Дефиниција декоратора

Понекад је потребно модификовати постојећу функцију без мењања њеног изворног кода.
Декоратор је функција која узима једну функцију као улаз а враћа другу функцију.
Синтакса декоратора: @ime_dekoratora
Од следеће линија кода се поставља дефиниција функције која се треба декорисати.

Дефиниција особина помоћу декоратора

Други начин да се дефинише особина је помоћу декоратора.
У примеру се користе две различите методе (иако се обе зову ime()) али којима претходе различити декоратори.

```
class Zivotinja():
    def __init__(self, uneti_ime):
        self.sakriveno_ime = uneti_ime

    @property
    def ime(self):
        print('aktivan geter')
        return self.sakriveno_ime

    @ime.setter
    def ime(self, uneti_ime):
        print('aktivan seter')
        self.sakriveno_ime = uneti_ime

pas = Zivotinja('Reks')
print(pas.ime)              #aktivira se samo geter
pas.ime = 'Sava'
print(pas.ime)              #aktivira se seter pa geter
print(pas.sakriveno_ime)
```

Даје:
aktivan geter
Reks
aktivan seter
aktivan geter
Sava
Sava
Види се да је у коду манипулације класом заштићен атрибут sakriveno_ime, јер се под тим именом не спомиње ниједна референца.
Да би се ускладиштеним атрибутом skriveno_ime могло манипулисати, користи се особина ime.

Именовање приватних атрибута

Пајтон има конвенцију именовања за атрибуте који не би требало да су видљиви изван дефиниција своје класе, а то је да име почиње са две доње црте (__).
Ако се у претходном примеру атрибут sakriveno_ime претвори у __ime, неће доћи до никакве промене у коду.
Али ако се покуша приступ атрибуту __ime:
print(pas.__ime) #AttributeError: 'Zivotinja' object has no attribute '__ime'
Ова порука о грешци не упућује да је атрибут __ime приватан, али свеједно довољно сакрива атрибут од спољног утицаја.
Ипак постоји начин и овде да се приђе приватом атрибуту:
pas._Zivotinja__ime чиме се добија тренутна вредност атрибута __ime.

Израда лабораторијских вежби:

Задатак 054: Креирати надкласе Возило и БорбеноСредство и подкласу Тенк. Класа Возило има два атрибута име и оружије. Класа БорбеноСредство има атрибут оружије. Проверити функционисање вишеструког наслеђивања подкласе Тенк од обе надкласе.

```
class Vozilo:
    def __init__(self, ime, oruzije):
        self.ime = ime
        self.oruzije = oruzije

class BorbenoSredstvo:
    def opis(self):
        print(self.ime, "ima", self.oruzije)

class Tenk(Vozilo, BorbenoSredstvo):
    print("xx")

x = Tenk("M1", "top 72mm")
x.opis()
```

Даје:

Xx

M1 ima top 72mm

Задаци за самосталан рад:

50. Креирати ланчано наслеђивање класе Тим, две подкласе Тестирање и Развој и подкласом која наслеђује од две подкласе Возач. Проверити способност последње подклас еу ланцу на користи методе из свих класа у ланцу наслеђивања.

51. Креирати надкласе Посао и Алат и подкласу Радник која наслеђује обе класе. Проверити вишеструко наслеђивање ако се инстанцира објекат класе Радник про, са атрибутима: програмер, тастатура и Јана, и исписује следећу поруку на екрану: Јана је програмер и на послу користи тастатуру. Ако се инстанцира објекат класе Радник пев са атрибутима: певач, микрофон, Мики и исписује на екрану: Мики је певач и на послу користи микрофон.

52. Креирати три класе у ланцу наслеђивања и проверити да ли последња класа у ланцу наслеђује методе из обе класе.

53. Поставити гетер, сетер и делетер у класи Osoba. У методама користити штампање вредности атрибута ime.

54. Употребом декоратера израчунати пречник круга ако се унесе полупречник истог круга.