# **Table of Contents**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Item** | **Page Number** |
| 1 | Introduction to the Project |  |
| 2 | Libraries Used |  |
| 3 | Functions in Code |  |
| 4 | Database |  |
| 5 | Python Code |  |
| 6 | Output |  |
| 7 | Tools and References |  |

# **Introduction to the Project**

**Project Name:** File Modification System Using Python and MySQL

**Description:**

This project aims at creating a system that performs file modification functions like renaming the file, deleting it from the system (without the use of recycle bin) and creating a file with extensions ‘.txt’ to create a dummy file. The scope of rename and delete functions go beyond ‘.txt’ and can be performed on any file on your computer.

Furthermore, it creates a record in the MySQL database that summarises the names of files created/ deleted/ renamed, time the action was taken and the type of action performed.

**Concepts Used:**

* Libraries
* Functions
* File Handling
* Python - My SQL Connector

**Motivations:**

File handling provides a mechanism to store the output of a program in a file and to perform various operations on it. The functions of file modification functions like the ones implemented in the project provide a better idea of how internal functions in computers work.

**Challenges faced during the project:**

It was quite challenging to work as a team remotely when every part of the code was integral for the execution of the system. We also had to research a lot of errors that showed up when we tried new functions in our code.

**Learning Outcomes:**

* Gained a better understanding of file modification functions
* Applied operating system related modules - os, sys in the code
* Increased familiarity with Python with MySQL database
* Gained experience with new error types, related to modules and file handling functionalities

# **Libraries**

**Libraries Used:** os, sys, time, mysql.connector

**os**

Python OS module allows us to use the operating system dependent functionalities and to interact with the underlying operating system in several different ways. For example, we can work with files, change the environment variables, and we can move files around, etc. This is as same as overriding all the os built-in functionalities in a module and using them in a file I/O and system handling.

Functions used from the os library

**os.rename()** - to rename a file or a folder.In arguments pass the original file name first and then the new name of the file.

**os.path()** - sub-module of OS module in Python used for common pathname manipulation

**os.path.exists()** - () method in Python is used to check whether the specified path exists or not. This method can be also used to check whether the given path refers to an open file descriptor or not.

**os.remove()**  - removes the path of a file. It takes path string as a variable.

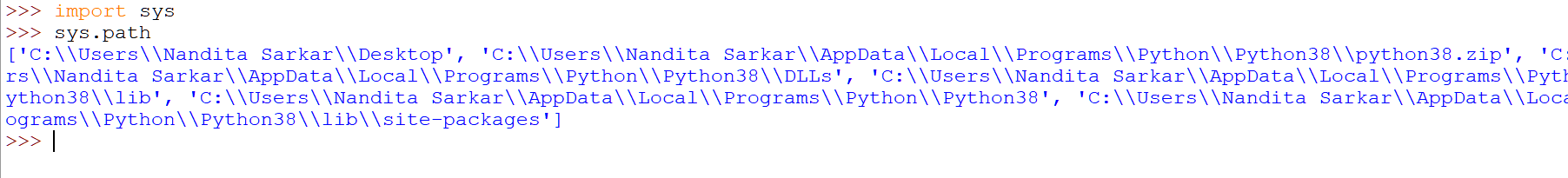
**sys**

Provides easy functions that allow us to interact with the interpreter directly. The functions python sys module provides allows us to operate on underlying interpreter, irrespective of it being a Windows Platform, Macintosh or Linux.

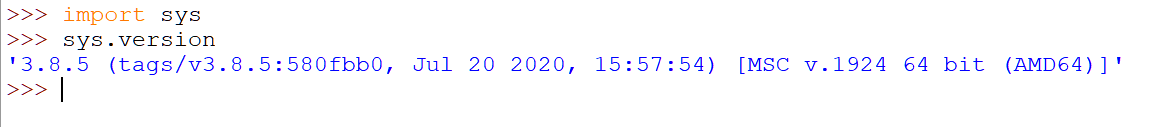
Functions used from the sys library

**sys.exit()** - This method makes the Python interpreter exits the current flow of execution abruptly.

**sys.path** - This function just displays the python path set in current system.



**sys.version -** This attribute displays a string containing the version number of the current Python interpreter.



**Datetime**

Supplies classes for manipulating dates and times.

Functions used from the datetime library

**datetime.datetime**  - Provides current date and time

**mysql.connector**

MySQL Connector/Python enables Python programs to access MySQL databases, using an API that is compliant with the Python Database API Specification v2.0 (PEP 249). It is written in pure Python and does not have any dependencies except for the Python Standard Library.

Functions used from the mysql.connector library

# **Functions**

**Functions in the code:** os, sys, time, mysql.connector

1. **main()**

The main() function is the body of the python project. It consists of the pathways to all the other functions and calls functions into it. Global variables file and time have also been declared into it. main() also consists of code for choices provided in the menu() function. All the choices in menu() are executed using main().

1. **menu()**

The menu() function has three choices:

* Rename a file
* Delete a file
* See file records

Upon choosing ‘Rename a file’, the user will be able to rename their file as long as the file

exists in the same location as the python code. The file extension also must be entered.

Alternatively, if the file doesn’t exist, a dummy file can be created.

The ‘Delete a file’ option allows users to delete files of any extension of their choice. The

requirements for deleting a file are the same as for renaming a file. When a file is deleted

using this method, it doesn’t go to the recycle bin. Instead it directly gets removed from

the memory of the computer. The file gets de-linked from the computer memory.

Different computer systems may utilise different methods to de-link the file deleted

using this method. If the coder aims to send the file first to the recycle bin and then

delete, the winshell library must be used.

The third option is to see the history of all file records : renamed, deleted and created.

When the user chooses this option, they are prompted with another menu that asks them

whether they want to see specific action types (deleted, renamed, created). There is also a

fourth option for seeing all the file types in sequential order.

If an unknown key an input as a response to the menu, the user is returned with an error

message.

1. **dummyfile()**

When this function is run, a ‘dummyfile.txt’ file is created in the same path as the location of the python file.

1. **file()**

In case the user wants to try this code on an existing file on their computer, they must choose ‘Y’ when the main() asks them so. Upon doing the same, they will be asked to enter the filename with extension. It is important to note that the file must exist in the same location as the python file because it uses relative, and not absolute directories. Absolute directories are being worked on and will be implemented in FilleManager v2.

1. **confirm()**

The confirm() function is a checking tool to verify if the user wants to execute the function. If they pressed a certain option accidentally, they have the chance to recover that mistake by giving a negative response at the confirm prompt. Upon doing so, the python code will stop functioning and needs to be re-run.

If the user plans to execute the action that they chose in the menu, they must proceed by typing ‘Y’ / ‘y’ in the response prompt.

The confirm() function not only gives the user a chance to re-think their choice, but also allows the computer to use less code so that the database doesn’t have to delete any data after it has committed.

That being said, once a data has been logged into the database, it cannot be deleted. Therefore, the user must use the confirm() function as an opportunity to consider their choices.

# **Database**

**Functions in the code:** os, sys, tim

# 

# 

# 

# 

# 

# 

# 

# 

# 

**Python Code**

import os

import sys

import datetime

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="root",

password="anneysha"

)

cursor = mydb.cursor()

cursor.execute(" create database if not exists mydb ")

cursor.execute(" use mydb ")

cursor.execute("CREATE TABLE if not exists filerecords(Start VARCHAR(255), Stop VARCHAR(255), Timestamp VARCHAR(255), Type VARCHAR(255))")

def main():

global file

global time

file()

menu()

choice = int(input(" Please choose between 1, 2 and 3. "))

if choice == 1:

old\_name = input("enter file name")

new\_name = input("enter new name")

if not os.path.exists(new\_name):

print("File name is unique. Renaming ....")

os.rename(old\_name,new\_name)

print("File rename successful!")

else:

print("File name exists.")

#Current\_Date = datetime.datetime.today().strftime ('%d-%b-%Y')

#cursor.execute(" insert into filerecords values(?, ?, ?, 'Renamed')".format(old\_name, new\_name, Current\_Date))

#mydb.commit()

elif choice == 2:

print (" Your file is going to be deleted.......... ")

confirm()

if os.path.exists("test.txt"): # checking if file exists

os.remove("test.txt") # file is deleted

print(f" Your file {file} has been deleted. ")

Current\_Date = datetime.datetime.today().strftime ('%d-%b-%Y')

#renamedict = {'Start': file, 'Stop': 0, 'Timestamp': Current\_Date, 'Type': 'Deleted the file'}

#cursor.execute(" insert into filerecords values(?, '0', ?, 'Deleted')".format(file, time))

#mydb.commit()

else:

print("The file does not exist")

elif choice == 3:

print("What records would you like to see?")

print(" 1. Renamed log ")

print(" 2. Deleted log ")

print(" 3. Created log ")

print(" 4. Full log ")

ask = input(" Please enter your choice 1, 2, 3, or 4: ")

if ask == '1':

print(" You requested to see the Renamed Files log. ")

cursor.execute(" select \* from filerecords where Type = 'Renamed' ")

print(" Requested records have been printed. ")

elif ask == '2':

print(" You requested to see the Deleted Files log. ")

cursor.execute(" select \* from filerecords where Type = 'Deleted' ")

print(" Requested records have been printed. ")

elif ask == '3':

print(" You requested to see the Created Files log. All the dummyfiles have been listed. ")

cursor.execute(" select \* from filerecords where Type = 'Created' ")

elif ask == '4':

print(" You requested to see the full log. ")

cursor.execute(" select \* from filerecords ")

else:

print(" Invalid key ")

else:

print(" Entered choice is not listed in menu. Please try again. ")

menu()

def menu():

print (" M E N U ")

print (" 1. Rename your file ")

print (" 2. Delete a file ")

print (" 3. See file records ")

#############Sample file creation for demo#############

## Delete the triple strings to create dummy file, else ignore ##

def dummyfile():

global time

file = open("test.txt", 'w')

text = input("Enter text here")

a = file.write(text)

file.close()

print(" Dummy file has been created. ")

Current\_Date = datetime.datetime.today().strftime ('%d-%b-%Y')

#cursor.execute(" insert into filerecords values('0', 'test.txt', {Current\_Date}, 'Created')".format(Current\_Date))

#mydb.commit()

#######################################################

def file():

askfile = input(" Do you wish to create a dummy file incase you don't want to modify your existing files? (Y/N) ")

if askfile.lower() == 'y':

dummyfile()

elif askfile.lower() == 'n' :

file = input(" Enter file name with extension, like .txt, .csv, .pdf etc. ")

return file

else:

pass

'''

def store():

count = 1

rec = "insert into filerecords values({count}, {start}, {file}, {time}, {type})"

cursor.execute(rec)

mydb.commit()

'''

def confirm():

a = input(" Press Y to confirm your action. ")

if a.upper() == 'Y':

pass

else:

print(" The program has been killed. Please restart the code. ")

sys.exit()

main()

mydb.close()

# 

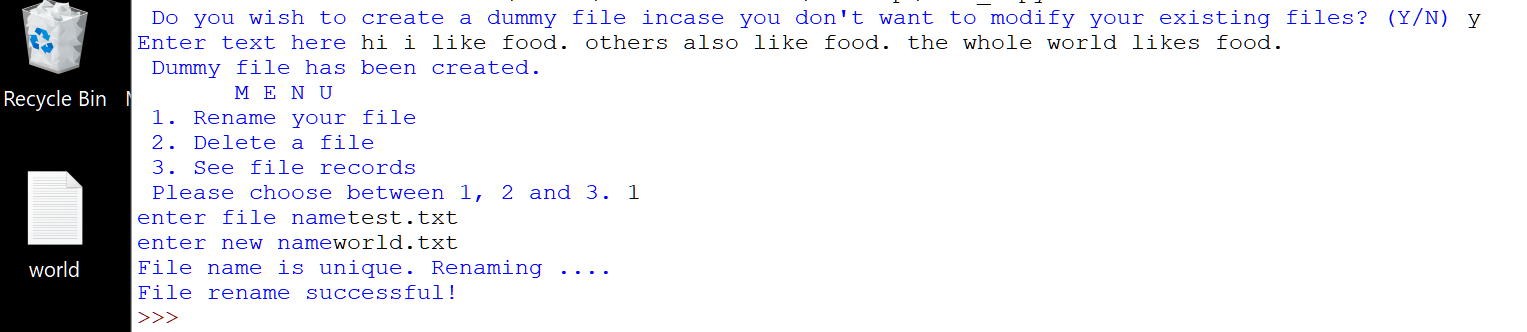
# 

# **Output**

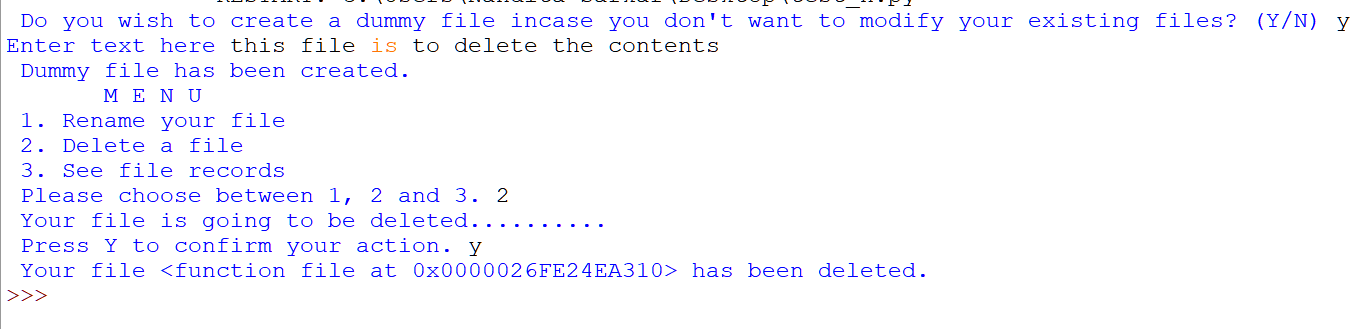
1. Creating a dummy file named ‘test.txt’ and trying to rename it into an existing name: **Python prompts that the file name exists.**

# 

1. Creating a dummy file named ‘test.txt’ and trying to rename it into a unique name: **File is renamed.**

****

1. Deleting a file: **File is deleted and shows file ID.**



1. Attempting to delete a file that doesn’t exist: **Python prompts that the file doesn’t exist.**

