Универзитет у Београду

Грађевински факултет

Катедра за геодезију и геоинформатику

ГИС ПРОГРАМИРАЊЕ

Материјали за вежбе

3

Наставник: Жељко Цвијетиновић

Асистент: Јован Ковачевић

Београд, 2017.

САДРЖАЈ

1. Функције

1.1. Синтакса, основне карактеристике, стандардне функције

```
# Sintaksa def nazivFunckije():
# Koriste se kada je potrebno modelirati ponavljajuće
operacije
# Brojne prednosti (čitljiviji kod, manja složenost, lakše
održavanje...)

# primer jednostavne funkcije
def zbir(a,b): #definicija funkcije
    '''
    Funkcija koja služi za
    računanje zbira dva broja''' # docstring
    return a + b

# docstring - prvi red, između trostrukih navodnika -
neobavezan
# help(zbir)
print zbir(5,10) # poziv funkcije

# Prenos parametara funkciji
# Svi parametri se prenose po referenci
# Promena vrednosti parametra unutar funkcije prouzrokuje
njegovu promenu i izvan

def dodajElement(element, lista):
    if element > 0:
        lista.append(element)
    return lista

clan=""
lista=[]
while clan != "N": # unos se prekida tasterom 'N'
    clan=input("unesi broj: ")
    if clan != "N":
        dodajElement(clan,lista)
        print "unutar petlje", lista
print "van petlje", lista

# Funkcije mogu prihvatiti parametre promenljive dužine
# Koristi se '*' ispred argumenta čime se prikazuje njegova
promenljivost
def stampaUlaznihArgumenata(arg1, *promenljivaLista):
    '''Stampa ulaznih argumenata'''
    print "arg1=", arg1
    for arg2 in promenljivaLista:
        print "arg2=", arg2
    return

# arg1 se zadaje, a promenljiva lista je #dužine nula (0)"
stampaUlaznihArgumenata(10)
# arg1 se zadaje, a promenljiva lista  je #dužine 2"
stampaUlaznihArgumenata(1, 2, 3)
```

1.2. Анонимне функције

```
# ANONIMNE FUNKCIJE
# Sintaksa funkcija = lambda [arg1 [,arg2,.....argn]]:izraz
# Proizvoljan broj ulaznih parametara,
# ali samo jedna izlaznu vrednost u formi izraza.

zbir2 = lambda a, b: (a + b)
print zbir2(10,22)
```

2. Класе

2.1. Синтакса, конструктор, атрибути, методе

```
# Sintaksa class Naziv():
# Apstrakcije iz realnog sveta
# Skup objekata okarakterisanih istim atributima i ponašanjem

import math
class Tacka:
    ''' Klasa za tačke u 2D ravni''' # docstring
    # poziva se iz konstruktora pri kreiranju objekta
    def __init__(self, x = 0, y = 0):
        self.x = x # definisanje atributa x
        self.y = y # definisanje atributa y

    # ispisivanje koordinata tačaka
    def koordinate(self):
        print('({:<f}, {:<f})'.format(self.x, self.y))

    # racuna rastojanje do druge tacke t
    def rastojanje_do(self, t):
        dx = self.x - t.x
        dy = self.y - t.y
        return math.sqrt(dx ** 2 + dy ** 2)

    def tip_tacke(self, tip):
        self.tip = tip

# primer
t1 = Tacka(10,20) # kreiranje novog objekta klase "Tacka"
print t1.x, t1.y # pristup atributima
print t1.koordinate() # poziv metode (bez argumenata)
print t1.rastojanje_do(Tacka(0,0)) # poziv metode (sa
argumentima)

# Atributi se dinamički definišu
t1.tip_tacke("orijentaciona")
print t1.tip
```

2.2. Магичне методе, преоптерећење оператора

```
# Magične metode
# Metode čija imena počinju i završavaju se sa "__" npr.
# Skup metoda koje se najčešće pozivaju indirektno
#  __init__() - inicijalizuje objekat u početno stanje
# __str__() - tekstualna reprezentacija objekta
#  __eq__() - poređenje objekata po sadržaju
# __add__() - sabiranje dva objekta

# Primer: tekstualna reprezentacija objekta, za print(t) ili
str(t)
# Dodati unutar klase "Tacka"
def __str__(self):
    return '({:<f}, {:<f})'.format(self.x, self.y)

# Preopterećenje operatora

# Poređenje da li su dve tacke jednake
# Dodati unutar klase "Tacka"
def __eq__(self, t):
    if isinstance(t, Tacka):
        return self.x == t.x and self.y == t.y

A = Tacka(1, 0)
B = Tacka(0, 1)
X = Tacka(1,0)

print A == B # interno se poziva metoda __eq__() iz klase
objekta A
print A == X
```

2.3. Класни атрибути и статичке методе

```
# Definišu se van svih metoda unutar klase

import math
class Tacka:
    ''' Klasa za predstavljanje tacaka u 2D ravni''' #
docstring
    broj = 0 # klasni atribut

    @staticmethod
    def broj_tacaka():
        print "Broj kreiranih tacaka je
{0:d}".format(Tacka.broj)

    # poziva se iz konstruktora pri kreiranju objekta
    def __init__(self, x = 0, y = 0):
        self.x = x # definisanje atributa x
        self.y = y # definisanje atributa y

        # statistika vezana za klasu (sve tačke)
        Tacka.broj += 1
```

```
    # ispisivanje koordinata tačaka
    def koordinate(self):
        print('({:<f}, {:<f})'.format(self.x, self.y))

    # racuna rastojanje do druge tacke t
    def rastojanje_do(self, t):
        dx = self.x - t.x
        dy = self.y - t.y
        return math.sqrt(dx ** 2 + dy ** 2)

    def tip_tacke(self, tip):
        self._tip = tip

# test primer
t21 = Tacka()
Tacka.broj_tacaka()
t22 = Tacka(1, 1)
t23 = Tacka(3, 3)
Tacka.broj_tacaka()
```

2.5. Сакривање промењливих

```
# Ako ime počinje sa jednom ili dve donje crte (npr. _a, __a),
# smatra se da se radi o privatnom polju

import math
class Tacka:
    ''' Klasa za tacke u 2D ravni''' # docstring
    broj = 0 # klasni atribut

    @staticmethod
    def broj_tacaka():
        print "Broj kreiranih tacaka je
{0:d}".format(Tacka.broj)

    # poziva se iz konstruktora pri kreiranju objekta
    def __init__(self, x = 0, y = 0, tip = "detaljna"):
        self.x = x # definisanje atributa x
        self.y = y # definisanje atributa y
        self._tip = tip

        # statistika vezana za klasu (sve tačke)
        Tacka.broj += 1

    # ispisivanje koordinata tačaka
    def koordinate(self):
        print('({:<f}, {:<f})'.format(self.x, self.y))

    # racuna rastojanje do druge tacke t
    def rastojanje_do(self, t):
        dx = self.x - t.x
        dy = self.y - t.y
        return math.sqrt(dx ** 2 + dy ** 2)

    def promeni_tip_tacke(self, tip):
        self._tip = tip
```

```
    def daj_tip_tacke(self):
        return self._tip

# Sakrivanje "_" je više konvencija, korisnik ipak može da
pristupi atributu
t31 = Tacka()
print t31.daj_tip_tacke()
t31._tip = "medjna"
print t31.daj_tip_tacke()
# Za to se može koristiti "__"
```

2.6. Наслеђивање

```
# Nasleđivanje
# Sintaksa: class IzvedenaKlasa(Roditelj_klasa):
class Tacka_Orijentaciona(Tacka):
    '''Tip tačke koja ima i pikselske koordinate'''

    def __init__(self, x, y, red, kolona):
        Tacka.__init__(self, x, y, "orijentaciona")
        self.red = red
        self.kolona = kolona

    def promeni_tip_tacke(self, tip):  # prepisivanje funkcije
        print "Tip tačke je već definisan"

# Primer
ot = Tacka_Orijentaciona(10, 10, 100, 125)
print ot.red
ot.promeni_tip_tacke("Detaljna")
```

3. Задаци за вежбу

1. Написати у модулу *Sfera* класу ***Sfera*** такву да садржи:
    - Атрибуте: полупречник и координате центра (*xCentar*, *yCentari*, *zCentar*);
    - Конструктор сфере са задатим центром и полупречником;
    - Дефинисати подразумеване вредности атрибута приликом иницијализације (јединична сфера са центром у координатном почетку);
    - Класну промењљиву која броји објекте класе;
    - Статички метод који враћа укупан број креираних објеката класе;
    - Инстанцни метод за рачунање запремине лопте.

    Написати у модулу *TestSfera* главни програм такав да се у њему извршава:

    - Испис броја објеката који су формирани (позивом одговарајућег метода) пре него што је формиран иједан објекат;
    - Креирање објеката:
        - sfera – објекат у координатном почетку, полупречника 4.0;
        - globus (12, 1.0, 1.0, 1.0);
        - bilijarska_lopta (10.0, 10.0, 0.0);
        - jedinicna_sfera – објекат у координатном почетку, полупречника 1.0;
    - Испис броја објеката који су формирани;
    - Испис запремине сваког претходно креираног објекта.

2. Написати у модулу *Geometrija* класу ***Tacka*** такву да садржи:
    - Инстанцне промењљиве x и y (координате тачке);
    - Конструктор на основу задатих x и y координата тачке;
    - Метод за померање Тацке по x и y оси за x*_pomeraj, y_pomeraj;*
    - Метод за рачунање растојања до задате тачке.

    Написати у модулу *Geometrija* класу ***Duz*** такву да садржи:

    - Инстанцне промењљиве, почетна и крајња тачка дужи
    - Конструктор на основу задате почетне и крајње тачке дужи;
    - Методу за креирање дужи на основу задате *x* и *y* координате почетне тачке и *x* и *y* координате крајње тачке;
    - Метод за рачунање дужине дужи

    Написати у модулу *TestGeometrija* главни програм такав да се у њему извршава:

    - Креирање две тачке (почетак (*x_poc*, *y_poc*) и крај (*x_zav*, *y_zav*)) на основу уноса корисника са стандардног улаза;
    - Креирање дуж чија су почетна и крајња тачка баш ове задате тачке;
    - Креирање друге дуж са задатим координатама почетне и крајње тачке;
    - Испис које дужи су формиране;
    - Померање тачке крај за (*dx, dy*) на основу уноса корисника са стандардног улаза;
    - Испис нове дуж.

3. Написати *python* модул/пакет за потребе извођења основних статистичких анализа површи терена. У фази израде програма, површ је потребно задати као скуп (псеудо)случајних тачака на основу корисничког уноса броја тачака са стандардног улаза (главни програм који тестира развијене функционалности модула/пакета). О креираној површи је потребно водити метаподатке о ономе ко анализира површ, броју тачака површи. Над креираном површи треба обезбедити могућност рачунања средње вредности површи, рачунање минималне и максималне вредности координата тачака површи тј. минималног обухватног правоугаоника. За сваку од тачака потребно је водити метаподатке о броју тачке који се додељује произвољно али теба да је јединствен. Омогућити функционалност рачунања најближе тачке некој тачки, као и интерполацију вредности висине дуж правца између две тачке а која ће бити корисна код каснијег рачунања положаја изохипси. Написати главни програм који тестира претходно развијене функционалности.

4. Написати у модулу *Inzenjer* класу ***Inzenjer*** такву да садржи:

   Атрибуте:

   - Име инжењера;
   - Презиме инжењера;
   - Матични број лица;
   - Лиценца.

   Методе:

   - Одговарајуће *getter* и *setter* методе (нема јавних атрибута);
   - Испис информација о инжењеру.

   Написати у модулу *Inzenjer* класу ***GeodetskiInzenjer*** која наслеђује класу ***Inženjer*** а садржи још:

   Атрибути:

   - Број година радног стажа

   Методе:

   - Одговарајуће *getter* и *setter* методе за ненаслеђене атрибуте (нема јавних атрибута)
   - Испис свих информација о геодетском инжењеру
   - Исписи информације о лиценци уколико је има, уколико је нема треба само исписати да инжењер нема лиценцу

   Написати у модулу *Inzenjer* класу ***ElektrotehnickiInzenjer*** која наслеђује класу ***Inzenjer*** а садржи још:

   Атрибути:

   - Број пројеката на којима је до сада радио

   Методе:

   - Одговарајуће *getter* и *setter* методе за ненаслеђене атрибуте (нема јавних атрибута)
   - Испис свих информација о инжењеру електротехнике
   - Исписи информације о лиценци уколико је има, уколико је нема треба само исписати да инжењер нема лиценцу

5. Написати у модулу *Osoba* одговарајуће класе које ће омогућити обраду података о категоријама особа у складу са следећим описом:

Податке о особи чини име, презиме, датум рођења и адреса становања. Ђак је особа за коју се зна назив основне школе, одељење у које иде и година уписа школе. Запослен је особа за коју се зна назив компаније и департмана у ком је запослен, датум закључења и прекид радног односа са свим компанијама у којима је претходно радио/ради. За сваку од категорија особа омогућити унос и испис свих информација о конкретној особи. За сваког ђака потребно је омогућити добијање информације о томе који разред школе ђак тренутно похађа, као и да ли је можда обновио неку годину током школовања (није важно коју!). За сваког запосленог потребно је омогућити добијање информација о укупном радном стажу запосленог изражену у месецима (претпоставља се да месец има 30 дана!).

Написати главни програм који тестира написане класе. Подаци о особама уносе са стандардног улаза и исписују на стандардни излаз.