Rad sa fajl sistemom

Kao i svaki drugi programski jezik, Python podržava upravljanje fajlovima pomoću funkcija za kreiranje, otvaranje i čitanje podataka i pisanje podataka u fajlove.

U primerima ćemo raditi sa fajlom Example.txt sadržine:

```
'Lorem Ipsum is simply dummy text of the printing and typesetting industry.
Lorem Ipsum has been the industry's standard dummy text ever since the 1500s,
when an unknown printer took a galley of type and scrambled it to make a type
specimen book.'
```

Otvaranje fajla

Sintaksa za otvaranje fajla je:

```
f_open = open(file_name, mode)
```

Od ovih argumenata jedini obavezan argument je ime samog fajla (file_name) koji želimo da otvorimo. Raspoloživi modovi za otvaranje fajla su:

Arg	Opis
r	read mode – mod za čitanje fajla; ako fajl ne postoji, doći će do greške
r+	mod za istovremeno čitanje fajla i pisanje u fajl
a	append mode – dopisivanje u fajl; ako fajl postoji, biće otvoren i podaci će biti dodati na trenutni sadržaj fajla; ako fajl ne postoji, biće kreiran
w	write mode – upis u fajl; ako fajl postoji i otvoren je na ovaj način, njegov sadržaj će biti obrisan; ako fajl sa datim imenom ne postoji – biće kreiran
x	ekskluzivno otvaranje fajla; ako fajl postoji, izbaciće grešku; u protivnom, progam će nastaviti
t	tekstualni mod
b	binarni mod

Tabela 20.1. Raspoloživi modovi za otvaranje fajlova

Ako se ne prosledi mod za čitanje fajla, podrazumevani mod je t. Naredba: `f_open('Example.txt')` je ista kao i `f_open('Example.txt', 'rt')`.

Otvoriti fajl znači precizirati njegovu lokaciju u ulaznom parametru `file_name`. To se može uraditi na dva načina:

- prvi je preciziranje apsolutne putanje (čitava putanja do njegove lokacije: `f_open = open('C:/Users/Desktop/Example.txt');`
- drugi je preciziranje relativne putanje (lokacija u odnosu na direktorijum odakle se skript pokreće): `f_open('./Example.txt')`.

Zatvaranje fajla

Nakon otvaranja fajla i čitanja/pisanja, bitno je zatvoriti fajl. Ukoliko fajl ostane otvoren u programu, drugi programi neće moći da mu pristupe. Istina je da Python sadrži načine da automatski zatvori fajl na kraju samog programa, ali ne treba se oslanjati na tu funkcionalnost i odgovornost je na programeru da ručno zatvori svaki otvoreni fajl. Najlakši način zatvaranja fajla je naredbom `.close()`:

```
f_open = open('./Example.txt', 'r')
f_open.close()
```

Iako je ovo sintaksički potpuno tačno, nije dobra praksa i treba koristiti ili `with` naredbu ili `try-finally` odlomak.

Zatvaranje fajla pomoću try-finally bloka koda

Sintaksa za ovaj način zatvaranja je:

Primer

```
try:
    f_open = open('./Example.txt', 'r')
finally:
    f_open.close()
```

Ovakav pristup nam omogućava da se fajl, ako se desi da dođe do neke greške u radu sa fajlom ili do nepredviđenog prekida programa, sam zatvori u okviru `finally` naredbe.

Na prostom primeru čitanja fajla i pokušaja deljenja nulom uočavamo ovu prednost:

Primer upotrebe try-finally bloka

```
try:
    f_open = ('./Example.txt','r')
    print(1/0)

finally:
    f_open.close()
    print("File is closed")
```

Vidimo da je fajl ipak zatvoren, iako je došlo do greške u programu.

Naredba with

Korišćenje ove naredbe nam omogućava implicitno zatvaranje fajla bez obaveze da mi to pišemo. Prethodni `try-finally` primer bi izgledao ovako:

Primer upotrebe with naredbe

```
with open("./Example.txt", 'r') as f:
    f.read()
```

Nakon ovog bloka koda, fajl će automatski biti zatvoren.

Čitanje fajla

Čitanje fajla se zasniva na korišćenju nekoliko metoda, u zavisnosti od toga kako želimo da pročitamo sam fajl.

Metoda read([<size>])

Sintaksa ove metode, u kombinaciji sa naredbom with, izgleda ovako:

```
with open ('./Example.txt', 'r') as f:
    print(f.read())
```

Ovim ćemo ispisati čitavu sadržinu fajla Example.txt. Ova metoda takođe prima i opcioni argument tipa int, koji nam omogućava da pročitamo tek toliko karaktera. Ukoliko u već pomenutom slučaju prosledimo `print(f.read(5))`, dobićemo: Lorem, jer su to prvih pet karaktera u fajlu. Ako bismo odmah nakon ove naredbe ponovo poželeli da pozovemo `read()` metodu, program bi nastavio da čita deo fajla nakon ovih pet karaktera:

Primer potrebe metode read[<size>]	
Kod	**Rezultat**

`with open ('./Example.txt', 'r') as f:`
`  print(f.read(5))`
`  print(f.read())`
	`Lorem`
	`Ipsum is simply dummy text of the printing and typesetting industry.`
	`Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.`

Tabela 20.2. Primer metode read[<size>]

Ovo se dešava jer Python pamti trenutnu poziciju u fajlu. Pa tako, ako bismo fajl pročitali do kraja i želeli ponovo da ga iščitamo, moramo tu poziciju vratiti na početak pomoću metoda `seek()` ili `tell()`.

Metode seek() i tell()

Ove dve metode nam pomažu u pozicioniranju pokazivača u fajlu.

- `tell()` nam pokazuje gde se trenutno u fajlu taj pokazivač nalazi;
- `seek()` nam omogućava da taj pokazivač pomerimo na željenu poziciju.

Primer upotrebe metoda seek() i tell()	
Kod	**Rezultat**

`with open ('./Example.txt', 'r') as f:`
`  print(f.read(5))`
`  print(f.tell())`
`  f.seek(0)`
`  print(f.read())`
	`Lorem`
	`5`
	`Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book`

Tabela 20.3. Primer upotrebe metoda seek() i tell()

Prvo smo ispisali prvih pet karaktera u fajlu. U tom trenutku i pokazivač je na petoj poziciji. Naredbom `f.tell()` smo uvideli da je trenutna pozicija 5 i naredbom `f.seek(0)` pokazivač sa pete pozicije ponovo vraćamo na početak (nultu, početnu poziciju, indeks) i naredbom `f.read()` se uveravamo da je fajl ponovo pročitan od početka.

Metode readline() i readlines()

Metoda `readline()` će pročitati trenutnu liniju, sve do graničnika za novu liniju (karakter \n). Ako se ponovo pozove ta metoda nad istim fajlom, biće pročitana naredna linija.

Metoda `readlines()` čita ceo fajl (od trenutnog pokazivača u fajlu) do kraja fajla. Ova metoda, za razliku od prethodne koja vraća string, vraća listu čiji su elementi svaka linija posebno.

Primer upotrebe metoda readline() i readlines()	
Kod	**Rezultat**

`with open ('./Example.txt', 'r') as f:`
`  print(f.readline())`
`  f.seek(0)`
`  print(f.readlines())`
	`Lorem Ipsum is simply dummy text of the printing and typesetting industry.\n`
	`['Lorem Ipsum is simply dummy text of the printing and typesetting industry.\n', "Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book."]`

Tabela 20.4. Primer upotrebe metoda readline() i readlines()

U ovom ispisu primećujemo rezultat prve naredbe (`print(f.readline())`) gde je pročitana prva linija. Nakon nje smo vratili pokazivač na početak fajla i izvršili naredbu (`print(f.readlines())`) koja nam je vratila listu stringova gde je svaki element zasebna linija fajla.

Pitanje

Ako smo tek otvorili fajl i zadali naredbu print(f.read(-1)):

- dobićemo ValueError: negative seek position -1
- ispisaće se poslednji karakter
- **ispisaće se ceo fajl**

Objašnjenje:

Ovom naredbom ćemo zapravo ispisati ceo fajl. Bilo koju negativnu vrednost da prosledimo funkciji read(), ispisaćemo ceo fajl.

Upis u fajl

Postoje dva načina upisa u fajl koji se zasnivanju na pomenutim modovima za otvaranje fajla. Jedan je dodavanje podataka u već postojeći fajl (append, mod a), a drugi pisanje preko fajla (trenutni sadržaj fajla će biti ispražnjen, a novi upisan na njegovo mesto – mod w).

Sintaksa za upis u fajl pod `write (w)` modom je sledeća:

Primer

```
with open ('./Example.txt', 'w') as f:
    f.write("Test")
```

Na ovaj način, ako bismo nakon izvršenja programa otvorili fajl, videli bismo da je naš tekst obrisan i da sada taj fajl sadrži samo string Test. Ako bismo, na primer, umesto ulaznog fajla imali fajl `./Example2.txt`, iako želimo da vršimo operaciju pisanja u fajl koji ne postoji – fajl će prvo biti kreiran.

Sintaksa za upis u fajl pod `append (a)` modom je sledeća:

Primer

```
with open ('./Example.txt', 'a') as f:
    f.write("Test")
```

Na ovaj način, nakon otvaranja našeg fajla vidimo da je string Test dodat odmah nakon poslednje rečenice. Ako želimo da dodamo tekst u narednom redu, naredbu `f.write("Test")` promenićemo u `f.write("\nTest")`. Kao što je prilikom upisa u fajl modom w važilo pravilo: ako fajl ne postoji, prvo će biti kreiran, isto važi i u ovom slučaju.

Primer

Potrebno je učitati podatke iz jednog fajla, ispisati ih na standardni izlaz(komandnu liniju) i upisati ih u drugi fajl tako da se svaka reč nalazi u novom redu.

Ako fajla Ulaz.txt izgleda ovako:

Ovo je jedna recenica.

Program bi trebao da upiše podatke u fajl Izlaz.txt kao:

Ovo
je
jedna
recenica

Rešenje

```
with open('Ulaz.txt','r') as f1:
   podaci=f1.read()

print(podaci)

podaci=podaci.split(' ')

with open('Izlaz.txt','w') as f2:
   for red in podaci:
      f2.write(red+'\n')
```

Atributi za rad sa fajl objektima

Python nam takođe omogućava i nekoliko atributa za rad sa fajl objektima:

- `file.name` – vraća u program ime fajla koji smo otvorili;
- `file.mode` – vraća u program informaciju o modu otvaranja fajla;
- `file.close` – vraća `True` ili `False` vrednost u zavisnosti od toga da li je fajl zatvoren ili ne.

Rezime

- Funkcija open() nam služi za otvaranje fajla.
- Modovi u kojima se fajl može otvoriti su čitanje, pisanje i dodavanje.
- Metodom seek() pomeramo trenutnu poziciju pokazivača u fajlu.
- Metodom tell() dobavljamo trenutnu poziciju pokazivača u fajlu.
- Za čitanje fajla možemo koristiti metode read(), readline() i readlines().
- Za upis u fajl koristimo metodu write().
- With naredbu koristimo kada ne želimo da brinemo o greškama koje mogu nastati tokom čitanja fajla, a bitno nam je da taj fajl zasigurno zatvorimo.