File Handling in Python

Read Discuss Courses Practice

File handling in Python is a powerful and versatile tool that can be used to perform a wide range of operations. However, it is important to carefully consider the advantages and disadvantages of file handling when writing Python programs, to ensure that the code is secure, reliable, and performs well.

Python File Handling

Python too supports file handling and allows users to handle files i.e., to read and write files, along with many other file handling options, to operate on files. The concept of file handling has stretched over various other languages, but the implementation is either complicated or lengthy, but like other concepts of Python, this concept here is also easy and short. Python treats files differently as text or binary and this is important. Each line of code includes a sequence of characters and they form a text file. Each line of a file is terminated with a special character, called the EOL or End of Line characters like comma {,} or newline character. It ends the current line and tells the interpreter a new one has begun. Let's start with the reading and writing files.

Advantages of File Handling

- **Versatility**: File handling in Python allows you to perform a wide range of operations, such as creating, reading, writing, appending, renaming, and deleting files.
- **Flexibility**: File handling in Python is highly flexible, as it allows you to work with different file types (e.g. text files, binary files, CSV files, etc.), and to perform different operations on files (e.g. read, write, append, etc.).
- **User–friendly**: Python provides a user-friendly interface for file handling, making it easy to create, read, and manipulate files.
- **Cross-platform**: Python file-handling functions work across different platforms (e.g. Windows, Mac, Linux), allowing for seamless integration and compatibility.

Disadvantages of File Handling

- **Error-prone:** File handling operations in Python can be prone to errors, especially if the code is not carefully written or if there are issues with the file system (e.g. file permissions, file locks, etc.).

- **Security risks**: File handling in Python can also pose security risks, especially if the program accepts user input that can be used to access or modify sensitive files on the system.
- **Complexity**: File handling in Python can be complex, especially when working with more advanced file formats or operations. Careful attention must be paid to the code to ensure that files are handled properly and securely.
- **Performance**: File handling operations in Python can be slower than other programming languages, especially when dealing with large files or performing complex operations.

For this article, we will consider the following “**geeks.txt**” file as an example.

```
Hello world
GeeksforGeeks
123 456
```

Working of open() Function in Python

Before performing any operation on the file like reading or writing, first, we have to open that file. For this, we should use Python’s inbuilt function open() but at the time of opening, we have to specify the mode, which represents the purpose of the opening file.

Trending Now Data Structures Algorithms Topic-wise Practice Python Machine Learning Data Science Ja

```
f = open(filename, mode)
```

Where the following mode is supported:

1. **r:** open an existing file for a read operation.
2. **w:** open an existing file for a write operation. If the file already contains some data then it will be overridden but if the file is not present then it creates the file as well.
3. **a:** open an existing file for append operation. It won’t override existing data.
4. **r+:** To read and write data into the file. The previous data in the file will be overridden.
5. **w+:** To write and read data. It will override existing data.
6. **a+:** To append and read data from the file. It won’t override existing data.

Working in Read mode

There is more than one way to read a file in Python. Let us see how we can read the content of a file in read mode.

Example 1: The open command will open the file in the read mode and the for loop will print each line present in the file.

Python3

```
# a file named "geek", will be opened with the reading mode.
file = open('geek.txt', 'r')

# This will print every line one by one in the file
for each in file:
    print (each)
```

Output:

```
Hello world

GeeksforGeeks

123 456
```

Example 2: In this example, we will extract a string that contains all characters in the file then we can use **file.read()**.

Python3

```
# Python code to illustrate read() mode
file = open("geeks.txt", "r")
print (file.read())
```

Output:

```
Hello world
GeeksforGeeks
123 456
```

Example 3: In this example, we will see how we can read a file using the **with statement**.

Python3

```
# Python code to illustrate with()
with open("geeks.txt") as file:
    data = file.read()

print(data)
```

Output:

```
Hello world
GeeksforGeeks
123 456
```

Example 4: Another way to read a file is to call a certain number of characters like in the following code the interpreter will read the first five characters of stored data and return it as a string:

Python3

```
# Python code to illustrate read() mode character wise
file = open("geeks.txt", "r")
print (file.read(5))
```

Output:

```
Hello
```

Example 5:

We can also split lines while reading files in Python. The split() function splits the variable when space is encountered. You can also split using any characters as you wish.

Python3

```
# Python code to illustrate split() function
with open("geeks.txt", "r") as file:
    data = file.readlines()
    for line in data:
        word = line.split()
        print (word)
```

Output:

```
['Hello', 'world']
['GeeksforGeeks']
['123', '456']
```

Creating a File using the write() Function

Just like reading a file in Python, there are a number of ways to write in a file in Python. Let us see how we can write the content of a file using the write() function in Python.

Working in Write Mode

Let's see how to create a file and how the write mode works.

Example 1: In this example, we will see how the write mode and the write() function is used to write in a file. The close() command terminates all the resources in use and frees the system of this particular program.

Python3

```
# Python code to create a file
file = open('geek.txt','w')
file.write("This is the write command")
file.write("It allows us to write in a particular file")
file.close()
```

Output:

```
This is the write commandIt allows us to write in a particular file
```

Example 2: We can also use the written statement along with the with() function.

Python3

```
# Python code to illustrate with() alongwith write()
with open("file.txt", "w") as f:
    f.write("Hello World!!!")
```

Output:

```
Hello World!!!
```

Working of Append Mode

Let us see how the append mode works.

Example: For this example, we will use the file created in the previous example.

Python3

```
# Python code to illustrate append() mode
file = open('geek.txt', 'a')
file.write("This will add this line")
file.close()
```

Output:

```
This is the write commandIt allows us to write in a particular fileThis will add this line
```

There are also various other commands in file handling that are used to handle various tasks:

```
rstrip(): This function strips each line of a file off spaces from the right-hand side.
lstrip(): This function strips each line of a file off spaces from the left-hand side.
```

It is designed to provide much cleaner syntax and exception handling when you are working with code. That explains why it's good practice to use them with a statement where applicable. This is helpful because using this method any files opened will be closed automatically after one is done, so auto-cleanup.

Implementing all the functions in File Handling

In this example, we will cover all the concepts that we have seen above. Other than those, we will also see how we can delete a file using the remove() function from Python os module.

Python3

```python
import os

def create_file(filename):
    try:
        with open(filename, 'w') as f:
            f.write('Hello, world!\n')
        print("File " + filename + " created successfully.")
    except IOError:
        print("Error: could not create file " + filename)

def read_file(filename):
    try:
        with open(filename, 'r') as f:
            contents = f.read()
            print(contents)
    except IOError:
        print("Error: could not read file " + filename)

def append_file(filename, text):
    try:
        with open(filename, 'a') as f:
            f.write(text)
        print("Text appended to file " + filename + " successfully.")
    except IOError:
        print("Error: could not append to file " + filename)

def rename_file(filename, new_filename):
    try:
        os.rename(filename, new_filename)
        print("File " + filename + " renamed to " + new_filename + " successfully.")
    except IOError:
        print("Error: could not rename file " + filename)

def delete_file(filename):
    try:
        os.remove(filename)
        print("File " + filename + " deleted successfully.")
    except IOError:
        print("Error: could not delete file " + filename)


if __name__ == '__main__':
    filename = "example.txt"
    new_filename = "new_example.txt"

    create_file(filename)
    read_file(filename)
    append_file(filename, "This is some additional text.\n")
    read_file(filename)
    rename_file(filename, new_filename)
    read_file(new_filename)
    delete_file(new_filename)
```

Output:

```
File example.txt created successfully.
Hello, world!

Text appended to file example.txt successfully.
Hello, world!
This is some additional text.

File example.txt renamed to new_example.txt successfully.
Hello, world!
This is some additional text.

File new_example.txt deleted successfully.
```

Last Updated : 27 Jun, 2023

375

Similar Reads

1. reStructuredText | .rst file to HTML file using Python for Documentations
2. Python - Copy contents of one file to another file
3. Python program to reverse the content of a file and store it in another file
4. Create a GUI to convert CSV file into excel file using Python
5. Python - Get file id of windows file
6. How to save file with file name from user using Python?
7. How to convert PDF file to Excel file using Python?
8. How to convert CSV File to PDF File using Python?
9. How to create a duplicate file of an existing file using Python?
10. Python - Copy all the content of one file to another file in uppercase

Related Tutorials

1. Pandas AI: The Generative AI Python Library

2. OpenAI Python API - Complete Guide
3. Python for Kids - Fun Tutorial to Learn Python Programming
4. Data Analysis Tutorial
5. Flask Tutorial

Previous Next

Article Contributed By :

GeeksforGeeks

Vote for difficulty

Current difficulty : Easy

Easy Normal Medium Hard Expert

Improved By : Shivaansh Agarwal, samadhan2601, harishhari3112004, icecrac34r, porasp9gp6, anikettchavan, tanya_sehgal

Article Tags : python, python-io, Python

Practice Tags : python, python, python-io

Improve Article Report Issue

GeeksforGeeks

A-143, 9th Floor, Sovereign Corporate
Tower, Sector-136, Noida, Uttar Pradesh -
201305

feedback@geeksforgeeks.org

Company

About Us

Legal

Careers

In Media

Contact Us

Advertise with us

Explore

Job-A-Thon For Freshers

Job-A-Thon For Experienced

GfG Weekly Contest

Offline Classes (Delhi/NCR)

DSA in JAVA/C++

Master System Design

Master CP

Languages

Python

Java

C++

PHP

GoLang

SQL

R Language

Android Tutorial

Data Structures

Array

String

Linked List

Stack

Queue

Tree

Graph

Algorithms

Sorting

Searching

Greedy

Dynamic Programming

Pattern Searching

Recursion

Backtracking

Web Development

HTML

CSS

JavaScript

Bootstrap

ReactJS

AngularJS

NodeJS

Computer Science

GATE CS Notes

Python

Python Programming Examples

Operating Systems

Computer Network

Database Management System

Software Engineering

Digital Logic Design

Engineering Maths

Django Tutorial

Python Projects

Python Tkinter

OpenCV Python Tutorial

Python Interview Question

Data Science & ML

Data Science With Python

Data Science For Beginner

Machine Learning Tutorial

Maths For Machine Learning

Pandas Tutorial

NumPy Tutorial

NLP Tutorial

Deep Learning Tutorial

DevOps

Git

AWS

Docker

Kubernetes

Azure

GCP

Competitive Programming

Top DSA for CP

Top 50 Tree Problems

Top 50 Graph Problems

Top 50 Array Problems

Top 50 String Problems

Top 50 DP Problems

Top 15 Websites for CP

System Design

What is System Design

Monolithic and Distributed SD

Scalability in SD

Databases in SD

High Level Design or HLD

Low Level Design or LLD

Top SD Interview Questions

Interview Corner

Company Wise Preparation

Preparation for SDE

Experienced Interviews

Internship Interviews

Competitive Programming

Aptitude Preparation

GfG School

CBSE Notes for Class 8

CBSE Notes for Class 9

CBSE Notes for Class 10

CBSE Notes for Class 11

CBSE Notes for Class 12

English Grammar

Commerce

Accountancy

Business Studies

Economics

Management

UPSC

Polity Notes

Geography Notes

History Notes

Science and Technology Notes

Income Tax

Finance

Economics Notes

Important Topics in Ethics

UPSC Previous Year Papers

SSC/ BANKING

SSC CGL Syllabus

SBI PO Syllabus

SBI Clerk Syllabus

IBPS PO Syllabus

IBPS Clerk Syllabus

Aptitude Questions

SSC CGL Practice Papers

Write & Earn

Write an Article

Improve an Article

Pick Topics to Write

Write Interview Experience

Internships

Video Internship