ASSIGNMENT-9 (PYTHON)

1. To what does a relative path refer?

A relative path refers to the location of a file or directory relative to the current working directory.

In computing, the file system organizes files and directories in a hierarchical structure, starting from the root directory. The current working directory is the directory from which a program is currently running or executing.

A relative path specifies the location of a file or directory relative to the current working directory, rather than specifying the complete path from the root directory.

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

/Users/nehatahalramani/Library/Mobile Documents/com~apple~TextEdit/Documents/pythonetest.rtf

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

The functions os.getcwd() and os.chdir() are part of the os module in Python, which provides functions for interacting with the operating system.

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

os.chdir(path): This function changes the current working directory to the one specified by the path argument. It takes a string argument representing the path of the directory to change to.

By using `os.getcwd()` and `os.chdir()`, you can navigate and manipulate the current working directory within your Python scripts. This is useful for accessing files, creating directories, or organizing your project's file structure.

4. What are the . and .. folders?

1. . (dot): The dot (.) directory represents the current directory. It is a reference to the directory in which the dot entry is located. For example, if you execute a command or open a terminal in a directory, that directory is referred to as ..

2. `..` (dot-dot): The dot-dot (`..`) directory represents the parent directory of the current directory. It is a reference to the directory one level above the directory in which the dot-dot entry is located. For example, if you navigate into a subdirectory from another directory, the subdirectory's `..` refers to the parent directory.

These entries are used to navigate the file system hierarchy easily, especially in command-line interfaces and when specifying file paths. For instance, you can use `.` to refer to the current directory in path operations, and you can use `..` to refer to the parent 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 and the base name can be identified as follows:

- Directory name: `c:\bacon\eggs`
- Base name: `spam.txt`

The directory name refers to the path of the directory containing the file, while the base name refers to the name of the file itself.

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

The open() function in Python is used to open files for reading, writing, or both, depending on the mode argument passed to it.

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

If an existing file is opened in write mode ('w') using the open() function in Python, the file's existing contents are truncated, meaning they are completely erased, and the file is treated as empty. Then, the file is ready to accept new data written to it. If the file does not exist, a new file will be created.

It's essential to be cautious when opening a file in write mode because any existing data in the file will be lost. If you want to preserve the existing data and append new content to the end of the file, you should use append mode ('a') instead of write mode.

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

read():

The read() method reads the entire contents of the file as a single string, including newline characters (\n) and any other characters present in the file.
It returns a single string containing the entire contents of the file.
If the file is empty or at the end of the file, read() returns an empty string ('').

readlines():

The readlines() method reads all lines from the file and returns them as a list of strings. Each string in the list represents a single line from the file, including the newline character (\n) at the end of each line.
If the file is empty or at the end of the file, readlines() returns an empty list ([]).

9. What data structure does a shelf value resemble?

A shelf value in Python resembles a dictionary data structure.

Python's shelve module provides a persistent, dictionary-like interface for storing and retrieving objects. It allows you to persistently store arbitrary Python objects (such as lists, dictionaries, etc.) to a file on disk and retrieve them later.

A shelf behaves like a dictionary in that it allows you to store key-value pairs, where keys are unique and hashable, and values can be arbitrary Python objects. You can use familiar dictionary methods like keys(), values(), items(), get(), pop(), etc., with shelf objects.