Pythonov modul os i rad s datotekama: Izrada radnog priručnika s primjerima

Rumora, Mislav

Sveučilište u Rijeci, Fakultet informatike i digitalnih tehnologija

Sveučilišni prijediplomski studij Informatika

Mislav Rumora

Pythonov modul os i rad s datotekama: Izrada radnog priručnika s primjerima

Završni rad

Mentor: Doc. dr. sc. Vanja Slavuj

Rijeka, rujan 2023.

FAKULTET INFORMATIKE I DIGITALNIH TEHNOLOGIJA
SVEUČILIŠTE U RIJECI

Rijeka, 17. lipnja 2022. godine

Zadatak za završni rad

Pristupnik: Mislav Rumora

Naziv završnog rada: Pythonov modul os i rad s datotekama: Izrada radnog priručnika s primjerima

Naziv završnog rada na eng. jeziku: Python os module and file management: A manual with examples

Sadržaj zadatka: Programski jezik Python nudi programsko sučelje za rad s operacijskim sustavom računala. U sklopu ovog završnog rada potrebno je proučiti, opisati i na vlastitim primjerima prikazati funkcionalnosti koje su povezane s operacijskim sustavom i ponuđene u okviru modula naziva os programskoga jezika Python. Naglasak je potrebno staviti na odabrane funkcije i podatke koji se odnose na rad s datotekama i direktorijima (npr. stvaranje objekata u stablu direktorija, meta podatci i atributi datoteka, zapis sadržaja u datoteke, dozvole pristupa i sl.), kao i na njihovu primjenu u originalnim primjerima, uz detaljna objašnjenja programskoga koda.

Mentor

Doc. dr. sc. Vanja Slavuj

Voditelj za završne radove

doc. dr. sc. Miran Pobar

Zadatak preuzet: (datum)

(potpis pristupnika)

Sadržaj

Sažetak

Operacijski sustav je skup sustavnih programa koji omogućuju komunikaciju između korisnika i računala te on tako upravlja radom računala, njegovim programima i podatcima (tutorialspoint., 2023). Prema posljednjim istraživanjima, najkorišteniji operacijski sustavi današnjice su Windows, MacOS i Linux. Operacijskim sustavima možemo upravljati pomoću programskog jezika Python koji je zbog svoje jasne i sažete sintakse stekao reputaciju programskog jezika namijenjenog početnicima. To objašnjava i zašto je od 2022. službeno stekao titulu najpopularnijeg programskog jezika.

Jedna od značajki koja pridonosi Pythonovoj popularnosti je njegova svestranost – može se koristiti u različite svrhe, od razvoja softvera do strojnog učenja, te ima široku standardnu biblioteku s više od 200 modula, od kojih je jedan i modul os.

Modul os čini interakciju između korisnika i korisničkog programa jednostavnijom, zahvaljujući svojim funkcijama koje prikupljaju informacije o sustavu i sudjeluju u upravljanju datotekama, direktorijima, procesima i I/O tokovima. Datoteke i direktoriji su temeljne značajke operacijskog sustava. Datoteke služe za pohranu programa i podataka bitnih korisniku, a direktoriji služe za organizaciju datoteka.

Cilj ovog rada je opisati korištenje Pythonovog modula os prilikom rada s datotekama i direktorijima operacijskog sustava te prikazati i objasniti dane primjere.

Ključne riječi: datoteka, direktorij, Linux, modul os, operacijski sustavi, Python.

1. Uvod

Operacijski sustav je softverski sloj programa koji služi kao sučelje između korisnika i hardvera te kontrolira izvršavanje programa. Svrha operacijskog sustava je osigurati okruženje u kojem korisnik može izvršavati programe na jednostavan i učinkovit način (GeeksforGeeks, 2022). Operacijski sustav bavi se dodjelom resursa i usluga u računalu. Neki od resursa koje dodjeluje operacijski sustavi su: memorije, procesori, uređaji i informacije. Sustavni programi pomažu operacijskom sustavu poboljšati upravljanje uključivanjem programa koji mu služe za upravljanje datotečnim sustavom i modula za upravljanje memorijom i I/O programima.

Većina računalnih sustava ima dva načina izvođenja operacija: kernelski način rada (engl. *kernel mode*) i korisnički način rada (engl. user mode) (Tanenbaum, 2015). Izvođenje sustavnih programa odvija se u kernelu. Kernel sustavnom programu omogućuje kontrolu osnovne razine na svim hardverskim uređajima računala, kao i ostalim resursima računala. Glavna uloga kernela je čitanje podataka iz memorije i pisanje podataka u memoriju te obrađivanje načina na koji monitor, miš i tipkovnica primaju i šalju podatke. Uz sustavne programe, bitnu ulogu imaju i aplikativni programi. Aplikativni programi služe korisnicima za rad na računalu, a za razliku od sustavnih programa rade u korisničkom načinu rada koji im ograničava pristup dijelovima računalnog sustava. Programiranjem sustavnih programa moguće je proširiti i unaprijediti funkcije operacijskog sustava. Ako je potrebno upravljati funkcijama operacijskog sustava, moguće je koristiti jedan od mnogih programskih jezika koji za to imaju potporu. Operacijski sustavi najčešće su programirani programskim jezicima C/C++ ili Python (freeCodeCamp, 2023).

U ovom radu opisano je korištenje Pythonovog modula os prilikom rada s datotekama i direktorijima operacijskog sustava, a na primjeru operacijskog sustava Linux. Prikazani su i objašnjeni primjeri funkcija koje se odnose na rad s datotekama i direktorijima, kao što su operacije stvaranja, brisanja, dohvaćanja atributa i druge. U početnom dijelu rada objašnjen je pojam operacijskog sustava i temeljne funkcije na kojima se zasniva. U drugom poglavlju rada dan je uvod u programski jezik Python i njegov modul os. Treće poglavlje rada sadrži opis rada s funkcijama Pythonovog modula os za upravljanje datotekama i direktorijima. U završnom poglavlju rada bit će iznesene završne misli te zaključak.

2. Python i modul os

Python je objektno orijentirani, interpretativni programski jezik visoke razine i opće namjene kojeg je kasnih 1980-ih godina počeo razvijati danski programer Guido van Rossum. Prva verzija jezika objavljena je 1991. godine. Inspiriran jezicima poput ABC-a i Modula-3, Rossum je htio stvoriti programski jezik koji bi programerima omogućio da koncepte izraze u što manje linija programskog koda, što bi rezultiralo učinkovitijim i lakšim procesom razvijanja. Python je zamijenio Javu kao najčešće korišteni jezik te je stekao reputaciju programskog jezika namijenjenog početnicima zbog svoje jednostavnosti koja i novim i iskusnim korisnicima omogućuje da više pažnje usmjere prema potpunom shvaćanju koncepata programiranja, umjesto da se zamaraju ostalim detaljima poput sintakse (teradata., 2022).

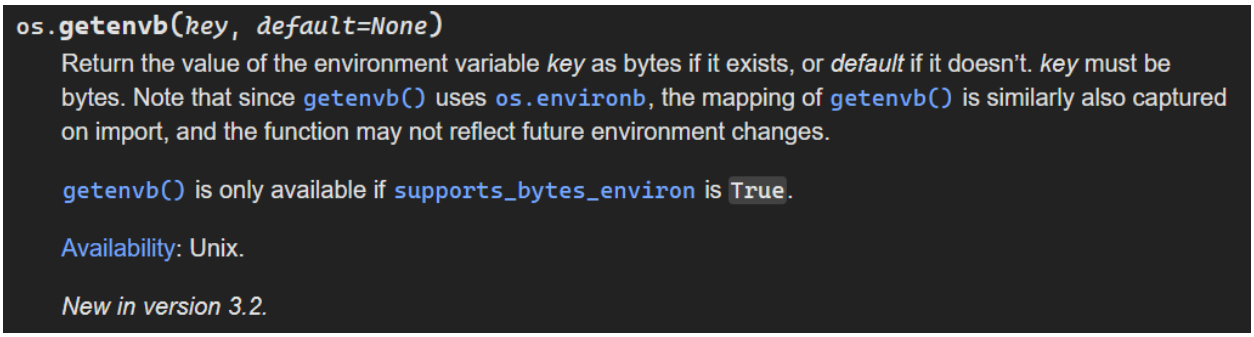
Pythonova kompatibilnost s više platformi omogućuje programerima pisanje programskog koda koji se može izvoditi na različitim operacijskim sustavima uključujući Windows, macOS i Linux. Od 2003. godine, Python je stalno rangiran među prvih deset najpopularnijih programskih jezika u TIOBE Programming Community Indexu, gdje je od prosinca 2022. godine najpopularniji programski jezik, našavši se ispred C, C++ i Jave (TIOBE indeks, 2023.). Svojom jasnom i sažetom sintaksom Pythonov kod nalikuje pseudo-kodu, što ga čini lakim za održavanje i razumijevanje. Ova karakteristika znatno smanjuje vrijeme potrebno za pisanje, ispravljanje pogrešaka i poboljšavanje programa te omogućuje programerima da se usredotoče na rješavanje problema umjesto da se bore sa složenom sintaksom.

Velika prednost Pythona je njegova svestranost. Zahvaljujući svojim ugrađenim strukturama podataka visoke razine može se koristiti u razvoju na poslužiteljskoj strani (engl. *server-side web development*)[1], razvoju softvera, matematici, skriptiranju sustava, automatizaciji, strojnom učenju ili kao jezik za povezivanje postojećih komponenti (GeeksforGeeks, 2022.). Dodatno, Python sadrži široku standardnu biblioteku s više od 200 modula kojima su sve klase, funkcije i varijable odvojene u posebne cjeline. Pravila pisanja koda u Pythonu zapisana su u priručnicima PEP (*Python Enhancement Proposals*). Posljednji dokument objavljen je 2021. godine pod nazivom PEP-8 (Python Software Foundation, PEP 8 – Style Guide for Python Code, 2023).

[1] Server side web development odnosi se na programe koji se pokreću na strani web poslužitelja za stvaranje dinamičkih web aplikacija.

Pythonov modul os je ugrađeni modul koji pruža široki raspon funkcija za interakciju s operacijskim sustavom. Omogućuje izvođenje raznih operacija povezanih sa stvaranjem i upravljanjem procesima, memorijom, rada s datotekama i direktorijima, I/O tokovima i slično. Pythonov modul os, uz ugrađene funkcije, sadrži i one koje nisu sastavni dio njegove jezgre. Takve funkcije organizirane su u module, a prije uporabe modula potrebno ih je najaviti ključnom riječju `import` (Slavuj, 2023). Funkcije koje modul os koristi trebale bi biti jednake za sve operacijske sustave, no nije nužno uvijek tako.

U dokumentaciji modula os prvo je opisana temeljna sintaksa pojedinih funkcija i metoda, nakon toga opisane su njezine funkcionalnosti ili vrijednosti koje funkcija vraća, slijede dustupnosti funkcije za pojedine operacijske sustave i na kraju je dan podatak o dostupnosti s obzirom na verziju Pythona (Slavuj, 2023). Prikaz opisa odabrane metode iz doumentacije dan je na Slici 1.


os.**getenvb**(*key*, *default=None*)

Return the value of the environment variable *key* as bytes if it exists, or *default* if it doesn't. *key* must be bytes. Note that since `getenvb()` uses `os.environb`, the mapping of `getenvb()` is similarly also captured on import, and the function may not reflect future environment changes.

`getenvb()` is only available if `supports_bytes_environ` is `True`.

Availability: Unix.

New in version 3.2.

Slika 1 Primjer zapisa iz službene dokumentacije Pythona

3. Rad s datotekama i direktorijima

Temeljna značajka svih operacijskih sustava su datoteke i direktoriji: računalne aplikacije trebaju spremati i dohvaćati podatke. Operacijski sustavi ne mogu funkcionirati bez korištenja datotečnog sustava (engl. *file system*). Windows 8 i novije verzije istog operacijskog sustava rabe NFTS upravljač datotečnog sustava dok su starije verzije koristile FAT-16 te FAT-32 (Tanenbaum, 2015). Linux rabi različite sustave datoteka kao što su ext2, ext3 i ZFS. Bez upravljača datotečnog sustava uređaj za pohranu bi sadržavao veliki broj podataka koji bi bili pohranjeni jedan uz drugoga pa ih operacijski sustav ne bi mogao razlikovati (freeCodeCamp, 2023). Odvajanje podataka i spremanje istih u datoteke omogućuje lakšu orijentaciju prilikom traženja.

Datoteke su logičke jedinice podataka koje stvaraju procesi (Tanenbaum, 2015). Procesi su zaduženi za stvaranje novih kao i čitanje već postojećih datoteka. Te datoteke su najčešće pohranjene na diskovima i ostalim uređajima za pohranu koji sadrže veliki broj različitih datoteka. Datotečni sustav upravlja stvaranjem, imenovanjem, strukturiranjem, zaštitom i drugim operacijma nad datotekama. U pravilu, datoteka bi trebala biti izbrisana samo ako je korisnik odluči izbrisati.

Imenovanje datoteka je vjerojatno najbitnija karakteristika rada s datotekama. Kada proces kreira datoteku i dodijeli joj naziv, ta datoteka će postojati i ako proces prestane s izvođenjem. Može joj se pristupiti putem drugih procesa, sve dok se zna njezino ime. Pravila imenovanja datoteka razlikuju se od operacijskog sustava do operacijskog sustava, ali svi trenutni operacijski sustavi dozvoljavaju korištenje slova i brojeva sve do duljine od 255 znakova. Mnogi operacijski sustavi podupiru imenovanje datoteka u dva dijela koji su međusobno odvojeni točkom (.), na primjer `dat.py` (Tanenbaum, 2015). Nazivi datoteka nakon točke (.) nazivaju se ekstenzije datoteka. Ekstenzije datoteka najčešće sadrže jedno do tri slova koja označuju tip datoteke. Korisnik na Windows operacijskom sustavu može izravno pristupiti programu koji je dodijeljen datoteci s ekstenzijom, dok na UNIX-u i operacijskim sustavima nalik UNIX-u nije tako jednostavno, jer ekstenzije služe samo kao podsjetnik korisniku koju informaciju mora prenijeti računalu. Primjeri postojećih ekstenzija su dani u Tablici 1, zajedno s njihovim objašnjenjima.

Tablica 1 Primjer ekstenzija datoteka

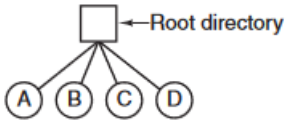
Ekstenzije	Objašnjenje
.bak	Sigurnosna kopija datoteke (engl. *Backup file*)
.c	C izvorni program (engl. *source program*)
.gif	*Graphic Interchange Format*
.hlp	Help datoteka
.html	Prezentacijski jezik za izradu web stranica (engl. *HyperText Markup Language*)
.jpg	Slika kodirana sa JPEG standardom
.mp3	MPEG layer 3 audio format
.o	Objekt datoteke (izlaz kompajlera)
.pdf	Portable Document Format datoteka
.ps	PostScript datoteka
.tex	Unos za TEX formatting program
.txt	Tekstualna datoteka
.zip	Sažeta arhiva

Operacijski sustavi za svaku datoteku pohranjuju njezin naziv i podatke koje sadrži, kao i druge podatke o datoteci, primjerice: vlasnik datoteke, tip, trenutna veličina i vrijeme zadnje izmjene podataka. Ti podatci nazivaju se atributi datoteka ili meta podatci.

Datoteke služe za spremanje podataka te ih je kao takve moguće kasnije dohvatiti po potrebi. Datotečni sustavi nude različite operacije koje se mogu vršiti nad njima, kao što su stvaranje, brisanje, čitanje, otvaranje, zatvaranje, pisanje, dodavanje, preimenovanje, dohvaćanje atributa (engl. *get attributes*) i postavljanje atributa (engl. *set attributes*) (Silberschatz, Galvin, & Gagne, 2008)

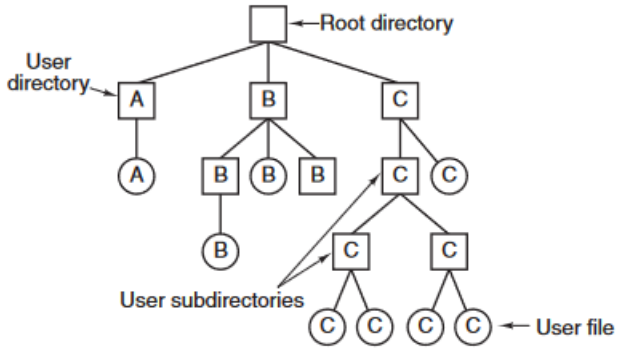
Direktoriji (ili mape) se koriste kako bi lakše organizirali i upravljali datotekama. Temeljni direktorij u datotečnom sustavu naziva se korijenski direktorij (engl. *root directory*). Takav direktorij sadrži sve poddirektorije i sve datoteke na jednom mjestu. Jednokorijenski operacijski

sustavi mogu sadržavati sve datoteke u jednom korijenskom direktoriju, dok višekorijenski sadrže više takvih direktorija na više particija. Jednokorijenski operacijski sustav za više particija imaju jedan korijenski direktorij, dok višekorijenski opercijski sustavi za više particija imaju više korijenskih direktorija. Jednokorijenski operacijski sustavi slični Unixu mogu imati particije u direktorijima `/`, `/home`, `/media`, … Višekorijenski operacijski sustav kao što je Windows particije označava kao `C:\`, `D:\`, `E:\` (Miletić, 2023). Primjer jednokorijenskog operacijskog sustava vidljiv je na Slici 2.



Slika 2 Primjer jednokorijenskog operacijskog sustava (Preuzeto: Tanenbaum (2015))

Pohranjivanje svih datoteka i podataka u jedan direktorij učinilo bi pronalazak željene datoteke gotovo pa nemogućim, stoga je metoda stabla, odnosno hijerarhijski sustav bolji način organizacije direktorija i datoteka. Primjer hijerarhijske organizacije vidljiv je na Slici 3.



Slika 3 Primjer hijerarhijske organizacije direktorija (Preuzeto: Tanenbaum (2015))

Želimo li pronaći traženu datoteku ili direktorij, ako nam je datotečni sustav organiziran metodom stabla, može se koristiti jedan od dva načina navođenja putanje (engl. *paths*): apsolutne adrese (engl. *absolute path name*) ili relativne adrese (engl. *relative path name*). Apsolutni način adrese

definira put kroz direktorije počevši od korijenskog direktorija prema traženome. Komponente su odvojene separatorom / kod operacijskih sustava temeljenih na UNIX-u i separatorom „\“ na operacijskom sustavu Windows. Primjer apsolutne adrese izgledao bi ovako: `/usr/home/mislav/cilj`, s kojim bi došli od korijenskog direktorija (prikazan prvim znakom / u navedenoj putanji) do traženog. Relativna adresa ne treba početi od korijenskog direktorija već od radnog direktorija: primjerice, dovoljno je vratiti se na višu razinu u stablu direktorija koji se u većini modernih operacijskih sustava označava sa „dotdot“ (`..`) (Tanenbaum, 2015.). Primjer relativne adrese izgledao bi ovako: `../../mislav/cilj`.

Operacije koje možemo vršiti nad direktorijima slične su onima koje se vrše nad datotekama. Kao kod datoteka, na direktorije se mogu primijeniti sljedeće operacije: stvaranje, brisanje, čitanje, otvaranje i zatvaranje. Operacije koje se mogu vršiti samo nad direktorijima su operacije *link* i *unlink*. Operacija *link* omogućuje datoteci pojavljivanje u više od jednog direktorija. dok operacija *unlink* služi kako bi vezu datoteke makli iz određenog direktorija.

3.1 Popis direktorija, dokumenata i atributa

Pythonov modul os sadrži brojne funkcije koje mogu biti od koristi prilikom prikazivanja sadržaja direktorija kao i filtriranja željenih rezultata (Python Software Foundation, os - Miscellaneous operating system interfaces, 2023). Želimo li izlistati sve datoteke i poddirektorije koje sadrži odabrani direktorij, koristimo jednu od dvije funkcije: `os.listdir()` ili `os.scandir()`. U starijim verzijama Pythona jedina dostupna metoda za ispis sadržaja direktorija na zaslon bila je funkcija `os.listdir()`. Na slici 3 slijedi primjer korištenja funkcije `os.listdir()`.

```
import os

podatci = os.listdir('Direktorij')
for primjer in podatci:
    print(primjer)
```

Slika 4 Primjer ispisa sadržaja direktorija korištenjem funkcije os.listdir()

Prikaz u ljusci (engl. *shell*) programa bi izgledao ovako:

```
mislav@DESKTOP-MO58PAP:~$ /usr/bin/python /home/mislav/primjer1/test2.py
pod_direktorij1
pod_direktorij2
pod_direktorij3
primjer1.py
tekst.txt
```

U primjeru je vidljiv ispis svih poddirektorija i datoteka koje se nalaze u direktoriju kojeg pretražujemo (radni direktorij). Da bi lakše pročitali sadržaje u direktoriju koristimo `for` petlju, kako bi nam podatci bili čitljiviji odnosno izlistani jedan ispod drugog, a ne jedan za drugim.

Verzija Pythona 3.5 predstavila je funkciju `os.scandir()` kao alternativu funkciji `os.listdir()`. Pozivanjem funkcije `os.scandir()` vraća se iterator, za razliku od funkcije `os.listdir()` koja vraća listu. U Pythonu iterator je objekt koji sadrži određeni broj vrijednosti koje se mogu pregledati, dok je lista niz pripadajućih elemenata. Na Slici 5 slijedi primjer ispisa sadržaja direktorija.

```
import os
podatci = os.scandir('Direktorij/')
podatci
```

Slika 5 Primjer ispisa sadržaja direktorija korištenjem funkcije os.scandir()

Prikaz u ljusci bi izgledao kao:

```
mislav@DESKTOP-MO58PAP:~$/usr/bin/python /home/mislav/primjer1/test2.py
<posix.ScandirIterator at 0x7f9a40590420>
```

Iz primjera je vidljivo da iterator pokazuje na sve dokumente u ciljnom direktoriju, ali ne ispisuje njihova imena. Taj problem moguće je rješiti rabeći sličnu metodu kao kod funkcije

`os.listdir()`. U primjeru koristimo `for` petlju kako bi ispisali sve unesene podatke u varijabli `podatci`, dok će `name` ispisati imena tih datoteka i direktorija. Na Slici 6 prikazan je primjer ispisa sadržaja.

```
import os

podatci = os.scandir('Direktorij/')
for unos in podatci:
        print(unos.name)
```

Slika 6 Primjer ispisa sadržaja direktorija korištenjem funkcije os.scandir()

Na ovaj način ispis ovog primjera u ljusci trebao bi biti isti kao u prijašnjem primjeru s funkcijom `os.listdir()`:

```
mislav@DESKTOP-MO58PAP:~$ /usr/bin/python /home/mislav/primjer1/test2.py
pod_direktorij1
pod_direktorij2
pod_direktorij3
primjer1.py
tekst.txt
```

U slučaju potrebe filtriranja direktorija i dokumenata u poddirektoriju mogu se također koristiti funckije `os.listdir()` i `os.scandir()`. Filtriranjem program ispisuje željene podatke koji odgovaraju kriterijima pretrage (npr. pretražuju se samo datoteke). Funkcija `os.listdir()` za razliku od `os.scandir()` koristi `os.path` kako bi mogla filtrirati putanju da ispiše traženi pojam. Ako je potrebno izlistati na zaslon sve podirektorije, potrebno je samo iskoristiti `os.path.isdir()` (vraća vrijednost `True` u slučaju da direktorij postoji), a ako je potrebno izlistati datoteke, koristi se `os.path.isfile()` (vraća vrijednost `True` u slučaju da datoteka postoji). (Python Software Foundation, os.path – Common pathname manipulations, 2023.)

Funkcija `os.path.join()` koristi se kako bi povezali „put“ i „unos“ kod pretraživanja direktorija. Primjer je vidljiv na Slici 7.

```
import os

put = 'Direktorij/'
for unos in os.listdir(put):
    if os.path.isfile(os.path.join(put, unos)):
        print(unos)
```

Slika 7 Primjer lista direktorija korištenjem funkcije os.listdir() s metodom os.path.isdir()

```
mislav@DESKTOP-MO58PAP:~$ /usr/bin/python /home/mislav/primjer1/test2.py
primjer.txt
tekst.txt
```

Kod filtriranja korištenjem funkcije `os.scandir()` želimo li ispisati sve datoteke dovoljno je dodati klasu `.is_file()` ako je potrebno izlistati sve datoteke na zaslon, a ako je potrebno izlistati samo poddirektorije, onda klasu `is_dir()`. Na Slici 8 prikazan je primjer ispisa sadržaja koristeći se funkcijom `os.scandir()` i klasom `.is_file()`.

```
import os

put = 'Direktorij/'
with os.scandir(put) as podatci:
    for unos in podatci:
        if unos.is_dir():
            print(unos.name)
```

Slika 8 Primjer ispisa sadržaja direktorija korištenjem funkcije os.scandir() sa klasom .is_file()

```
mislav@DESKTOP-MO58PAP:~$ /usr/bin/python /home/mislav/primjer1/test2.py
pod_direktorij1
pod_direktorij2
```

```
pod_direktorij3
```

U ovom primjeru poziva se klasa `.is_dir()` na svaki unos kojeg vraća funkcija `os.scandir()`. Poddirektorij će biti ispisan ako unos vrati vrijednost `True`.

3.2 Popis atributa datoteka

Python olakšava dohvaćanje atributa datoteka, kao što su veličina datoteke, vrijeme zadnje izmjene, vrijeme kreiranja, podatci o dozvolama i vlasniku datoteke. Ti atributi dohvaćaju se pomoću funkcija `os.stat()` i `os.scandir()`. Velika prednost kod `os.scandir()` u odnosu na `os.listdir()` je to što `os.scandir()` dohvaća listu datoteka u direktoriju s njihovim atributima. Kod `os.listdir()` potrebno je prvo dohvatiti listu datoteka, a zatim željene atribute za svaki dokument. Na Slici 9 slijedi primjer dohvaćanja atributa datoteka.

```python
import os
with os.scandir('Direktorij/') as sadrzaj:
    for unos in sadrzaj:
        if unos.is_file():
         info = unos.stat()
         print("Vrijeme posljednji izmjene datoteke: ",info.st_atime)
         print("Veličina datoteke: ", info.st_size)
         print("ID vlasnika datoteke: ", info.st_uid)
         print("Broj čvrstih veza:", info.st_nlink)
```

Slika 9 Primjer dohvaćanja atributa datoteka

```
mislav@DESKTOP-MO58PAP:~$ /usr/bin/python /home/mislav/primjer1/test2.py
Vrijeme posljednji izmjene datoteke:  1693055104.7732656
Veličina datoteke:  0
ID vlasnika datoteke:  1001
Broj čvrstih veza: 1
Vrijeme posljednji izmjene datoteke:  1694356487.3074312
Veličina datoteke:  308
ID vlasnika datoteke:  1001
Broj čvrstih veza: 1
```

U prikazanom primjeru svaki unos u `scandir()` iterator koristi `.stat()` metodu. Ta metoda koristi se kod dohvaćanja podataka o datoteci ili direktoriju na kojeg iterator pokazuje. Koristeći `.stat()` dohvaćamo željene informacije o datoteci. U gornjem primjeru dodali smo klasu `is_file()` kako bi prikazali samo datoteke u direktoriju te uz metodu `.stat()` koristimo atribut `st_atime` kako bi se prikazao posljednji datum izmjene datoteke u sekundama, atribut *`st_size`* koji prikazuje veličinu datoteke u bajtovima, atribut `st_uid` koji prikazuje ID vlasnika datoteke te `st_nlink` koji prikazuje broj čvrstih veza. Osim navedenih atributa koje smo koristili u primjeru postoje i drugi kao što su: `st_mode` (prikazuje tip datoteke i dozvole datoteke), `st_ino` (prikazuje vrijednost datoteke ovisno o platformi koja se koristi), `st_gid` (prikazuje grupni ID vlasnika datoteke), st_dev (prikazuje identifikator uređaja na kojem je datoteka pohranjena), `st_atime_ns` (prikazuje vrijeme zadnjeg pristupa datoteci u nanosekundama kao cijeli broj), `st_mtime_ns` (prikazuje vrijeme zadnje promjene datoteke u nanosekundama kao cijeli broj), `st_ctime_ns` (ovisno o operacijskom sustavu, na Windows operacijskom sustavu prikazuje vrijeme stvaranja datoteke u nanosekundama, a na Unix operacijskom sustavu prikazuje vrijeme zadnje izmjene meta podataka izražen u sekundama).

3.3 Stvaranje datoteka

Datoteka je temljenja značajka programskog jezika Pyhton. Upravljanje datotekama korištenjem programskog jezika Python daje korisnicima mogućnosti kreiranja datoteka sa različitim dozvolama koristeći 6 različitih metoda pristupa (engl. *access modes*) (freeCodeCamp, 2023). Te metode su:

1. Read Only ('r') – metoda otvara datoteku koja se može samo čitati.
2. Read and Write ('r+') – metoda otvara datoteku koja se može čitati i uređivati.
3. Write Only ('w') – metoda otvara datoteku samo za čitanje.
4. Write and Read ('w+') – metoda otvara datoteku koja se može uređivati i čitati.
5. Append Only ('a') – metoda otvara datoteku u kojoj se može pisati. U slučaju da datoteka ne postoji ona se kreira.
6. Append and Read ('a+') – metodom se datoteka može čitati i uređivati. Ako ne postoji nova datoteka, bit će kreirana

Primjer kreiranja *Write Only* datoteke prikazan je na Slici 10.

```
with open('novitekst.txt', 'w') as f:
    upis = 'kreirali smo ovu datoteku koristeći metodu write only'
    f.write(upis)
```

Slika 10 Primjer kreiranja datoteke

3.4 Stvaranje direktorija

Nakon određenog vremena pisanja programa u Pythonu, treba kreirati direktorije za pohranjivanje programa (koji su u datotekama). Kreiranje direktorija omogućeno je funkcijama `os.mkdir(put, mode=0o777)` i `os.makedirs(put, mode=0o777, exist_ok=False)`. Funkcija `os.mkdir()` koristi se kod kreiranja imenovanog puta direktorija sa specifičnim načinom pristupa (engl. *mode*), gdje zadani brojevi imaju vrijednost `0o777`. Vrijednost `0o777` mijenja dozvole datoteke tako da korisnik, grupa i ostali datoteku mogu pisati, čitati i pokretati. U oktalnom zapisu brojevi su predstavljeni na način 0oUGO gdje slovo U predstavlja korisnika, G grupu, a O ostale (SmallBASIC, 2023). Svaki broj u zapisu predstavlja određene dozvole. Dozvole su vidljive u Tablici 2.

Tablica 2 Prikaz dozvola u oktalnom zapisu

Vrijednost	Dozvole
0	nema dozvola
1	x (pokretanje)
2	w (pisanje)
3	w+x
4	r (čitanje)
5	r+x
6	r+w
7	r+w+x

Na primjer, ako je potrebno da samo korisnik može čitati i pisati u datoteci prilikom stvaranja datoteke specifičnom načinu pristupa (engl. *mode*), a pripadnici korisničke grupe i ostali mogu samo čitati sadržaj, dozvole će imati vrijednost `0o644`.

U slučaju kreiranja direktorija koji već postoji, funkcija `os.mkdir()` vraća grešku *FileExistError*. Način na koji se može rukovati pojavom greške je koristeći izjavu (engl. *statement*) `try/except` s kojom se u bloku `try` pokušava kreirati direktorij. Ako se pojavi `FileExistError`, pokrenut će se `except` blok koji će ispisati poruku o pogreški na zaslon. Želimo li izbjeći ispisivanje poruke, moguće je koristiti izjavu `pass`. Primjer stvaranja datoteke prikazan je na Slici 11.

```
import os

ime_direktorija = 'novi'

try:
    os.mkdir(ime_direktorija)
except FileExistsError:
    print('Direktorij već postoji.')
```

Slika 11 Primjer stvaranja datoteka

Funkcija `os.makedirs()` ponaša se slično kao `os.mkdir()`. Te dvije funkcije razlikuju se po tome što `os.makedirs()` ne treba samo kreirati individualne direktorije, već se može koristiti i za kreiranje cjelokupne grane stabla, odnosno može kreirati sve potrebne direktorije kako bi osiguralo da putanja postoji. Navedena funkcija sadrži naziv putanje kao obavezni argument, dok su argumenti `mode` i *`exist_ok`* opcionalni. Argumentom `mode` određujemo sve dozvole direktorija koje će se dodijeliti korisnicima pri njegovu stvaranju, dok argument `exist_ok` određuje radnju koja će se izvesti ako direktorij već postoji. U slučaju da je `exist_ok` postavljen na vrijednost `False` i direktorij postoji, funkcija će vratiti grešku, ali ako je `exist_ok` postavljen na vrijednost `True`, funkcija neće vratiti grešku. Primjer korištenja funkcije `os.makedirs()` vidljiv je na Slici 12 te je prikaz stvorenih direktorija vidljiv na Slici 13.

```
import os


os.makedirs('Škola/Razredi/Učenici', mode=0o777)
```

Slika 12 Primjer stvaranja više direktorija naredbom os.makedirs()

```
./Škola:
Razredi

./Škola/Razredi:
Učenici

./Škola/Razredi/Učenici:
```

Slika 13 Prikaz stvorenog direktorija i poddirektorija korištenjem naredbe ls -R

U ovom primjeru kreirana je struktura direktorija `Škola/Razredi/Učenici` te su svim korisnicima dane dozvole čitanja, pisanja i uređivanja. Zadani mode je `0o777`, a dozvole datoteka postojećih nadređenih direktorija se ne mijenjaju.

3.5 Kretanje po stablu direktorija i obrada datoteka

Tijekom programiranja jedan od čestih zadataka je „prolazak“ stablom direktorija i obrada datoteka u stablu. Python omogućuje oba zadatka s njegovom ugrađenom funkcijom `os.walk(top, topdown=True)`. Funkcija `walk` izvodi „prolazak“ kroz direktorij (koji je naveden putanjom) i svim poddirektorijima koje sadrži. Funkcija vraća listu svih datoteka i poddirektorija koje sadrže direktoriji koje prođe i provjeri. Argumentom *topdown* generiranom stablu određuje se redoslijed prolaska kroz direktorije, bilo to od vrha prema dnu ili od dna prema vrhu. Primjer funkcije `os.walk()` vidljiv je na Slici 14.

```
import os

for putanja, naziv, datoteke in os.walk('/home/mislav/Direktorij', topdown= True):

    print('Pronađene datoteke: %s ', %putanja)
    for naziv_datoteke in datoteke:
        print(naziv_datoteke)
```

Slika 14 Primjer funkcije os.walk()

```
mislav@DESKTOP-MO58PAP:~$ /usr/bin/python /home/mislav/primjer1/test2.py
Found directory: /home/mislav/Direktorij
primjer.txt
```

```
tekst.txt
primjer1.py
Found directory: /home/mislav/Direktorij/Sub_Direktorij1
neki_tekst.txt
Found directory: /home/mislav/Direktorij/Sub_Direktorij2
datoteka.py
Found directory: /home/mislav/Direktorij/Sub_Direktorij3
```

Funkcija `walk` u petlji pri svakoj iteraciji vraća tri vrijednosti:

1. Imena svih trenutnih datoteka
2. Listu svih poddirektorija u trenutnom direktoriju
3. Listu svih trenutnih datoteka u direktoriju

Primjer navednoga vidljiv je u ranijem primjeru. U slučaju da je potrebno pregledati sve datoteke u obrnutom rasporedu, argumentu `topdown=True` treba promijeniti vrijednost navedenog atributa u `topdown=False`. Na taj način program će početi ispisivati sve poddirektorije i njihove sadržaje prije sadržaja korijenskog direktorija. Ovakav način ispisa koristan je u situacijama u kojima želimo rekurzivno brisati datoteke i direktorije. Rekurzivno brisanje počinje od poslijednje datoteke ili direktorija stabla. Taj način je bitan kod brisanja direktorija jer nam svaki direktorij mora biti prazan kako bi bio obrisan. Prema zadanim postavkama, funkcija `os.walk` ne „prolazi“ do simboličnih poveznica koje se izvršavaju u direktoriju. Takvo ponašanje može biti riješeno uvođenjem argumenta `followlinks=True`.

3.6 Brisanje datoteka i direktorija

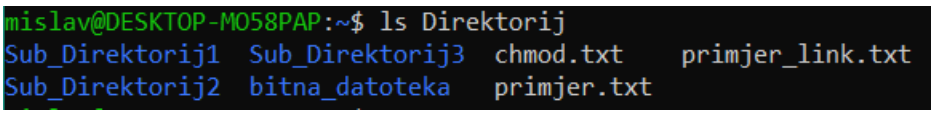
Korištenjem funkcija kao što su `os.remove()` ili `os.unlink()`, Pythonov modul os nudi mogućnost brisanja datoteka. Slijedi primjer brisanja datoteke na Slici 15 i Slici 16.

```
import os

datoteka = 'tekst.txt'
lokacija = "/home/mislav/Direktorij/"
```

```
path = os.path.join(lokacija, datoteka)

os.remove(path)
print('Datoteka: %s je uspješno obrisana!', %datoteka)
```

Slika 15 Primjer brisanja datoteke funkcijom os.remove()

```
mislav@DESKTOP-MO58PAP:~$ /usr/bin/python /home/mislav/primjer1/test2.py
Datoteka: tekst.txt je uspješno obrisana!"
```



Slika 16 Prikaz brisanja datoteke

U gornjem primjeru prvo smo datoteci koju želimo izbrisati definirali ime te smo definirali putanju na kojoj se ta datoteka nalazi. Koristeći `os.path.join()` povezali smo „lokaciju“ direktorija i „datoteku“ koju želimo izbrisati. Na kraju, koristeći se funkcijom `os.remove(path)`, obrisana je željena datoteka.

Funkcija `os.unlink()` radi na sličan način kao i funkcija `os.remove()`. Pozivanjem obiju funkcija briše se željena datoteka. Mora se napomenuti da će obje funkcije vratiti grešku OSError ako putanja pokazuje na direktorij umjesto datoteke. Takve slučajeve možemo izbjeći korištenjem funkcije `os.path.isfile()` kako bi provjerili je li ciljni element zapravo datoteka. Ako datoteka postoji, ona će biti izbrisana, a ako je datoteka koju želimo izbrisati zapravo direktorij, program će vratiti poruku o pogreški. Na Slici 17 prikazan je primjer pojave greške.

```
import os

datoteka = 'home/mislav/Direktorij/popis.txt'

if os.path.isfile(datoteka):
```

```
    os.remove(datoteka)
else:
    print('Greška: %s nije važeće ime datoteke!', %datoteka)
```

Slika 17 Primjer greške koristeći funkciju os.remove()

```
mislav@DESKTOP-MO58PAP:~$ /usr/bin/python /home/mislav/primjer1/test2.py
Greška: home/mislav/Direktorij/popis.txt nije važeće ime datoteke!
```

Kod brisanja direktorija koriste se funkcije `os.rmdir()` i `os.removedirs()`. Ako je potrebno izbrisati jedan direktorij, treba koristiti funkciju `os.rmdir(put)`. Ako je potrebno izbrisati direktorij funkcijom, treba uzeti u obzir da on mora biti prazan. U slučaju da direktorij nije prazan, program vraća grešku OSError. Na Slici 18 prikazan je primjer korištenja funkcije `os.rmdir()`.

```
import os

direktorij = 'prazni_direktorij'
lokacija = '/home/mislav/Direktorij/'

path = os.path.join(lokacija, direktorij)

os.rmdir(path)
print('Direktorij: %s je uspješno izbrisan' %direktorij )
```

Slika 18 Primjer brisanja direktorija funkcijom os.rmdir()

```
mislav@DESKTOP-MO58PAP:~$ /usr/bin/python /home/mislav/primjer1/test2.py
Direktorij: prazni_direktorij je uspješno izbrisan!
```

Način na koji se može rukovati programom u slučaju pojave greške je korištenjem izjave (engl. *statement*) `try/except`. U bloku `try` pokušava se izbrisati direktorij koji nije prazan ili korisnik nema dopuštenje brisanja direktorija, pokrenut će se blok `except` koji će ispisati poruku na zaslon. Na Slici 19 slijedi primjer rukovanja OSError greškom .

```
import os

direktorij = 'prazni_direktorij'
lokacija = '/home/mislav/Direktorij/'

path = os.path.join(lokacija, direktorij)

try:
    os.rmdir(path)
    print('Direktorij: %s je uspješno izbrisan!' %direktorij)
except OSError as error:
    print(error)
    print('Direktorij: %s ne može biti izbrisan!' %direktorij )
```

Slika 19 Primjer rukovanja OSError greškom

```
mislav@DESKTOP-MO58PAP:~$ /usr/bin/python /home/mislav/primjer1/test2.py
[Errno 2] No such file or directory: '/home/mislav/Direktorij/prazni_direktorij'
Direktorij: prazni_direktorij ne može biti izbrisan!
```

Ako je potrebno izbrisati više direktorija, treba koristiti funkciju `os.removedirs()`. Funkcija `os.removedirs()` rekurzivno briše sve prazne direktorije u putanji, tj. funkcija počinje brisanje direktorija od posljednjeg direktorija u putanji. Direktoriji će biti izbrisani ako ne sadrže druge direktorije ili datoteke, odnosno ako su prazni. Na Slici 20 slijedi primjer brisanja više direktorija.

```
import os


direktorij = '/home/mislav/Direktorij/Prazni_Dir/Prazni_Dir2'
```

```
os.removedirs(direktorij)

print('Prazni direktoriji u putanji %s su uspješno izbrisani!', %direktorij)
```

Slika 20 Primjer brisanja više direktorija funkcijom os.removedirs()

```
mislav@DESKTOP-MO58PAP:~$ /usr/bin/python /home/mislav/primjer1/test2.py
Prazni direktoriji u putanji /home/mislav/Direktorij/Prazni_dir/Prazni_Dir2 i su
uspješno izbrisani!
```

Rukovati greškom može se na isti način kao i kod funkcije `os.rmdir()`. Na Slici 21 slijedi primjer rukovanja OSError.

```
import os

direktorij = '/home/mislav/Primjer1/Direktorij/primjer.txt'


try:
    os.removedirs(direktorij)
    print("Direktorij je uspješno izbrisan!")

except OSError as error:
    print(error)
    print('Direktorij ne može biti izbrisan.')
```

Slika 21 Primjer rukovanja sa OSError u funkciji os.removedirs()

```
mislav@DESKTOP-MO58PAP:~$ /usr/bin/python /home/mislav/primjer1/test2.py
[Errno 2] No such file or directory'/home/mislav/Primjer1/Direktorij/primjer.txt'
Direktorij ne može biti izbrisan.
```

3.7 Ostale mogućnosti rada s datotekama i direktorijima

Osim navedenih funkcija, Pythonov modul os nudi i druge funkcije i mogućnosti koje mogu pomoći prilikom rada s datotekama i direktorijima. Neke od funkcija su: `os.getcwd()`,

`os.chdir()`, `os.link()`, `os.access()`, `os.umask()`, `os.rename()`, `os.chmod()`, itd.

Funkcija `os.getcwd()` vraća trenutnu adresu radnog direktorija apsolutnog oblika. Na Slici 22 slijedi primjer.

```
import os

td = os.getcwd()

print('Trenutni direktorij je: ', td)
```

Slika 22 Primjer funkcije os.getcwd()

```
mislav@DESKTOP-MO58PAP:~$ /usr/bin/python /home/mislav/primjer1/test2.py
Trenutni direktorij je:  /home/mislav/primjer1
```

Funkcija `os.chdir(putanja)` mijenja radni direktorij u onaj zadan adresom putanje. Slijedi primjer na Slici 23 te lokacija datoteke prije korištenje Python skripte na Slici 24 i nakon korištenja na Slici 25.

```
import os

os.chdir('/home/mislav/primjer1/Direktorij')

print('Radni direktorij je promijenjen')
```

Slika 23 Primjer korištenja funkcije os.chdir()

```
mislav@DESKTOP-MO58PAP:~$ /usr/bin/python /home/mislav/primjer1/test2.py
Radni direktorij je promijenjen
```

```
mislav@DESKTOP-MO58PAP:~/primjer1$ pwd
/home/mislav/primjer1
```

Slika 24 Lokacija prije korištenja python skripte

```
mislav@DESKTOP-MO58PAP:~/primjer1/Direktorij$ pwd
/home/mislav/primjer1/Direktorij
```

Slika 25 Lokacija nakon korištenja python skripte koristeći funkciju os.chmod()

Funkcija `os.link(izvor, destinacija)` kreira čvrstu vezu (engl. *hard link*) kojom se definirana datoteka veže s imenom u putanji. Takva veza omogućuje da više različitih datoteka pokazuju na isti skup podataka. Argumentom `izvor` određuje se datoteka koju je potrebno povezati s drugim direktorijem koji je označen s argumentom `destinacija`. Primjer kreiranja čvrste veze vidljiv je na Slici 26 i Slici 27.

```
import os

izvor = '/home/mislav/Direktorij/primjer_link.txt'

lokacija = '/home/mislav/Direktorij/bitna_datoteka/primjer_link(link).txt'

os.link(izvor, lokacija)

print('Čvrsta veza je kreirana!')
```

Slika 26 Primjer kreiranja čvrste veze funkcijom os.link()

```
mislav@DESKTOP-MO58PAP:~$ /usr/bin/python /home/mislav/primjer1/test2.py
Čvrsta veza je kreirana!
```

```
mislav@DESKTOP-MO58PAP:~/primjer1/Direktorij/bitna_datoteka$ ls
'primjer_link(link).txt'
```

Slika 27 Primjer kreirane čvrste veze

Funkcija `os.access(put, mode)` koristi argument `mode` kako bi provjerila dozvole datoteke. Funkcija provjerava postojanje `(os.F_OK)`, čitljivost `(os.R_OK)`, upisivost

(os.W_OK) i izvedivost (os.X_OK) odabranedatoteke. Funkcija vraća boolean vrijednosti True ili False. Slijedi primjer korištenja funkcije na Slici 28.

```
import os

put1 = os.access('test1.py', os.F_OK)
print("Postojanje:",put1)

put2 = os.access('test1.py', os.R_OK)
print("Čitljivost:", put2)

put3 = os.access('test1.py', os.W_OK)
print("Upisivost:", put3)

put4 = os.access('test1.py', os.X_OK)
print("Izvedivost:", put4)
```

Slika 28 Primjer korištenja funkcije os.access()

```
mislav@DESKTOP-MO58PAP:~$ /usr/bin/python /home/mislav/primjer1/test2.py
Postojanje: True
Čitljivost: True
Upisivost: True
Izvedivost: False
```

Funkcija os.chmod(put, mode) koristi se prilikom promjene vrijednosti puta odabreane datoteke ili direktorija u numeričku vrijednost. Neke od vrijednosti modea su: stat.S_ISUID (postavlja korisnički ID prilikom pokretanja), stat.S_ISVTX (pohranjuje tekstualnu sliku nakon pokretanja), stat.S_IRWXU (samo vlasniku omogućuje čitanje, pisanje i pokretanje datoteke), stat.S_IRWXO (ostalim korisnicima omogućuje čitanje,pisanje i pokretanje datoteke) i mnoge druge. Primjer korištenja funkcije vidljiv je na Slici 29, te stanje dozvola na Slici 30 i Slici 31.

```python
import os, stat

os.chmod('Direktorij/novitekst.txt', stat.S_IRWXU)
print('Datotekom može upravljati samo vlasnik!')

os.chmod('Direktorij/novitekst.txt', stat.S_IRWXO)
print('Datotekom mogu upravljati svi korisnici!')
```

Slika 29 Primjer funkcije os.chmod()

```
mislav@DESKTOP-MO58PAP:~$ /usr/bin/python /home/mislav/primjer1/test2.py
Datotekom može upravljati samo vlasnik!
Datotekom mogu upravljati svi korisnici!
```

```
-rwx------ 1 mislav mislav   54 Sep 12 20:05 novitekst.txt
```

Slika 30 Primjer stat.S_IRWXU

```
---------- 1 mislav mislav   54 Sep 12 20:05 novitekst.txt
```

Slika 31 Primjer stat.S_IRWXO

Funkcija `os.umask(mask)` koristi se prilikom postavljanja početnih dozvola nad datotekom ili direktorijem (dakle, nad novostvorenim elementima datotečnog sustava). Funkcija prima jedan argument u oktalnom zapisu kojim se dodaju nove dozvole, dok program vraća vrijednost prijašnje korisničke maske. Važno je napomenuti da će program vratiti vrijednost maske u dekadskom zapisu. Moguće dozvole dane su ranije u Tablici 2. Slijedi primjer korištenja funkcije `os.umask()` na Slici 32.

```python
import os

mask = 0o777

umask = os.umask(mask)

print("Trenutna korisnička maska: ", mask)
```

```
print("Prijašnja korisnička maska:", umask)
```

Slika 32 Primjer korištenja funkcije os.umask()

```
mislav@DESKTOP-MO58PAP:~$ /usr/bin/python /home/mislav/primjer1/test2.py
Current umask: 511
Previous umask: 18
```

Funkcija `os.rename(staro_ime, novo_ime)` koristi se za preimenovanje željene datoteke ili direktorija. Ako je potrebno imenovati više datoteka ili direktorija rekurzivno, koristi se funkcija `os.renames(stara_imena, nova_imena)`. Slijedi primjer funkcije na Slici 33 te ispis sadržaja direktorija prije pokretanja skripte na Slici 34 i nakon pokretanja skripte na Slici 35.

```
import os

staro_ime = 'Direktorij/myfile.txt'

novo_ime = 'Direktorij/novimyfile.txt'

os.rename(staro_ime, novo_ime)
print('Datoteka je uspješno preimenovana!')
```

Slika 33 Primjer funkcije os.rename()

```
mislav@DESKTOP-MO58PAP:~$ /usr/bin/python /home/mislav/primjer1/test2.py
Datoteka je uspješno preimenovana!
```

```
mislav@DESKTOP-MO58PAP:~/primjer1/Direktorij$ ls
bitna_datoteka  kreiranje_append_datoteke.txt  novitekst.txt    pod_direktorij2  prazni_direktorij  test2.py
data.txt        myfile.txt                     pod_direktorij1  pod_direktorij3  primjer1.py
```

Slika 34 Primjer sadržaja drijektorija prije preimenovanja datoteke myfile.txt

```
mislav@DESKTOP-MO58PAP:~/primjer1/Direktorij$ ls
bitna_datoteka  kreiranje_append_datoteke.txt  novitekst.txt    pod_direktorij2  prazni_direktorij  test2.py
data.txt        novimyfile.txt                pod_direktorij1  pod_direktorij3  primjer1.py
```

Slika 35 Primjer sadržaja direktorija nakon pokretanja python skripte

4. Zaključak

Cilj ovog rada bio je pojasniti pojam operacijskog sustava, opisati programski jezik Python te poseban naglasak staviti na prikaz i opis rada njegovog modula os prilikom upravljanja datotekama i direktorijima.

Zadatak operacijskog sustava je upravljati resursima računala i dodjelom usluga računalu. Pomoću programskog jezika Python i korištenjem njegovog modula os moguće je upravljati značajkama operacijskog sustava.

Značajke bez kojih operacijskih sustavi ne mogu funkcionirati su datoteke i direktoriji. Datoteke i direktoriji operacijskom sustavu služe za pohranu i dohvaćanje podataka, dok su za stvaranje i čitanje datoteka zaduženi procesi. Datotečni sustavi nude različite operacije koje se mogu vršiti nad elementima datotečnog sustava. Iz primjera opisanih u ovome radu vidljive su mogućnosti i načini rada funkcija koje služe za upravljanje datotekama i direktorijima.

Zaključno, modul os iznimno je koristan jer je, poput samog Pythona, svestran te sadrži mnoge različite funkcije koje mogu pomoći tijekom izvršavanja zadataka. Uz to, jednako ga mogu koristiti i iskusni programeri, ali i oni koji to tek žele postati.

5. Literatura

freeCodeCamp. *File Handling in Pyhton – How to Create, Read, Write to a file.* URL: https://www.freecodecamp.org/news/file-handling-in-python/ (Pristupljeno 10. rujna 2023.)

freeCodeCamp. *What is a File System? Types of Computer File Systems and How They Work.* URL: https://www.freecodecamp.org/news/file-systems-architecture-explained/ (Pristupljeno 15. kolovoza 2023.)

Geeksforgeeks. *OS Module in Python with examples*. URL: https://www.geeksforgeeks.org/os-module-python-examples/ (Pristupljeno 22. kolovoza 2023.)

Miletić, V. *Python: općenite usluge operacijskog sustava: osnovna sučelja*. URL: https://group.miletic.net/hr/nastava/materijali/python-modul-os/ (Pristupljeno 15. kolovoza 2023.)

Python Software Foundation. *os — Miscellaneous operating system interfaces.* URL: https://docs.python.org/3/library/os.html#files-and-directories (Pristupljeno 17. kolovoza 2023.)

Python Software Foundation. *os.path – Common pathname manipulations.* URL: https://docs.python.org/3/library/os.path.html (Pristupljeno 17. kolovoza 2023.)

Python Software Foundation. *PEP 8 -- Style Guide for Python Code.* URL: https://www.python.org/dev/peps/pep-0008/ (Pristupljeno 17. kolovoza 2023.)

Slavuj, V. *Rad s procesima (nastavni materijali)*. Rijeka. (Pristupljeno 15. kolovoza 2023.)

Silberschatz, A., Galvin, P. B., & Gagne, G. (2008). *Operating system concepts (9th edition).* Hoboken: Wiley.

SmallBASIC. URL: https://smallbasic.github.io/pages/file.html (Pristupljeno 9. rujna 2023.)

Tanenbaum, A. S. (2015). *Modern Operating Systems (fourth edition)*. Pearson.

teradata. *What is python?* URL: https://www.teradata.com/Glossary/What-is-Python (Pristupljeno 15. kolovoza 2023.)

TIOBE Software BD. *Tiobe index*. URL: https://www.tiobe.com/tiobe-index/ (Pristupljeno 15. kolovoza 2023.)

tutorialspoint. *Operating System – Overview.* URL: *https://www.tutorialspoint.com/operating_system/os_overview.html* (Pristupljeno 17. kolovoza 2023.)

Popis slika

Popis tablica