Uvod u programski jezik Pajton - 5. deo

Uvod u merno-informacione sisteme

Objektno-orijentisano programiranje (OOP)

- Obrađeni tipovi:
  - Numerički tipovi: celobrojni, realni, kompleksni
  - Istinitosni tipovi
  - Skupovi
  - Rečnici
  - Sekvence: liste, torke, stringovi
- Šta raditi ako se želi realitovati sopstveni tip?
- OOP nastoji da objekat opiše atributima i ponašanjem
- U Pajtonu je sve objekat!
- Pripadnost klasi se može proveriti funkcijom *type*

OOP u poređenju sa strukturnim programiranjem

- Strukturno programiranje:
  - Dekompozicija koda deljenjem na funkcije (moguće i module)
  - Kompleksniji način ponovnog korišćenja koda
  - Nije moguće kontrolisati pristup promenljivim
- Objektno-orijentisano programiranje :
  - Kod se skladišti u klasama odn. objektima
  - Korisnici mogu koristiti i nadograđivati postojeće klase
  - Moguće kontrolisati pristup promenljivim (sigurniji pristup)
- Objektno-orijentisano programiranje uključuje strukturno programiranje i pridodaje dodatne koncepte poput enkapsulacije, nasleđivanja, apstrakcije itd.

Klase i objekti

- Klasa predstavlja skup pravila na osnovu kojih će se definisati objekat
- Klasa se definiše ključnom rečju *class*

```
1 class Knjiga:
2     pass
```

- Pravljenje objekta se u terminologiji OOP zove **instanciranje**

```
1 knjiga = Knjiga()
2 print(type(knjiga))#<class '__main__.Knjiga'>
```

- Kako razlikovati poziv klase od poziva metode?
- PEP8 preporučuje definiciju klase početnim slovom u kamiljoj notaciji (*CamelCase*) npr. *KlasaKnjiga*, dok se promenljive i funkcije pišu malim početnim slovom i svaka reč se odvaja donjom crtom (*underscore*) npr. *ovo_je_funkcija*

Klase i objekti

- Moguće je instancirati više objekata iste klase
- Instancirani objekti su nezavisni

```
1 knjiga_1 = Knjiga()
2 knjiga_2 = Knjiga()
3 print(id(knjiga_1))
4 print(id(knjiga_2))
```

- Prazna klasa nema praktičnog značaja jer ona predstavlja model po kom će se praviti objekti
- Svakoj klasi pridodaju se atributi i metode

Atributi i metode

- Atributi opisuju stanje objekta tj. predstavljaju promenljive pridodate objektu
- Metode opisuju ponašanje objekta i menjaju njihovo stanje (atribute) tj. predstavljaju funkcije pridodate objektu
- Prisutp atributima i metodama vrši se korišćenjem tačke (*objekat.metoda(argumenti)* ili *objekat.atribut*)
- Specijalna metoda za inicijalizaciju objekta (***__init__***)
- Ova metoda se naziva konstruktorom (***__init__***), što nije tačno
- Konstruktor je ***__new__*** metoda, a ona poziva ***__init__*** metodu
- Svaka metoda ima prvi parametar ***self***
- ***self*** predstavlja objekat nad kojim je pozvana metoda

```python
class Knjiga:
    def __init__(self, ime, broj_stranica):
        self.ime = ime
        self.broj_stranica = broj_stranica
        self.trenutna_stranica = 0
    def promeni_stranicu(self, stranica):
        if stranica >= 0 and stranica<=self.broj_stranica:
            self.trenutna_stranica = stranica
    def citaj(self):
        print(f'Citam knjigu "{self.ime}"')
        print(f'Citam stranicu {self.trenutna_stranica}')

knjiga = Knjiga('Naucite Pajton', 212)
knjiga.promeni_stranicu(20)
knjiga.citaj()
knjiga.trenutna_stranica = 35
knjiga.citaj()
```

Atributi i metode

- Metoda se poziva nad objektom

```
1 knjiga.citaj()
2 Knjiga.citaj(knjiga)
```

- Zbog kompleksnosti, drugi način nije preporučljiv
- Kao i kod funkcija, parametri metoda mogu imati podrazumevane vrednosti

```
1 class Knjiga:
2     def __init__(self, ime, broj_stranica=200):
3         self.ime = ime
4         self.broj_stranica = broj_stranica
5         self.trenutna_stranica = 0
6
7 knjiga_1 = Knjiga('Naucite Pajton', 212)
8 knjiga_2 = Knjiga('Uvod u Pajton')
```

- I objekti mogu biti atributi drugih klasa

```
1  class Knjizara:
2      def __init__(self, knjige = []):
3          self.knjige = knjige
4
5      def dodaj_knjigu(self, knjiga):
6          self.knjige.append(knjiga)
7
8      def prikazi_naslove(self):
9          for i in self.knjige:
10             print(f"Knjiga: {i.ime}")
11
12 knjiga_1 = Knjiga('Price za laku noc', 320)
13 knjiga_2 = Knjiga('Hari Poter i kamen mudrosti', 288)
14 knjizara = Knjizara([knjiga_1, knjiga_2])
15 knjizara.dodaj_knjigu(Knjiga("Gospodar prstenova",1200))
16 knjizara.prikazi_naslove()
```

Statičke metode i statičke promenljive

- Sastičke metode i statičke promenljive postoje nezavisno od objekata
- Mogu se koristiti nezavisno od objekata
- Statičke metode nemaju parametar ***self***
- Pristup se vrši ***ImeKlase.staticka_promenljiva*** ili ***ImeKlase.staticka_metoda***
- Kako bi se naglasilo da je metoda statička korsiti se dekorater *@staticmethod*
- Dekorateri se koriste da izmene ponašanje neke metode ili klase
- Dekorater *@staticmethod* omogućuje poziv metode bez poziva nad objektom
- Kada bi se izostavio dekorater javila bi se greška

```
1  class Knjizara:
2      sve_knjige = 0
3      def __init__(self, knjige = []):
4          self.knjige = knjige
5          Knjizara.sve_knjige+=len(knjige)
6
7      def dodaj_knjigu(self, knjiga):
8          self.knjige.append(knjiga)
9          Knjizara.sve_knjige+=1
10
11     def prikazi_naslove(self):
12         for i in self.knjige:
13             print(f"Knjiga: {i.ime}")
14
15     @staticmethod
16     def broj_knjiga():
17         print(f'Ukupan broj knjiga je {Knjizara.sve_knjige}
               u svim knjizarama')
```

```
Knjizara.broj_knjiga()

knjiga_1 = Knjiga('Price za laku noc', 320)
knjiga_2 = Knjiga('Hari Poter i kamen mudrosti', 288)
knjizara = Knjizara([knjiga_1, knjiga_2])

Knjizara.broj_knjiga()

knjizara.dodaj_knjigu(Knjiga("Gospodar prstenova", 1200))
knjizara.prikazi_naslove()

Knjizara.broj_knjiga()
```

Magične metode

- Sve metode i atributi objekta, dobijaju se pozivom ***dir()***

```
1 knjiga = Knjiga('Price za laku noc', 320)
2 print(dir(knjiga))
```

- Prilikom generisanja klase iako se ne vide neke metode će biti definisane
- Ove metode se prepoznaju po dve donje crte na početku i kraju naziva
- Npr. ako se pokuša ispisati objekat dobija se poruka

```
1 knjiga = Knjiga('Price za laku noc', 320)
2 print(knjiga)#<__main__.Knjiga object at 0x7fe7c57dd150>
```

- Za ispis podataka zadužena je metoda ***__str__***

Ime magične metode	Opis
__add__(self, obj2)	Poziva korišćenjem operatora +
__sub__(self, obj2)	Poziva korišćenjem operatora -
__floordiv__(self, obj2)	Poziva korišćenjem operatora //
__truediv__(self, obj2)	Poziva korišćenjem operatora /
__mod__(self, obj2)	Poziva korišćenjem operatora %
__pow__(self, obj2)	Poziva korišćenjem operatora **
__lt__(self, obj2)	Poziva korišćenjem operatora <
__le__(self, obj2)	Poziva korišćenjem operatora <=
__eq__(self, obj2)	Poziva korišćenjem operatora==
__ne__(self, obj2)	Poziva korišćenjem operatora !=
__gt__(self, obj2)	Poziva korišćenjem operatora >
__ge__(self, obj2)	Poziva korišćenjem operatora>=

Ime magične metode	Opis
__str__(self)	Kada se objekat konvertuje u string
__hash__(self)	Računa hash vrednost objekta
__int__(self)	Poziva se prilikom konverzije u int
__float__(self)	Poziva se prilikom konverzije u float
__complex__(self)	Konverzija u kompleksnu vrednost
__iadd__(self, obj)	Kada se pozove operator +=
__isub__(self, obj)	Kada se pozove operator -=
__abs__(self)	Kada se pozove abs()
__round__(self,n)	Pozivom funkcije round()
…	…

Nadjačavanje metoda

- Magičnim metodama se može redefinisati funkcionalnost
- Ovo je moguće raditi nad svim metodama i naziva se nadjačavanje metode

```
1 class Knjiga:
2     def __init__(self, ime, broj_stranica):
3         self.ime = ime
4         self.broj_stranica = broj_stranica
5
6 knjiga=Knjiga('Autostoperski vodic kroz Galaksiju',632)
7 print(knjiga)#<__main__.Knjiga object at 0x7fe7c5760c40>
```

- Automatski generisana metoda __str__ ispisaće podatke o objektu (memorijsku lokaciju i kojoj klasi pripada)
- Ovo se može izmeniti nadjačavanjem metode

```python
class Knjiga:
    def __init__(self, ime, broj_stranica):
        self.ime = ime
        self.broj_stranica = broj_stranica

    def __str__(self):
        return f'Ime knjige je: {self.ime}'

knjiga=Knjiga('Autostoperski vodic kroz Galaksiju',632)
print(knjiga)
```

```
1  class Knjiga:
2      def __init__(self, ime, broj_stranica):
3          self.ime = ime
4          self.broj_stranica = broj_stranica
5
6      def __gt__(self, obj2):
7          if self.broj_stranica > obj2.broj_stranica:
8              return f'{self.ime} je veca knjiga'
9          else:
10             return f'{self.ime} nije veca knjiga'
11
12
13 knjiga_1=Knjiga('Autostoperski vodic kroz Galaksiju',632)
14 knjiga_2=Knjiga('Hari Poter i kamen mudrosti', 288)
15 print(knjiga_1>knjiga_2)
16 print(knjiga_2>knjiga_1)
```

Enkapsulacija

- Jedan od koncepata OOP
- Enkapsulacija predstavlja restrikcije pristupa metodama i atributima (izmenama ili očitavanjima)
- Zaštita se vrši dodavanjem:
  - donje crte ispred imena za *protected* (moguć je pristup samo u okviru klase ili klasa naslednica, ali u Pajtonu je moguć pristup i izvan klase)
  - dve donje ispred imena za *private* (moguć je pristup samo unutar klase)
  - samo ime bez donje crte na početku *public* (moguć pristup i spolja i unutar klase)

```python
class Knjiga:
    def __init__(self, ime, broj_stranica, isbn=None):
        self.ime = ime
        self.isbn = isbn
        self.broj_stranica = broj_stranica

knjiga = Knjiga('Autostoperski vodic kroz Galaksiju', 632,
    '978-86-10-00700-8')
print(knjiga.isbn)#978-86-10-00700-8
knjiga.isbn = '111-22-33-44444-5'
print(knjiga.isbn)#111-22-33-44444-5
```

Problem može se izmeniti atribut koji ne bi trebao biti promenljiv!

```python
class Knjiga:
    def __init__(self, ime, broj_stranica, isbn=None):
        self.ime = ime
        self.__isbn = isbn
        self.broj_stranica = broj_stranica

knjiga = Knjiga('Autostoperski vodic kroz Galaksiju', 632,
    '978-86-10-00700-8')
print(knjiga.__isbn)#'Knjiga' object has no attribute '
    __isbn'
knjiga.__isbn = '111-22-33-44444-5'#'Knjiga' object has no
    attribute '__isbn'
```

```python
class Knjiga:
    def __init__(self, ime, broj_stranica, isbn=None):
        self.ime = ime
        self.__isbn = isbn
        self.broj_stranica = broj_stranica

    def prikazi_isbn(self):
        return self.__isbn

knjiga = Knjiga('Autostoperski vodic kroz Galaksiju', 632,
    '978-86-10-00700-8')
print(knjiga.prikazi_isbn())#978-86-10-00700-8
```

Moguće je samo očitati ali ne i promeniti vrednost.

Enkapsulacija

- Kako bi se olakšalo korišćenje privatnih atributa moguće je koristiti dekoratere za pristup atributima
- Pomoću dekoratora izbegava se korišćenje dodatnih metoda već je moguće pristupiti samom atributu
- Metoda koja vraća vrednost (*getter*) dobija se korišćenjem ***@property*** dekoratera
- Metoda koja postavlja vrednost (*setter*) dobija se korišćenje ***@ime_atributa.setter***

```python
class Knjiga:
    def __init__(self, ime, broj_stranica, isbn=None):
        self.ime = ime
        self.__isbn = isbn
        self.broj_stranica = broj_stranica
    @property
    def isbn(self):
        return self.__isbn
    @isbn.setter
    def isbn(self, isbn):
        self.__isbn=isbn

knjiga=Knjiga('Autostoperski vodic kroz Galaksiju',632,'978-86-10-00700-8')
print(knjiga.isbn)#978-86-10-00700-8
knjiga.isbn = '111-22-33-44444-5'
print(knjiga.isbn)#111-22-33-44444-5
```

Hvala na pažnji!