

# **OS CLASS ASSIGNMENT**

# **Submitted to:**

# Mam Farzana Jabeen

# **Submitted from:**

# Awais Bin Adil (288166)

# Kaynat Sajid (289051)

# Urooj Fatima Raza (289365)

# Zainab Asif (295486)

# **File System Structure:**

We choose Dictionaries in Python for our system.

Dictionaries are used to store data values in **key: value** pairs. Dictionary items are ordered, changeable, and does not allow duplicates. Dictionary items are presented in key: value pairs, and can be referred to by using the key name.

Our Dictionary has root/super dictionary, which has key: value pairs, which has further dictionaries. Each directory has further files in it, where files store the its content.

* Dictionaries –
* Dir1-
* File 1
* File 3
* Dir2
* File 2

We have a Class File, which has functions and constructors that our used in functions later on.

## Meta Data:

Each File has its own Size, Name, Content and Length.

# **Reason:**

* We initially tried to implement the file system using **B trees** but since there are no pointers in python, using B trees in python is very complicated. So we switched to python dictionaries because they are in-built in python and eliminates the use of pointers.
* Using python dictionaries also allows us to use **built in functions** which greatly simplified things.
* Dictionaries use **Hash tables** as a result CRUD (create, read, update and delete) functions take constant time O (1). You can look up a key in a Python dictionary very fast. The speed of a task like looking up keys is measured by looking at how many operations it takes to finish. Looking up a key is done in constant time vis-a-vis looking up an item in a large list which is done in linear time.

# **System Functionalities**

The file system was created using Python 3.7.6. Our File System allows us to perform 13 functions.

1. Create a file
2. Delete a file
3. Make directory
4. Change directory
5. Move a file
6. Open a file
7. Close a file
8. Write to a file
9. Read from a file
10. Move within a file
11. Truncate a file
12. Memory Map
13. Exit

## Create a file

Our data structure (dictionary) has initially has 2 directories and you can create files in either one of them. The function create\_a\_file () takes two arguments i.e the name and content of the file and you can choose what directory you want to create the file in.

## Delete a file

As the name indicates this allows us to delete a file in any directory. You are required to input the name of the file you want to delete.

## Make directory

You can create a new directory through this feature. You are required to input the name of the directory you want to create.

## Change directory

This function allows us to change the current directory we are residing in. This is similar to using the cd command in windows terminal.

## Move a file

This function allows you to move a file one directory to another and required you to input the name a directory of the file you want to move and the destination directory.

## Open a file

This function allows us to return the file as an object.

## Close a file

This function allows us to close the opened file.

## Write to a file

This function allows us to append text at the end of a file or you can also write text at a specific point in the text.

## Read from a file

This function allows you to read the entire file as displays it on the console. Or you can read from a specific part of the file.

## Move within a file

This function allows you to move text from one position to another. You are required to input the position you want to move the text from and the position you want to move it too and also the size of the text.

## Truncate a file

User inputs the size of the file he wants to keep and the rest of the file is truncated.

## Memory Map

The memory map displays the directories and files currently existing in the file system and also their size and length.

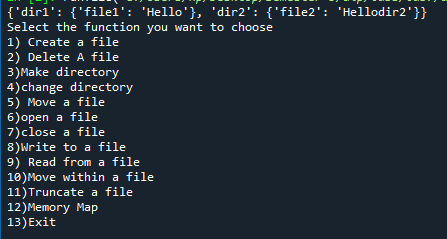
## Exit

This quits the loop which shows the main menu.

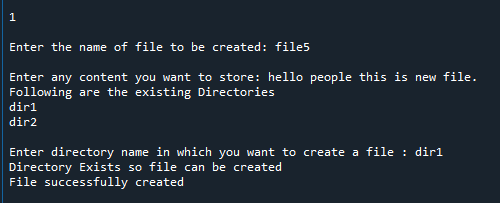
# **Limitations:**

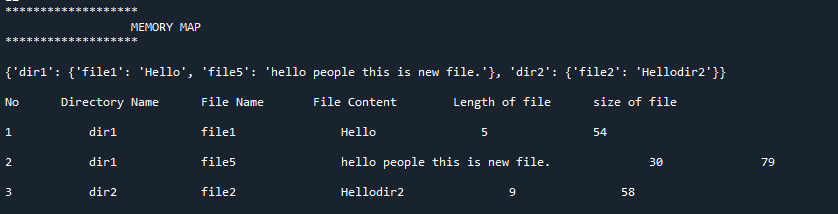
* File operations such as creating, deleting, moving files are working **logically**, but not physically created or altered in memory.
* Our function, **change directory** changes a current directory, it operates correctly. But it does not show current path, which makes it harder for user to identify which directly he currently is in. This is not a logical error, but can improve user experience.

# **Output:**

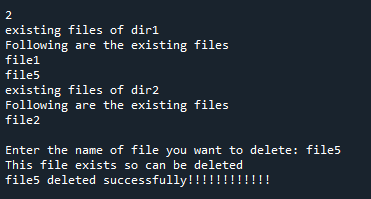
****

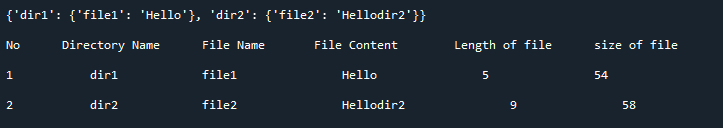
Create a File:



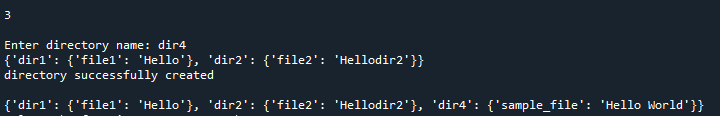


Delete file:



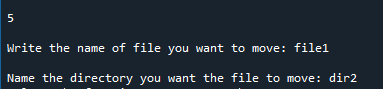


Make Directory:

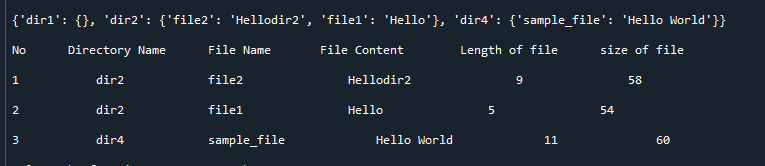


Change Directory:

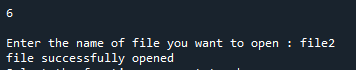
# 



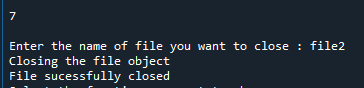
Move a File:



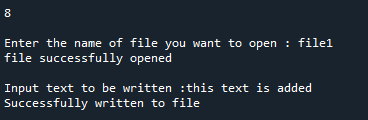
Open File:

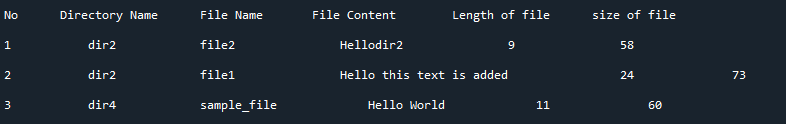


Close File:

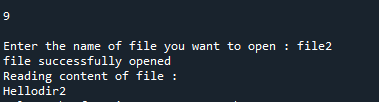


**Write to a File**

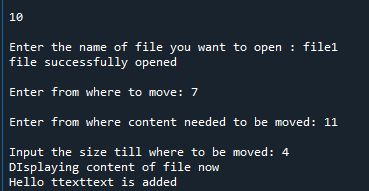




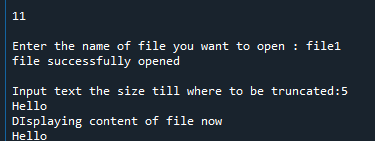
Read from a File:



Move within a File:



Truncate a File:



Memory Map:

# 

References:

<https://www.w3schools.com/python/python_dictionaries.asp>

<https://www.tutorialspoint.com/python/python_dictionary.htm>

<https://www.geeksforgeeks.org/python-dictionary/>

<https://www.youtube.com/watch?v=rZjhId0VkuY>