Advantages of File Handling in Python:

1. Flexibility: Python provides a wide range of functions and methods that can be used for file handling, making it very flexible and easy to work with different types of files.
2. Portability: Python can be used on different operating systems, which means that file handling can be done on different platforms.
3. Efficiency: Python's file handling is efficient and can handle large files with ease.
4. Versatility: Python can handle different types of files, such as text files, binary files, CSV files, and more.
5. Security: Python's file handling modules provide features for secure file operations such as permissions, encryption, and decryption.

Disadvantages of File Handling in Python:
1. Complexity: File handling in Python can be complex, especially for beginners, as it involves working with file objects, file pointers, modes, and exceptions.
2. Overhead: File operations can be slow and can consume a lot of resources if not done properly.
3. Risk of Data Loss: If a file is opened in write mode, the contents of the file will be overwritten, which can result in the loss of data if not done carefully.
4. Security: While Python provides modules for secure file operations, it is up to the programmer to ensure that proper security measures are in place.
5. Error-prone: File handling can be error-prone, as there are many things that can go wrong, such as file not found errors, permission errors, and others.

In summary, while file handling in Python is a powerful and flexible feature, it requires proper planning and execution to avoid errors and ensure the security of files.

Features of Python Files:

In Python, files are objects that provide a way to interact with external files in a program. Here are some of the key features of files in Python:

1. File Objects: In Python, files are represented as objects of the `file` class. These objects have various attributes and methods that can be used to read, write, and manipulate files.

2. Modes: When working with files in Python, you can open them in different modes depending on the type of operation you want to perform. The available modes are: 'r' for reading, 'w' for writing, 'a' for appending, and 'x' for creating a new file.

3. Opening and Closing Files: To work with files in Python, you need to open them first using the `open()` function. Once you're done with the file, you should close it using the `close()` method.

4. Reading and Writing Files: Python provides several methods for reading and writing to files, such as `read()`, `write()`, and `readline()`. These methods allow you to read and write text, binary data, and more.

5. File Position: In Python, you can move the file position indicator to a specific location in the file using the `seek()` method. This allows you to read or write data from a specific location in the file.

6. Context Managers: Python provides context managers, which allow you to automatically close a file when you're done with it. This helps to ensure that resources are properly managed and can help avoid memory leaks.

7. File Compression: Python provides modules for working with compressed files, such as `gzip`, `bz2`, and `zipfile`. These modules allow you to read and write compressed files without having to decompress them first.

Overall, Python provides a robust and flexible set of features for working with files. Whether you're working with text files, binary files, compressed files, or other types of files, Python provides the tools you need to interact with them in a programmatic way.