/// by rOm1mPL
/// rOm1mPL.herokuapp.com

Коментарі до файлів

1) **design.py** – створення і настройка користувацького інтерфейсу (UserInterface), туди входять: блок для вставки шляху до папки для перейменування та дві кнопки: шифрування назв папок та розширфування відповідно(вони викликаються із python.py коли натснуто кнопку).
2) **python.py** – оголошення двох функцій відповідно для виконання певних дій коли натискается конкретна кнопка – запис в локальну базу даних python та/або кодування/розкодування папок.
3) **main.py** – основний файл, в ньому викликається фукнція для запуск у інтерфейсу.

Коментарі до коду

1) **design.py**:

Імпорт необхідних для роботи програми бібліотек(всі вони є у файлі requirements.txt)

```
from PyQt5 import QtCore, QtGui, QtWidgets
from python import rename_folders,
set_folders_name, resize_image
from PIL import ImageGrab
import sys
```

Авто-масштабування вікна інтерфейсу та картинки заднього фону під поточний монітор

```
# get current screen width and height
screen = ImageGrab.grab()
screen_w, screen_h = screen.size
if screen_w > 1920:
screen_w = 1920
if screen_h > 1080:
screen_h = 1080
if screen_w < 800:
screen_w = 800
```

```
if screen_h < 600:
screen_h = 600
window_w, window_h = screen_w // 2 + 170, screen_h
// 2 + 100
# resize background image
resize_image(window_w, window_h)
```

Встановлення довжин та широт для 3 блоків: шлях до папок та дві кнопки - в залежності від довжин вікна інтерфейсу

```
# set LinePath width and height relative to our
screen
LinePath_w, LinePath_h = (window_w * 80) // 100,
window_h // 10
# set RenameFolders width and height relative to
our screen
RenameFolders_w, RenameFolders_h = (
window_w * 50) // 100, (window_h * 15) // 100
# set SetFoldersName width and height relative to
our screen
SetFoldersName_w, SetFoldersName_h = (
window_w * 50) // 100, (window_h * 15) // 100
```

Встановлення координат для 3 блоків: шлях до папок та дві кнопки - в залежності від довжин вікна інтерфейсу

```
# set LinePath x and y(the upper left coordinate
of rectangle) relative to our screen
LinePath_x, LinePath_y = (window_w -
LinePath_w) // 2, (window_h * 15) // 100
# set RenameFolders x and y(the upper left
coordinate of rectangle) relative to our screen
RenameFolders_x, RenameFolders_y = (
window_w - RenameFolders_w) // 2, (window_h *
35) // 100
# set SetFoldersName x and y(the upper left
coordinate of rectangle) relative to our screen
SetFoldersName_x, SetFoldersName_y = (
```

```
window_w - SetFoldersName_w) // 2, (window_h * 55)
// 100
```

Створення класу для основного вікна програми, в якому оголошуються дві функції: для настройки 3 блоків(згаданих раніше) та для встановлення їхнього тексту.

```
# create class for MainWindow
class Ui_MainWindow(object):
def setupUi(self, MainWindow):
# main window setting
# set object name
MainWindow.setObjectName("MainWindow")
# set rectangle with and height
MainWindow.resize(window_w, window_h)
MainWindow.setStyleSheet("""
background: white;
""")
self.centralwidget = QtWidgets.QWidget(MainWindow)
# set styles
self.centralwidget.setStyleSheet("""
background: url("data/background.jpg") no-repeat
center;
""")
# set obj name
self.centralwidget.setObjectName("centralwidget")

# Line for path setting
self.PathLine =
QtWidgets.QLineEdit(self.centralwidget)
# set rectangle start, end x and y
self.PathLine.setGeometry(QtCore.QRect(
LinePath_x, LinePath_y, LinePath_w, LinePath_h))
# set styles
self.PathLine.setStyleSheet("""
font: Times New Roman;
color: white;
font-size: 26px;
```

```
border: 2px solid rgb(0, 153, 255);
""")
# set obj name
self.PathLine.setObjectName("PathLine")

# Button for rename folders
self.RenameFolders = QtWidgets.QPushButton(
self.centralwidget)
# set rectangle start, end x and y
self.RenameFolders.setGeometry(
QtCore.QRect(RenameFolders_x, RenameFolders_y,
RenameFolders_w, RenameFolders_h))
# set styles
self.RenameFolders.setStyleSheet("""
font: bold Times New Roman;
color: white;
font-size: 32px;
border: 2px solid rgb(255, 51, 133);
""")
# set obj name
self.RenameFolders.setObjectName("RenameFolders")
MainWindow.setCentralWidget(self.centralwidget)

# Button for return folder's names
self.SetFoldersName = QtWidgets.QPushButton(
self.centralwidget)
# set rectangle start, end x and y
self.SetFoldersName.setGeometry(
QtCore.QRect(SetFoldersName_x, SetFoldersName_y,
SetFoldersName_w, SetFoldersName_h))
# set styles
self.SetFoldersName.setStyleSheet("""
font: bold Times New Roman;
color: white;
font-size: 32px;
border: 2px solid rgb(255, 51, 133);
""")
# set obj name
```

```
self.SetFoldersName.setObjectName("SetFoldersName"
)
MainWindow.setCentralWidget(self.centralwidget)

self.retranslateUi(MainWindow)
QtCore.QMetaObject.connectSlotsByName(MainWindow)

# set buttons text and main window title

def retranslateUi(self, MainWindow):
_translate = QtCore.QCoreApplication.translate
MainWindow.setWindowTitle(_translate("RenameFolder
s", "RenameFolders"))
self.RenameFolders.setText(_translate("MainWindow"
, "Rename Folders"))
self.SetFoldersName.setText(
_translate("MainWindow", "Set Folders Name"))
```

Створення основного класу для програми в якому прописані дії, які слід виконувати коли кнопка натиснута, це все прописується в функціях.(в нашому випадку виклик відповідної функції, які ми імпортували на початку, з файлу python.py)

```
class MyApp(QtWidgets.QMainWindow, Ui_MainWindow):
def __init__(self):
super().__init__()
# set window icon
self.setWindowIcon(QtGui.QIcon('data/icon.ico'))

# set up settings
self.setupUi(self)
# if clicked run func 'rename_folders'
self.RenameFolders.clicked.connect(self.rename_fol
ders)
# if clicked run func 'set_folders_name'
self.SetFoldersName.clicked.connect(self.set_folde
rs_name)
```

```
def rename_folders(self, file_path):
# turn off button 'RenameFolders'
self.RenameFolders.setEnabled(False)
# del spaces in our path line
if self.PathLine.text().strip() != '':
# call our func to rename folders from other
script
if rename_folders(file_path=self.PathLine.text()):
# turn on button
self.RenameFolders.setEnabled(True)
else:
self.RenameFolders.setEnabled(True)

def set_folders_name(self, file_path):
# turn off button 'RenameFolders'
self.RenameFolders.setEnabled(False)
# del spaces in our path line
if self.PathLine.text().strip() != '':
# call our func to rename set back names from
other script
if
set_folders_name(file_path=self.PathLine.text()):
# turn on button
self.RenameFolders.setEnabled(True)
else:
self.RenameFolders.setEnabled(True)
```

Запуск всього інтерфейсу з усіма настройками.

```
# start main app
def start_app():
app = QtWidgets.QApplication(sys.argv)
window = MyApp()
window.show()
app.exec_()
```

2) **python.py**:

Імпорт необхідних для роботи програми бібліотек(всі вони є у файлі requirements.txt)

```
import os
import sqlite3
from PIL import Image
from random import randrange
```

Функція для кодування всіх папок по шляху, який передається з design.py – з блоку шляху, підключення до бази данних(якщо не існує то створюється), створення таблиці dependencies та обхід по всім папкам в циклі, перейменування їх та запис змін до БД.

```
# rename folders for button event
def rename_folders(file_path):
try:
# connect to database(if a database doesn't exist
- it will create it)
connection = sqlite3.connect('database.db')
# from connection get object cursor
cursor = connection.cursor()
# Database - DB
# create table in DB
cursor.execute(
"CREATE table if NOT EXISTS
dependencies(FolderName TEXT, FolderKey TEXT);")
# commit changes
connection.commit()

# get folder's names from path
list_of_folders_name = [item for item in
os.listdir(
file_path) if
os.path.isdir(os.path.join(file_path, item))]
list_of_renamed_folders = []
```

```
# rename every folder in loop
for folder_name in list_of_folders_name:
# choose unique random name for folder
random_name = str(randrange(123456, 987655))
while random_name in list_of_folders_name and
random_name in list_of_renamed_folders:
random_name = str(randrange(123456, 987655))
# take all folder's names from DB
cursor.execute("SELECT * FROM dependencies")
folder_names = [item[1] for item in
cursor.fetchall()]
connection.commit()
try:
if folder_name in folder_names:
# update folder name if it's in DB
cursor.execute(
f"UPDATE dependencies SET FolderKey =
{random_name} WHERE FolderKey = {folder_name};")
connection.commit()
else:
# add new folder name if it's not in DB
cursor.execute(
f"INSERT INTO dependencies (FolderName, FolderKey)
VALUES ('{folder_name}', '{random_name}');")
connection.commit()
except:
# add new folder name if it's not in DB
cursor.execute(
f"INSERT INTO dependencies (FolderName, FolderKey)
VALUES ('{folder_name}', '{random_name}');")
connection.commit()
# rename folder's names from our path
os.rename(f"{file_path}/{folder_name}",
f"{file_path}/{random_name}")
list_of_renamed_folders.append(random_name)
return True
except Exception as error:
```

```
print(error)
finally:
if connection:
# after all those operations close connection to
DB
connection.close()
```

Функція для розкодування всіх назв папок по шляху, який передається з design.py – з блоку шляху, підключення до бази данних і з тих данних, які є в БД зміна імен папок в циклі. Після всіх операцій видалення БД.

```
# return folder's names on path from DB
def set_folders_name(file_path):
try:
# Database - DB
# connect to DB
connection = sqlite3.connect('database.db')
# from connection get object cursor
cursor = connection.cursor()
# get folder's names from path
list_of_folders_name = [item for item in
os.listdir(
file_path) if
os.path.isdir(os.path.join(file_path, item))]
# turn back names from DB
for folder_name in list_of_folders_name:
# take name from BD where name = now folder's name
cursor.execute(
f"SELECT FolderName FROM dependencies WHERE
FolderKey = {folder_name};")
new_name = cursor.fetchall()[0][0]
# rename back to first folder's name
os.rename(f"{file_path}/{folder_name}",
f"{file_path}/{new_name}")
# commit all changes
connection.commit()
try:
```

```
# remove DB - no need
os.remove('database.db')
except:
pass
return True
except Exception as error:
print(error)
finally:
# after all those operations close connection to
DB
if connection:
connection.close()
```

Функція для зміни розміру картинки заднього фону, під поточний монітор

```
# func for resize image for current screen width
and height
def resize_image(width, height):
img = Image.open(r'data/background.jpg')
new_img = img.resize((width, height),
Image.ANTIALIAS)
new_img.save("data/background.jpg", "JPEG")
```

3) **main.py**:

Імпорт UI функції

```
from design import start_app
import os
```

Виклик основної функції з design.py

```
def main():
# start main app
start_app()
# if a database exists - remove it
try:
os.remove('database.db')
except:
pass
```

```
if __name__ == '__main__':
main()
```