

Tartalomjegyzék

- 1 Ismétlés
- 2 Öröklődés
- 3 Feladatok
 - ◆ 3.1
Síkidomok
 - ◆ 3.2
Alkalmazottak

Ismétlés

Műveletek átdefiniálása, osztályok részletebben:

```
class Vektor(object):
    def __init__(self, x, y):
        self.x = x
        self.y = y
    def __add__(self, jobb):
        return Vektor(self.x + jobb.x, self.y + jobb.y)
    def __sub__(self, jobb):
        return Vektor(self.x - jobb.x, self.y - jobb.y)
    def __mul__(self, szam):
        return Vektor(self.x * szam, self.y * szam)
    def hossz(self):
        return sqrt(self.x ** 2 + self.y ** 2)
    def __repr__(self):
        return "(" + str(self.x) + ", " + str(self.y) + ")"
```

- `__init__`: inicializálás
- `__add__`: összeadás
- `__sub__`: kivonás
- `__mul__`: szorzás
- `__div__`: osztás
- `__pow__`: hatványozás
- `__len__`: hossz lekérése
- `__repr__`: karakterekkel való reprezentálás

Öröklődés

```
class A(object):
    def __init__(self, x):
        print "A létrehozva"
        self.v1 = x
    def vlketszeres(self):
        return self.v1 * 2

class B(A):
    def __init__(self, x, y):
        print "B létrehozva"
        super(B, self).__init__(x)
        self.v2 = y
    def v2haromszoros(self):
        return self.v2 * 3
```

```
objA = A(5)
print "====="
```

```
objB = B(1, 4)

print "====="
print objA.vlketszeres()

print "====="
print objB.vlketszeres()
print objB.v2haromszoros()
```

A **super()** függvénynek adunk egy osztályt, és egy objektumot, tekinthetjük úgy a működését, hogy a neki adott objektum azon részét adja vissza ami az adott osztály (**B**) ősoosztályának (**A**) felel meg.

Egy szemléletesebb példa.

Feladatok

Síkidomok

Kezdeti file

- Használni fogjuk az ismétlésben látott **Vektor** osztályt
 - Előre megírtam az **Alakzat** alaposztályt, és az ő **Negyszog** leszármazottját.
 - Ezek alapján írjátok meg a **Hurnegyszog** osztályt, ő is a 4 csúcsot kapja, hasonlóan a négyszöghöz, de neki van képlete a területére: wikipedia
 - A **Tegla** osztály a téglalap bal alsó sarkát kapja, és az x és y tengellyel párhuzamos oldalainak hosszát. Ebből kellene meghívni a **Hurnegyszog** ősének az `__init__`-jét. (Ki kell számolni a csúcsait, vigyázzunk hogy a körüljárási irány megmaradjon.)
 - A **Negyzet** osztály olyan mint a **Tegla**, de csak 1 oldalhosszt kap.
-
- Az **Ellipszis** a középpontját és a kis és nagy féltengelyének hosszát kapja. Területe, a kerületét nem kötelező kiszámolni...
 - A **Kor** a középpontját kapja és a sugarának hosszát. Neki már a kerületét is könnyen tudjuk számolni.
 - Ha valakinek marad ideje, akkor a szabályos N szögnek is írjon osztályt.

Alkalmazottak

Kezdeti file

- Előre megírtam az **Alkalmazott** alaposztályt, mely valamilyen cég egy alkalmazottját írja le, van neve és fizetése. A fizetését változtathatjuk a **fizetesValtozas** függvénnyel és az **eltanacsolas** függvénnyel eltanácsoljuk (a fizetését 0-ra állítjuk).
- A **Manager** osztályt is megírtam egyszerű puska gyanánt, a managerek fix 300ezer fizetésről kezdenek.
- Írjátok meg a **Programozo** osztályt, mely az **Alkalmazott**-ból származik le, fix 200ezer fizetésről

kezd, és van még egy plusz adattagja, **szint** néven, ami azt jelzi, hogy hanyadik szinten dolgozik az épületben. Természetesen neve és fizetése is van mint az **Alkalmazott**-nak.

- Legyen még neki egy **athelyezes** nevű függvénye, mely egy szintet kap, és átállítja a **szint** adattagját, hogy mostmár ezen a szinten dolgozzon.

- Írjátok meg a **VezetoProgramozo** osztályt, mely a **Programozo**-ból származik le, fix 400ezer fizetése van, neki is meg van adva melyik szinten dolgozik, és van még egy plusz adattagja **beosztottakSzama** névvel, ami kezdetben mindig 0.
- Van egy **ujBeosztott** függvénye, mely eggyel növeli a beosztottainak a számát, és a fizetését 50ezerrel.
- Van továbbá neki is **athelyez** függvénye, de ez nem úgy működik mint a **Programozo**-nál, mert átállítja a szintjét, visszaveszi a fizetését az alapra, és a **beosztottakSzama**-t visszaveszi 0-ra.