Informatika 2, 2. ZH (2020-04-28)

A megoldásokat a ***hazi@math.bme.hu*** *címre küldjétek el 19:15-ig!*

1. Írjunk egy `Shape` nevű osztályt, ami absztrakt alakzatokat reprezentál a síkon. És hozzá két leszármazott osztályt: `Line` és `Circle`. *(8 pont)*

A `Line` osztály egyeneseket reprezentáljon, konstruktora három számot kapjon, egy pontjának x és y koordinátáit valamint az egyenes meredekségét (az x tengellyel bezárt szögének tangensét), ami lehet ∞ is, ha az egyenes függőleges.

A `Circle` osztály köröket reprezentáljon, konstruktora három számot kapjon, a középpontjának x és y koordinátáit valamint a sugarát. Ha a sugár nem pozitív, akkor emeljünk `ValueError` kivételt.

Mindkét alakzatnak legyen `move` metódusa, amivel egy adott vektorral eltolhatjuk. Ez `None`-t adjon vissza, de közben tolja el az alakzatot. A `self`-en kívül két argumentuma legyen: az eltolás vektor x és y koordinátái.

Mindkét alakzatnak legyen `__str__` metódusa is, amit kiírásra lehet használni. Ez az alakzat egyenletét adja vissza string-ként. Az egyenlet bármilyen alakú lehet, amíg az egy kétváltozós egyenlet és a megoldásai pontosan a kívánt alakzatot alkotják. Egyenesnél érdemes ezt: meredekséggel kifejezett egyenlet

Példa:

```
>>> l = Line(0, 0, 1)
>>> print(l)
y-0=1*(x-0)
>>> x = Circle(0, 0, 1)
>>> x.move(1, 0)
>>> print(x)
(x-1)^2+(y-0)^2=1
```

Figyeljünk az egyenes egyenletére, ha a meredeksége végtelen.

Segítség:

```
class Shape:
    pass

class Line(Shape):
    def __init__(self, x, y, m):
        pass

class Circle(Shape):
    def __init__(self, x, y, r):
        pass
```

2. Elméleti kérdések *(4 pont)*

1. Mutassunk példát egy öröklődésre és nevezzük meg az osztályokat az öröklődésben betöltött szerepük szerint (rokoni viszonyok)!

2. Hogyan lesznek kiírhatóak (`print`-tel) a saját osztályunk példányai? Mely metódust kell ehhez megírni és hogyan?

3. Mi a különbség metódus (tagfüggvény) és adattag (tagváltozó) között? Példával vagy magyarázattal szemléltessük.

4. Mutassunk egy példát a `str.format` metódus használatára!

3. Írjunk egy `Collatz2` nevű iterálható osztályt, amivel egy pozitív egészből indított Collatz sorozaton lehet iterálni, amíg el nem érjük az 1-et, de mindig az adott elem **négyzetét** adja vissza az iterátor. Az iterálás sosem lehet üres, mert az utolsó elem mindig 1. A konstruktorának egy paramétere legyen (a self-en kívül): `x`. Ellenőrizzük, hogy `x` egész és pozitív, ha nem az akkor emeljünk `ValueError` kivételt. *(4 pont)*
emlékezzünk, hogy a Collatz sorozat képzési szabálya a következő:

$$x_{n+1} = \begin{cases} \frac{x_n}{2} & \text{ha } x_n \text{ páros} \\ 3x_n + 1 & \text{ha } x_n \text{ páratlan} \end{cases} \tag{1}$$

Az osztálynak így kell működnie:

```
for i in Collatz2(3):
    print(i, end=" ")
```

Kimenet:

```
9 100 25 256 64 16 4 1
```

Mert az eredeti sorozat ez lenne:

```
3 10 5 16 8 4 2 1
```

Figyeljük meg, hogy a pozitív számok halmazán a négyzetre-emelés bijekció egyetlen fixponttal (ami az 1), ezért a négyzet-sorozatnak is van egy (1)-hez hasonló képzési szabálya. Viszont ha megvan az eredeti feladat kódja, akkor azzal még egyszerűbben is megoldható a feladat!

4. Az alábbi kódban 4 hiba található, találjuk meg és javítsuk ki a hibákat! *(4 pont)*
Mindegyik sorban legfeljebb egy hiba van.

```
class A:
    x = 3
    def __init__(self):
        self.x = []
    def __str__(self):
        return self.x * A.x

class B:
    def add(self, x)
        self.x.append(x)

b = B(3)
b.add(3)
print(b)
```