

Objektum Orientált Programozás

OSZTÁLY

[class]

Készítette: Vastag Atila

2020

A korai programozási nyelvek nem az adatokra, hanem a műveletekre helyezték a hangsúlyt, mert akkoriban még főleg matematikai számításokat végeztek a számítógépekkel. Ahogy aztán a számítógépek széles körben elterjedtek, megváltoztak az igények, az adatok pedig túl komplexekké váltak ahhoz, hogy a procedurális módszerrel kényelmesen és hatékonyan kezelni lehessen őket.

Képzeljünk el egy olyan függvényt, mely egy logikailag összefüggő adatok halmazát kellene, hogy paraméterül kapja.

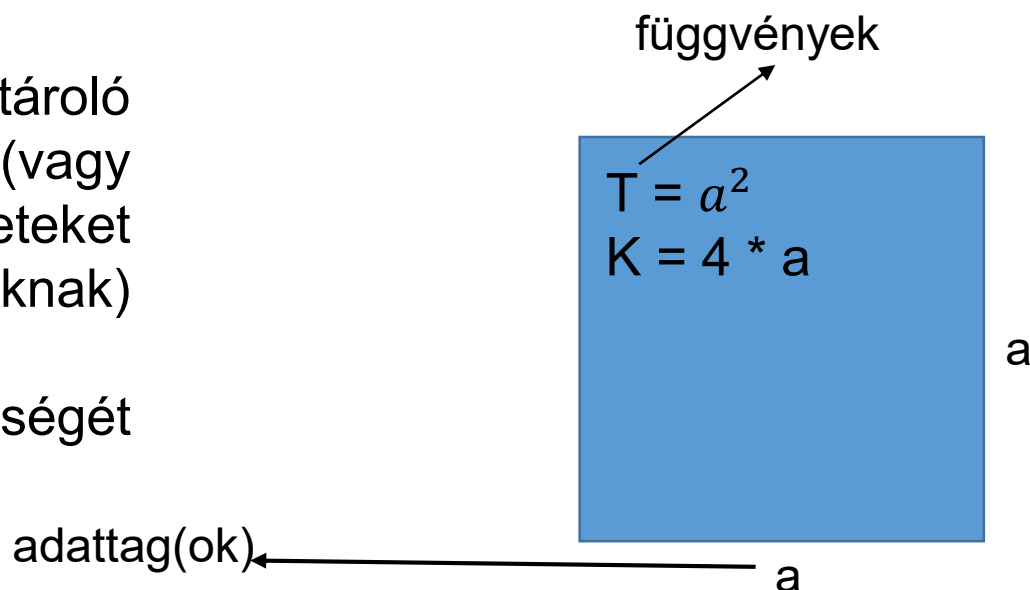
Ezért az OOP már nem a műveleteket helyezi a középpontba, hanem az egyes adatokat (adatszerkezeteket) és a közöttük levő kapcsolatokat (hierarchiát).

Az OOP világában egy osztály olyan adatok és műveletek összessége, amellyel leírhatjuk egy modell (vagy entitás) tulajdonságait és működését.

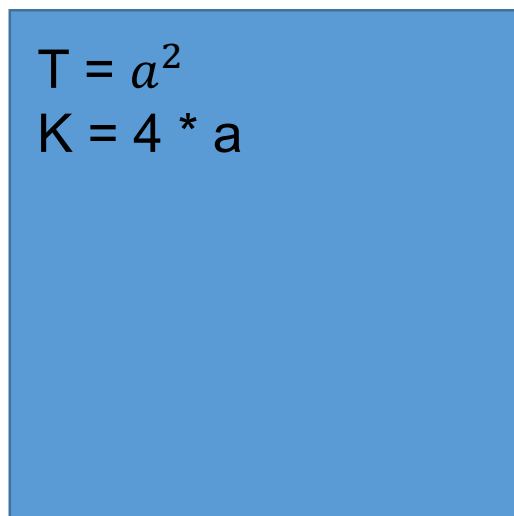
Egy objektumnak az életciklusa során megváltozhat az állapota, tulajdonságai. Ezt az állapotot valahogy el kell tudnunk tárolni, illetve biztosítani kell a szükséges műveleteket a tulajdonságok megváltoztatásához.

A tulajdonságokat tároló változókat adattagoknak (vagy mezőnek), a műveleteket függvényeknek (metódusoknak) nevezzük.

A műveletek összességét felületnek is hívjuk.



Hogyan különböztessük meg, hogy mikor adattag és mikor függvény (metódus)?



a

```
class Negyzet:
    def __init__(self, a: float = 0):
        super().__init__()
        self.a: float = a  adattag(ok)

    def terület(self) -> float:
        return self.a * self.a

    def kerulet(self) -> float:
        return 4 * self.a  függvény(ek)
```

Ha az osztály olyan adattaggal rendelkezik, mely nem követel számítást akkor adattagról van szó (oldal, sugár, ...).

Amennyiben fel kell használni az osztály valamely adattag tulajdonságát műveletek sorozatában, hogy eredményt kapjunk (kerület, terület kiszámítása) akkor függvényről (metódus) van szó.

Amikor programot írunk, akkor az adott osztályból (osztályokból) létre kell hoznunk egy (vagy több) példányt, ezt példányosításnak nevezzük. Az osztály és példány közötti különbségre jó példa a recept (osztály) és a sütemény (példány).

```
negyzet: Negyzet = Negyzet(10)
```

A `negyzet` objektum `Negyzet` típusú, ez azt jelenti, hogy a `negyzet` az objektum egy konkrét példánya a `Negyzet` osztálynak. A `Negyzet` osztály nem használható példányosítás nélkül, csak annak a példánya(i)!!!

A példányosításkor lefut a **"konstruktor"**, `__init__`, (később lesz szó róla), megfelelő nagyságú hely foglalódik a memóriában, ezután pedig megtörténik az adattagok inicializálása is (`self.a: float = a`).

Osztályt a **class** kulcsszó segítségével deklarálhatunk:

```
class Test:
    globalisValtozo: float = 0

    def __init__(self):
        super().__init__()
        self.lokalisValtozo: float = 0
```

`globalisValtozo` – statikus, azaz nem változik az értéke objektum példányonként. Minden példányosított objektumnál ugyanaz az értéke.

`lokalisValtozo` – változik az értéke objektum példányonként. Minden példányosított objektumnál egyedi az értéke.

```
x: Test = Test()
```

```
y: Test = Test()
```

```
x.globalisValtozo = 10
```

```
x.lokalisValtozo = 20
```

```
print(x.globalisValtozo)    #10
```

```
print(x.lokalisValtozo)    #20
```

```
y.globalisValtozo = 100
```

```
y.lokalisValtozo = 200
```

```
print(y.globalisValtozo)    #100
```

```
print(y.lokalisValtozo)    #200
```

```
x.globalisValtozo = -10
```

```
x.lokalisValtozo = -20
```

```
y.lokalisValtozo = -200
```

```
print(x.globalisValtozo)    # -10
```

```
print(x.lokalisValtozo)    # -20
```

```
print(y.globalisValtozo)    # -10
```

```
print(y.lokalisValtozo)    # -200
```

Az osztály konkrét példányán ponttal elválasztva lehet elérni az adattagokat és a függvényeket.

```
from Negyzet import Negyzet
```

```
alakzat: Negyzet = Negyzet()
```

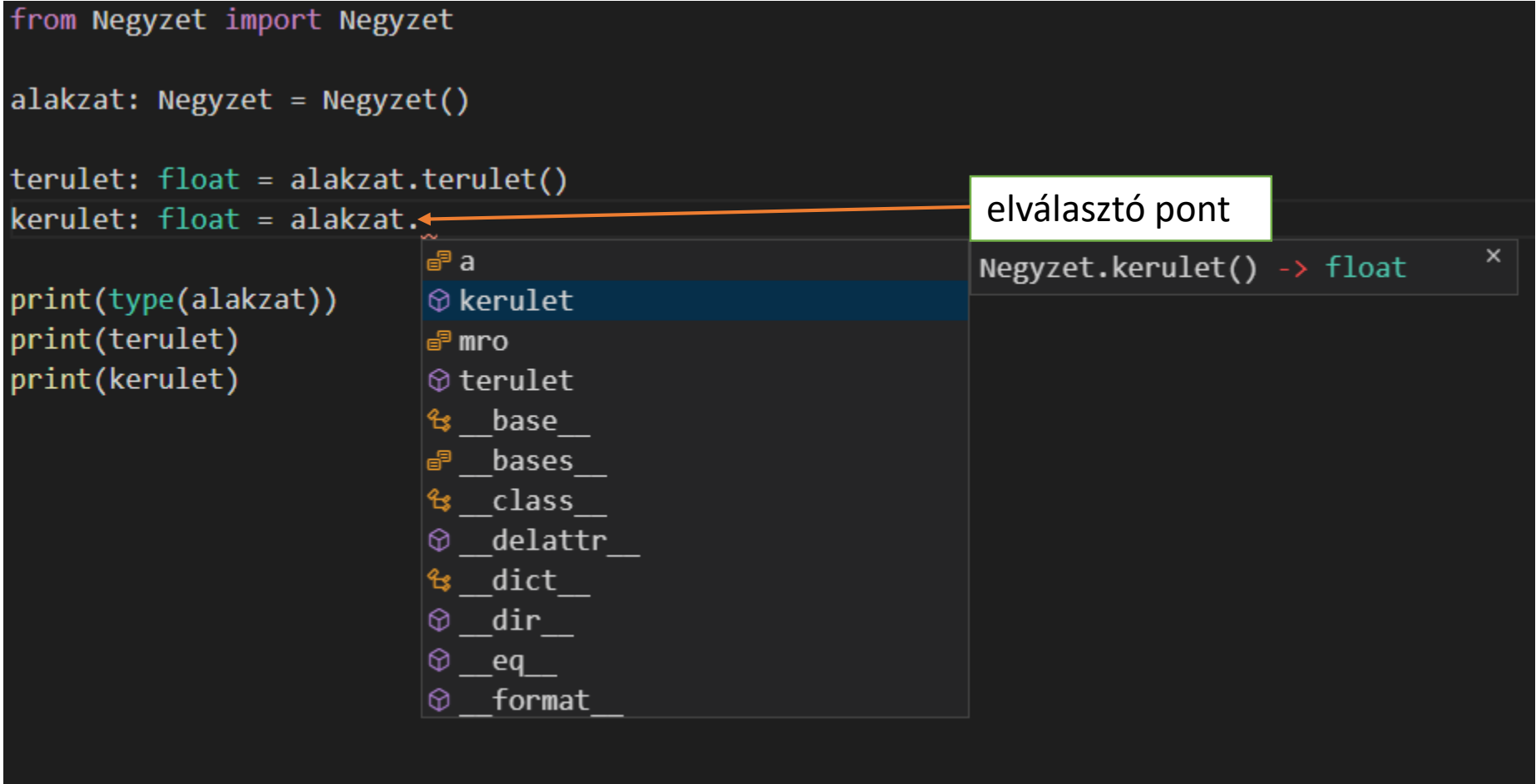
```
terulet: float = alakzat.terulet()
```

```
kerulet: float = alakzat.
```

```
print(type(alakzat))
```

```
print(terulet)
```

```
print(kerulet)
```



- a
- kerulet
- mro
- terulet
- __base__
- __bases__
- __class__
- __delattr__
- __dict__
- __dir__
- __eq__
- __format__

elvásztó pont

Negyzet.kerulet() -> float

Egy osztály csak egy entitásért kell, hogy felelős legyen.

Például:

A **Negyzet** osztálynak csak a négyzettel, mint geometriai alakzattal, kapcsolatos feladatokat kell ellátnia. A **Negyzet** osztálynak nem szabad, hogy foglalkozzon a körrel, háromszöggel ... bármely más geometria alakzattal és azok tulajdonságaival.

objektum = adat + kód

ettől objektum, az objektum; mert e kettőnek elválaszthatatlan egészen értjük az objektumot !

Az objektum egyik alkotóeleme az adat, vagy adatszerkezet. Deklarált adatokat jelent. E részben tulajdonképpen a valóságot ábrázoljuk. (úgymond: a tárgy méreteit).

A másik a kód, amelyen olyan eljárások és függvények összességét értjük, amelyek leírják az objektum viselkedésmódját.