Test report

We used unittest, assertEqual and assertIn in test part.

In each part this test will create a user/admin instance.

It will have an expected result according to testing function.

it will call the service function by the user/admin instance. The return answer will be saved in result.

Result and expected result will be compared by assertEqual.

At the end of each test all created files and folders will be deleted to not have any problem in next test running.

1. test_create_folder: This is a basic test for creating folder which tests if any user can create any folder in his own directory. This test first registered an admin automatically with username = user1 and password = pass1, and automatically create a folder for it in a Admin folder.

   Then test what will happen if this user wants to create a folder in his own directory. We made an object from Admin class, then we call create folder function with this object, put the answer in result and compare it with expected result.

2. test_change_folder: This is a test for changing folder which tests if any user can change his current folder and enter to an existing folder. Here first the test makes a folder for a admin, then make an object from Admin class, call the function with the object and test it.

3. test_back_folder: This is a test for changing folder to back folder which tests if a user can walk back the previous folder from his current folder. Here first the test makes a folder for a admin, then make an object from Admin class, call the function with the object and test it.

4. test_print_list: This is a test for print_list which print current directory files and folder with size and date of creations. In this test we just test if the current directory files and folders get printed correctly. The difference of this test with last ones is that: the expected result is a list containing files and folders also the asserIn is used to check if any of results are the expected result list or not. For testing, the test creates a folder for a admin. And also a file and a folder in that. Then

5. test_read_file: This is a test for read_file functions. The test creates a folder for an admin. Then makes admin instance, creates a file in the admins folder and writes some text in it, calls the function by the object, puts the answer in result and compare it with expected result with assertEqual.

6. test_write_file: This is a test for write_file functions. The test creates a folder for an admin. Then makes admin instance, calls the function by the object, and write some input in a file which is not exist, so with is function: the file will be create and the input will be write on the created file. Then we try to read the input of file and compare it with expected result with assertEqual.

7. test_delete: This is a test for delete functions. The function just can be used by an admin. And deleted user would not be in registered_info.json also its own folder should be deleted. And also it should be deleted from signed_info. So it creates to object of Admin, makes a folder for each of them, for the second one it makes a file and a folder inside previous folder. Add their name to signedin_info.json and registered_info.json then calls the delete function with the first object and gives the second object name as a username who should be deleted. The function would automatically deletes the second user from json files and delete the its folder (any file and folder in it would be deleted)