# Tartalomjegyzék



# Örökl?dés

## Alakzatok

Írjunk egy **Shape** osztályt.

- Legyen **x** és **y** változója, ezek tárolják az alakzat pozícióját a síkon.
- Legyen egy **move** metódusa, aminek egyetlen **v** paramétere van, egy kételem? lista, a vektor, amivel el kell mozgatni az alakzatot.

Definiáljuk a **Shape** osztály leszármazottaiként az

- **Ellipse** ellipszis, legyen meg a kis- és nagytengelye (**a,b**)
- **Rectangle** téglalap, legyen meg az oldalak hossza (**a,b**)

osztályokat. Mindkét esetben a pozíciójuk a súlypontjukat jelentse.

Írjunk mindkét osztályhoz egy **area** függvényt, ami kiszámítja az alakzat területét!

Definiáljuk az **Ellipse** osztály **equation** metódusát, ami kiírja az adott ellipszis egyenletét!

# Reservation

A következ? feladatokban egy olyan rendszeren dolgozunk, ami egy moziban fogja nyomon követni a különböző emberek helyfoglalásait. Ehhez létre fogunk hozni egy `Reservation` nevű osztályt, ami egy

ember foglalását tárolja, és ezt az osztály ellátjuk minden szükséges metódussal a kényelmes használathoz.

# Reservation_1

Ebben a feladatban hozzuk létre a **konstruktort**. A foglalás osztály két tagváltozót tartalmazzon majd, és ezek legyenek a konstruktor paraméterei is, ebben a sorrendben:

- **name**, a mozi foglalás lefoglalójának neve
- **seats**, egy lista, ami az üléseket tartalmazza amiket *name* lefoglalt. Minden ülés egy karakterlánccal van leírva, pl. "http://wiki.math.bme.huE12"http://wiki.math.bme.hu

Egyelőre csak elég a konstruktort megírni, semmi más nem kell az osztályba még.

**Segítség**

- Ne felejtsük el, hogy (mint minden metódusnál) a konstruktor definiálásakor az első paraméter a *self* legyen.

# Reservation_2

Ebben a feladatban adjunk egy metódust a *Reservation* osztályhoz. A metódus neve legyen **not_taken**, és egy paramétere legyen (a *self*-en kívül):

- **lehetosegek**, ami egy lista, ami lehetséges üléseket tartalmaz.

A *not_taken* metódus térjen vissza a *lehetosegek* lista azon elemeivel, amik ebben a *Reservation* objektumban nincsenek lefoglalva. Az elemek maradjanak ugyanabban a sorrendben mint a *lehetosegek* listában voltak.

**Segítség**

- Ne felejtsük el, hogy az aktuális objektum foglalt helyeit a *self* referencia *seats* tagváltozóján keresztül érhetjük el.

# Reservation_3

Ebben a feladatban kipróbáljuk, milyen használni az eddig megalkotott *Reservation* osztályt. Írjuk meg a **reservation_3** nevű függvényt, aminek paraméterei:

- **foglalasok**, egy lista, ami az éppen aktuális foglalásokat tartalmazza, *Reservation* típusú objektumok formájában
- **osszes_ules**, egy lista, az adott mozi terem összes ülését tartalmazza

A függvény azoknak az üléseknek a listájával térjen vissza, amik még szabadak, tehát amik még egyik foglalásnak sem részei.

**Segítség**

- Használjuk a *Reservation* osztály *not_taken* nevű metódusát.

## Reservation_4

Ebben a feladatban olyan metódust írunk, ami módosítja az objektum tartalmát. Az eddigi *Reservation* osztályunkban hozzunk létre egy új metódust, aminek a neve **change**, és paramétere (a *self*-en kívül):

- **uj_helyek**, egy lista, ami azt tartalmazza hogy mik a helyek amiket foglalni akar az illető a változtatás után.

A metódus változtassa meg az objektumot úgy, hogy átírja a **self.seats** változót, de ne térjen vissza semmivel.

## Reservation_5

Most használjuk a *Reservation* objektumainkat kicsit bonyolultabb feladatra. Írjuk meg a **reservation_5** nevű függvényt, aminek paraméterei:

- **foglalasok**, egy lista, ami az éppen aktuális foglalásokat tartalmazza, *Reservation* típusú objektumok formájában
- **osszes_ules**, egy lista, az adott mozi terem összes ülését tartalmazza
- **nev**, egy karakterlánc, az éppen foglalni próbáló vendég neve
- **darab**, egy egész szám, hogy hány helyet szeretne lefoglalni

A függvény keressen *darab* olyan helyet, amik az *osszes_ules* listában egymás után szerepelnek, és még nincsenek lefoglalva. Ha talál ilyeneket, akkor azt a lehetőséget vegye, amelyiknek az ülései a legkorábban szerepelnek az *osszes_ules* listában, és adja vissza a foglalást egy *Reservation* típusú objektumban. Ha nem talál ilyen üléseket, akkor *None*-t adjon vissza.

#### Segítség

- Használjuk ez el?z? feladatok megoldását

# Sakk(Szorgalmi)

Definiáljuk a **Piece** osztályt. Ez reprezentál egy sakkbábut, tároljuk a pozícióját a táblán két koordinátával, a színét (black/white), illetve a **__str__** írja ki, hogy hol áll (A2, G3 etc.)!

- Definiáljuk a bábu leszármazottjaként a **King** és a **Pawn** osztályokat!
- Legyen a leszármazottaknak (King, Pawn) is **__str__** függvénye úgy, hogy az már a figura típusát is kiírja. (Nem muszáj a leszármazott osztályban megvalósítani, lehet az ?sosztályban is)
  - ♦ Érdemes megnézni a sakk figurák unicode kódját: [1]
  - ♦ Vagy az egybet?s angol nevüket
- Minden leszármazottnak legyen egy **.move(pos)** metódusa, ahol a **pos** egy sztring (A3, G2 etc.)! Mozgassuk el a bábut, ha szabályos a lépés! Ha a lépés szabályos volt, térjünk vissza **True**-val, egyébként **False**-szal!
- Definiáljuk a **PieceMoveError** osztályt. Ha szabálytalan a lépés, dobjunk egy ilyen exceptiont és kezeljük le!

Legyen most egy **Board** osztályunk! Tároljuk listában a figurákat!

- Implementáljunk egy **add** metódust amivel hozzá tudunk adni bábúkat a táblához.
- Legyen a **Board** osztálynak egy **move(player, pos1, pos2)** metódusa, ami a **pos1** pozícióban álló bábut a **pos2** helyre mozgatja, ha a lépés szabályos!

    - ♦ Érdemes el?ször egy **.occupied(pos)** metódust implementálni a Board osztályban.
    - ♦ A normál szabályok mellett vegyük figyelembe, hogy áll-e ott más bábu! Ha az a bábu sajátunk, a lépés szabálytalan, ha az ellenfél bábuja, akkor távolítsuk el a pályáról, hiszen leütöttük!
- Írjuk meg a **Knight, Rook, Bishop** és **Queen** osztályokat!
    - ♦ Kezdjük a **Rook**-kal, ez a legegyszer?bb! Ügyeljünk rá, hogy a ne lépjünk át másik bábun, hiszen ez szabálytalan!
    - ♦ Másodikként a **Knight** osztályt írjuk meg! A ló ugorhat, tehát a szabályok írásánál ezt nem kell figyelembe venni.
    - ♦ A **Bishop** után a **Queen** lépését könny? összerakni.
- A **Board** osztályon belül definiáljunk egy **check** metódust. Térjen vissza azzal a színnel, amelyik király sakkban áll, vagy üres string-gel ha semelyik.
    - ♦ Definiáljuk át a lépést, hogy az csak akkor legyen szabályos, ha a lép? játékos királya nem áll sakkban a lépés után!
- Legyen **__str__** függvénye a **Board**-nak!

Miután mindez megvan, közel állunk ahhoz, hogy tudjunk sakkozni. Definiáljuk a **start** metódust, ami kezd?állapotba teszi a táblát.