Tartalomjegyzék

El?z? óra

Széls?értékek

Írjuk meg a minimum() és maximum() függvényt, aminek a bemenete egy lista és kiemenete a lista legkisebb, illetve legnagyobb eleme.

Legnagyobb, legkisebb

Írjunk egy két paraméteres függvényt `szelsoertek` néven, els? paramétere legyen: *l*, ami egy számokat tartalmazó lista, a második pedig egy **True** vagy **False** érték

A függvény térjen vissza a lista legnagyobb elemek indexeivel, ha a második paraméter **True**, egyébként pedig legkisebb elemének indexeivel.

Tuple

Definiáljunk egy függvényt lookup() néven, aminek 2 argumentuma van. A második argumentuma egy lista, ami 2 hosszú tuple-ket tartalmaz, az els? argumentum pedig a kulcs. A lookup(kulcs, lista) hívás térjen vissza az els? olyan tuple második tagjával, aminek az els? tagja megegyezik a kulcs bemenettel. Ha nincs ilyen tuple a listában, akkor térjen vissza None-nal.

Legnagyobb közös osztó

Definiáljuk az lnko() függvényt, aminek paramétere két természetes szám és visszatér a legnagyobb közös osztójukkal. (Ehhez felhasználható a prím_faktorizáció() függvény.)

F?

Leghosszabb konstans sorozat

Írjunk egy függvényt, aminek bemenete egy lista és visszatér a leghosszabb olyan sorozat hosszával, amiben egy tag ismétl?dik.
Például:

```
f([1,2,2,2,3,2])
3
```

Azonos elemek

Definiáljuk az azonos() függvényt, aminek a bemenete egy lista, és olyan listák listájával tér vissza, amelyek az eredeti lista egymás után következ? azonos elemeit tartalmazzák.
Például:

```
azonos([1,2,2,3,3,3,3,4,2,2])
[[1],[2,2],[3,3,3,3],[4],[2,2]]
```

Átlaghoz legközelebbi

Írjunk függvényt, mely a kapott valós számokat tartalmazó listában megkeresi a lista elemeinek átlagához legközelebbi számot, és ezzel tér vissza.
Tipp: Használjuk az abszolútérték meghatározására az abs() függvényt.
Üres lista esetén adjon vissza **None**-t.

Nehezebbek

Szavak

1.

Definiáljuk a kiejtés() függvényt, aminek bemenete egy sztring és a program térjen vissza **True**-val, ha ugyanannyi magánhangzó van benne, mint mássalhangzó, egyébként pedig **False**-szal. Az egyszer?sítés kedvéért használhatjuk az angol abc-t.

2.

Írjuk meg a nehéz() függvényt, ami egy sztringr?l eldönti, hogy nehéz szónak számít-e a sztring. A függvény térjen vissza **True**-val ha két magássalhangzó egymás után következik benne és térjen vissza **False**-szal egyébként. Az egyszer?sítés kedvéért használhatjuk az angol abc-t.

Pascal

A Pascal-háromszög a binomiális együtthatók háromszög formában való elrendezése. Részletes leírás található pl. a magyar wikipédián. A lényeg, hogy minden elem a felette lev? kett? összege és a legels? sorban csak egy 1-es van és úgy tekintjük, hogy a sor többi eleme mind nulla, ami nincs kiírva. 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!

Névgenerátor

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 többször 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.

Próbáljuk a feladatot listagenerálással is megoldani!