Tartalomjegyzék

- 1 Iterálhatók
- 2 Polimorfizmus
  - 2.1 Alakzatok a vásznon
    - 2.1.1 Alakzatok
    - 2.1.2 Örökl?dés és konstruktorok
    - 2.1.3 Vászon
  - 2.2 Sakk
    - 2.2.1 CLI

Iterálhatók

Íjunk olyan iterálható osztályt, mint a **range**, de ne egy listát járjon be, hanem csak az aktuális elemet tárolja.

```
class Range:
     def __init__( ... ):
         ...
     def __iter__( ... ):
         ...
     def __next__( ... ):
         ?
```

- konstruktora egy számot vagy sztringet kapjon. Addig a számig lehessen iterálni rajta, nullától, egyesével.
- Ha a szám nem pozitív, akkor 0 hosszan lehessen iterálni rajta.
- Ha sztringet kap a konstruktor és az nem értelmezhet? egészként, akkor emeljünk **ValueError** kivételt.
  - Ha értelmezhet? egészként, akkor alakítsuk át egésszé és számoljunk azzal.
- Ha **"http://wiki.math.bme.huinf"http://wiki.math.bme.hu** sztringet kap a konstruktor, akkor végtelen sokáig lehessen rajta iterálni!

Polimorfizmus

Alakzatok a vásznon

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!

Örökl?dés és konstruktorok

Ha a *leszármazott osztályban* (például Ellipse) az *?sosztály konstruktorát* (jelen esetben Shape) akarjuk hívni, akkor erre két módszer is lehetséges:

```
class B(A):
    def __init__(self, x, y, a, b):
         A.__init__(self, x, y)
         # vagy
         super().__init__(x, y)
```

Az els? változatban

```
A.__init__(self, x, y)
```

megmondjuk, hogy az **A** osztály konstruktorát akarjuk meghívni (amit már korábban megírtunk).
A második változatban

```
super().__init__(x, y)
```

azt mondjuk, hogy a **B** mindenkori ?sosztályát hívjuk meg, ami most éppen **A**.

Vászon

Definiáljuk a **Canvas** (vászon) osztályt.

- Egyetlen tagváltozója legyen a **shapes**, ami alakzatok listáját tárolja.
- Definiáljuk egy **add** metódust, amivel újabb **Shape**-et adunk a vászonhoz!
- Oldjuk meg, hogy az osztályunk iterálható legyen! Ehhez definiáljuk az **__iter__(self)** metódust, valamit a **__next__(self)** metódust, ahogy az el?adáson láttuk.
- Definiáljuk a **crop** metódust a következ?képp: a bemeneti paraméter két pont koordinátája, ezek egy téglalap bal fels? és jobb alsó pontjai. A függvény térjen vissza azon alakzatok listájával, amelyek a vásznon vannak és teljesen belefférnek az így definiált téglalapba. Ehhez a feladathoz az **Ellipse** és a **Rectangle** osztálynak is szüksége lesz egy **box()** metódusra, ami a legkisebb tartalmazó doboz bal fels? és jobb alsó sarkait adja vissza.

Sakk

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.

CLI

- Játszunk a sakk-kal úgy, hogy `input`-tal kérünk a felhasználótól bemenetet, egyszer a világostól, egyszer a sötétt?l és aszerint lép a tábla.
    - Ehhez érdemes implementálni az algebrai notációt (Bf5, Qc3, Ne2, Kcd4, Kxd5 stb.) és lépjünk eszerint!
- Minden lépés után írjuk ki a táblát (ha megírtuk a `__str__`-et akkor ez csak egy `print`).
- Próbáljuk ki parancssorból.