

Fakultet tehničkih nauka, DRA, Novi Sad

Predmet:
Organizacija podataka

Rad sa datotekama na programskom jeziku *Python*

TEKSTUALNE DATOTEKE

Tekstualne datoteke

- Tekstualne datoteke služe za permanentno (stalno) skladištenje podataka u tekstualnom formatu na nekom medijumu masovne memorije
- Tekst u datoteci mora odgovarati nekom kodnom standardu
 - najčešće ASCII ili UTF-8

Otvaranje datoteke

- Funkcija *open()* vraća file object koji predstavlja Pajtonovu apstrakciju nad datotekom
- Najčešće se koristi sa dva argumenta *open(filename, mode)*
 - `f = open('workfile', 'w')`

Režimi otvaranja datoteke

- Parametar mode predstavlja režim u kom se datoteka otvara
 - 'r' - otvara datoteku u režimu za čitanje
 - 'w' - otvara datoteku u režimu za pisanje s tim što prethodni sadržaju datoteke (ako ga je bilo) uništi
 - 'a' - otvara datoteku u režimu dodavanja na kraj sadržaja
 - 'r+' režim omogućava i čitanja i pisanje
 - podrazumevana vrednost mode parametra je 'r'

Čitanje iz datoteke

- Metoda *read(size)* fajl objekta čita size karaktera iz datoteke i pročitano vrednost vraća kao string
- Ukoliko se size izostavi onda *read()* čita celokupan sadržaj datoteke
- Tekstualnu datoteku moguće je čitati i liniju po liniju koristeći metodu *readline()* fajl objekta

Čitanje iz datoteke

- Kroz redove datoteke moguće je iterirati na sledeći način:
 - `for line in f:`
 `print(line, end='')`
- Lista svih redova se može dobiti preko `list(f)` ili `f.readlines()`
- Preporučuje se korišćenje ***with*** konstrukcije prilikom rada sa datotekama
 - `with open('workfile') as f:`
 `read_data = f.read()`
- ***with*** garantuje uredno zatvaranje datoteke čak i u slučaju podizanja izuzetka

Pisanje u datoteku

- Metoda *write(content)* fajl objekta upisuje prosleđeni sadržaj u datoteku koja je prethodno otvorenu u režimu pisanja
 - `f.write('This is a test\n')`
- *write()* vraća broj upisani karaktera kao povratnu vrednost
- Objekte koji nisu string tipa neophodno je prvo sa *str()* pretvoriti u string pa potom upisati u tekstualnu datoteku

Primer 1

- Kreirati tekstualno datoteku i popuniti je sa 5 redova proizvoljnog teksta
- Kreiranu datoteku pročitati:
 - karakter po karakter
 - red po red
 - odjednom
- Sadržaj datoteke prepisati u novu datoteku u obrnutom redosledu:
 - redova
 - karaktera

Zadatak 1

- Učitati proizvoljnu tekstualnu datoteku i prebrojati:
 - karaktere
 - slova
 - cifre
 - reči
 - rečenice

Zadatak 2

- Iz ulazne datoteke učitati podatke o automobilima u niz
- Za unetu kubikažu vozila pronaći najnovije vozilo sa kubikažom ne većom od zadate
- Primer datoteke:

AlfaRomeo	1200	2005
BMW	2200	2011
AUDI	1900	2010
Zastava	1600	1998
Ferrari	2500	2013

Zadatak 3

- Za zadati karakter c i prirodan broj d formirati tekstualnu datoteku sa simbolom “karo” od karaktera c , a širine i visine $2d+1$ karaktera
- Primer: $c=\#$, $d=3$

#

###

#####

#####

#####

###

#

BINARNE DATOTEKE

Otvaranje, čitanje i pisanje

- Binarne datoteke otvaraju se na isti način kao i tekstualne, samo je neophodno u *mode* parametru dodati karakter 'b'
 - `f = open('workfile', 'rb+')`
- Za čitanje se koristi `read(size)` metoda koja vraća objekat *bytes klase*
- Za pisanje se koristi `write(content)` gde je *content* parametar tipa *bytes*
 - `f = open('workfile', 'rb+')`

Ugrađena klasa *bytes*

- Omogućava kreiranje objekata koji sadrže nepromenljiv niz bajtova
 - `x = b"some content"`
- Podržava samo ASCII vrenosti kao literale, svi ostali znakovi se zadaju korišćenjem eskejp sekvenci
- Za promenljivi niz bajtova koristi se klasa *bytearray*
- Klase *bytes* i *bytearray* podržavaju većinu standardnih string metod:
 - `replace`, `removeprefix`, `join` ...

Navigacija kroz binarnu datoteku

- Za razliku od tekstualni datoteka, koje se najčešće čitaju i pišu sekvencijalno (od početka do kraja), prilikom rada sa binarnim datotekama neretko je potrebno pozicionirati se na određenu poziciju u okviru datoteke
- *tell()* metoda fajl objekta vraća trenutnu poziciju internog indikatora unutar datoteke, a to je broj bajtova udaljenosti u odnosu na početak datoteke

Navigacija kroz binarnu datoteku

- Metoda *seek(offset, whence)* omogućava promenu poziciju u datoteci
 - *offset* je broj bajtova u odnosu na referentnu tačku
 - *whence* je referentna tačka, a može biti:
 - 0 - početak datoteke (podrazumevana vrednost)
 - 1 - trenutna pozicija
 - 2 - kraj datoteke
- Primeri:
 - *f.seek(5)* # Go to the 6th byte in the file
 - *f.seek(-3, 2)* # Go to the 3rd byte before the end

Serijalizacija i deserijalizacija Pajton objekata

- Modul *pickle* implementira binarni protokol za serijalizaciju i deserijalizaciju Pajton objekata
- *pickle* podržava serijalizaciju i deserijalizaciju
 - logičkog i brojevni tipova
 - stringova, torki
 - listi, setova, rečnika koji sadrže elemente koji su nekog od gore navedenih tipova

Serijalizacija i deserijalizacija Pajton objekata

- Funkcija *`pickle.dump(obj, file)`* omogućava serijalizaciju objekta u binarnu datoteku
- Za obrnuti postupak koristi se funkcije *`pickle.load(file)`*

Primer 2

- Vrednost celobrojne promenjive upisati u binarnu datoteku
- Pročitati kreiranu datoteku
- Pročitanu vrednost udvostručiti i upisati u novu datoteku

Primer 3

- Vrednost string promenjive upisati u binarnu datoteku
- Pročitati kreiranu datoteku
- Na pročitano vrednost dodati nekoliko ne-ASCII (UTF-8) karaktera i upisati string u novu datoteku
- Pročitati UTF-8 datoteku i pročitati samo poslednjih 6 bajta

Primer 4

- Objekat tipa rečnik serijalizovati u datoteku
- Pročitati kreiranu datoteku
- Izmeniti deserializovani rečnik
- Ispisati izmenjeni rečnik u novu datoteku

Zadatak 4

- Ulaznu CSV datoteku učitati kao niz torki (tuple)
- Učitani niz serializovati u binarnu datoteku
- Primer ulazne datoteke:

Grad, Uslovi, MinTemp, MaxTemp

Beograd, Oblačno, 6, 12.5

Novi Sad, Oblačno, 5, 11

Subotica, Sunčano, 6, 12

Zrenjanin, Kišovito, 4, 11

Nevesinje, Sunčano, 6, 13

Zadatak 5

- Data je binarna BMP [slika](#)
- Na 2. bajtu je veličina fajla (4 bajta)
- Na 10. bajtu je veličina zaglavlja fajla (4 bajta)
- Bajtove iz zaglavlja kopirati u novi fajl
- Ostale bajtove učitati u int promenljivu x (jedan po jedan)
- Ispisati komplementarne vrednosti $(255-x)$ u novi fajl