# File Operations

File operations in Python involve reading from or writing to files. Here are some common file operations:

### Reading from a File:

1. **Reading the Entire File:**
   * Use the read() method to read the entire content of the file.

with open('example.txt', 'r') as file:

content = file.read()

print(content)

1. **Reading Line by Line:**
   * Use a loop to read the file line by line using the readline() method.

with open('example.txt', 'r') as file:

for line in file:

print(line.strip()) # strip() removes the newline character

1. **Reading Lines into a List:**
   * Use the readlines() method to read all lines into a list.

with open('example.txt', 'r') as file:

lines = file.readlines()

for line in lines:

print(line.strip())

### Writing to a File:

1. **Writing to a File:**
   * Use the write() method to write content to a file.

with open('example.txt', 'w') as file:

file.write('Hello, World!')

1. **Appending to a File:**
   * Use the write() method with append mode ('a') to add content to the end of a file.

with open('example.txt', 'a') as file:

file.write(' Appending content.')

# Files in python and How to open a file:

A file is a container in computer storage devices used for storing data. Text files are structured as a sequence of lines, where each line includes a sequence of characters. Each line is terminated with a special character, called the EOL or **End of Line** character.

Open a file

Read or write (perform operation)

Close the file

**1. Open a File**

To open a file for reading, you can use the **open()** function:

**Syntax:**

file=open(“filename.txt”,”mode”)

file.close()

**File Access modes : -**

Access mode tells the interpreter and developer which way the file will be used. This is optional parameter and the default file access mode is read (r).

|  |  |
| --- | --- |
| Modes | Description |
| r | Opens a file for reading only. The file pointer is placed at the beginning of the file. This is the default mode. |
| rb | Opens a file for reading only in binary format. The file pointer is placed at the beginning of the file. This is the default mode. |
| r+ | Opens a file for both reading and writing. The file pointer placed at the beginning of the file. |
| rb+ | Opens a file for both reading and writing in binary format. The file pointer placed at the beginning of the file. |
| w | Opens a file for writing only. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing. |
| wb | Opens a file for writing only in binary format. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing. |
| w+ | Opens a file for both writing and reading. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing. |
| wb+ | Opens a file for both writing and reading in binary format. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing. |
| a | Opens a file for appending. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing. |
| ab | Opens a file for appending in binary format. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing. |
| a+ | Opens a file for both appending and reading. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing. |
| ab+ | Opens a file for both appending and reading in binary format. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for rea |

**Example of File Access Mode:**

**Read Mode (**'r'**):**

* + Opens the file for reading. If the file does not exist, it raises a FileNotFoundError.

with open('example.txt', 'r') as file:

content = file.read()

1. **Write Mode (**'w'**):**
   * Opens the file for writing. If the file already exists, it truncates the file to zero length. If the file does not exist, it creates a new file.

with open('example.txt', 'w') as file:

file.write('Hello, World!')

1. **Append Mode (**'a'**):**
   * Opens the file for writing, but appends to the end of the file rather than truncating it. If the file does not exist, it creates a new file.

with open('example.txt', 'a') as file:

file.write(' Appending content.')

1. **Binary Mode (**'b'**):**
   * Opens the file in binary mode, allowing the reading or writing of binary data (e.g., images, videos).

pythonCopy code

with open('binary\_file.bin', 'rb') as file:

binary\_data = file.read()

1. **Text Mode (**'t' **or default):**
   * Opens the file in text mode (default). This is similar to opening the file without specifying a mode. It processes the file as a text file, and it is the default mode if no mode is specified.

with open('text\_file.txt', 'rt') as file:

text\_data = file.read()

1. **Read and Write Mode (**'r+'**):**
   * Opens the file for both reading and writing. The file pointer is placed at the beginning of the file.

with open('example.txt', 'r+') as file:

content = file.read()

file.write('Appending content.')

1. **Write and Read Mode (**'w+'**):**
   * Opens the file for both writing and reading. If the file exists, it truncates the file to zero length.

with open('example.txt', 'w+') as file:

file.write('Hello, World!')

file.seek(0) # Move the file pointer to the beginning

content = file.read()

1. **Append and Read Mode (**'a+'**):**
   * Opens the file for both appending and reading. It places the file pointer at the end of the file.

with open('example.txt', 'a+') as file:

file.write(' Appending content.')

file.seek(0) # Move the file pointer to the beginning

content = file.read()

**To Read a File**

file=open(“demo.txt”,”r”)

variable=file.read() # read is used to print all statements

print(file)

file.close()

file=open(“demo.txt”,”r”)

variable=file.readline() # readline is used to print one bye one statement

print(file)

file.close()

**To Write a File**

file=open(“demo.txt”,”w”)

file.write(“Replace statement)

file.close()

**To append a File**

file=open(“filename.txt”,”a”)

file.write(“Message”)

file.close()

#Check at txt file

**Python JSON**

**When you want to save more complex data types like nested lists and dictionaries, parsing and serializing by hand becomes complicated.** Rather than having users constantly writing and debugging code to save complicated data types to files, Python allows you to use the popular data interchange format called **JSON (JavaScript Object Notation).** The standard module called json can take Python data hierarchies, and convert them to string representations; this process is called serializing. Reconstructing the data from the string representation is called deserializing.

**dump()**=The json.dump() function in Python is used to serialize (convert) Python objects into a JSON formatted string and write it to a file-like object

**load()**=The json.load() function in Python is used to deserialize (convert) a JSON-formatted string or data from a file-like object into a Python object

**Example:**

import json

# Creating a list of dictionaries

data1 = [{1: 'denu', 2: 'pinchu', 3: 'elizabeth', 4: 'denova'}]

# Writing the list of dictionaries to a file named 'json.txt'

with open('json.txt', 'w') as my\_file3:

json.dump(data1, my\_file3) # Writing JSON data to the file

# Reopening the file in read mode

with open('json.txt', 'r') as my\_file3:

# Loading the JSON-formatted string from the file into a Python object

b1 = json.load(my\_file3)

# Reopening the file for printing its content

with open('json.txt', 'r') as my\_file3:

print("File content:")

print(my\_file3.read())

# The file is automatically closed when using the 'with' statement

Explanation:

1. **Writing to a file (**json.dump**):**

with open('json.txt', 'w') as my\_file3:

json.dump(data1, my\_file3) # Writing JSON data to the file

This writes the list of dictionaries (data1) to the file 'json.txt' using json.dump().

1. **Reading from a file (**json.load**):**

with open('json.txt', 'r') as my\_file3:

b1 = json.load(my\_file3)

This reads the content of the file back into the Python object b1 using json.load().

1. **Reopening the file for printing its content:**

with open('json.txt', 'r') as my\_file3:

print("File content:")

print(my\_file3.read())

#### ****'with'**** Statement

**It is possible to open a file using** with **statement. It will automatically close the file after the nested block of code. The advantage of using a** with **statement is that it is guaranteed to close the file no matter** how **the nested block exits. If an exception occurs before the end of the block, it will close the file before the exception is caught by an outer exception handler.**

# CSV File Handling in Python

A csv stands for "comma separated values", which is defined as a simple file format that uses specific structuring to arrange tabular data. It stores tabular data such as spreadsheet or database in plain text and has a common format for data interchange. A csv file opens into the excel sheet, and the rows and columns data define the standard format. CSV files are used to store a large number of variables – or data. They are incredibly simplified spreadsheets – think Excel – only the content is stored in plaintext.

**CSV - Reader:-**

To pull data from a CSV file, you must use the reader function to generate a reader object. The reader function is designed to take each line of the file and make a list of all columns.

Example:

with open('apple.csv', 'r', newline='') as file:

    reader = csv.reader(file)

    # Iterate through each row in the CSV file

    for row in reader:

        print(row)

**CSV-Write:**

When you have a set of data that you would like to store inside a CSV file, it’s time to do the opposite and use the write function. Believe it or not, this is just as easy to accomplish as reading them. The writer() function will create an object suitable for writing. To iterate the data over the rows, you will need to use the writerow()function.

With Data Filed:

**Example 2:**

import csv

# Open the CSV file in append mode

with open('apple.csv', 'a', newline='') as file:

writer = csv.writer(file)

# Tuple to be added to the CSV file

tup = ("Bob", 20)

# Write the tuple as a new row in the CSV file

writer.writerow(tup)

print("Data has been appended to 'apple.csv'.")