**Python Assignment - 9**

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

=> A relative path refers to a file or directory location that is specified relative to the current working directory or another known reference point. It describes the location of a file or directory relative to the current location in the file system.

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

=> The prefix for an absolute path depends on the operating system. Here are the prefixes used by common operating systems:

1. Unix-like Systems (e.g., Linux, macOS):

* Absolute paths start with a forward slash (/). For example: /home/user/file.txt

2. Windows:

* Absolute paths start with a drive letter followed by a colon (:) and a backslash (\). For example: C:\Users\User\file.txt

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

=>

1. os.getcwd():

* os.getcwd() returns a string representing the current working directory (CWD) of the Python script or application.
* The current working directory is the directory in the file system from which the script or application is currently running.
* This function does not take any arguments.

2. os.chdir(path):

* os.chdir(path) changes the current working directory to the specified path.
* The path argument is a string representing the desired directory path to which the CWD should be changed.
* If the path is a relative path, it is interpreted relative to the current working directory.
* This function returns None

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

=> In the context of file systems, the . (dot) and .. (dot-dot) folders are special directory references that have specific meanings:

1. **‘.’ (dot) folder**:

* The . folder is a reference to the current directory itself.
* It is used to indicate the current working directory in file system paths.
* For example, if you are in the /home/user directory, the path ./file.txt refers to the file file.txt in the current directory.

2. **‘..’ (dot-dot) folder**:

* The .. folder is a reference to the parent directory of the current directory.
* It is used to indicate the parent directory in file system paths.
* For example, if you are in the /home/user directory, the path ../file.txt refers to the file file.txt in the parent directory of /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 and the base name can be identified as follows:

1. Directory Name: C:\bacon\eggs

* The directory name refers to the path of the directory that contains the file.
* In this case, the directory name is C:\bacon\eggs.

2. Base Name: spam.txt

* The base name refers to the file name itself, without the directory path.
* In this case, the base name is spam.txt.

**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:

1. Read Mode ('r'):

* This is the default mode when no mode argument is provided.
* It opens the file for reading.
* If the file does not exist, it raises a FileNotFoundError.
* Example: open('file.txt', 'r')

2. Write Mode ('w'):

* It opens the file for writing.
* If the file exists, it truncates the contents and starts fresh.
* If the file does not exist, it creates a new file.
* Example: open('file.txt', 'w')

3. Append Mode ('a'):

* It opens the file for appending data at the end of the file.
* If the file exists, it moves the file pointer to the end of the file.
* If the file does not exist, it creates a new file.
* Example: open('file.txt', 'a')

**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 following consequences occur:

1. The existing file's content is truncated:

* When opening a file in write mode, if the file already exists, its existing content will be deleted or truncated. The file will be emptied, and any previous data will be lost.

2. A new empty file is created if the file doesn't exist:

* If the file specified in write mode does not exist, a new empty file with the given name will be created.

3. The file is positioned at the beginning:

* After opening the file in write mode, the file pointer is positioned at the beginning of the file, ready to write data.

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

=> The differences between read() and readlines() are:

1. read():

* The read() method reads the entire content of a file as a single string.
* It reads from the current position of the file pointer until the end of the file.
* It returns a string containing the entire content of the file.
* Example: file.read()

2. readlines():

* The readlines() method reads the content of a file line by line and returns a list of strings.
* It reads from the current position of the file pointer until the end of the file.
* Each line is represented as a string element in the returned list.
* Example: file.readlines()

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

=> In Python, a shelf value, as obtained from the shelve module, resembles a dictionary data structure. It behaves similarly to a dictionary and provides key-value storage for persistent objects.

A shelf is a persistent, dictionary-like object that stores data on disk. It allows you to store and retrieve Python objects using keys, just like a dictionary. The shelve module utilizes a disk-based hash table implementation to store the data.