Függvények 2

A repository-dban hozz létre egy „fuggvenyek2” nevű mappát és ebbe dolgozz! A feladatokat az alábbiak alapján nevezd el: *feladat[sorszám].py*, például: feladat1.py.

1. Írjunk olyan függvényt, amely paraméterként sztringek listáját kapja. Meg kell vizsgálnia a listában található sztringeket, és megválaszolnia ezt a kérdést: van-e olyan sztring, amelyik „a” betűvel kezdődik! Ennek is legyen logikai típusú a visszatérési értéke.
   Ügyelj arra, hogy a függvény helyesen működjön abban az esetben is, ha üres sztring van a listában!
   Példák:
   - **["körte", "alma", "barack"]** – van „a” betűvel kezdődő szó.
   - **["dinnye", "papaja", "", "zeller"]** – nincs „a” betűvel kezdődő szó.
   - **[]** – nincs „a” betűvel kezdődő szó.

   Teszteljük a függvényt ezekkel a példákkal, kiírva a listákat és a függvény visszatérési értékét is a főprogramban!
2. Írjunk egy függvényt, amely egy egész számokat tartalmazó listát kap bemenetként, és két listát ad vissza. Az egyik lista tartalmazza a bemenet páros, a másik a bemenet páratlan számait. Példakód a függvény használatára:

```
bemenet = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
kimenet = szetvalogat(bemenet)
parosak = kimenet[0]
paratlanok = kimenet[1]
print("Páros számok:", parosak)
print("Páratlan számok:", paratlanok)
```

   A feladat megoldásához készítsünk egy segédfüggvényt, ami képes megállapítani egy adott számról, hogy páros-e, vagy sem!
3. Készítsünk egy függvényt, amely paraméterként számok listáját és egy számot kap. A függvény végigiterál a listán és minden elemhez hozzáadja a számot. A függvénynek ne legyen visszatérési értéke!
4. Készítsünk egy függvényt, ami egy paraméterként kapott szám összes előfordulását eltávolítja egy paraméterként kapott listából. A függvénynek ne legyen visszatérési értéke!
5. Készítsünk egy aritmetika.py nevű modult, amely tartalmazza az alapvető matematikai műveleteket: összeadás, kivonás, szorzás, osztás. Ezután készítsünk egy menüvezérelt főprogramot, amely importálja ezt a modult és használja a műveleteket. Minden bemenetet kérjünk be a felhasználótól.

6. Készítsünk egyszerű konzolos játékot! A játékban a játékos feladata keresztülhaladás egy 6x6 mezőből álló táblán. A játékos a tábla bal szélén kezd, és át kell haladnia a tábla jobb széléig. A játékos arról a mezőről, ahol áll, mindig felfelé, jobbra, balra, vagy lefelé mozoghat a síkban, de a tábláról nem léphet le. Egy mezőn lehet akna, vagy lehet ártalmatlan terület, de ez csak akkor derül ki, ha a játékos rálép. Példa a játékmező megjelenésére:

```
# # # # # #
# # # # # #
# # # # # #
O O J # # #
# O # # # #
# # # # # #
```

   a. A *#* az eddig felfedezetlen mezőket jelenti.
   b. Az *O* a játékos által már felfedezett mezőket jelenti.
   c. A *J* a játékos jelenlegi helyzetét jelenti.

A játék pályáit .txt fájlokban tároljuk. Ezekben a következő jelöléseket használjuk:

   a. *A* betű: akna
   b. *M* betű: veszélytelen mező

Ha a játékos aknamezőre lép, akkor a játéknak vége. A játék elkészítéséhez a következő adatstruktúrákat fogjuk felépíteni és karbantartani:

   a. Egy 6x6-os 2D listát, ami a játékmezőt reprezentálja. Ennek a listának minden eleme egy karakter: *A,* ha a mező felfedezetlen akna; *M,* ha a mező felfedezetlen veszélytelen mező; *O*, ha a mező már felfedezett veszélytelen mező.
   b. Egy 2 elemű listát, ami a játékos koordinátáit tartalmazza. Az első elem jelenti a sort, a második az oszlopot. A bal felső sarok koordinátája 0, 0.

A játék megírásához készítsük el az alábbi függvényeket:

   a. Egy függvényt, ami paraméterként egy fájlnevet kap, beolvassa az abban található pályát, ebből felépít egy játékmezőt reprezentáló tömböt.
   b. Egy függvényt, ami paraméterként kap egy pályát reprezentáló 2D listát, és egy listát a játékos koordinátáival. A függvény írja ki konzolra a játékmezőt, úgy, hogy a játékos tartózkodási helyére J betűt kerüljön, a még felfedezetlen mezők helyére pedig # karaktert. A függvénynek nincs visszatérési értéke.
   c. Egy függvényt, ami paraméterként listában megkapja a játékos koordinátáit, felfelé lépteti egyet a táblán, de ügyel arra, hogy a játékos nem léphet le a pályáról. Térjen vissza a játékos új koordinátáival.

d. Egy függvényt, ami paraméterként listában megkapja a játékos koordinátáit, lefelé lépteti egyet a táblán, de ügyel arra, hogy a játékos nem léphet le a pályáról. Térjen vissza a játékos új koordinátáival.
e. Egy függvényt, ami paraméterként listában megkapja a játékos koordinátáit, jobbra lépteti egyet a táblán, de ügyel arra, hogy a játékos nem léphet le a pályáról. Térjen vissza a játékos új koordinátáival.
f. Egy függvényt, ami paraméterként listában megkapja a játékos koordinátáit, balra lépteti egyet a táblán, de ügyel arra, hogy a játékos nem léphet le a pályáról. Térjen vissza a játékos új koordinátáival.
g. Egy függvényt, ami paraméterként a táblát és a játékos koordinátáit kapja, és ellenőrzi, hogy a játékos éppen bombás mezőn áll-e. A függvény logikai értékkel térjen vissza.
h. Egy függvényt, ami paraméterként a játékos koordinátáit kapja, és ellenőrzi, hogy a játékos elérte-e a tábla jobb szélét, vagyis nyert-e

Ezután készítsük el a fő programot: töltsünk be egy pályát, ezt ciklikusan írjuk ki a pályát a konzolra. Inicializáljuk a játékos koordinátáit a következőkkel: [3; 0]. Ezután a játék végéig ciklikusan kérjünk be egy irányt a felhasználótól (pl. wasd); az iránynak megfelelően módosítsuk a koordinátáit; ellenőrizzük, hogy aknába lépett-e; ellenőrizzük, hogy nyert-e. Ha nyert, akkor írjuk ki és töltsünk be egy új pályát. Ha bombába lépett, akkor írjuk ki, hogy vesztett, majd lépjünk ki a programból.

7. Ha szükséges, refaktoráljuk a 3. feladatban elkészített programunkat: kerüljön külön modulba a négy irányba léptető függvény. Hozzunk létre modulokat. Gondoljuk át, hogy milyen függvényeknél lehetne mellékhatást használni visszatérési érték helyett, úgy, hogy ne rontsa az átláthatóságot! Használjunk main() függvényt.