**AMALIY MASHG‘ULOT UCHUN O‘QUV MATERIALLARI**

**1-Mavzu:** “Python dasturlash tili” faniga kirish va asosiy tushunchalari.

**11-mashg‘ulot**. Fayllar va kataloglar bilan ishlash.

O‘quv savollari:

1. Faylni ochish. Fayllar bilan ishlovchi metodlar.
2. os modulining imkoniyatlari.
3. Fayl va katalog yo‘lini o‘zgartirish.
4. Katalog va fayl bilan ishlovchi funksiya va metodlar.

**1. Faylni ochish. Fayllar bilan ishlovchi metodlar.**

Python dasturlash tilidagi fayllar bilan o‘zaro aloqada bo‘lish foydalanuvchiga ilova tomonidan qayta ishlangan ma’lumotlarni saqlash imkoniyatini beradi va foydalanuvchi keyinchalik istalgan qulay vaqtda faylga murojaat qilishi mumkin bo‘ladi. Ushbu python dasturlash tilining asosiy funktsiyalari fayllarni yaratish, faylga ma’lumotlarni yozish va fayldan ma’lumotlarni o‘qishni ancha osonlashtiradi.

**Faylni yaratish va ochish**

Fayl bilan ishlash uchun albatta avval uni yaratish kerak. Buni standart operatsion tizim vositalari yordamida kerakli katalogga o‘tish va **\*.txt** formati bilan yangi hujjat yaratish orqali amalga oshirish ham mumkin. Biroq, shunga o‘xshash harakat Python dasturlash tilida **open()** metodi yordamida amalga oshiriladi. Bu metodga **fayl nomi** va uni **qayta ishlash rejimi** parametrlar sifatida berilishi kerak.

Quyidagi kod fayl o‘zgaruvchisini yangi hujjatga ishora qilishini ko‘rsatadi. Agar ushbu dasturni ishga tushirilsa, u manba kodi saqlanadigan papkada **test.txt** matn faylini yaratadi.

file\_txt = **open**("test.txt", "**w**")

file\_txt.**close**()

Agar **test.txt** nomli fayl kod katalogida allaqachon mavjud bo‘lsa, dastur yangi hujjat yaratmasdan u bilan ishlashni davom ettiradi. Ko‘rib turganingizdek, fayl nomi **open()** metodining birinchi parametridir. Undan so‘ng ma’lumotlarni **qayta ishlash usulini** ko‘rsatadigan maxsus belgi parametr sifatida keladi. Bu yerda ikkinchi parametr sifatida kelgan “w” - fayl faqat yozish uchun ochilayotganini bildiradi.

Endi ma’lumot yo‘qolmasligini ta’minlash uchun fayl ustida turli amallarni bajarilgandan so‘ng, uni **close()** funktsiyasidan foydalanib yopish kerak bo‘ladi.

Oldingi misolda faylga kirish uchun nisbiy yo‘l ishlatilgan bo‘lib, u qattiq diskdagi ob’yektning joylashuvi haqida to‘liq ma’lumotni o‘z ichiga olmaydi. Ularni o‘rnatish uchun **open()** funksiyaga birinchi argument sifatida mutlaq yo‘lni ko‘rsatilishi kerak. Bunday holda, test.txt hujjati dastur papkasida emas, balki D diskidagi qaysidir bir katalokga joylashgan bo‘ladi.

file\_txt = **open**(**r**"D:\test\test.txt", "**w**")

file\_txt.**close**()

Bu yerda faylga yo‘lni ko‘rsatishda satrdan oldin **r** belgisi qo‘yiladi. Aks holda, kompilyator "\t" ketma-ketligini maxsus operator sifatida ko‘rib chiqadi va xatolik kelib chiqadi.

**Faylni ochilish rejimlari**

Barcha dasturlash tillaridagi kabi Python dasturlash tilida ham faylni ochishda ishlatiladigan maxsus belgilardan foydalanladi. Ular faylni ochish rejimi deb ataladi va faylni qanday maqsadda ochishni aniq beradi. Ularning barchasi quyidagi jadvalda keltirilgan, unda ularning belgisi va qisqacha mazmuni keltirilgan:

5-jadval

| **Belgi** | **Mazmuni** |
| --- | --- |
| «r» | O‘qish uchun ochiq (standart) |
| «w» | Yozish uchun ochiladi. Agar fayl ko‘rsatilgan kotalogda mavjud bo‘lmasa, yangisi yaratiladi. Yozilgan ma’lumot hujjatdagi barcha ma’lumotlarni o‘rniga yoziladi |
| «a» | Yozish uchun ochiladi. Yozilgan ma’lumot hujjatning oxiriga qo‘shiladi |
| «b» | Faylni ikkilik rejimda ochish |
| «t» | Faylni matn rejimida ochish (standart) |
| «+» | O‘qish va yozish uchun bir vaqtning o‘zida ochish. Ushbu amal alohida o‘zi ochish rejimi sifatida ishlatilmaydi. a+, r+, w+ kabi ishlatiladi. |

**Open()** metodining ikkinchi argumentidan foydalanib, turli xil fayl rejimlarini birlashtirish mumkin. Masalan, yozma ma’lumotlarni ikkilik rejimda o‘qish uchun "rb" ni belgilash kerak.

Yana bir misol: "r+" va "w+" o‘rtasidagi farq shundaki, “w+” - agar fayl mavjud bo‘lmasa, yangi fayl yaratiladi. Birinchi holda, xatolik kelib chiqadi. "r+" va "w+" dan foydalanish faylni o‘qish va yozish uchun ochadi.

**Usullari**

**open()** funksiyasi tomonidan qaytarilgan ob’yekt mavjud faylga tavsifni o‘z ichiga olgan to‘rtta parametrni qaytaradi. Quyidagi jadvalda ularning barchasi nomlari va ma’nolari keltirilgan:

6-jadval

| **Nomi** | **Ma’nosi** |
| --- | --- |
| name | fayl nomini qaytaradi |
| mode | ochilgan rejimni qaytaradi |
| closed | fayl yopiq bo‘lsa true ni, ochiq bo‘lsa false qiymatini qaytaradi |
| softspace | agar faylning chiqishi alohida bo‘sh joy belgisini o‘z ichiga olmasa, true qiymatini qaytaradi |

Faylning xususiyatlarini ko‘rsatish uchun kirish operatoridan, ya’ni nuqtadan foydalanish va keyin buni **print()** funktsiyasiga parametr sifatida berish kifoya bo‘ladi.

**Masalan** :

f = **open**(r"D:\test\test.txt", "**w**")

print(f.**name**)

f.**close**()

**Natijasi:**

D:\test\test.txt

**Faylga ma’lumot yozish**

Faylga ma’lumot yozish **write()** usuli yordamida amalga oshiriladi. Usul mavjud faylga ishora qiluvchi ob’yektda chaqiriladi. Shuni esda tutish kerakki, buni amalga oshirish uchun avval hujjatni ochish funktsiyasidan foydalanib ochish va "w" belgisi bilan yozish rejimini belgilash kerak bo‘ladi. **Write()** usuli argument sifatida matn fayliga yoziladigan ma’lumotlarni qabul qiladi. Quyidagi kod misolida "Salom dunyo!" satrini kiritilishi ko‘rsatilgan:

file\_txt = **open**("test.txt", "**w**")

file\_txt.**write**("Salom dunyo!")

file\_txt.**close**()

Agar ilgari yozilgan ma’lumotlarga yangi ma’lumotlarni qo‘shish kerak bo‘lsa, u holda ochish rejimi sifatida "a" belgisini ko‘rsatib, **open** funksiyasini qayta chaqirish kerak. Aks holda, **test.txt** faylidagi barcha ma’lumotlar butunlay o‘chirib tashlanadi. Quyidagi kod misolida faylga ma’lumot qo‘shish uchun matnli hujjatni ochadi. Shundan so‘ng unda " Salom dunyo!" satri hujjat oxiriga joylashtiriladi. Shunday qilib, **test.txt** faylida "Salom dunyo! Salom dunyo!" satri paydo bo‘ladi. Bularning barchasidan so‘ng, faylni majburiy yopish haqida unutmang.

file\_txt = **open**("test.txt", "a")

file\_txt.**write**(" Salom dunyo!")

file\_txt.**close**()

Matn fayliga ma’lumotlarni yozishning eng oddiy protsedurasi shunday amalga oshiriladi. Shuni ta’kidlash kerakki, Python dasturlash tilida hujjatlar bilan yanada ilg‘or ishlash uchun ko‘plab qo‘shimcha vositalar mavjud bo‘lib, ular yaxshilangan yozishni ham o‘z ichiga oladi.

**Ikkilik ma’lumotlarni yozing**

Ikkilik ma’lumotlarni yozishda "wb" rejimidan foydalanish kerak. Utf8 kodlashda satr yozishga misol:

file\_txt = **open**(‘test.dat’, ‘**wb**’)

file\_txt.**write**(bytes(‘наш строк’, **‘utf8’**))

file\_txt.**close**()

**Fayldan o‘qish**

Fayldan ma’lumotni o‘qish uchun mavjud faylga ishora qiluvchi ob’yektda **read()** metodini chaqirish kerak bo‘ladi. Matn faylini ochishda **open()** funksiyaning ikkinchi parametri sifatida "r" belgidini ko‘rsatishni ham unutmaslik kerak.

Quyidagi misolda o‘qish **test.txt** dan ma’lumotni **print()** funksiyasiga qaytaradi, so‘ngra ma’lumotni ekranga chop etadi. Avvalgidek, dastur hujjatni **close()** usuli bilan yopish bilan yakunlanadi. **Read()** metodi butun son parametrini ham qabul qilishi mumkin, bu o‘qish uchun belgilar sonini uzatish uchun ishlatiladi. Misol uchun, agar 5 ni kiritilsa, dastur faqat *“Salom”* satrini o‘qiydi.

**try**:

file\_txt = **open**("test.txt", "**r**")

print(file\_txt.**read**())

file\_txt.**close**()

**except** **FileNotFoundError**:

print(‘Fayl topilmadi!’)

**except** **IOError**:

print(‘Qandaydir xatolik!’)

Iltimos, ochishda agar ko‘rsatilgan fayl topilmasa xatolik yuz berishi mumkinligini unutmang. Shunday qilib, try bloki bilan orab olish kerak edi. Pythonda buni oldini olish uchun **with** konstruktoridan foydalanish mumkin. Bu holda xatoliklarni ko‘rib chiqish va hatto faylni yopish ham shart emas.

**Ikkilik ma’lumotlarni o‘qish**

Agar ma’lumotlar ikkilik turda bo‘lsa, **open()** funksiyasining ikkinchi parametri sifatida "rb" dan foydalanish kerak:

**try**:

f\_txt = **open**("test.dat", "**rb**")

b = f\_txt.**read**(1)

str = ""

**while** True:

b = f\_txt.**read**(1)

**if** b == ’b’:

**break**

str += b.**hex**()

print(str)

f\_txt.**close**()

**except** **IOError**:

print(‘error’)

**Natijasi:**

81d182d180d0bed0bad0b081d182d180d0bed0bad0b0

Bu yerda fayl bayt bo‘yicha o‘qiladi. Har bir bayt o‘n oltilik sanoq sistemasi ko‘rinishdagi satr shaklida natija beradi. **Print**() funksiyasi olingan qatorni chop etadi.

**with as konstrultori**

Matnli fayllarni qayta ishlashni biroz avtomatlashtirish uchun bir qator **with as** iboralaridan foydalanish tavsiya etiladi. Ushbu operatrolardan foydalanilsa, yopilishi kerak bo‘lgan hujjatda **close()** usulini chaqirishning hojati yo‘q, chunki bu avtomatik ravishda sodir bo‘ladi. Buni quyidagi qism dasturda ko‘rib chiqamiz. Odatdagidek, ekranga satr ma’lumotlarini chop etish uchun **print()** usuli qo‘llaniladi:

**with** open(‘test.txt’, ‘r’) **as** file\_txt:

print(file\_txt.**read**())

Bundan tashqari, bu holda istisno bilan shug‘ullanish kerak emas. Agar ko‘rsatilgan nomga ega fayl bo‘lmasa, **with** operatoridagi ichki kodli satrlar bajarilmaydi.

Python dasturlash tilining ko‘rib chiqilgan xususiyatlari yordamida foydalanuvchi ma’lumotlarni fayllarga o‘qish va yozishning asosiy operatsiyalarini osongina bajarishi mumkin.

**2. os modulining imkoniyatlari**

Python dasturlash tilining standart kutubxonasidagi OS moduli odatda o‘rnatilgan OT va shaxsiy kompyuter fayl tizimi bilan ishlash uchun qo‘llaniladi. Unda qattiq diskda saqlanadigan fayllar va kataloglar bilan ishlash uchun juda ko‘p foydali o‘rnatilgan metodlar mavjud. OS moduli bilan ishlaydigan dasturlar OT turiga bog‘liq bo‘lmasdan, boshqa platformalarga osongina ko‘chiriladi.

**OS moduli nima?**

Pythondagi OS moduli operatsion tizim bilan ishlash uchun kerak bo‘ladigan turli funktsiyalar kutubxonasidir. Unga kiritilgan usullar **operatsion tizim turini aniqlash**, **muhit o‘zgaruvchilariga kirish**, **katalog** va **fayllarni boshqarish** imkonini beradi:

* berilgan yo‘lda ob’yektning mavjudligini tekshirish;
* baytlarda o‘lchamlarni aniqlash;
* o‘chirish;
* nomini o‘zgartirish va boshqalar.

OS funksiyalarini chaqirayotganda, ularning ba’zilari joriy opratsion tizim tomonidan qo‘llab-quvvatlanmasligi mumkin.

OS moduli metodlaridan foydalanish uchun kutubxonani “import os” orqali chaqirish kerak. Ushbu ko‘rsatmani dasturning boshida joylashtirish tavsiya etiladi.

OS modulidagi mavjud metodlardan foydalanuvchi turli maqsadlarda foydalanishi mumkin. Quyida operatsion tizim haqida ma’lumot olish imkonini beruvchi eng mashhur metodlar keltirilgan. Shuningdek, kompyuterning qattiq diskidagi xotirada saqlangan fayl va papkalar haqida ma’lumot olish mumkin.

**OT haqida ma’lumot olish**

Joriy OT nomini bilish uchun faqat **name()** metodidan foydalaning. O‘rnatilgan platformaga qarab, u o‘zining qisqa nomini satr ko‘rinishida qaytaradi. Quyidagi dastur Windows 10 kompyuterida ishga tushirildi, shuning uchun **name()** funktsiyasining natijasi sifatida **nt** qaytarildi. Buni odatiy print() funksiyasi bilan ko‘rish mumkin.

import **os** *#os modulini chaqirish*

print(**os**.**name**) *#OT nomini bilish*

**Natijasi:**

nt

**environ()** metodidan foydalanib, kompyuterning konfiguratsiyasi bilan bog‘liq ma’lumotlarni olish mumkin. Undan **os** kutubxonasini chaqirish orqali foydalanuvchi   konsolga yoki satr o‘zgaruvchisiga chiqadigan muhit o‘zgaruvchilari bilan katta lug‘at turidagi ma’lumot oladi. Shunday qilib, tizim diskining nomini (C:\....), katalogining manzilini, tizim nomini va boshqa ko‘plab ma’lumotlarni bilib olish mumkin. Quyidagi misol **environ**() metodidan foydalanishni ko‘rsatadi.

import **os**

print(**os**.**environ**)

**environ**({‘ALLUSERSPROFILETXT’: ‘C:\\ProgramData’, …})

**getenv** funktsiyasidan foydalanib, turli xil muhit o‘zgaruvchilariga kirish mumkin. Buning uchun quyidagi misoldagi kabi kerakli o‘zgaruvchi nomini argument sifatida beriladi. Bunda **print** displeydagi TMP haqidagi ma’lumotlarni konsolga chop etadi.

import **os**

print(os.**getenv**("TMP"))

**Natijasi:**

C:\Users\IT\AppData\Local\Temp

**Ishchi katalogni o‘zgartirish**

Odatda dasturning ishchi katalogi uning manba kodi bilan hujjatni o‘z ichiga olgan katalog bo‘ladi. Buning yordamida, agar u ushbu papkada joylashgan bo‘lsa, faylga mutlaq yo‘lni aniqlashning iloji bo‘lmaydi. Qattiq diskdagi ishchi katalogning to‘liq manzilini qaytaruvchi **getcwd** funksiyasidan foydalanib joriy katalog haqida ma’lumot olinadi. Quyidagi qism dastur, agar ushbu usulning natijasini chop etish uchun o‘tkazsa nima bo‘lishini ko‘rsatadi. Ko‘rib turganingizdek, ishchi katalog C tizim diskidagi dastur katalogidir.

import **os**

print(**os**.**getcwd**())

**Natijasi:**

C:\Users\IT\source\repos\program

Agar so‘ralsa, ishchi katalogni **os** kutubxonasidan **chdir** usuli yordamida xohishga ko‘ra sozlash mumkin. Buni amalga oshirish uchun uni yangi katalogning mutlaq manzilini parametr sifatida o‘tkazish kerak. Agar ko‘rsatilgan yo‘l haqiqatda mavjud bo‘lmasa, dastur xatolik qaytaradi va dastur ishlashi to‘xtatiladi. Quyidagi kod misoli D diskidagi papka deb nomlangan yangi ishchi katalogga o‘tishni ko‘rsatadi.

import **os**

os.**chdir**(r"D:\folder")

**Natijasi:**

D:\folder

**3. Fayl va katalog yo‘lini o‘zgartirish**

Ko‘pincha, faylning to‘liq nomi mavjud bo‘lganda, uning kengaytmasini bilish kerak bo‘ladi. Yoki foydalanuvchi kerakli formatni kiritgan yoki kiritmaganligi noma’lum bo‘lsa, kerakli formatni qo‘shish kabi masalalar mavjud. Ushbu mavzu fayl nomi bilan ishlashning asosiy usullari haqida bo‘ladi.

**Faylga mutlaq yo‘l**

Pythonda faylga mutlaq yo‘lni bilish uchun OS modulidan foydalanish kerak bo‘ladi. **Path** sinfida faylga absolyut yo‘lni qaytaruvchi - **abspath** metodi mavjud:

import **os**

r = os.**path**.**abspath**(‘file.txt ‘)

print(r)

**Natijasi:**

C:\python\file.txt

Buning uchun standart **pathlib** kutubxonasidan ham foydalanish mumkin. U Python 3.4 dan boshlab asosiy kutubxonalarga kiritilgan. Undan oldin uni pip install pathlib buyrug‘i yordamida o‘rnatish kerak edi. U turli xil operatsion tizimlarda fayl tizimi yo‘llari bilan ishlash uchun mo‘ljallangan va bu muammoni hal qilish uchun juda mos keladi.

import **pathlib**

r = pathlib.**Path**(‘file.txt ‘)

print(r)

**Natijasi:**

C:\python\file.txt

**Fayl nomi**

To‘liq yo‘l qatoridan fayl nomini olish uchun os modulining **basename()** metodidan foydalaniladi.

import **os**

file\_name = os.**path**.**basename**(r’C:\python\file.txt ‘)

print(file\_name)

**Natijasi:**

file.txt

Bu yerda maxsus belgilarni oddiy belgilar sifatida ishlatish uchun qatordan oldin **r** belgisi qo‘yilgan.

**Kengaytmasiz fayl**

Endi Pythonda kengaytmasiz fayl nomini qanday topish mumkinligini aniqlaylik. Bunday hollarda os modulining **splittext** metodidan foydalaniladi:

from **os** import **path**

full\_file\_name = path.**basename**(r’C:\python\file.tar.gz ‘)

name = **path**.**splitext**(full\_file\_name)[0]

print(name)

**Natijasi:**

file.tar

Ko‘rinib turibdiki, **gz** arxivatorining oxirgi kengaytmasi o‘chirilgan, siqilmagan **tar** arxivining kengaytmasi esa fayl nomida qolgan.

Agar faqat nom kerak bo‘lsa, birinchi nuqtadan keyin kelgan qabul qilingan satrning barcha belgilarini olib tashlash mumkin. Nuqta belgisini ham olib tashlaylik.

Oldingi misolni quyidagi kod bilan to‘ldiraylik:

index = name.**index**(‘.’)

print(**name**[:**index**])

**Natijasi:**

file

**Fayl kengaytmasi**

Pythonda fayl kengaytmasini xuddi shu **splitext** funksiyasidan foydalanib olish mumkin. Ushbu metod  o‘zidan kortej tipli ma’lumot qaytaradi. Kortejning birinchi elementi fayl nomi bo‘lsa, ikkinchi element kengaytmadir. Bunday holda, bizga ikkinchi element kerak. Ikkinchi elementning indeksi birga teng, chunki kortej elementlari ham massivlar elementlari kabi noldan indexlanadi.

from **os** import **path**

full\_file\_name = path.**basename**(r’C:\python\file.tar.gz ‘)

name = path.**splitext**(full\_file\_name)[1]

print(name)

**Natijasi:**

.gz

Xuddi shunday, **pathlib** kutubxonasining **suffix** metodidan foydalanish ham mumkin:

from **pathlib** import **Path**

print(**Path**(r’C:\python\file.tar.gz‘).**suffix**)

**Natijasi:**

.gz

Ammo ushbu holatlarda ikkita kengaytma mavjud. Ularni **suffixes()** funktsiyasi yordamida aniqlab olish mumkin. U elementlari kengaytmalar bo‘lgan ro‘yxatni qaytaradi. Quyida kengaytmalar ro‘yxatini olish misoli keltirilgan:

from **pathlib** import **Path**

print(**Path**(r’C:\python\file.tar.gz‘).**suffixes**)

**Natijasi:**

[‘.tar’, ‘.gz‘]

To‘liq yo‘ldan fayl nomi yoki kengaytmasini olish yoki faylga mutlaq yo‘lni olish uchun **os** va **pathlib** kutubxonalaridan foydalanish tafsiya etiladi.

**4. Fayllar bilan ishlovchi metodlar**

Python dasturlash tilining asosiy xususiyatlari nafaqat matnli fayllardagi ma’lumotlarni qayta ishlash, balki ularni har tomonlama boshqarish imkonini beradi. Buning uchun bir nechta maxsus kutubxonalar mavjud bo‘lib, ularning o‘rnatilgan funktsiyalari kompyuterdagi fayllar bilan **nusxa ko‘chirish**, **o‘chirish**, **nomini o‘zgartirish** va boshqa turdagi operatsiyalarni ta’minlaydi.

**Fayl mavjudligini tekshirish**

Kompyuterning qattiq diskida uning yo‘qligi bilan bog‘liq bo‘lishi mumkin bo‘lgan matnli hujjat bilan ishlashda oddiy xatoliklarga yo‘l qo‘ymaslik OS kutubxonasining **exists()** metodidan foydalaniladi. Ushbu funksiya orqali ko‘rsatilgan yo‘lda fayl mavjudligini tekshirish mumkin va natijas sifatida mantiqiy True yoki False qiymatlarni qaytaradi. Quyidagi Python misoli D qattiq diskining **test** nomli katalogida **test.txt** va **test10.txt** fayllari mavjudligini tekshiradi. PRINT() funksiyasi esa D da faqat birinchi hujjat mavjudligini ko‘rsatadi.

import os

print(os.path.exists("D:\\test\\test.txt"))

print(os.path.exists("D:\\test\\test10.txt"))

**Natijasi:**

True

False

Ba’zan, hujjatlar bilan ishlashda, berilgan yo‘l bo‘ylab nafaqat ob’yektning mavjudligini tekshirish, balki fayl yoki papka ekanligini tekshirish ham kerak bo‘ladi. Bunday holda OS kutubxonasining **isfile** funksiyasi dasturchiga kompyuterning qattiq diskidagi ma’lum bir manzilda olingan ob’yekt papka emas, balki fayl ekanligiga ishonch hosil qilish imkoniyatini beradi. Bu usul **path** sinfida ham mavjud. Quyidagi misolda **isfile** funksiyasi **test.txt** faylini va D diskning test papkasini argument sifatida qabul qilganda qanday natija ko‘rsatadi. PRINT() funksiyasining chiqishidan ko‘rinib turibdiki, birinchi holatda True, keyin esa False ko‘rsatiladi:

import **os**

print(os.**path**.**isfile**("D:\\test\\test.txt"))

print(os.**path**.**isfile**("D:\\test\\folder"))

**Natijasi:**

True

False

**Faylni nusxalash**

**shutil** kutubxonasi qattiq diskdagi ob’yektlarning nusxalarini yaratish uchun bir nechta foydali funktsiyalarni o‘z ichiga oladi. Faylni manba katalogiga tezda nusxalash uchun **shutil** kutubxonasini qo‘shgandan so‘ng **copyfile()** usulidan foydalanish kerak. Bu yerda birinchi argument asl hujjat, ikkinchi argument esa taklif qilingan yangi fayldir. Esda tutingki, **meta ma’lumotlar** emas, faqat faylning tarkib nusxalanadi. Quyidagi misolda D diskidagi test.txt dan test2.txt ga ma’lumotlarni ko‘chiriladi. **copyfile()** funktsiyasi yangi yaratilgan hujjatning manzilini ham qaytaradi.

import **shutil**

shutil.**copyfile**("D:\\test\\test.txt", "D:\\test\\test2.txt")

**shutil** modulidan o‘rnatilgan **copy()** usuli Pythonga faylni asl nomini saqlab qolgan holda belgilangan papkaga nusxalash imkonini beradi. Quyidagi kod misoli **test.txt** dan ma’lumotni D diskida joylashgan **folder** deb nomlangan katalogdagi ob’yektga qanday ko‘chirishni ko‘rsatadi. **Copy**() funktsiyasi bilan oldingi holatda bo‘lgani kabi, faqat ichki ma’lumotlar uzatiladi, lekin hujjatni yaratish va tahrirlash sanasi haqida ma’lumotlar qaytarilmaydi.

import **shutil**

shutil.**copy**("D:\\test\\test.txt", "D:\\folder")

Matn faylidan ma’lumotni, shuningdek, u haqidagi barcha ma’lumotlarni to‘liq nusxalash uchun tayyor nusxa **copy2** usulidan foydalanish kerak. Uni ishlatish usuli **copy** funksiyasi bilan bir xil. Birinchi parametr o‘rniga bu asl faylning manzili, ikkinchi parametr esa yangi hujjatning joylashuvi va nomini belgilaydi. Quyida kontent va metamaʼlumotlar folser papkasidan **test2.txt** ga koʻchiriladigan misol keltirilgan:

import **shutil**

shutil.**copy2**("D:\\test\\test.txt", "D:\\folder\\test2.txt")

**Fayl o‘chirish**

Ob’yektning nomi va diskdagi aniq joylashuvini bilib olinsa, undan qutulish juda oson. Yuqorida aytib o‘tilgan operatsion tizim kutubxonasidan **remove** usuli bu vazifani hal qilishga yordam beradi. Buning uchun keraksiz faylning to‘liq manzilini unga parametr sifatida beriladi.

Quyida D diskining **test** katalogidagi **test.txt** faylini oʻchirish uchun remove metodidan qanday foydalanishga misol keltirilgan:

import **os**

os.**remove**("D:\\test\\test.txt")

**Fayl hajmini bilib olish**

Fayl hajmini baytlarda qaytaradigan OS modulidagi **getsize** standart funksiyasidan foydalanib, qattiq diskdagi istalgan ob’yektning aniq hajmini aniqlash mumkin. Bu yerda **getsize** funksiyasi argumenti hujjatning kompyuter xotirasidagi joylashuvidir. **getsize** funksiyasi ma’lumotlariga ko‘ra, **test.txt** faylining o‘lchami 7219. **Print()** funksiyasi buni ekranga chop etadi:

import **os**

print(os.**path**.**getsize**("D:\\test\\test.txt"))

**Natijasi:**

7219

Pythonda fayl hajmini hisoblashning yana bir usuli - uni **open** funksiyasi orqali ochish va **seek** funksiyasini chaqirish. U faylning boshidan oxirigacha ma’lumotlarni o‘qish uchun maydonni parametr sifatida uzatilishi kerak. Natijada, matnli faylga yo‘l orqali **tell** usulini chaqirish kerak va keyin uning ishining natijasini konsolga chiqarish uchun print() funksiyasi uchun yuborish kerak.

import **os**

**with** open("D:\\test\\test.txt") **as** **file**:

**file**.**seek**(0, **os.SEEK\_END**)

print(file.tell())

**Natijasi:**

7219

**Fayl nomini o‘zgartirish**

Hujjat nomini nafaqat tizim vositalari, balki OS modulining tayyor funksiyalari yordamida ham o‘zgartirish mumkin. Buning uchun **rename()** metodidan foydalanish maqsadga muvofiq bo‘ladi. Bu metod asl va yangi fayl nomlarini parametr sifatida qabul qiladi. Quyidagi misolda D diskining **test** katalogidagi **test.txt** fayli nomi test1.txt deb o‘zgartirish ko‘rsatilgan:

import **os**

os.**rename**("D:\\test\\test.txt", "D:\\test\\test1.txt")

Xuddi shunday, **shutil** modulidan **copy** metodi yordamida Pythonda fayl nomini o‘zgartirish mumkin. Buning uchun ushbu kutubxonani dasturga import qilish, funktsiyaga hujjatning joylashuvi va yangi nomini berish kifoya qiladi:

import **shutil**

**shutil**.**move**("D:\\test\\test.txt", "D:\\test\\test1.txt")

Shunday qilib, Python tilidagi fayllar bilan ishlash bo‘yicha asosiy operatsiyalar bir nechta o‘rnatilgan kutubxonalar, jumladan **os** va **shutil** yordamida amalga oshiriladi. Ushbu modullarning funktsiyalari diskda **fayl mavjudligini tekshirish**, uni bir nechta **turli rejimlarda nusxalash**, shuningdek **o‘chirish**, **nomini o‘zgartirish** va **hajmini ko‘rsatish** imkonini beradi

**Nazorat savollari:**

1. Mutlaq fayl yo‘li qanday olinadi?
2. Fayl nomini qanday olish mumkin?
3. Fayl kengaytmasini qanday olish mumkin?
4. Fayl kengaytmasini o‘zgartirish qanday amalga oshiriladi?
5. Qanday turdagi fayllar mavjud?
6. Fayl yaratishning qanday usullari mavjud?
7. Faylni ochishning qanday rejimlari mavjud?
8. Faylni faqat yozish uchun ochish qanday amalga oshiriladi?
9. Faylni faqat o‘qish uchun ochish rejimi qanday?
10. Faylni faqat o‘qish va yozish uchun ochish rejimi qanday?
11. Faylni ochish rejimlariga “+” belgisi qo‘shilsa qanday o‘zgarish bo‘ladi?
12. with as operatorining qo‘llanilishi qanday?