PYTHON DASTURLASH TILIDA FAYLLAR BILAN ISHLASH

Ahtamqulov Muhriddin Ahtamqul o‘g‘li
Samarqand davlat universiteti Urgut filiali assistenti e-mail:
muhriddinahtamqulov@mail.com

Taniyeva Farzona Xolmo‘min qizi
Samarqand davlat universiteti Urgut filiali talabasi
e-mail:ftaniyeva@gmail.com
Tel: +998934958828

ANNOTATSIYA

Ushbu maqolada Python dasturlash tilida fayllar bilan ishlash, fayllarning turlari, ularni yaratish va saqlash, matnli fayllar va ular ustida ishlash atroflicha yoritib o‘tilgan, hamda python dasturlash tilida fayllar bilan ishlashning afzalliklari va kamchiliklari, matn va binar fayllar haqida fikr yuritilgan.

Kalit so‘zlar: *Python dasturlash tili, fayllar, txt, open (), read (), ochish, o‘qish, yopish, afzalliklari, kamchiliklari, ikkilik fayl.*

ABSTRACT

This article covers working with files in the Python programming language, file types, their creation and storage, text files and working with them, as well as the advantages and disadvantages of working with files in the Python programming language, text and binary files.

Keywords: *Python programming language, files, txt, open (), read (), open, read, close, advantages, disadvantages, binary file.*

АННОТАЦИЯ

В данной статье рассматривается работа с файлами на языке программирования Python, типы файлов, их создание и хранение, текстовые файлы и работа с ними, а также преимущества и недостатки работы с файлами на языке программирования Python, текстовыми и двоичными файлами. .

Ключевые слова: *язык программирования Python, файлы, txt, open(), read(), открыть, прочитать, закрыть, преимущества, недостатки, бинарный файл.*

KIRISH

Hozirgacha biz asosan bitta fayl ichida ishladik, ya’ni bizning barcha kodimiz bitta .py faylida saqlangan va o‘sha fayldan ishlaydi. Biroq, haqiqiy dunyoda bizning ko‘plab dasturlarimiz bir nechta fayllarda saqlanadi. Bundan tashqari, biz ba’zi sevimli kod parchalari va funksiyalarini keyinchalik foydalanish uchun fayllarda

saqlashimiz mumkin. Bu biz dasturchilar – hozir sizni o‘z ichiga olgan tarzda ishlaymiz.

Python dasturlash tili fayllar bilan ishlash uchun juda qulay vositalardan biridir. U bizga fayllarni ochish, o‘qish, yozish va yopish imkoniyatini beradi. Shuningdek, Python fayllar bilan ishlashda sizga turli xil xususiyatlar, qulayliklar va imkoniyatlarni taklif etadi.

MUHOKAMA VA NATIJALAR

Pythonda turli xil fayl turlari bilan ishlash imkoniyati mavjud bo‘lib, shartli ravishda ularni ikki turga bo‘lish mumkin: matn va binar fayllar. Matn fayllari, masalan, kengaytmasi cvs, txt, html, umuman, matn shaklida ma’lumot saqlaydigan barcha fayllarlarni o‘z ichiga oladi. Binar fayllar tasvirlar, audio va video fayllar va boshqalardan iborat. Fayl turiga qarab u bilan ishlash biroz farq qilishi mumkin. Python dasturlash tili tarkibida fayllarga murojat qilish uchun fayl nomiga to‘g‘ridan to‘g‘ri murojat qilib bo‘lmaydi, fayllarga murojat qilish uchun fayllarni dastur bilan bog‘lash uchun alohida o‘zgaruvchi qabul qilinadi.

*Fayllarni faollashtirish.*Python dasturlash tilida fayllar bilan ishlashda .txt kengaytmali fayllardan foydalanish maqsadga muvofiq bo‘ladi. Dastur tarkibida fayllarga murojat qilish uchun albatta oldin .txt kengaytmali faylni yaratib alohida joyga saqlab quyish kerak bo‘ladi. Dasturlash tilida ishlatiladigan fayllar nomi ikki turda bo‘ladi.

1) Fizik nomi;

2) Mantiqiy nomi.

Faylning fizik nomi kompyuterda .txt kengaytmali nom bilan saqlangan nomi hisoblanadi. Faylning mantiqiy nomi esa dastur tarkibida faylning fizik nomi bilan bog‘lashga xizmat qiladigan nomi hisoblanadi[1].

Python dasturlash tilida faylni ochish uchun “open ()” funksiyasi ishlatiladi. Bu funksiya fayl nomi va ochish rejimini qabul qiladi. Python fayllarni turli xil rejimlarda ochishi mumkin: faqat o‘qish uchun, yozish uchun, qo‘shish uchun, binary rejimda, va hokazo.

Quyida fayllarni ochishning usullari ko‘ramiz:

a) **“r”**- O‘qish - Standart qiymat. Faylni o‘qish uchun ochadi, agar fayl mavjud bo‘lmasa, xatolik beradi.

b) **“a”**- Qo‘shish - faylni qo‘shish uchun ochadi, agar u mavjud bo‘lmasa, uni yaratadi.

c) **“w”**- Write - faylni yozish uchun ochadi, agar u mavjud bo‘lmasa, uni yaratadi.

d) **“x”**- Yaratish - Belgilangan faylni yaratadi, agar fayl mavjud bo‘lsa, xatoni qaytaradi.

Fayl ochish quyidagicha bo‘ladi. Agar bizda qandaydir .txt fayl bo‘lsa, uni o‘qish rejimida ochish bunday qilamiz:

```
f = open("fayl_nomi.txt")
```

Eslatma uchun aytish kerakki, fayl nomini yozib unga murojaat qilmoqchi bo‘lganimizda fayl papkada joylashgan bo‘lsa, manzilini to‘liq ko‘rsatishimiz maqsadga muvofiq.

```
f = open("D:\fayllarim\fayl_nomi.txt")
```

Hozir biz faylni ochib o‘qish uchun ochib ko‘ramiz. Avval .txt kengaytmali biror faylga 4 – 5 qatorli matn kiritamiz, uni python faylimiz joylashgan papkaga bitr nom bilan saqlaymiz. Uni open () funksiyasi bilan ochamiz va read () funksiyasi bilan o‘qiymiz:

```
f = open("fayl_nomi.txt", "r")
print(f.read())
```

Agar fayldan ma’lum bir qismni o‘qish kerak bo‘lsa, read () funksiyasi fayldagi butun matnni o‘qiydi. Ammo bizga uning faqatgina ma’lum bir qismi kerak bo‘lsa, uni belgilab ko‘rsatishimiz kerak. Quyidagi misolimizdagi kod matnning dastlabki 10 ta harf yoki belgisini ekranga chiqaradi:

```
f = open("fayl_nomi.txt", "r")
print(f.read(10))
```

Python dasturlash tilida fayl elementlarini teskari tartibda o‘qish imkoniyati mavjud, bu imkoniyatni reversed () funksiyasi amalga oshiradi.

Python dasturlash tilida ma’lumotlarni ikki turda yozish mumkin. Python dasturlash tilida birinchi tur bo‘yicha faylga ma’lumot yozish uchun

<fayl mantiqiy nomi>=open(‘fayl fizik nomi’, ‘w’),

ikkinchi tur bo‘yicha faylga ma’lumot yozish uchun

<fayl mantiqiy nomi>=open(‘fayl fizik nomi’, ‘a’) buyruqlari oldin yozilishi shart undan so‘ng uning tarkibiga ma’lumot yozish mumkin. Python dasturlash tilida faylga ma’lumot yozishning umumiy ko‘rinishi quyidagicha bo‘ladi:

<faylni mantiqiy nomi> .write(o‘zgaruvchi)

Fayl tarkibiga ma’lumot yozilganda uning oldingi ma’lumotlari o‘chirilib, yangi ma’lumotlar yoziladi yoki fayl oxiridan yoziladi [2].

Hozir biz faylni ochib o‘qish uchun faylni ochib ko‘ramiz. Avval .txt kengaytmali biror faylga 4-5 qatorli papka kiritamiz. Uni python faylimiz joylashgan

papkaga biror nom bilan saqlaymiz. Uni open () funksiyasi bilan ochamiz va read () funksiyasi bilan o‘qiymiz:

```
File  Edit  Format  Run  Options  Window  Help
f = open("fayl_nomi.txt", "a")
f.write("Matnga qo'shimcha qo'shdik.")
f.close()
# Endi faylni o'qiymiz
f = open("fayl_nomi.txt", "r")
print(f.read())
f.close()
```

Fayl bilan ishlab bo‘lgach albatta uni yopish kerak. Buni close () funksiyasi bilan amalga oshiramiz. Yuqoridagi kodimizda faylni ochib dastlabki ikkita qatorni o‘qigan edik. Endi o‘sha faylni yopamiz.

```
File  Edit  Format  Run  Options  Window  Help
f = open("fayl_nomi.txt", "r")
print(f.readline())
print(f.readline())
f.close()
```

Faylni o‘chirish uchun os moduliga murojaat qilamiz va undagi os.remove() funksiyasidan foydalanamiz. Masalan, biror faylimiz bor. Uni nomini bilamiz. Uni o‘chirish quyidagicha bo‘ladi:

```
File  Edit  Format  Run  Options  Window  Help
import os
os.remove("fayl_nomi.txt")
```

Os modulini ishga tushirish uchun avval uni xotiradan yuklash olishimiz kerak bo‘ladi. Yuklab olish uchun pusk+R tugmalari birgalikda bosiladi va hosil bo‘lgan oynaga cmd deb yozilib enter tugmasi bosiladi. Hosil bo‘lgan interfeysga py –m pip install os deb yozib enter tugmasi bosiladi.

Agar faylni o‘chirmasdan uning ichidagi ma’lumotlarni, element, kodlarni o‘chirmoqchi bo‘lsak ya’ni tozalamoqchi bo‘lsak quyidagi kod yoziladi:

```
f=open('test.txt', 'w')
```

Har bir narsaning afzalliklari va kamchiliklari bor bo‘lgani kabi Python dasturlash tilida fayllar bilan ishlashning ham shunday taraflari bor.

Pythonda fayllar bilan ishlashning afzalliklari

Ko‘p qirralilik: Python-da fayllar bilan ishlash sizga fayllarni yaratish, o‘qish, yozish, qo‘shish, nomini o‘zgartirish va o‘chirish kabi keng ko‘lamli operatsiyalarni bajarishga imkon beradi.

Moslashuvchanlik: Python-da fayllar bilan ishlash juda moslashuvchan, chunki u sizga turli xil fayl turlari (masalan, matnli fayllar, ikkilik fayllar, CSV fayllar va boshqalar) bilan ishlash va fayllar ustida turli operatsiyalarni bajarish (masalan, o‘qish, yozish, qo‘shish, va boshqalar.).

Foydalanuvchi uchun qulay: Python fayllar bilan ishlash uchun qulay interfeysni taqdim etadi, bu fayllarni yaratish, o‘qish va boshqarishni osonlashtiradi.

O‘zaro platformalar: Python fayllar bilan ishlash funktsiyalari turli platformalarda (masalan, Windows, Mac, Linux) ishlaydi, bu uzluksiz integratsiya va muvofiqlikni ta‘minlaydi.

Pythonda fayllar bilan ishlashning kamchiliklari

Xatoga moyil: Python-da fayllarni qayta ishlash operatsiyalari xatolarga moyil bo‘lishi mumkin, ayniqsa kod ehtiyotkorlik bilan yozilmagan bo‘lsa yoki fayl tizimi bilan bog‘liq muammolar mavjud bo‘lsa (masalan, fayl ruxsatnomalari, fayllarni blokirovka qilish va h.k.).

Xavfsizlik xavflari: Python-da fayllar bilan ishlash, ayniqsa, dastur tizimdagi nozik fayllarga kirish yoki o‘zgartirish uchun foydalanish mumkin bo‘lgan foydalanuvchi ma‘lumotlarini qabul qilsa, xavfsizlikka xavf tug‘dirishi mumkin.

Murakkablik: Python-da fayllar bilan ishlash, ayniqsa, yanada rivojlangan fayl formatlari yoki operatsiyalari bilan ishlashda murakkab bo‘lishi mumkin. Fayllar to‘g‘ri va xavfsiz ishlov berilishini ta‘minlash uchun kodga diqqat bilan e‘tibor qaratish lozim.

Ishlash: Python-da fayllar bilan ishlash operatsiyalari boshqa dasturlash tillariga qaraganda sekinroq bo‘lishi mumkin, ayniqsa katta fayllar bilan ishlashda yoki murakkab operatsiyalarni bajarishda [4].

Biz Pythonda ko‘proq txt ko‘rinishidagi fayllar ustida amallar bajarishni o‘rganganmiz. Keling Pythonda ikkilik faylni qanday o‘qish mumkinligini ko‘rib chiqamiz.

Ikkilik fayllar o‘qib bo‘lmaydigan ma’lumotlarni o‘z ichiga oladi. Aksariyat ma‘lumotlar ikkilik sifatida saqlanadi, chunki bu fayllar saqlash va qayta ishlashda juda samarali. Agar faylda inson tomonidan o‘qilmaydigan ma’lumotlar mavjud bo‘lsa, u ikkilik fayl deb ataladi. Ikkilik matn ma’lumotlarini belgilar ketma-ketligi

kabi saqlamaydi. Buning o‘rniga u ma’lumotlarni baytlar shaklida saqlaydi. Ikkilik fayldagi baytlar turli xil ma’lumotlarni, masalan, tasvirlar, video, audio, matn va boshqalarni ifodalaydi [5].

Ikkilik fayldan foydalanishning bir qancha sabablari bor. Ikkilik fayllar bo‘sh joyni tejaydi, ya’ni ular haqiqiy matnni o‘z ichiga olganlarga qaraganda kamroq joy egallaydi. Ular, shuningdek, tezroq o‘qish va yozish tezligini ta’minlaydi. Ikkilik fayl saqlangan ma’lumotlarning aniqligini saqlaydi, shuning uchun hech qanday ma’lumot yo‘qolmaydi, matnli fayl esa ba’zi ma’lumotlarni yo‘qotishi mumkin. Ikkilik fayl murakkab ma’lumotlar tuzilmalarini saqlashga imkon beradi. Lekin nima uchun ikkilik faylni ishlatish kerak? Buning bir qancha sabablari bor. Hajmi kichrayganligi va uni matn sifatida tahlil qilish talab etilmagani uchun tez o‘qish va yozish operatsiyalari zarur bo‘lgan ilovalar uchun ishlatiladi. Audio, video va tasvirlar kabi ma’lumotlar ma’lumotlar sifatini saqlash uchun ikkilik fayllar sifatida saqlanadi.

```
# Open a file named 'data.bin' in binary write mode ('wb')
with open('data.bin', 'wb') as file:
    # Write a sequence of bytes to the file
    # The byte sequence includes various UTF-8 encoded characters:
    # \xC2\xA9 represents the © symbol
    # \x20 is a space character
    # \xF0\x9D\x8C\x86 represents the 𝌆 character (a musical symbol)
    # \x20 is a space character
    # \xE2\x98\x83 represents the ☃ symbol (a snowman)
    file.write(b'\xC2\xA9\x20\xF0\x9D\x8C\x86\x20\xE2\x98\x83')
```

Yuqoridagi kod bajarilganda, berilgan ikkilik ma’lumotlar tizimingizdagi **“data.bin”** fayliga yoziladi. Endi **“data.bin”** ikkilik faylini o‘qish uchun quyidagi kod yordamida **“rb”** ga teng rejimda open () usuli yordamida faylni yana oching .

```
# Open a file named 'data.bin' in binary write mode ('wb')
with open('data.bin', 'wb') as file:
    # Write a sequence of bytes to the file
    # The byte sequence includes various UTF-8 encoded characters:
    # \xC2\xA9 represents the © symbol
    # \x20 is a space character
    # \xF0\x9D\x8C\x86 represents the 𝌆 character (a musical symbol)
    # \x20 is a space character
    # \xE2\x98\x83 represents the ☃ symbol (a snowman)
    file.write(b'\xC2\xA9\x20\xF0\x9D\x8C\x86\x20\xE2\x98\x83')

# Open the same file 'data.bin' in binary read mode ('rb')
with open('/content/data.bin', 'rb') as file:
    # Read the binary data from the file
    binary_data = file.read()
    # Print the binary data to the console
    print(binary_data)
```

```
b'\xc2\xa9 \xf0\x9d\x8c\x86 \xe2\x98\x83'
```

Yuqoridagi rasm shuni ko‘rsatadiki, fayl o‘qilganda qizil o‘q bilan ko‘rsatilgan ikkilik ma’lumotlarni qaytaradi. Bundan tashqari, har bir ikkilik ma’lumotlar nuqtasi qanday ma’lumotlarni ko‘rsatishini ko‘rishingiz mumkin. Ammo barcha ikkilik ma’lumotlar fayldan o‘qilganligi sababli, *ikkilik fayldan faqat ma’lum bayt ma’lumotlar kerak bo‘lsa nima bo‘ladi?* Xavotir olmang; *read* ***(num_bytes)*** usuli butun son turidagi argumentni qabul qiladi, bu fayldan o‘qimoqchi bo‘lgan baytlar sonini bildiradi. Misol uchun, agar sizga ikkilik fayldan faqat ikki bayt kerak bo‘lsa, quyida ko‘rsatilgandek read () usulini chaqiring.

```
# Open the same file 'data.bin' in binary read mode ('rb')
with open('/content/data.bin', 'rb') as file:
    # Read the 2 bytes of binary data from the file
    binary_data = file.read(2)
    # Print the binary data to the console
    print(binary_data)
```

```
b'\xc2\xa9'
```

Fayl file.Read(2) sifatida o‘qilganda, u faqat yuqoridagi rasmda ko‘rsatilgan **“data.bin”** faylidan 2 bayt ma’lumotni qaytaradi.

Agar siz kompyuteringizda mavjud bo‘lgan fayllarni o‘qimoqchi bo‘lsangiz, fayllarni qanday o‘qishni bilish juda foydali, lekin *Python yordamida ikkilik fayllarni o‘qish* ikkilik fayllar bilan ishlashga imkon beradi, ya’ni siz uni o‘qib chiqqandan

so‘ng ushbu faylni boshqarasiz. Siz yangi ma’lumotlarni yozishingiz yoki keraksiz ma’lumotlarni olib tashlashingiz mumkin [6].

Fayllar bilan ishlash jarayonida xatoliklar ro‘y berishi mumkin. Masalan, faylni ochish uchun urinishda fayl mavjud bo‘lmaganida yoki faylga yozish uchun urinishda fayl yozish uchun ochilmaganida xatoliklar ro‘y beradi. Bu tushunchalarni hisobga olgan holda, Python dasturchilarining xatoliklarni boshqarish va dasturiy ta’minotning to‘g‘ri ishlashini kafolatlash uchun xatoliklarni qayta ishlashga e‘tibor berishlari kerak.

XULOSA.

Xulosa qilib aytadigan bo‘lsak, Python dasturlash tili yordamida fayllar bilan ishlashning amaliy foydasi katta. Fayllar orqali, dasturchilar ma’lumotlarni doimiy saqlash, ularga kirish va ularni tahrirlash imkoniyatiga ega. Bu, ma’lumotlar bazasi bilan ishlashga o‘xshash, lekin fayllar orqali ishlash ancha oddiy va tezroq. Fayllar bilan ishlash dasturchilarga ma’lumotlarni tez va qulay tarzda saqlash, ularga kirish va ularni tahrirlash imkoniyatini beradi. U bizga fayllarni ochish, o‘qish, yozish, va yopish imkoniyatini beradi, shuningdek, fayllar bilan ishlashda xatoliklarni boshqarish imkoniyatini ham beradi. Fayllar bilan ishlashning samaradorligi va qulayligi, Python dasturlash tilini boshqa dasturlash tillaridan ajralib turadi. Python dasturlash tili orqali fayllar bilan ishlash, ma’lumotlarni saqlash va ulardan foydalanishning eng samarali usullaridan biri hisoblanadi.

ADABIYOTLAR (REFERENCES)

1. Axatov Akmal, Nazarov Fayzullo Python tilida dasturlash asoslari. O‘quv qo‘llanma. – Samarqand: SamDU nashri, 2020 yil, – 143 bet.
2. Abbosbek Ibragimov- Python asoslari
3. Ermatova Zarina Qaxramonovna, “Python dasturlash tilida fayllar bilan ishlash”// International Conference on Journal of technical research and development// 1st nov. 2023
4. File Handling In Python Harsh Pandey Software Developer Published on Mon Mar11 2024; https://flexiple.com/python/file-handling-python/.
5. How to Read Binary File in Python May 28, 2024 by Bijay Kumar; https://pythonguides.com/python-read-a-binary-file.
6. https://uzbekdevs.uz/darsliklar/python/python-da-fayl.