeretnénk hogy a hallgatók megtudhassák az internetr?l a eredményüket, de nem akarjuk mindenkiét kitenni egy nagy táblázatban, mert lehet hogy van aki nem akarja hogy a többiek tudják hogy teljesített. Ezért létrehoztunk egy honlapot, ahol beírhatja bárki a NEPTUN-kódját, és a honlap kiírja az osztályzatát.

Ennek részeként írjunk egy python függvényt, ami a százalékos eredményekb?l és NEPTUN-kódból ki tudja adni a megfelel? hallgató osztályzatát. A függvény neve legyen `osztalyzas`, két paramétere legyen

- *szazalekok*, egy python szótár (dict), ami minden NEPTUN-kódhoz tartalmazza az adott hallgató százalékos eredményét.
- *neptunkod*, a lekérdezett NEPTUN-kód.

A függvény egy 1 és 5 közötti számmal térjen vissza, a hallgató osztályzatával. A jegyekhez szükséges elért százalék rendre 40, 55, 70, 85, a 2-eshez, 3-ashoz, 4-eshez illetve 5-öshöz.

Ötösök

Miután kijavítottuk a vizsgát, és megvannak a százalékos eredmények, szeretnénk kisz?rni bel?le az ötösök listáját, hogy megdicsérhessük ?ket az el?adáson.

Írjunk egy python függvényt, ami ki tudja sz?rni az eredményekb?l az ötösöket. A függvény neve legyen `otosok`, és egy paramétere legyen

- *szazalekok*, egy python szótár (dict), ami minden NEPTUN-kódhoz tartalmazza az adott hallgató százalékos eredményét.

A függvény egy listát adjon vissza, amiben az ötöst elért hallgatók NEPTUN-kódjai vannak. Akkor ötös egy hallgató, ha legalább 85 százalékot elért.

Gólkirály

Koppány és barátai minden hétvégén játszanak egy barátságos focimeccset az egyik helyi focipályán. Szeretnék tudni az év végén hogy ki rúgta közülük a legtöbb gólt, hogy egy kicsit megünnepelhessék az illet? teljesítményét. Koppány megkért minket hogy írjunk egy python függvényt, ami segít a meccs végén elkönyvelni valaki góljait.

A függvény neve legyen `golkiraly`, és három paramétere legyen

- *eredmenyek*, az év folyamán eddig l?tt gólok száma, szótár formájában, amiben mindenkinek a nevéhez hozzá van rendelve hogy eddig mennyit l?tt
- *jatekos* és *darab* az elkönyvelend? gólok löv?jének neve, és a góljainak száma

A függvény adja vissza az *eredmenyek*-nek megfelel?en módosított változatát, azaz ha eddig nem szerepelt benne játékos, akkor most szerepeljen benne *darab*-bal, ha eddig is szerepelt benne, akkor pedig növeljük meg a hozzá könyvelt gólok számát *darab*-bal.

Leltár frissítés

Bemenetként két dolgot kapunk:

- *leltar* egy lista, aminek elemei egy bolt leltárában található tárgyak, a sorszámukkal jelképezve. Ugyanaz a sorszám többször is szerepelhet, ha abból a tárgyból több is van raktáron.
- Némely raktáron lev? tárgynak megváltozott a sorszáma. *ujszam* egy szótár, ami a megváltozott sorszámokat tartalmazza, a régihez rendelve az újat. Tehát pl. ha egy tárgy sorszáma változott, régen 3 volt, és most 4 lett, akkor azt a "http://wiki.math.bme.hu{3: 4}"http://wiki.math.bme.hu szótár jelképezné.

Adjuk vissza a kijavított leltár listát, amiben minden átszámozott tárgyat átszámoztunk. (A sorrendet nem szabad módosítani.)

Segítség

- Szótárakról leírás az el?z? féléves el?adásokból itt.
- Le lehet ellen?rizni hogy egy elem szerepel-e a szótárban kulcsként a `x in d` kifejezéssel.

- Egy lehetséges megoldás végigmenni a listán és (egy elágazást használva) csak azokat az elemeket átírni amik változnak.
- VAGY Listaértelmezéssel is meg lehet oldani. Listaértelmezésen belül m?ködik az `x if c else y` kifejezés.

Közel

A `kozel` nev? függvény megírása a feladat. A függvény paramétere:

- *l*, egy lista, ami *n*-dimenziós egységvektorokat tartalmaz.

Úgy lehet megállapítani hogy mennyire kicsi a szög két *n*-dimenziós egységvektor között, hogy a skalárszorzatukat vesszük, és ha kisebb a szög, nagyobb a skaláris szorzat. (A skaláris szorzatnak az inverz koszinusza lenne a tényleges szög, de most ennél a feladatnál elég a skalárszorzattal foglalkozni.) Az *l* listában megadott egységvektorok közül keressük meg a két legközelebb lev?t, és adjuk vissza a skaláris szorzatukat.

A `kozel` függvény megírásához segít ha megírjuk külön a `skalar_szorzat` nev? függvényt, úgyhogy el?ször írjuk meg azt. Ennek a függvénynek a paraméterei:

- *a* és *b*, két vektor, listák formájában.

Adjuk vissza a skaláris szorzatukat, ami úgy kapható meg, hogy mind az *n* dimenzióban összeszorozzuk a megfelel? koordinátájukat, és ezek összegét vesszük.

Segítség

- Ajánlom hogy a </tt>skalar_szorzat</tt> függvényt teszteljétek valamelyik el?adáson tanult módszerrel, miel?tt megpróbáltok nekiállni a `kozel` függvénynek.

Teszteléshez használható, hogy tudjuk, hogy mer?leges vektorok (pl. `[0, 1]` és `[1, 0]`) skalár szorzata 0, egy irányba mutató vektorok skalár szorzata pedig a hosszuk szorzata.

- A `kozel`-ben egy kétszeres ciklus kell, hogy minden párt megvizsgáljunk, és megtaláljuk a maximálisat.
- Mivel egységvektorokról van szó, a skaláris szorzat értéke mindig legalább -1.

Prímszám-e

Készítsünk egy szótárat, melyben a prímszámok vannak, mint kulcsok 2-t?l 100-ig, az értékek pedig igaz-hamis értékek, hogy az adott prím Mersenne-prím-e.

Ehhez 2 segéd függvényt érdemes írni:

- Visszaadja 2-t?l n-ig a prímek listáját (ezt már volt korábban)
- Megmondja egy prímszámról, hogy Mersenne prím-e: adjunk a számhoz 1-et, majd azt, hogy 2 hatvány-e tesztelhetjük úgy, hogy vizsgáljuk milyen maradékot ad 2-vel osztva, ha 0-t, osztjuk 2-vel, és vizsgáljuk tovább.

el?z? fel következ?