File Handling in Python

A file is a collection of data or information stored on a computer or storage device (like a hard disk, SSD, or USB drive) that is identified by a name and a file extension (like .txt, .jpg, .pdf).

File handling in Python refers to the process of working with files, which are used to store data on a computer's storage system.

Files can be used to store a wide range of information, such as:

- Text
- Numbers
- Images
- And more

In Python, you can perform various operations on files, such as:

- Creating files
- Reading content
- Writing data
- Updating existing content

Example 1: See all the files in your directory

In [40]:
```
!dir
```

```
 Volume in drive D is New Volume
 Volume Serial Number is A09E-98C2

 Directory of D:\decode ai\github\Decode-AI\Section 01 - Decode Python for ML A2Z\1.
13 File Handling in Python

20-07-2025  08:02    <DIR>          .
19-07-2025  23:15    <DIR>          ..
19-07-2025  12:41    <DIR>          .ipynb_checkpoints
20-07-2025  08:02            16,727 1.13_File_handling_in_Python.ipynb
               1 File(s)         16,727 bytes
               3 Dir(s)  10,434,174,976 bytes free
```

In [41]:
```
# The os module in Python provides functions to interact with the operating system,
# environment variables, and process management.
import os
```

In [42]:
```
def show_files():
    # Set the folder path
    folder = "D:\\decode ai\\github\\Decode-AI\\"
```

```
    # Get list of all files and folders in the folder
    items = os.listdir(folder)

    # If the folder is not empty, print each item
    if items:
        for item in items:
            print(item)
```

In [43]:
```
show_files()
```

```
.git
about.txt
CNAME
google37f70b56e827a2c8.html
input.txt
output.txt
README.md
Section 0 - Getting Started
Section 01 - Decode Python for ML A2Z
Section 02 - Decode DSA with Python A2Z
Section 03 - Decode Calculus and Optimization A2Z
Section 04 - Decode Linear Algebra A2Z
Section 05 - Decode Statistics and Probability A2Z
Section 06 - Decode Machine Learning A2Z
Section 07 - Decode Deep Learning A2Z
Section 08 - Decode Generative AI A2Z
Section 09 - Decode AI Development Tools A2Z
Section 10 - Decode ML Model Deployment A2Z
Section 11 - Decode Recommendation Engine A2Z
Section 12 - Decode ML Design Interview A2Z
```

Example 2: Read the file and output to console

Syntax: `with open(file_path, 'r') as file`

In [44]:
```
def read_numbers_from_file():
    # Set the path of the file to read
    file_path = "D:\\decode ai\\github\\Decode-AI\\input.txt"

    try:
        # Open the file in read mode
        with open(file_path, 'r') as file:
            # Read each line, convert to integer, and print
            for line in file:
                number = int(line.strip())
                print(number)

    except FileNotFoundError:
        print("File not found:", file_path)
```

In [45]:
```
read_numbers_from_file()
```

```
1
2
3
4
5
6
7
8
9
10
```

Modes Summary

Mode	Action	File Must Exist	Creates New File	Truncates Existing File
`"r"`	Read	✅	❌	❌
`"w"`	Write	❌	✅	✅
`"a"`	Append	❌	✅	❌
`"r+"`	Read and Write	✅	❌	❌
`"w+"`	Write and Read	❌	✅	✅
`"x"`	Create (fail if exists)	❌	✅	❌
`"b"`	Binary (combine with above)	❌	❌	❌

Example 3: Read the file and output to another file

Q. you have a file with some list of values. you need to read the file and copy all values to another file.

In [34]:
```python
def copy_and_show_file():
    # File paths
    input_file = "D:\\decode ai\\github\\Decode-AI\\input.txt"
    output_file = "D:\\decode ai\\github\\Decode-AI\\output.txt"

    try:
        # Copy content from input.txt to output.txt
        with open(input_file, 'r') as in_file, open(output_file, 'w') as out_file:
            for line in in_file:
                out_file.write(line)

        # Read and print content from output.txt
        with open(output_file, 'r') as out_file:
            for line in out_file:
                print(line.strip())

    except IOError as error:
        print("Error while reading or writing file:", error)
```

In [35]:
```
copy_and_show_file()
```

```
1
2
3
4
5
6
7
8
9
10
```

Example 4: Read, do operations and output to another file

Q. you have a file with some list of values. you need to read the file and store the square of all values to another file.

In [36]:
```
def process_numbers():
    # File paths
    input_path = "D:\\decode ai\\github\\Decode-AI\\input.txt"
    output_path = "D:\\decode ai\\github\\Decode-AI\\output.txt"

    try:
        # Read numbers from input file and write number + square to output file
        with open(input_path, "r") as in_file, open(output_path, "w") as out_file:
            for line in in_file:
                number = int(line.strip())
                square = number * number
                out_file.write(f"{number} {square}\n")

        # Read and print output file content
        with open(output_path, "r") as out_file:
            for line in out_file:
                print(line.strip())

    except IOError as error:
        print("File error:", error)
```

In [37]:
```
process_numbers()
```

```
1 1
2 4
3 9
4 16
5 25
6 36
7 49
8 64
9 81
10 100
```

Example 6: You are given an input file - input.txt which contains a paragraph about "DecodeAiML". Find no of occurences of "DecodeAiML"

In [38]:
```python
def count_word_in_file():
    # Path to the file
    file_path = "D:\\decode ai\\github\\Decode-AI\\about.txt"
    search_word = "DecodeAiML"
    count = 0

    try:
        # Open the file and count the word
        with open(file_path, 'r') as file:
            for line in file:
                words = line.split()
                for word in words:
                    if word == search_word:
                        count += 1

        # Print the total count
        print(f"Number of occurrences of '{search_word}':", count)

    except IOError as error:
        print("Error reading the file:", error)
```

In [39]:
```python
user_function()
```

```
Number of occurrences of 'DecodeAiML': 2
```

Common File Extensions in Machine Learning

Machine learning workflows involve a variety of file formats for data storage, model persistence, configuration, and visualization. Below are the commonly used file extensions grouped by their role.

---

1. Data Files

Extension	Description	Common Usage
`.csv`	Comma-Separated Values	Tabular datasets
`.tsv`	Tab-Separated Values	Tabular data with tab delimiter
`.xlsx`	Excel Spreadsheet	Data analysis, feature tracking
`.json`	JavaScript Object Notation	Structured data, configurations
`.xml`	Extensible Markup Language	Annotated datasets (e.g., for NLP tasks)
`.txt`	Plain Text	Simple data, labels, notes
`.h5`	HDF5 (Hierarchical Data Format)	Storing large datasets or models
`.npz`	NumPy Compressed Archive	Storing NumPy arrays

---

2. Model Files

Extension	Description	Used By
`.pkl` / `.pickle`	Python Pickle File	Scikit-learn, XGBoost, custom models
`.joblib`	Serialized models with Joblib	Scikit-learn, faster for large models
`.h5`	HDF5-based Keras model format	TensorFlow/Keras
`.pt` / `.pth`	PyTorch Model	PyTorch
`.onnx`	Open Neural Network Exchange	Model interoperability (PyTorch ↔ ONNX)
`.tflite`	TensorFlow Lite Format	Deploying ML models on mobile/IoT

---

3. Image/Audio/Video Files

Extension	Description	Usage
`.jpg` / `.jpeg`	JPEG Image	Image classification, CV tasks
`.png`	Portable Network Graphic	Images with transparency
`.bmp`	Bitmap Image	Raw image format
`.wav`	Waveform Audio File	Speech/audio processing
`.mp3`	Compressed Audio Format	Audio classification
`.mp4`	MPEG-4 Video	Action recognition, video analysis

---

4. Configuration and Code

Extension	Description	Usage
`.py`	Python Script	ML model code
`.ipynb`	Jupyter Notebook	Interactive model development
`.yaml` / `.yml`	YAML Config File	Model config (e.g., for PyTorch Lightning)

---

5. Compressed Files

Extension	Description	Usage
`.zip`	ZIP Archive	Dataset/model packaging

Extension	Description	Usage
`.tar.gz`	Gzipped TAR Archive	Distributing data/models
`.7z`	7-Zip Archive	High-compression packaging

Summary

Category	Examples
Data Files	`.csv`, `.json`, `.xlsx`, `.parquet`
Model Files	`.pkl`, `.pt`, `.h5`, `.onnx`
Media Files	`.jpg`, `.wav`, `.mp4`
Code & Config	`.py`, `.ipynb`, `.yaml`
Archives	`.zip`, `.tar.gz`

In [ ]: