Tartalomjegyzék

Függvény listákon

Írjunk egy listafv() nev? függvényt, aminek bemenete egy számokat tartalmazó lista, kimenete az a lista, amit úgy kapunk, hogy minden elemét köbre emeljük és kivonunk egyet.

Négyzetszámok

Definiáljunk egy függvényt, aminek 2 paramétere m és n természetes számok és visszaadja az $[m,n]$ intervallumban található négyzetszámok listáját.

Hányadik elem

Definiáljuk a listaindex() függvényt, aminek két paramétere van. Els? paramétere egy lista, a második paramétere egy lehetséges listabeli elem. A függvény térjen vissza, hogy a lista hányadik elemeként szerepel els?nek a függvény második paramétere. Ha nincs benne a listában, akkor térjen vissza **None**-nal.

Változtatás

Írjuk meg a listacsere() nev? függvény, aminek 3 paramétere van az els? egy lista, a másik kett? tetsz?leges. A listacsere(l, a, b) függvény adja vissza azt a listát, mely az l elemeit tartalmazza, de minden 'a' elemet 'b'-re cserél.

Hatvány

Írjunk egy függvényt max_exp() néven úgy, hogy m,n természetes számok esetén max_exp(m,n) térjen vissza a legnagyobb k természetes számmal, melyre $m^k \mid n$. Feltehet?, hogy $m > 1$.

Oszthatóság

1.

Írjunk egy 2 paraméter? függvényt osztható() néven a következ? módon: Az els? bemenete legyen egy lista, a második pedig egy természetes szám. A függvény térjen vissza a lista azon elemeinek listájával, amelyek oszthatók ezzel a természetes számmal.
Például:

```
osztható(list(range(30,50)),7)
[35, 42, 49]
```

2.

Definiáljunk egy osztók() függvényt, ami egy természetes szám valódi osztóinak listáját adja vissza.

Tökéletes számok

Írjunk programot, mely bekér egy pozitív egész számot és leellen?rzi, hogy tökéletes szám-e.

Listák

1.

Írjunk egy függvényt dupla() néven, ami bemenetnek 1 listát kap és visszatér **True**-val, ha van benne olyan elem, ami legalább kétszer szerepel, egyébként visszatér **False**-szal.

2.

Definiáljunk egy függvényt rendezett_e() néven, aminek egy paramétere egy egész számokból álló lista, és eldönti, hogy rendezett-e a lista. Ha rendezett, akkor térjen vissza **True**-val, egyébként **False**-szal. Ha listában egy elem is többször szerepel, akkor térjen vissza **None**-nal.

3.

Írjunk egy függvényt részrendezés() néven, aminek 2 paramétere egy lista és egy természetes szám. A függvény térjen vissza a lista els? *n* rendezett elemével.
Például:

```
részrendezés([6,4,5,2,1],3)
[1,2,4]
```

Prímfaktorizáció

Írjunk egy függvényt prím_faktorizáció() néven, aminek bemenete egy természetes szám és egy olyan listával tér vissza, amiben párok vannak. A pár els? tagja adja vissza, hogy melyik prím szerepel a faktorizációban, a második tagja pedig, hogy milyen hatvánnyal szerepel.
(Tipp: Felhasználható erre a célra a max_exp() függvény.)
Például:

```
prime_decomp(90)
```

```
[(2, 1), (3, 2), (5, 1)]
```

Lista a listában

Definiáljuk a listahossz() függvényt, aminek 2 bemenete van, egy lista, amiben listák vannak és a másik paramétere egy természetes szám. A listahossz(l,n) függvény adja vissza az l listában található azon listák számát, melyek n hosszúak.

Például:

```
listahossz([[1,2,3],[2,3],[1,2,6,5],[2,3,2]], 3)
2
```

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.)