THE ROLE AND IMPORTANCE OF USING FILES IN STORING LARGE VOLUMES OF DATA

assistant **Normatov Nizamiddin Kamoliddin o'g'li**
Jizzakh branch of the National University of Uzbekistan, Uzbekistan
normatov@jbnuu.uz
student **Sirojiddin Mukhtorov**
Jizzakh branch of the National University of Uzbekistan, Uzbekistan
mukhtorovs468@gmail.com

Abstract: *It is a practical tool needed for working with big data and for storing and transferring data generated in many fields to other services. Using files is one of the strategic and experiential methods of analyzing and processing data. In this section, we will learn how to retrieve large amounts of data from a file and save the required data to a file at the end of the program. Working with files allows both program users and the program to upload the data they want.*

Keywords: *Argument, text editor, format, function, method, console, block, command, object, archiving, optimization.*

INTRODUCTION. Files for storing large amounts of data are the most widely used means for storing variables, texts, images, audio, and video data. Files allow you to store, read, edit, delete, and other operations. Files are important when storing large amounts of data for the following reasons. Depending on their size and layers, data can support large volumes.

- Data Reprocessing: Files allow data to be reloaded, read, edited, and deleted. This makes it easy to perform various analyzes and operations on the data.
- Data protection against external influences: The data in the files can be influenced from the outside (access, change, deletion, destruction), it is possible to protect the file with a password and create the relevant files only for the necessary people.
- Use for writing public data: Files are convenient for public data storage and connection to other programs. It makes it easy to share data over the internet by sending a file.
- Long-term data storage: Files allow long-term data storage and archiving. This makes it easy to archive and restore.
- Ease of platform variability: Files are independent of the operating system, making it easy to store and share data across platforms.

LITERATURE ANALYSIS AND METHODS. In many cases, it is necessary to read large data directly from files during the program. It is especially natural for analysis programs to work with large tables stored in file format. But working with files is also useful in other cases, for example, when writing a program that automates the conversion of plain text into html. The first step in working with files is to copy the data in the file to the computer memory. There are several ways to do this, we will get acquainted with them below:

Read the file. To get started we need a file. Let's create a new pi.txt file and put the following text inside it:

3.1415926535
8979323846
2643383279

Download the pi.txt file. Our three-line file stores the value of the number p \pi p (with 30-digit precision). To read the file completely, we write the following code:

```
with open ('pi.txt') as file:
pi = file. read ()
```

Let's analyze the code:

- In the first line, we open the file using the open()function. Here we are giving the file name as an argument to the function. Here it is important that the file we are opening and our program are in the same folder.
- The open()function returns the file as an object, and with the as operator, we are naming our object file.
- In the second line, using the .read() method, we take the text we need from the content of the file object and load it into a new, PI variable.
- with operator is to close the file when we are done with the file. In the above example, after line 2, Python immediately closes the file.

The method we saw above is the safest way to work with the file. In fact, we could open files directly using file=open('pi.txt') and close the file using the file.close() command when we are done with the file:

```
file = open ('pi.txt')
PI = file. read ()
print (pi)
file. close ()
```

But this method is dangerous and not recommended. The point is that after we open the file using the open() function, our file remains open until we call the close() method. If we do not close the file in time, or if our program stops before the file is closed, the contents of the file may be damaged and data may be lost. For example, other programs (such as

Microsoft Word) can damage your file if your computer shuts down before closing the file, or if the program closes accidentally.

Therefore, when we call the open() function through with, our file remains open until the end of the with block, and when the with ends, the file is also closed. So we have to perform operations on the file in the with block.

Now let's plot the value of pi:

```
>>> print(pi)
3.1415926535
8979323846
2643383279
```

The text is output to the console exactly as it was stored in the file. While the data stored is a number, the value returned when we read from the file is text. To convert the text to a number, we do a little processing on it:

```
pi = pi. rstrip () # we remove the spaces at the end of the line
pi = pi. replace ('\n','') # we replace the carriage return
pi = float (pi) # convert the text to a float (decimal) number
print (pi)
```

Result: 3.141592653589793

The .replace() method is used to replace a character or character in text with another character or character.

Open a file inside a folder. If the file you are opening is not in the same folder as our program, but in a folder inside this folder, the folder name is written before the file name:

```
with open ('data/pi.txt') as file:
pi = file. read ()
```

If there are multiple folders, it is better to record the file name and the folders before it separately:

```
filename = 'data/math/numbers/pi.txt'
with open (filename) as file:
pi = file. read ()
```

While Windows uses the ``\'' character between folders, Python uses the ``/'' character. If the \ character is used, this character is written twice: C:\\python\\lessons\\data

Reading a file line by line. Sometimes it may be necessary to read the file line by line rather than the entire file. For example, when a file stores students' names or daily weather data, etc. In such cases, we use a for loop:

```
filename = 'data/students.txt'
with open (filename) as file:
```

```
for line in file:
print(line)
```

The result:

Alijon Valiyev
Hasan Olimov
Rahima Muminova

To save the lines as a list, we use the .readlines() method.

```
with open (filename) as file:
students = file . readlines ()
print (students)
```

Result: ['alijon valiyev\n', 'hasan olimov\n', 'rahima muminova\n', 'hamida aqilova']

Write the file. The most convenient way to save data is to write it to a file. After our program stops executing, the data in memory may be erased, but the data written to the file will remain. We can load the files back into memory in the future and continue our program from where we left off.

Above, we used the open() function to open a file, passing the filename as the only argument. In this case, the file is opened for reading only, it cannot be written to. When calling the open() function to write data to a file, we provide another argument in addition to the file name. The second argument specifies what exactly we want to open the file for. Arguments can be:

Argument	Application	Content
'w'	open('file.txt','w')	Open the file for writing. If the file does not exist, a new file will be created. If the file exists, its contents will be deleted
'r'	open('file.txt','r')	Open a file as read-only (not writable)
'w+'	open('file.txt','w+')	Open the file for reading and writing. If the file does not exist, a new file will be created. If the file exists, its contents will be deleted.
'r+'	open('file.txt','r+')	Open the file for reading and writing.
'a'	open('file.txt','a')	Open to add information to the file. If the file does not exist, a new file will be created.
'a+'	open('file.txt','a+')	Write to read and add data to a file. If the file does not exist, a new file will be created.

Write to a new file. We use the 'w' (write) argument when calling the open() function to write data to a new file . To add information to the opened file, we call the .write() method.

```
filename = 'ustozlar.txt' # the name of the file being opened (created).
with open (filename,'w') as file:
file. write ('anvar narzullaev') # information being written to the file
```

Add new data to the file. If we need to append data to an existing file, we use the 'a' (append) argument when calling the open() function. This will append the newly added data to the end of the file.

```
with open(filename,'a') as file:
file.write('Alijon Valiyev\n')
file.write('2000')
Added new information to the file
If the file we're opening doesn't exist, Python will create a new file.
```

LIST OF REFERENCES:

1. Obid o'g A. S. J. et al. Numpy Library Capabilities. Vectorized Calculation In Numpy Va Type Of Information //Eurasian Research Bulletin. – 2022. – T. 15. – C. 132-137.
2. Nizomiddin N. et al. TA'LIMDA DASTURLASH JARAYONINI BAHOLASHGA ASOSLANGAN AVTOMATLASHTIRILGAN TIZIMNI TADBIQ ETISH //International Journal of Contemporary Scientific and Technical Research. – 2023. – C. 24-28.
3. Ziyoda M., Nizommiddin N. RAQAMLI IQTISODIYOTDA SUN'IY INTELLEKT TEXNOLOGIYALARINI TURLI SOHALARDA AVTOMATLASHTIRISH VOSITALARI //International Journal of Contemporary Scientific and Technical Research. – 2023. – C. 246-250.
4. Чорркулов Г., Норматов Н., Мамараимов А. Роль анализа текстовых связей в электронных документах в информационной безопасности //Информатика и инженерные технологии. – 2023. – Т. 1. – №. 1. – С. 67-71.
5. Норматов Н., Мамараимов А. Ta'lim tizimida baholash tizimini avtomatlashtirishni joriy etish jarayonlari va foydalanish metodlari //Информатика и инженерные технологии. – 2023. – Т. 1. – №. 2. – С. 356-359.
6. Мамараимов А., Чорёркулов Г., Норматов Н. Tanib olish modullarini dasturiy amalga oshirish //Информатика и инженерные технологии. – 2023. – Т. 1. – №. 2. – С. 38-44.

7. Kamoliddin o‘g’li N. N. et al. ERWIN DASTURI YORDAMIDA IDEF0, IDEF3 VA DFD STANDAT DIAGARAMMALARIDAN FOYDALANIB TIZIM SIFATIDA YARATILGAN UNIVERSITETNING MONITORING BO ‘LIMI LOYIHASI //Новости образования: исследование в XXI веке. – 2023. – Т. 1. – №. 6. – С. 378-386.

8. Тавбоев С. А. и др. НЕКОТОРЫЕ МЕТОДЫ И ЗАДАЧИ ЦИФРОВОЙ ОБРАБОТКИ И РАСПОЗНАВАНИЯ ИЗОБРАЖЕНИЙ //International Journal of Contemporary Scientific and Technical Research. – 2022. – С. 334-339.

9. Abdurahimovich A. A., Kamoliddin o'g'li M. A. SANOQ SISTEMALARIDA VAQT TUSHUNCHASI //International Journal of Contemporary Scientific and Technical Research. – 2022. – С. 331-334.

10. Tavboyev Sirojiddin Akhbutayevich, Mamaraimov Abror Kamoliddin ugli, and Karshibaev Nizomiddin Abdumalikovich, “Algorithms for Selecting the Contour Lines of Images Based on the Theory of Fuzzy Sets”, TJET, vol. 15, pp. 31–40, Dec. 2022.

11. Obid o’g A. S. J. et al. Numpy Library Capabilities. Vectorized Calculation In Numpy Va Type Of Information //Eurasian Research Bulletin. – 2022. – Т. 15. – С. 132-137.

12. Naim o’g’li M. D., Shokir o’g’li B. Z. МЕТОДЫ ОБУЧЕНИЯ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ С УЧИТЕЛЕМ //Новости образования: исследование в XXI веке. – 2023. – Т. 1. – №. 9. – С. 1260-1264.

13. Мамараимов А., Мухторов Л., Рахимов А. Fanlarni o‘qitishda netsupport school ilovasidan foydalanishning pedagogik imkoniyatlari //Информатика и инженерные технологии. – 2023. – Т. 1. – №. 2. – С. 266-273.

14. Мамараимов А., Мухторов Л., Рахимов А. Fanlarni o‘qitishda netsupport school ilovasidan foydalanishning pedagogik imkoniyatlari //Информатика и инженерные технологии. – 2023. – Т. 1. – №. 2. – С. 266-273.

15. Choryorqulov G‘.H., & Qosimov N.S. (2023). ELEKTRON JADVAL MODELINING TAVSIFLANISHI. PEDAGOGS Jurnali, 30(3), 67–73.

16. Amrullayevich K. A., Obid o'g'li S. J. ELEKTRON TALIM MUHITIDA TALABALARDA AXBOROT BILAN ISHLASH KOMPETENTLIKNI SHAKLLANTIRISH //International Journal of Contemporary Scientific and Technical Research. – 2022. – С. 641-645.