1. To what does a relative path refer?

A relative path refers to the location of a file or directory in relation to the current working directory. It doesn't start from the root directory of the file system but rather from the directory where the user or program is currently located. Relative paths are generally shorter and easier to work with when you want to reference files and directories within the same project or directory structure.

2. What does an absolute path start with on your operating system?

An absolute path starts with the root directory of the file system on your operating system. The root directory is the highest-level directory in the file hierarchy. On Unix-like systems (including Linux and macOS), the absolute path starts with a forward slash ("/"). On Windows systems, it typically starts with a drive letter followed by a colon (e.g., "C:") and then a backslash ("").

3. What do the functions os.getcwd() and os.chdir() do?

* os.getcwd(): This function, part of the os module in Python, returns the current working directory (CWD) as a string. The current working directory is the directory from which the Python script is currently being executed.
* os.chdir(path): This function is used to change the current working directory to the specified path. You provide a path as an argument, and the CWD will be set to that path. This allows you to navigate and operate on files and directories in a different location within the file system.

4. What are the . and .. folders?

In the context of file systems, . (dot) refers to the current directory, and .. (double dot) refers to the parent directory. These are special directory notations used to navigate the directory structure.

* . (dot): When used in a path, it represents the current directory. For example, if you're currently in the directory /home/user/docs, ./file.txt would refer to the file file.txt in the current directory.
* .. (double dot): When used in a path, it represents the parent directory. For example, if you're in the directory /home/user/docs, ../notes.txt would refer to the file notes.txt in the parent directory (/home/user).

5. In C:\bacon\eggs\spam.txt, which part is the dir name, and which part is the base name?

In the path C:\bacon\eggs\spam.txt, the directory name (dir name) is C:\bacon\eggs, and the base name is spam.txt. The directory name represents the path to the folder containing the file, while the base name is the actual filename.

6. What are the three “mode” arguments that can be passed to the open() function?

The three mode arguments that can be passed to the open() function in Python are:

* 'r': Read mode. This mode allows you to read the contents of the file.
* 'w': Write mode. This mode allows you to write data to the file, overwriting its previous contents. If the file doesn't exist, a new file will be created.
* 'a': Append mode. This mode allows you to write data to the end of the file without overwriting existing content. If the file doesn't exist, a new file will be created.

7. What happens if an existing file is opened in write mode?

If an existing file is opened in write mode ('w'), the contents of the file are completely overwritten with the new data you write to it. If the file doesn't exist, a new file will be created. Be cautious when using write mode on an existing file, as you will lose the original content.

8. How do you tell the difference between read() and readlines()?

* read(): The read() method is used to read the entire contents of the file as a single string. It reads from the current file position until the end of the file. You can specify the number of characters to read as an argument, but if no argument is provided, it reads the entire file.
* readlines(): The readlines() method reads the lines of the file and returns a list of strings, where each string represents a line in the file. It reads from the current file position until the end of the file or until a specified number of bytes.

9. What data structure does a shelf value resemble?

A shelf value in Python resembles a dictionary data structure. The shelve module in Python provides a way to store and retrieve data using a persistent dictionary-like format. It allows you to store key-value pairs, similar to how you would store data in a dictionary, but the data is stored on disk rather than in memory. Shelves are often used for storing and managing data that doesn't fit entirely in memory.