el?z? fel következ?

Tartalomjegyzék

Feladatok

Kezd?

Átlaghoz legközelebbi

Írjunk függvényt, mely a kapott valós számokat tartalmazó listában megkeresi a lista elemeinek átlagához legközelebbi számot és ezzel tér vissza.

n hosszú növekv? részek

Írjunk függvényt, mely kap egy listát és egy egész számot (n). Megkeresi az összes olyan n hosszú részintervallumot, amire igaz, hogy az elemei növekv? sorrendben vannak. Ezeket a listákat beteszi egy f? listába és ezt adja vissza. Segítség: bontsuk részfeladatokra!

F?

Névkonfliktus

Egy házibuliban sok új ismeretlen emberrel találkozunk, ezért felírjuk a neveiket egy listába. Ha két új ismer?sünket is ugyanúgy hívják, abból probléma lehet, ezért szeretnénk egy `python` függvényt, ami szól ilyen esetben.

A függvény neve legyen `nevkonfliktus`, és egy paramétere legyen:

- *nevek*, egy lista, amiben a buliban lev? emberek beceneve szerepel.
- A függvény `True`-val térjen vissza ha van két ember akinek ugyanaz a beceneve, és `False`-al egyébként.

Segítség:

Vigyázzunk, hogy akkor ne jelezzünk ha valakinek önmagával ugyanaz a neve (ami mindenkire igaz), hanem csak akkor ha két különböz? embernek ugyanaz a neve.

Kiejtés

A föld bizonyos nyelvei a magyarok számára elég nehezen beszélhet?ek, például azért, mert az ? szavaikban sokkal több a mássalhangzó mint a mienkben. Magyarok számára például nehéz lehet kiejteni a szlovák *zmrzlina* szót, ami magyarul azt jelenti hogy *fagylalt*.

Írjunk `python` függvényt, ami megpróbálja megállapítani hogy egy szót nehéz-e kiejteni, a benne lev? mássalhangzók száma alapján.

- A függvény neve legyen `kiejtes`,
- egy paramétere legyen: *szo*, egy idegen nyelv? szó. Az egyszer?ség kedvéért az itt leírt változat csak angol kis bet?ket fog tartalmazni.
- A függvény `True`-val térjen vissza, ha legalább kétszer annyi mássalhangzó van a szóban mint magánhangzó, és `False`-al egyébként. A lehetséges bet?k közül az *a, e, i, o, u* számít magánhangzónak.

Fogások

A diétánk szigorúan meghatározza hogy hány kalóriás ebédet kell fogyasztanunk, így kicsit gondban vagyunk amikor étteremben eszünk és nem magunknak állítjuk össze az ételt. Szerencsére az étlapon mindenhez oda van írva a kalóriaértéke, így egy kicsit könnyebb a dolgunk. Azonban még mindig ki kell választanunk a megfelel? els? és második fogást ami megfelel a diétánknak.

A függvény neve legyen `fogasok`, és három paramétere legyen:

- *kaloria*, egy természetes szám, hogy hány kalóriát kell fogyasztanunk
- *elso*, egy lista, amiben a különböz? lehetséges els? fogások kalóriaértéke van, tehát természetes számokat tartalmaz
- *masodik*, szintén egy lista, ami a második fogások kalóriaértékét tartalmazza.
- A függvény `True`-val vagy `False`-al térjen vissza attól függ?en, hogy lehet-e olyan els? és második fogást választani amelyek összege pontosan a *kaloria*.

Pascal

A Pascal-háromszög a binomiális együtthatók háromszög formában való elrendezése. Részletes leírás található pl. a magyar wikipédián. A lényeg az, hogy az *n.* sor *k.* eleme az az *"http://wiki.math.bme.hun alatt a k"http://wiki.math.bme.hu* binomiális együttható, és minden elem a felette lev? kett? összege. Írjuk meg a `pascal` nev? függvényt, ami visszaadja a Pascal-háromszög els? néhány sorát listák listájaként. A függvény paramétere:

- *n*, hogy hány sort számoljunk ki

Így tehát pl. `pascal(4)`-nek a következ?t kell visszaadnia:

```
[[1],
[1, 1],
[1, 2, 1],
[1, 3, 3, 1]]
```

Az elemek kiszámolásához ne a binomiális együttható általános (faktoriálisos) képletét használjuk, hanem azt, hogy a felette lev? két elem összege!

Cserebere

A bemenetünk egyrészt egy szó a *szo* nev? változóban, másrészt cserék egy sorozata, amit végre kell hajtani a szón, a *cserek* nev? változóban. Így *szo* egy karakterlánc, *cserek* pedig egy lista, melynek minden eleme egy pár, ami a csere helyét, és az új bet?t tartalmazza.

Ezeket a cseréket kell elvégezni a *szo*-n és az így kapott sztringet visszaadni.

Név generátor

Egy olyan számítógépes játékon dolgozunk, amiben rendszeresen találkozunk más, a számítógép által megszemélyesített, karakterekkel. Ezeknek a karaktereknek számítógép véletlenszer?en választ nevet, azonban szeretnénk elkerülni hogy ugyanazt a nevet sokszor lássa a játékos. Ezért ahelyett hogy egy listányi nevet megadtunk volna a játék készítésekor, inkább külön-külön egy listányi el?tagot és utótagot adtunk meg, hogy ezekb?l rakja össze a karakterek neveit.

Írjuk meg a függvényt, ami az összes lehetséges nevet összerakja az adott el? és utótagok listája alapján.

- A függvény neve legyen `nev_generator`, és két paramétere legyen
- *elotagok*, egy lista, amiben az el?tagok szerepelnek karakterláncokként
- *utotagok*, egy lista, amiben az utótagok vannak hasonlóan.
- A adja vissza az összes lehetséges összerakott nevet egy listában.

Záró

cloudcoder-en a feladatok ajánlott sorrendje:

1. nevkonfliktus
2. ismetles
3. kiejtes
4. szorzotabla
5. fogasok
6. pascal
7. cserebere
8. nev_generator

el?z? fel következ?