**1. To what does a relative path refer?**

A relative path in Python refers to a file or directory path that is relative to the current working directory, rather than an absolute path that starts from the root directory. Relative paths make it easier to reference files and directories that are located in the same directory or in a subdirectory of the current directory.

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

In most operating systems, an absolute path starts with the root directory represented by a forward slash "/" (Unix/Linux/macOS) or a drive letter followed by a colon ":" and a backslash "" (Windows).

For example, on Unix/Linux/macOS, an absolute path might look like:

/home/user/documents/file.txt

On Windows, an absolute path might look like:

C:\Users\user\Documents\file.txt

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

In Python, the os module provides a way to interact with the operating system.

os.getcwd(): This function returns the current working directory (cwd) as a string in the form of an absolute path. The current working directory is the default location for most file operations and is the directory where the Python script is executed from.

os.chdir(path): This function changes the current working directory to the directory specified by the path argument. The path argument must be a string representing a valid directory path.

**4. What are the . and .. folders?**

In Python (and in most operating systems), . and .. are special directories that refer to the current directory and the parent directory, respectively.

. represents the current directory. For example, if your current working directory is /home/user/documents, then ./file.txt refers to the file file.txt in the /home/user/documents directory.

.. represents the parent directory. For example, if your current working directory is /home/user/documents, then ../file.txt refers to the file file.txt in the /home/user directory.

These special directories can be used in relative paths to navigate between directories without specifying the full path.

**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 part C:\bacon\eggs is the directory name and the part spam.txt is the base name.

In Python, you can use the os.path module to extract the directory name and base name from a file path.

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

The open function in Python can take an optional mode argument that specifies how the file is opened. The following are the three common mode arguments that can be passed to the open function:

'r' (Read-only mode): This is the default mode and opens the file for reading only. The file pointer is placed at the beginning of the file. If the file does not exist, an error is raised.

'w' (Write mode): This mode opens the file for writing. If the file already exists, its contents are truncated, and a new file is created if it does not exist.

'a' (Append mode): This mode opens the file for writing and appends new data to the end of the file. If the file does not exist, a new file is created.

There are other mode arguments as well, such as 'x' (Exclusive creation), 'b' (Binary mode), and 't' (Text mode), but the above three are the most commonly used.

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

If an existing file is opened in write mode ('w') in Python, the file's contents are truncated (deleted), and a new empty file is created. The file pointer is placed at the beginning of the file, so any data written to the file will overwrite the existing data. If the file does not exist, a new file is created.

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

The read and readlines methods are used to read the contents of a text file in Python. The main difference between them is the format in which the data is returned.

read method: The read method reads the entire contents of the file and returns it as a single string. The string may contain newline characters (\n) which separate lines of text in the file.

readlines method: The readlines method reads the entire contents of the file and returns a list of strings, where each string represents a line of text in the file. The newline characters (\n) are included at the end of each string.

In general, if you want to process the contents of a file line by line, the readlines method is the better choice. If you want to process the entire contents of the file as a single string, the read method is the better choice.

**9. What data structure does a shelf value resemble?**

In Python, a shelf value resembles a dictionary. A shelf is a persistent, dictionary-like object that can store key-value pairs and preserve their values across sessions. It's implemented as a binary file that can be accessed using the shelve module. The keys and values of a shelf can be of any Python object type, including numbers, strings, lists, and even other custom objects.