Универзитет у Београду

Грађевински факултет

Катедра за геодезију и геоинформатику

ГИС ПРОГРАМИРАЊЕ

Материјали за вежбе

2

Наставник: Жељко Цвијетиновић

Асистент: Јован Ковачевић

Београд, 2017.

САДРЖАЈ

1. Фундаменталне структуре података

1.1. Ниске

```
# Niske
# Mozemo ih pisati izmedju jednostrukih i dvostrukih navodnika
# Postoje velike slicnosti sa listama

niska1 = 'Ovo je neka niska.'
niska2 = "People are strange when you're a stranger ."

print niska1
print niska2
# Karakterima u niski mozemo pristupati koristeci notaciju []
kao kod listi

print niska2 [4]
print niska2 [6:10]

# Duzinu niske racunamo koristeci funkciju len(niska)
print len(niska1)

# Funkcija count
# Vraca broj koliko se puta
# podniska nalazi u niski (u intervalu od pocetak do kraj)
# niska.count(podniska , pocetak , kraj])

print niska2.count("strange")

# Funkcija find
# niska.find(podniska , pocetak , kraj)
# Vraca poziciju prvog pojavljivanja
# podniska u niski (u intervalu od pocetak do kraj),
# vraca -1 ukoliko se podniska ne nalazi u niski
print niska2.find("are")

# Funkcija join
# spaja listu niski separatorom
# niska separator.join([niska1 ,niska2 ,niska3 ,...])
print ' '.join(["Olovka",'pise','srcem.'])

# Korisne funkcije za rad sa niskama:
# niska.isalnum()
# isalpha()
# isdigit()
# islower()
# isspace()
# isupper()
# niska.split(separator) - razlaze nisku u listu koristeci
separator
# niska.replace(stara , nova [, n]) - zamenjuje svako
pojavljivanje niske stara
# niskom nova (ukoliko je zadat broj n, onda zamenjuje najvise
n pojavljivanja)
```

1.2. Листе

```
# LISTA
# Sintaksa: [el1 , el2 , ...]
# Liste mogu sadrzati razlicite tipove podataka
# Kreiranje liste
lista1 = list([66.25, 333, 333, 1, 1234.5])
lista2 = [1,2,3.4,"Another brick in the wall",True ,[5,False
,4.4,'Layla ']]
print lista1
print lista2

# Prazna lista
lista_prazna = []  # moze i lista_prazna = list()

# Broj elemenata liste: funkcija len()
print len(lista1), len(lista_prazna)

# Pristupanje elementima liste
# Indeksiranje elemenata pocinje od 0
print lista1 [0]
print lista2 [3][1]
print lista2 [0:3]
# Mozemo indeksirati liste unazad, pozicija -1 odgovara
poslednjem elementu
print lista1[-1]

# Nepostojeci element liste prijavljuje gresku tipa
IndexError: list index out of range
# print lista1 [1000]

# Ispitivanje da li se element nalazi u listi
# Koriscenjem "if" funkcije
if 1 in lista2:
    print "1 se nalazi u listi\n"

# Prolazak kroz listu
# "for" petlja
for el in lista1:
    print el

# Neke ugradjene funkcije za rad sa listama
# kopmletna lista
https://docs.python.org/2/tutorial/datastructures.html
# Ubacivanje elementa na kraj

# lista.append (3.14) - dodavanje elementa na kraju liste
# lista.remove(x) - izbacuje prvo pojavljivanje elementa x iz
liste
# lista.count(x) - vraca broj koliko puta se element x nalazi
u listi
# lista.index(x) - vraca indeks prvog pojavljivanja elementa x
u listi
# del lista[a:b] - brise elemente liste od pozicije a do b
# lista.insert(pozicija , element) - Ubacivanje elementa na
odredjenu poziciju u listi
# lista.pop(pozicija) - Izbacuje element na zadatoj poziciji i
vraca ga kao objekat
```

```
# Spajanje dve liste
lista = lista1+["Plava", "Zuta", "Crna"]
print lista

# Poredjenje listi
#Dve liste se porede tako sto se njihovi elementi porede redom
leksikografski
print "[1,2,3] < [1,2,5]"
print "\n['abc ','abc ','abc '] < ['abc ', 'ab', 'abcd ']"
print ['abc','abc','abc'] < ['abc', 'ab', 'abcd']
print "\n['a','b','c '] > ['a', 'b ']"
print ['a','b','c'] > ['a', 'b']
```

1.3. Скупови, n-торке

```
# SKUP
# Nesortirana kolekcija, bez duplih elemenata
# ne moraju svi elementi  da budu istog tipa
# Sintaksa: {el1 , el2 , ...}
# Kreiranje skupa
skup1 = {1, 12, 16, 1, 1, 55}

# Pravljenje skupa od liste
lista1 = [4,56,34,2,5,6,4,4,6]
skup2 = set(lista1)
skup3 = set(['a', 'c', 'r', 'd', 'b', 'm', 'z', 'l'])

# Prazan skup
skup_prazan = set()
# ne moze skup_prazan = {}, kreira se prazan recnik umesto
skupa

# N-torka
# Konacan niz (uredjena lista) od n objekata, od kojih je
svaki odredjenog tipa

torka1 = 12345, 54321, 'hello!'
torka2 = ('Pera', 'Peric', 100010)

# Pristupanje elementima u torki
print torka2 [1]
```

1.4. Речници

```
# RECNIK
# Recnik je kolekcija uredjenih parova oblika (kljuc ,
vrednost)
# Sintaksa: {kljuc:vrednost , kljuc:vrednost , ...}
# Kreiranje recnika
recnik1 = {'jack': 4098, 'sape': 4139}
recnik2 = dict([('sape', 4139), ('guido', 4127), ('jack',
4098)])

# Prazan recnik
recnik_prazan = {} # moze i recnik_prazan = dict()

# Pristupanje elementima u recniku
print recnik1['sape']

# Prolazak kroz recnik
for kljuc in recnik1:
    print " '{0:s}' =>
{1:s}\n".format(str(kljuc),str(recnik1[kljuc]))

# Korisne funkcije
# dict.keys() - vraca listu kljuceva iz kataloga
# dict.values() - vraca listu vrednosti iz kataloga
# dict.has_key(kljuc) - vraca True/False u zavisnosti od toga
da li se element
# sa kljucem kljuc nalazi u katalogu
```

2. Библиотека *NumPy*

```
import numpy as np

# Glavni objekat biblioteke je homogeni visedimenzioni niz
(matrica) - ndarray
# Svi elementi moraju biti istog tipa
# Dimenzija niza se naziva osa - axes

# Sintaksa np.array([e1, e2, e3, ...])

mat1 = np.array([2,3,4])
mat2 = np.array([(1.5,2,3), (4,5,6)])
print mat1
print mat2

# Osnovne informacije kreirane matrice
# ndarray.ndim - rang matrice
# ndarray.shape - oblik matrice
# ndarray.size - ukupan broj elemenata
# ndarray.dtype - tip podataka niza

# Osnovne operacije
A = np.array( [[1,1], [0,1]] )
B = np.array( [[2,0], [3,4]] )
print A*B  # mnozenje elemenata na istim pozicijama
print A.dot(B)  # proizvod matrica, moze i np.dot(A, B)
print A+B
```

```
# Prlazak kroz matricu
for niz in mat2: #podrazumeva po vertikalnoj osi
    print niz

for element in mat2.flat: # kroz sve elemente
    print(element)

# Nadovezivanje dve matrice
mat3 = np.vstack([mat1, mat2])
print mat3

# Provlacenje krive
# funkcija np.polyfit()
x = [1, 10, 4 ,5, 11, 5]
y = [5, 3, 4 ,1 , 5, 6]

fit_xy = np.polyfit(x, y, 1)
fit_fn_xy = np.poly1d(fit_xy)
print fit_xy, fit_fn_xy
```

3. Задаци за вежбу

1. Написати програм која рачуна суму парних елемената низа. Тестирати га позивом из главног програма.
2. Написати програм која рачуна суму елемената низа. Тестирати га позивом из главног програма.
3. Написати програм која рачуна производ елемената низа. Тестирати га позивом из главног програма.
4. Написати програм која од два низа креира нови низ тако да елементи тог низа представљају наизменична појављивања елемената првог и другог низа, односно другог и првог низа у зависности од унете опције корисника (нпр. за унос опције -п прво се ређају елементи првог па другог, а за унос опције -д прво се ређају елементи другог па првог). Тестирати га позивом из главног програма.
5. Написати програм који за унету реченицу приказује карактере који се појављују у њој.
6. Написати програм који врши фитовање полинома произвољног степена кроз сет унетих тачака (2Д простор). Корисник прво задаје број тачака кроз које провлачи полином, затим уноси X, Y координате тачака. Након што су унете све тачке, корисник дефинише степен полинома. Резултат у виду формуле полинома са одређеним коефицијентима приказује се на стандардном излазу.
7. Написати програм који имплементира игру Ајнц са једним играчем. Игра се са шпилом од 52 карте. На почетку играч уноси своје име након чега рачунар дели две карте играчу и две карте себи. У свакој следећој итерацији рачунар дели по једну карту играчу и себи. Циљ игре је сакупити карте које у збиру имају 21 поен. Карте са бројевима носе онолико бодова колики је број, док жандар, дама, краљ носе 10 бодова. Карта Ас може да носи 1 или 10 бодова, у зависности од тога како играчу одговара. Играч који сакупи 21 је победио. Уколико играч премаши 21 бод, победник је његов противник. https://en.wikipedia.org/wiki/Blackjack