1. To what does a relative path refer?

Ans. A relative path is a path that is specified relative to the current working directory of the program or user. It’s base path is where file is located.

For example, let's say we have a file example.txt in a directory called data, which is located in the same directory as our Python script. We can specify the path to the file using a relative path like this:

file\_path = "data/example.txt"

In this case, the relative path data/example.txt specifies the path to the file relative to the current working directory, which is the directory that contains the Python script.

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

Ans. In Python, as in most operating systems, an absolute path starts with the root directory of the file system.

In Unix-based systems (including Linux and macOS), the root directory is denoted by a forward slash (/). For example, an absolute path to a file called example.txt located in the /home/user/documents directory would be:

file\_path = "/home/user/documents/example.txt"

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

The os.getcwd() function returns the current working directory as a string. The current working directory is the directory in which the Python script is being executed

The os.chdir() function changes the current working directory to the specified path. It takes a path string as its argument and changes the current working directory to that path.

4. What are the . and .. folders?

The . directory (also called the current directory) refers to the current directory itself. For example, if we are in the /home/user/documents directory and you refer to the . directory, we are referring to the /home/user/documents directory.

The .. directory (also called the parent directory) refers to the parent directory of the current directory. For example, if we are in the /home/user/documents directory and we refer to the .. directory, you are referring to the /home/user directory.

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

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

The open() function in Python is used to open a file and returns a file object. It takes at least one argument, the file name (or path), and can also take an optional mode argument that specifies how the file should be opened.

There are three main mode arguments that can be passed to the open() function:

* "r" (read mode): Opens the file for reading (default). Raises an error if the file does not exist.
* "w" (write mode): Opens the file for writing. If the file already exists, it will be truncated (i.e. all contents will be deleted). If the file does not exist, a new file will be created.
* "a" (append mode): Opens the file for appending. If the file already exists, new data will be written to the end of the file. If the file does not exist, a new file will be created.
* "b": binary mode, for working with binary files
* "t": text mode, for working with text files (default)
* "x": exclusive creation mode, for creating a new file and raising an error if it already exists
* "+": read and write mode

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

If an existing file is opened in write mode using the open() function in Python, the contents of the file will be truncated (i.e., deleted) and any data written to the file will overwrite the existing content.

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

read() reads the entire contents of a file as a string and returns the result. This means that if we call read() on a file, we will get a single string containing all of the data in the file.

readlines(), on the other hand, reads the entire contents of a file and returns a list of strings, where each string in the list corresponds to a line in the file

9. What data structure does a shelf value resemble?

A shelf value resembles a dictionary in that it is a collection of key-value pairs. However, unlike a dictionary, a shelf is backed by a file on disk, which allows it to persist even after the Python interpreter is closed.