FILE HANDLING

CLASS XII

COMPUTER SCIENCE WITH PTHON

What is a File?

a **file** is a named collection of related data that is stored and retrieved as a single unit.

It's the fundamental way computers organize and persist information on storage devices.

Files can contain various types of data, including text, images, audio, video, program instructions, and more.

Each file has a name and resides within a directory (or folder) structure that helps organize the file system.

File Handling in Python

File handling in Python refers to the process of working with files on your computer's file system.

This includes various operations such as:

- **Opening a file:** Establishing a connection between your Python program and a specific file.
- **Reading from a file:** Retrieving data from an opened file.
- **Writing to a file:** Storing data into an opened file.
- **Appending to a file:** Adding new data to the end of an existing file.
- **Closing a file:** Terminating the connection to a file, releasing system resources.
- **Performing other operations:** Such as checking file existence, renaming, deleting, etc. (These are often done using the os module).

Types of Files

- Text File
- Binary file
- Comma Separated Values (CSV File)

Text File

- A **text file** is a fundamental type of computer file designed to store human-readable characters in a sequential manner. Think of it as a digital version of a simple notepad document.

Core Characteristics:

- **Plain Text Content:** The primary characteristic of a text file is that it contains plain text. This means it consists of characters like letters, numbers, punctuation marks, and a few special control characters (like tabs, spaces, and line breaks). It does not contain rich formatting like different fonts, sizes, colors, bolding, italics, embedded images, or other complex layout information found in word processing documents (e.g., .docx, .odt).

Core Characteristics:

- **Character Encoding:** Text files rely on a specific **character encoding** to represent each character as a numerical code that the computer can understand. Common encodings include:
  - **ASCII (American Standard Code for Information Interchange):** A basic encoding that represents 128 characters, primarily the English alphabet, numbers, and common symbols.
  - **UTF-8 (Unicode Transformation Format - 8-bit):** The most widely used modern encoding, capable of representing almost all characters from all writing systems in the world. It's the standard for most web pages and text files today.
  - **Other encodings:** Depending on the region and language, other encodings like Latin-1 (ISO-8859-1) might be used.

Core Characteristics:

- **Line Structure:** Text files are typically organized into lines. Each line is a sequence of characters that ends with a special newline character (\n). This character signals to text editors and other programs where one line ends and the next begins. Different operating systems might use slightly different newline conventions (e.g., Windows uses \r\n).
- **Simple Structure:** The internal structure of a text file is very simple: a sequence of characters. There's no complex metadata or formatting information embedded within the content itself.

Core Characteristics:

- **Universally Readable:** One of the biggest advantages of text files is their universal readability. Almost any text editor or word processor on any operating system can open and display the content of a plain text file. This makes them excellent for data exchange and storing information that needs to be accessible across different platforms.
- **Small File Size:** Due to the lack of formatting overhead, text files are generally smaller in size compared to documents with rich formatting.

Common Uses of Text Files:

- **Simple Documents and Notes:** Creating and storing basic text documents, notes, and reminders.
- **Configuration Files:** Many software applications use text files (like .ini, .conf, .txt, .json, .yaml) to store user preferences and program settings.
- **Log Files:** Systems and applications often write event logs to text files for debugging and monitoring purposes.
- **Source Code:** Programming languages often store their code in plain text files (e.g., .py, .java, .c, .html).
- **Data Storage and Exchange:** Simple datasets can be stored in text-based formats like CSV (Comma Separated Values) or TSV (Tab Separated Values) for easy import and export between applications.
- **Web Development:** Core web technologies like HTML, CSS, and JavaScript are written in plain text files.

Binary files

- **Binary files** are computer files that contain data stored in the form of sequences of bytes (0s and 1s) that are **not** primarily intended to be interpreted as human-readable text.
- Unlike text files, which are designed to store characters encoded in systems like ASCII or UTF-8, binary files can store a wide variety of data types, including numbers, images, audio, video, and executable code, in formats that are specific to the application or file type.

Core Characteristics:

- **Not Human-Readable (Directly):** When you open a binary file in a standard text editor, you'll typically see a jumble of seemingly random characters, symbols, and potentially unprintable characters. This is because the bytes in the file are not organized according to standard text encodings.
- **Specific Formats:** Binary files adhere to specific, often complex, formats that define how the data is structured and interpreted. These formats are usually proprietary or standardized for particular types of data. The application that created the binary file has the knowledge of this specific format.

Core Characteristics:

- **Direct Representation of Data:** Binary files can directly represent various data types in their native binary form. For example, an integer value can be stored directly as a sequence of bytes representing its binary equivalent, without needing to convert it to a sequence of text digits. This can lead to more efficient storage and faster processing.
- **No Implicit Line Structure:** Unlike text files, binary files do not inherently have a concept of "lines" delimited by newline characters. The structure is determined by the specific file format.

Core Characteristics:

- **Require Specific Applications for Interpretation:** To correctly understand and use the data in a binary file, you typically need the specific application or program that was designed to work with that particular file format. The application knows how to decode the bytes and present the information in a meaningful way (e.g., displaying an image, playing audio).
- **Efficiency and Performance:** Binary formats can often store complex data more efficiently and allow for faster read/write operations compared to text-based representations, especially for large amounts of structured data.

Examples of Binary File Types:

- **Executable Files (.exe, .app):** Contain compiled program code that the computer's processor can directly execute.
- **Image Files (.jpg, .png, .gif, .bmp):** Store pixel data, color information, and metadata in specific image encoding formats.
- **Audio Files (.mp3, .wav, .aac):** Contain encoded audio data representing sound waves.
- **Video Files (.mp4, .avi, .mov):** Store encoded video and audio streams, often with complex structures for synchronization and playback control.

Examples of Binary File Types:

- **Document Files (.doc, .docx, .pdf, .odt):** While they may contain text, they also include rich formatting, images, and other structured data stored in a binary format.
- **Database Files (.db, .sqlite, .mdb):** Store structured data in a specific binary format optimized for querying and management.
- **Compressed Archives (.zip, .rar, .tar.gz):** Contain one or more files compressed into a single binary file to reduce storage space.
- **Object Files (.obj, .o):** Contain compiled code that is later linked to create executable files.

CSV (Comma Separated Values) files

- A CSV file is a way to represent this exact same information using plain text.
- Each row of the table becomes a line in the CSV file, and the values within each row are separated by commas.

CSV (Comma Separated Values) files

Name	Age	City
Alice	30	New York
Bob	25	London
Charlie	35	Paris, France

Export to Sheets

Code snippet

```
Name,Age,City
Alice,30,"New York"
Bob,25,London
Charlie,35,"Paris, France"
```

Core Characteristics:

- **Plain Text:** The fundamental characteristic of a CSV file is that it's a plain text file. This means you can open it and read its contents using any basic text editor (like Notepad on Windows, TextEdit on macOS, or various code editors). You won't see any special formatting, fonts, or styles, just the raw characters.
- **Comma as a Delimiter:** The core of the format is the use of a comma (,) to separate individual values within each row. This comma acts as a delimiter, indicating where one piece of data ends and the next begins.

Core Characteristics:

- **Rows and Records:** Each **line** in a CSV file typically represents a single **record** or row of data from your table. In the example above, each line after the header represents information about one person.
- **Values and Fields:**The individual pieces of data within each row, separated by commas, are called **values** or **fields**. These correspond to the columns in your table (Name, Age, City in our example).
- **Optional Header Row:** Often, the **first line** of a CSV file contains a **header row**. This row lists the names of the columns, making it easier to understand the meaning of the data in each subsequent row. In our example, "Name,Age,City" is the header row.

Core Characteristics:

- **Handling Special Characters (and the Need for Quotes):**
- **Commas within data:** If a data value itself contains a comma (like "Paris, France"), it needs to be enclosed in double quotes (") to prevent the program reading the file from misinterpreting the comma as a delimiter between fields.
- **Double quotes within data:** If a data value contains double quotes, they are usually escaped by another double quote (e.g., ""Quoted Value"") or the entire value is enclosed in double quotes.
- **Line breaks within data:** Similar to commas, if a data value spans multiple lines, it's typically enclosed in double quotes.

Why is CSV such a popular format?

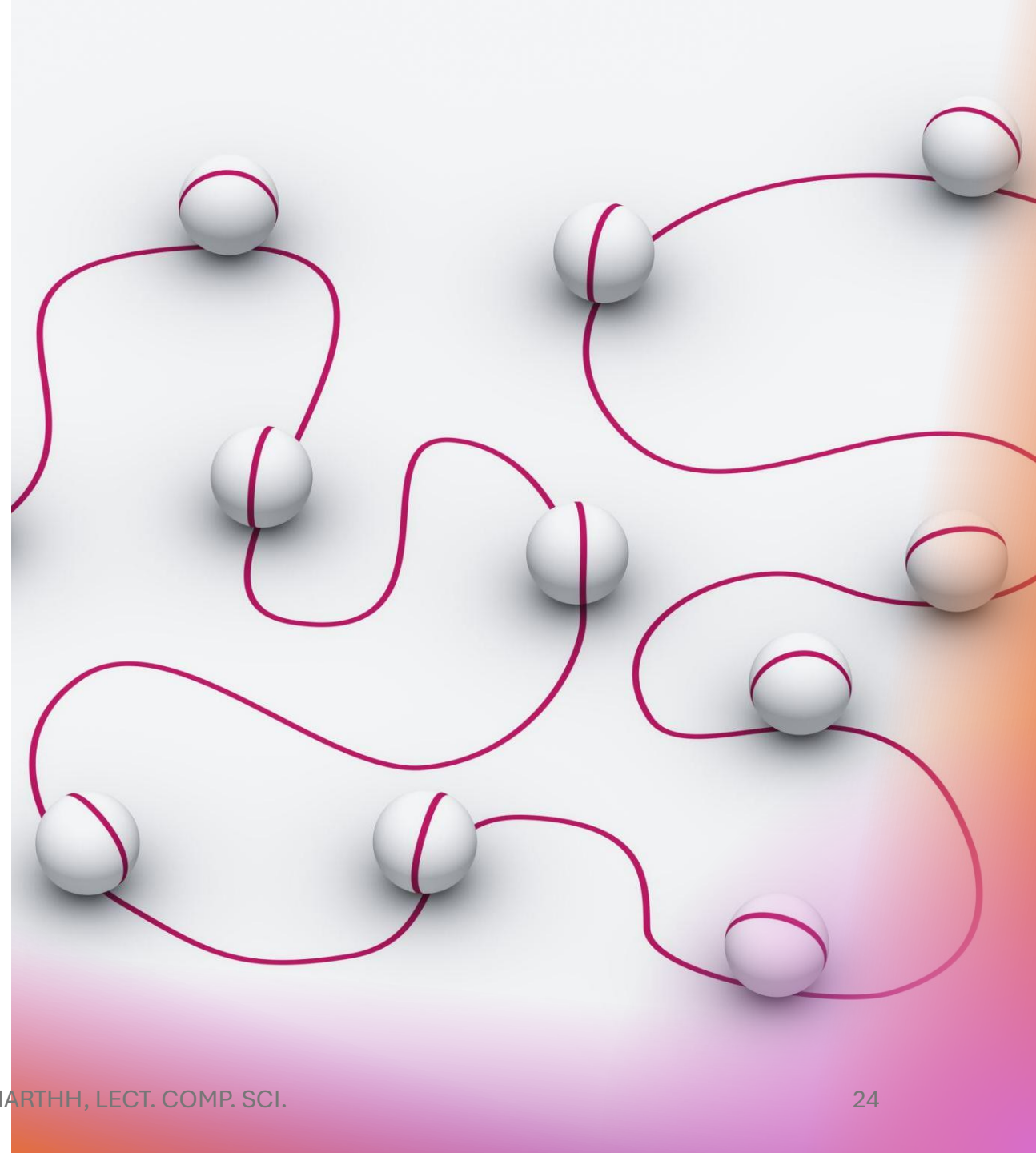
- **Simplicity:** The format is incredibly simple and easy to understand and implement by different software.
- **Compatibility:** Virtually every spreadsheet program (like Microsoft Excel, Google Sheets, LibreOffice Calc), database management system, and many programming languages have built-in support for reading and writing CSV files. This makes it an excellent format for data exchange between different applications and systems.
- **Portability:** Because it's plain text, CSV files are highly portable across different operating systems and hardware platforms.

Why is CSV such a popular format?

- **Human-Readable (to a degree):** While not as visually appealing as a formatted spreadsheet, the data in a CSV file is still relatively easy for humans to read and understand with a simple text editor.
- **Efficiency for Simple Data:** For straightforward tabular data without complex formatting, CSV files are often more efficient in terms of file size compared to more complex formats like Excel's .xlsx.

Path

- A **path** is a string of characters that uniquely identifies the location of a file or directory (folder) within a file system. Think of it as the address that tells your computer exactly where to find a specific item.
- There are two main types of paths:
  - Absolute Path (or Full Path)
  - Relative Path



Absolute Path (or Full Path)

- An absolute path specifies the exact location of a file or directory, starting from the root directory of the file system.
- It provides a complete and unambiguous way to locate an item, regardless of your current working directory.

Absolute Path (or Full Path) Example

Example (Windows): `C:\Users\YourUsername\Documents\MyFile.txt`

- `C:\` is the root directory of the C: drive.
- `Users` is a directory within the root.
- `YourUsername` is a subdirectory within `Users`.
- `Documents` is a subdirectory within `YourUsername`.
- `MyFile.txt` is the specific file located in the `Documents` directory.

Relative Path

- A relative path specifies the location of a file or directory relative to your current working directory.
- It doesn't start from the root directory. Instead, it assumes a starting point (your current location in the file system) and then describes how to get to the target item from there.
- Relative paths are often shorter and more convenient when working with files and directories within the same project or a related part of the file system.

Relative Path

Example (assuming your current working directory is `C:\Users\YourUsername` on Windows):

- To access `Documents\MyFile.txt`: The relative path is simply `Documents\MyFile.txt`.
- To access a file in a subdirectory of the current directory, like `Pictures\VacationPhotos\photo.jpg`: The relative path is `Pictures\VacationPhotos\photo.jpg`.
- To go up one level to `C:\Users` and then into `Public\SharedFiles`: The relative path is `..\Public\SharedFiles`.

Diff. Absolute Path and Relative Path

Feature	Absolute Path	Relative Path
Starting Point	Root directory (e.g., C:\ or /)	Current working directory
Completeness	Complete and unambiguous	Depends on the current location
Length	Generally longer	Often shorter
Portability	Less portable (drive letters differ)	More portable within a project structure
Use Cases	Specifying precise locations, system-wide	Working within a project, scripts that move