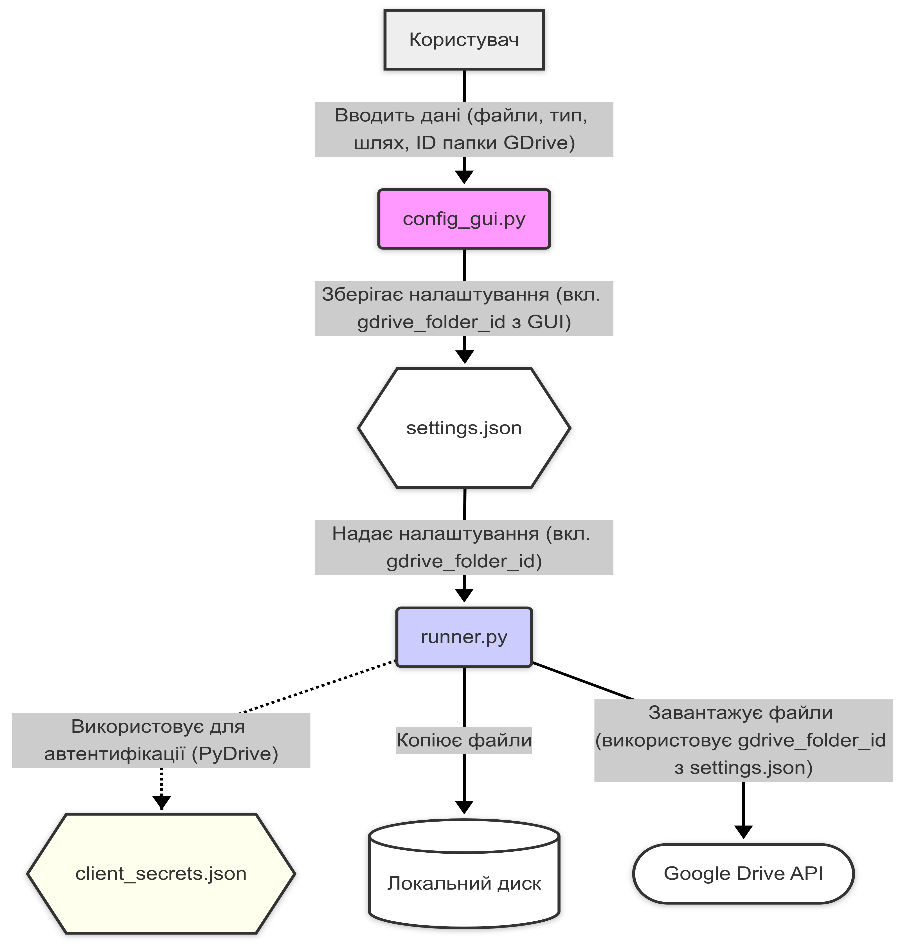
**РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ РЕЗЕРВНОГО КОПІЮВАННЯ ДАНИХ**

**3.1 Загальна архітектура розробленої системи**

Розроблена програмна система призначена для автоматизації процесу резервного копіювання файлів користувача. Вона складається з двох основних модулів: графічного інтерфейсу користувача для налаштування параметрів резервного копіювання (config\_gui.py) та виконавчого модуля, що безпосередньо здійснює процес копіювання (runner.py). Конфігураційні дані, включаючи обрані файли, тип сховища, шлях до локальної папки та ідентифікатор папки Google Drive (який тепер вводиться користувачем через GUI), зберігаються у файлі settings.json. Для автентифікації у Google Drive використовується файл client\_secrets.json, який обробляється бібліотекою PyDrive.

**Основні функціональні можливості системи:**

* Вибір файлів для резервного копіювання.
* Вибір місця збереження резервних копій: локальний диск або хмарне сховище Google Drive.
* Введення ідентифікатора цільової папки на Google Drive через графічний інтерфейс.
* Збереження та завантаження налаштувань користувача.
* Виконання резервного копіювання згідно зі збереженими налаштуваннями.

**

*Діаграма 1* **Загальна архітектура системи резервного копіювання**

**Користувач** взаємодіє з config\_gui.py для налаштування параметрів, включаючи введення ID папки Google Drive.

Файл config\_gui.py зберігає всі обрані налаштування, включаючи ID папки, у файл settings.json.

Файл runner.py читає налаштування, включаючи ID папки, з settings.json для виконання операції резервного копіювання.

Для автентифікації з Google Drive API, runner.py (через бібліотеку PyDrive) неявно використовує файл client\_secrets.json.

Резервні копії зберігаються або на **Локальний диск**, або завантажуються на **Google Drive** у папку, ID якої вказав користувач.

**3.2 Модуль конфігурації (config\_gui.py)**

Модуль config\_gui.py надає користувачеві графічний інтерфейс для налаштування параметрів резервного копіювання. Ключовою зміною в цій версії є додавання можливості для користувача вказувати ідентифікатор (ID) цільової папки на Google Drive безпосередньо в інтерфейсі програми. Цей модуль використовує бібліотеку tkinter.

**Файл: config\_gui.py (версія з полем введення ID папки)**

Python

# Файл: config\_gui.py

# GUI-програма для створення та збереження налаштувань резервного копіювання

# з полем для введення ID папки Google Drive

import tkinter as tk

from tkinter import filedialog, messagebox

import json

import os

SETTINGS\_FILE = "settings.json"

class ConfigApp:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("Налаштування резервного копіювання")

self.root.geometry("500x400") # Трохи збільшив висоту для нового поля

self.files = []

self.destination = ""

self.backup\_to\_drive = tk.BooleanVar()

self.gdrive\_folder\_id\_var = tk.StringVar() # Змінна для зберігання ID папки з поля вводу

# Завантаження попередніх налаштувань, якщо вони є

self.load\_initial\_settings()

tk.Label(root, text="Оберіть файли для резервного копіювання:").pack(pady=5)

tk.Button(root, text="Обрати файли", command=self.select\_files).pack()

tk.Label(root, text="Куди зберігати резервну копію?").pack(pady=10)

# Прив'язуємо команду для відображення/приховування поля ID

tk.Radiobutton(root, text="На диск", variable=self.backup\_to\_drive, value=False, command=self.toggle\_gdrive\_id\_input).pack()

tk.Radiobutton(root, text="У хмару (Google Drive)", variable=self.backup\_to\_drive, value=True, command=self.toggle\_gdrive\_id\_input).pack()

# Створюємо рамку для поля вводу ID папки, щоб керувати її видимістю

self.gdrive\_id\_frame = tk.Frame(root)

# Не пакуємо її одразу, видимість буде контролюватися toggle\_gdrive\_id\_input

tk.Label(self.gdrive\_id\_frame, text="ID папки Google Drive:").pack(side=tk.LEFT, padx=5)

self.gdrive\_id\_entry = tk.Entry(self.gdrive\_id\_frame, textvariable=self.gdrive\_folder\_id\_var, width=40)

self.gdrive\_id\_entry.pack(side=tk.LEFT)

# Розміщуємо рамку після радіокнопок

self.gdrive\_id\_frame.pack(pady=5)

tk.Button(root, text="Обрати папку призначення (для локального диска)", command=self.select\_destination).pack(pady=5)

tk.Button(root, text="Зберегти налаштування", command=self.save\_settings, bg="green", fg="white").pack(pady=15)

self.status = tk.Label(root, text="")

self.status.pack()

# Початкове налаштування видимості поля ID

self.toggle\_gdrive\_id\_input()

def load\_initial\_settings(self):

"""Завантажує ID папки Google Drive з settings.json при запуску, якщо файл існує."""

if os.path.exists(SETTINGS\_FILE):

try:

with open(SETTINGS\_FILE, "r", encoding="utf-8") as f:

settings = json.load(f)

# Встановлюємо значення для StringVar, яке автоматично оновить поле Entry

self.gdrive\_folder\_id\_var.set(settings.get("gdrive\_folder\_id", ""))

# Встановлюємо значення для backup\_to\_drive, щоб радіокнопка була коректною

self.backup\_to\_drive.set(settings.get("backup\_to\_drive", False))

# Зберігаємо інші налаштування, якщо потрібно їх використовувати при старті

self.files = settings.get("files", [])

self.destination = settings.get("destination", "")

except json.JSONDecodeError:

print(f"Попередження: Не вдалося розпарсити {SETTINGS\_FILE}. Будуть використані значення за замовчуванням.")

except Exception as e:

print(f"Попередження: Помилка при завантаженні початкових налаштувань: {e}")

def toggle\_gdrive\_id\_input(self):

"""Показує або приховує поле для введення ID папки Google Drive."""

if self.backup\_to\_drive.get(): # Якщо обрано Google Drive

self.gdrive\_id\_frame.pack(pady=5) # Показуємо рамку з полем вводу

else:

self.gdrive\_id\_frame.pack\_forget() # Приховуємо рамку

def select\_files(self):

self.files = list(filedialog.askopenfilenames(title="Оберіть файли"))

if self.files:

self.status.config(text=f"Обрано {len(self.files)} файлів.")

else:

self.status.config(text="Файли не обрано.")

def select\_destination(self):

self.destination = filedialog.askdirectory(title="Оберіть папку")

if self.destination:

self.status.config(text=f"Папка призначення: {self.destination}")

else:

self.status.config(text="Папку призначення не обрано.")

def save\_settings(self):

if not self.files:

messagebox.showerror("Помилка", "Не вибрано файлів.")

return

gdrive\_id = self.gdrive\_folder\_id\_var.get().strip() # Отримуємо ID з поля вводу та видаляємо зайві пробіли

if self.backup\_to\_drive.get(): # Якщо обрано Google Drive

if not gdrive\_id:

messagebox.showerror("Помилка", "Вкажіть ID папки Google Drive.")

return

elif not self.destination: # Якщо обрано локальний диск, але не вказано папку

messagebox.showerror("Помилка", "Вкажіть папку призначення для локального збереження.")

return

settings = {

"files": self.files,

"backup\_to\_drive": self.backup\_to\_drive.get(),

"destination": self.destination if not self.backup\_to\_drive.get() else "", # Зберігаємо папку призначення тільки для локального бекапу

"gdrive\_folder\_id": gdrive\_id if self.backup\_to\_drive.get() else "" # Зберігаємо ID тільки для Google Drive

}

try:

with open(SETTINGS\_FILE, "w", encoding="utf-8") as f:

json.dump(settings, f, indent=4)

self.status.config(text="Налаштування збережено.")

messagebox.showinfo("Готово", "Налаштування збережено!")

except Exception as e:

messagebox.showerror("Помилка збереження", f"Не вдалося зберегти налаштування: {e}")

if \_\_name\_\_ == "\_\_main\_\_":

root = tk.Tk()

app = ConfigApp(root)

root.mainloop()

**3.2.1 Імпорт необхідних бібліотек та глобальні налаштування**

Імпорт бібліотек (tkinter, filedialog, messagebox, json, os) залишається таким же, як і в попередній версії. SETTINGS\_FILE = "settings.json": визначає ім'я файлу для збереження налаштувань.

**3.2.2 Клас ConfigApp**

Основна логіка графічного інтерфейсу.

**3.2.2.1 Конструктор \_\_init\_\_(self, root)**

Конструктор ініціалізує головне вікно та його компоненти.

* self.root.geometry("500x400"): Розмір вікна збільшено для розміщення нового поля вводу.
* self.gdrive\_folder\_id\_var = tk.StringVar(): Створюється спеціальна змінна tkinter типу StringVar. Ця змінна буде пов'язана з полем введення для ID папки Google Drive, дозволяючи легко отримувати та встановлювати його значення.
* self.load\_initial\_settings(): Викликається новий метод для завантаження раніше збережених налаштувань (включаючи ID папки) при запуску програми.

Далі створюються та розміщуються елементи інтерфейсу:

* Елементи для вибору файлів та папки призначення залишаються без змін.
* Радіокнопки для вибору типу сховища ("На диск" або "У хмару (Google Drive)") тепер мають параметр command=self.toggle\_gdrive\_id\_input. Це означає, що при зміні вибору радіокнопки буде викликатися метод toggle\_gdrive\_id\_input.
* **Поле для введення ID папки Google Drive:**

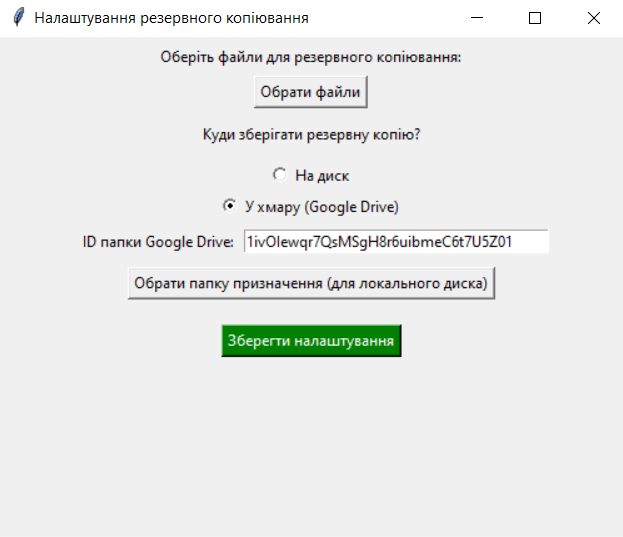
self.gdrive\_id\_frame = tk.Frame(root): Створюється об'єкт tk.Frame. Рамка використовується як контейнер для мітки та поля введення ID папки, що дозволяє керувати їх видимістю як єдиним цілим.

tk.Label(self.gdrive\_id\_frame, text="ID папки Google Drive:").pack(side=tk.LEFT, padx=5): Текстова мітка "ID папки Google Drive:", розміщена всередині рамки.

self.gdrive\_id\_entry = tk.Entry(self.gdrive\_id\_frame, textvariable=self.gdrive\_folder\_id\_var, width=40): Створюється поле введення (tk.Entry). Параметр textvariable пов'язує це поле зі змінною self.gdrive\_folder\_id\_var. Будь-які зміни в полі введення автоматично оновлюватимуть значення self.gdrive\_folder\_id\_var, і навпаки. width=40 встановлює ширину поля.

self.gdrive\_id\_frame.pack(pady=5): Рамка з міткою та полем вводу розміщується у головному вікні.

self.toggle\_gdrive\_id\_input(): Викликається наприкінці конструктора для встановлення початкової видимості поля введення ID папки залежно від стану радіокнопок (який міг бути завантажений з settings.json).

**

Зображення 2 Графічний інтерфейс модуля конфігурації

**3.2.2.2 Метод load\_initial\_settings(self)**

Цей новий метод відповідає за завантаження налаштувань з файлу settings.json при запуску програми.

Перевіряє, чи існує файл SETTINGS\_FILE.

Якщо файл існує, він читається, і дані JSON десеріалізуються.

self.gdrive\_folder\_id\_var.set(settings.get("gdrive\_folder\_id", "")): Значення для ID папки Google Drive (якщо воно є у файлі налаштувань) встановлюється у змінну self.gdrive\_folder\_id\_var. Це автоматично відобразить ID у відповідному полі введення. Якщо ключ gdrive\_folder\_id відсутній, встановлюється порожній рядок.

self.backup\_to\_drive.set(settings.get("backup\_to\_drive", False)): Встановлюється попередньо обраний стан радіокнопки.

Також завантажуються списки файлів (self.files) та шлях до локальної папки (self.destination).

Обробляються можливі помилки при читанні або парсингу файлу.

**3.2.2.3 Метод toggle\_gdrive\_id\_input(self)**

Цей метод керує видимістю рамки (self.gdrive\_id\_frame), що містить поле для введення ID папки Google Drive.

if self.backup\_to\_drive.get():: Якщо обрано опцію "У хмару (Google Drive)" (тобто self.backup\_to\_drive має значення True):

self.gdrive\_id\_frame.pack(pady=5): Рамка (разом з міткою та полем введення) відображається у вікні.

else:: Якщо обрано опцію "На диск":

self.gdrive\_id\_frame.pack\_forget(): Рамка приховується.

**3.2.2.4 Методи select\_files(self) та select\_destination(self)**

Ці методи залишаються функціонально такими ж, як і в попередній версії, відповідаючи за вибір файлів та папки призначення відповідно. Додано оновлення статусної мітки у випадку, якщо файли або папка не були обрані.

**3.2.2.5 Метод save\_settings(self)**

gdrive\_id = self.gdrive\_folder\_id\_var.get().strip(): Отримується значення з поля введення ID папки через змінну self.gdrive\_folder\_id\_var. Метод .get() повертає поточний текст з поля, а .strip() видаляє можливі зайві пробіли на початку та в кінці рядка.

* **Перевірка даних:**

Перевірка наявності обраних файлів (if not self.files:) залишається.

if self.backup\_to\_drive.get():: Якщо обрано збереження на Google Drive:

if not gdrive\_id:: Перевіряється, чи введено ID папки. Якщо поле порожнє, виводиться повідомлення про помилку.

elif not self.destination:: Якщо обрано локальне збереження (тобто self.backup\_to\_drive.get() повернуло False), перевіряється, чи вказано папку призначення.

* **Формування словника налаштувань:**

Python

settings = {

"files": self.files,

"backup\_to\_drive": self.backup\_to\_drive.get(),

"destination": self.destination if not self.backup\_to\_drive.get() else "",

"gdrive\_folder\_id": gdrive\_id if self.backup\_to\_drive.get() else ""

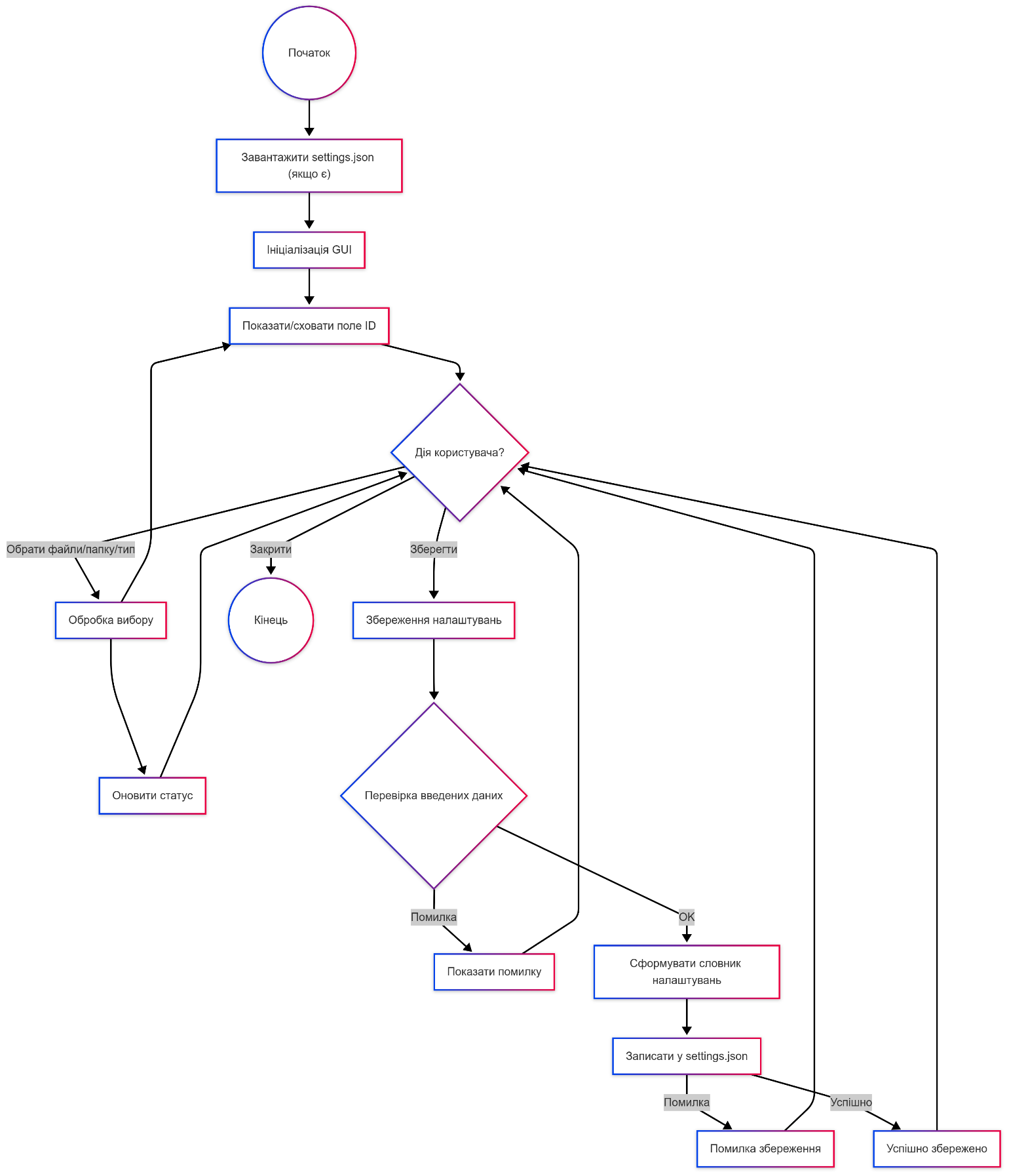
}

Словник settings тепер містить:

* + "gdrive\_folder\_id": значення, введене користувачем у поле self.gdrive\_id\_entry (якщо обрано Google Drive), або порожній рядок (якщо обрано локальний диск).
  + "destination": зберігається лише якщо обрано локальний бекап, інакше – порожній рядок.
* Збереження у файл JSON та повідомлення користувачу залишаються аналогічними, але додано блок try-except для обробки можливих помилок під час збереження файлу.

**3.2.3 Запуск додатку**

Блок if \_\_name\_\_ == "\_\_main\_\_": залишається без змін, він створює екземпляр ConfigApp та запускає головний цикл tkinter.

**

**Блок – схема 3-1 Логіка роботи модуля config\_gui.py**

**3.3 Модуль виконання резервного копіювання (runner.py)**

Модуль runner.py є виконавчим компонентом системи резервного копіювання. Його основне завдання – зчитування конфігураційних параметрів, збережених користувачем через графічний інтерфейс (config\_gui.py) у файлі settings.json, та здійснення процесу копіювання файлів відповідно до цих налаштувань. Модуль підтримує два режими резервного копіювання: на локальний диск та у хмарне сховище Google Drive.

**Файл: runner.py**

import json

import os

import shutil

from pydrive.auth import GoogleAuth # Потрібно встановити: pip install PyDrive

from pydrive.drive import GoogleDrive

SETTINGS\_FILE = "settings.json"

# BACKUP\_FOLDER\_ID = "1ivOIewqr7QsMSgH8r6uibmeC6t7U5Z01" # Більше не потрібен тут, будемо брати з settings.json

def load\_settings():

if not os.path.exists(SETTINGS\_FILE):

print(f"Файл налаштувань '{SETTINGS\_FILE}' не знайдено.")

return None

try:

with open(SETTINGS\_FILE, "r", encoding="utf-8") as f:

settings = json.load(f)

return settings

except json.JSONDecodeError:

print(f"Помилка: Не вдалося розпарсити JSON у файлі '{SETTINGS\_FILE}'.")

return None

except Exception as e:

print(f"Помилка при читанні файлу налаштувань '{SETTINGS\_FILE}': {e}")

return None

def upload\_to\_drive(files, gdrive\_folder\_id): # Функція тепер приймає ID папки

if not gdrive\_folder\_id:

print("Помилка: ID папки Google Drive не вказано в налаштуваннях.")

return

print(f"Спроба завантаження у папку Google Drive ID: {gdrive\_folder\_id}")

try:

gauth = GoogleAuth()

# Спроба завантажити збережені облікові дані, якщо вони є

# gauth.LoadCredentialsFile("mycreds.txt") # Розкоментуйте, якщо хочете зберігати/завантажувати облікові дані

# if gauth.credentials is None:

# Автентифікація, якщо облікових даних немає

gauth.LocalWebserverAuth() # Відкриє браузер для автентифікації

# elif gauth.access\_token\_expired:

# Оновлення, якщо термін дії токена минув

# gauth.Refresh()

# else:

# Авторизація, якщо облікові дані завантажено та вони дійсні

# gauth.Authorize()

# Збереження облікових даних для наступних запусків (опціонально)

# gauth.SaveCredentialsFile("mycreds.txt") # Розкоментуйте, якщо хочете зберігати/завантажувати облікові дані

drive = GoogleDrive(gauth)

successful\_uploads = 0

for file\_path in files:

if not os.path.isfile(file\_path):

print(f"Попередження: Файл '{os.path.basename(file\_path)}' не знайдено за шляхом '{file\_path}'. Пропускається.")

continue

file\_name = os.path.basename(file\_path)

try:

print(f"Завантаження файлу: {file\_name}...")

# Створення файлу на Google Drive у вказаній папці

file\_drive = drive.CreateFile({'title': file\_name, 'parents': [{'id': gdrive\_folder\_id, "kind": "drive#fileLink"}]})

file\_drive.SetContentFile(file\_path) # Встановлення вмісту файлу

file\_drive.Upload() # Завантаження файлу

print(f"Успішно завантажено в Google Drive: {file\_name}")

successful\_uploads += 1

except Exception as e\_file: # Обробка помилок для окремого файлу

print(f"Помилка під час завантаження файлу '{file\_name}' в Google Drive: {e\_file}")

if successful\_uploads > 0:

print(f"Успішно завантажено {successful\_uploads} файлів.")

else:

print("Жодного файлу не було завантажено.")

except FileNotFoundError: # Ця помилка може виникнути, якщо client\_secrets.json не знайдено

print(f"Помилка: Файл client\_secrets.json не знайдено. Будь ласка, переконайтеся, що він знаходиться у тій самій директорії, що і скрипт, або налаштуйте шляхи у GoogleAuth.")

except Exception as e:

print(f"Загальна помилка під час процесу завантаження в Google Drive: {e}")

def copy\_locally(files, destination):

if not destination:

print("Помилка: Локальна папка призначення не вказана.")

return

if not os.path.exists(destination):

try:

os.makedirs(destination)

print(f"Створено папку призначення: {destination}")

except Exception as e:

print(f"Помилка створення папки призначення '{destination}': {e}")

return

elif not os.path.isdir(destination):

print(f"Помилка: Вказаний шлях призначення '{destination}' не є папкою.")

return

successful\_copies = 0

try:

for file\_path in files:

if not os.path.isfile(file\_path):

print(f"Попередження: Файл '{os.path.basename(file\_path)}' не знайдено за шляхом '{file\_path}'. Пропускається.")

continue

file\_name = os.path.basename(file\_path)

dest\_path = os.path.join(destination, file\_name)

try:

shutil.copy2(file\_path, dest\_path)

print(f"Скопійовано на диск: {file\_name} -> {dest\_path}")

successful\_copies += 1

except Exception as e\_file: # Обробка помилок для окремого файлу

print(f"Помилка копіювання файлу '{file\_name}' на диск: {e\_file}")

if successful\_copies > 0:

print(f"Успішно скопійовано {successful\_copies} файлів.")

else:

print("Жодного файлу не було скопійовано.")

except Exception as e:

print(f"Загальна помилка копіювання на диск: {e}")

def main():

settings = load\_settings()

if not settings:

print("Завершення роботи через помилку завантаження налаштувань.")

return

files = settings.get("files", [])

use\_drive = settings.get("backup\_to\_drive", False)

destination = settings.get("destination", "")

gdrive\_folder\_id = settings.get("gdrive\_folder\_id", "") # Отримуємо ID папки з налаштувань

if not files:

print("Немає файлів для копіювання у налаштуваннях.")

return

if use\_drive:

if not gdrive\_folder\_id:

print("Помилка: Резервне копіювання на Google Drive обрано, але ID папки не вказано у файлі налаштувань.")

print("Будь ласка, відкрийте GUI, вкажіть ID папки Google Drive та збережіть налаштування.")

return

upload\_to\_drive(files, gdrive\_folder\_id) # Передаємо ID у функцію

else:

if not destination:

print("Не вказано папку призначення для локального збереження у налаштуваннях.")

return

copy\_locally(files, destination)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**3.3.1 Імпорт бібліотек та глобальні налаштування**

На початку модуля runner.py здійснюється імпорт необхідних бібліотек:

**json:** Стандартна бібліотека Python для роботи з даними у форматі JSON. Використовується для читання файлу налаштувань settings.json.

**os:** Стандартна бібліотека для взаємодії з операційною системою. Надає функції для роботи з шляхами файлів (наприклад, os.path.exists, os.path.isfile, os.path.basename, os.path.join, os.makedirs).

**shutil:** Стандартна бібліотека, що містить функції високого рівня для операцій з файлами. У даному проекті використовується shutil.copy2 для копіювання файлів на локальний диск зі збереженням метаданих.

**pydrive.auth.GoogleAuth:** Компонент бібліотеки PyDrive, відповідальний за процес автентифікації користувача для доступу до Google Drive API.

**pydrive.drive.GoogleDrive:** Компонент бібліотеки PyDrive, що надає інтерфейс для виконання операцій з файлами та папками на Google Drive (створення, завантаження тощо).

## 3.3.2 Функція load\_settings()

Функція load\_settings() призначена для завантаження конфігураційних даних з файлу settings.json.

def load\_settings():

if not os.path.exists(SETTINGS\_FILE):

print(f"Файл налаштувань '{SETTINGS\_FILE}' не знайдено.")

return None

try:

with open(SETTINGS\_FILE, "r", encoding="utf-8") as f:

settings = json.load(f)

return settings

except json.JSONDecodeError:

print(f"Помилка: Не вдалося розпарсити JSON у файлі '{SETTINGS\_FILE}'.")

return None

except Exception as e:

print(f"Помилка при читанні файлу налаштувань '{SETTINGS\_FILE}': {e}")

return None

**Логіка роботи функції:**

1. **Перевірка існування файлу:** За допомогою os.path.exists(SETTINGS\_FILE) перевіряється, чи існує файл налаштувань. Якщо файл не знайдено, виводиться відповідне повідомлення в консоль, і функція повертає None.
2. **Читання та десеріалізація:** Якщо файл існує, він відкривається для читання ("r") з кодуванням utf-8 (для коректної обробки можливих нелатинських символів у шляхах файлів або інших даних). Вміст файлу десеріалізується з формату JSON у словник Python за допомогою json.load(f).
3. **Повернення налаштувань:** У разі успішного читання та десеріалізації, функція повертає словник з налаштуваннями.
4. **Обробка помилок:** Функція включає обробку двох типів винятків:

json.JSONDecodeError: Виникає, якщо файл settings.json має некоректну структуру JSON і не може бути ‘розпарсений’. У цьому випадку виводиться повідомлення про помилку, і функція повертає None.

Exception: Загальний блок для перехоплення будь-яких інших можливих помилок під час читання файлу (наприклад, проблеми з правами доступу). Також виводиться повідомлення про помилку, і функція повертає None.

Ця функція забезпечує надійне завантаження конфігурації, необхідної для подальшої роботи модуля.

## 3.3.3 Функція upload\_to\_drive(files, gdrive\_folder\_id)

Функція upload\_to\_drive реалізує логіку завантаження обраних файлів на хмарне сховище Google Drive. Вона приймає два аргументи: files (список шляхів до файлів, які потрібно завантажити) та gdrive\_folder\_id (ідентифікатор цільової папки на Google Drive, куди будуть завантажені файли).

def upload\_to\_drive(files, gdrive\_folder\_id):

if not gdrive\_folder\_id:

print("Помилка: ID папки Google Drive не вказано в налаштуваннях.")

return

print(f"Спроба завантаження у папку Google Drive ID: {gdrive\_folder\_id}")

try:

gauth = GoogleAuth()

gauth.LocalWebserverAuth()

drive = GoogleDrive(gauth)

successful\_uploads = 0

for file\_path in files:

**Логіка роботи функції:**

**Перевірка ID папки:** На початку функції перевіряється, чи було передано gdrive\_folder\_id. Якщо ID папки відсутній (є порожнім рядком або None), виводиться повідомлення про помилку, і функція завершує роботу, оскільки без ID неможливо визначити місце завантаження на Google Drive.

**Автентифікація:**

gauth = GoogleAuth(): Створюється екземпляр класу GoogleAuth з бібліотеки PyDrive. Цей об'єкт відповідає за процес автентифікації.

gauth.LocalWebserverAuth(): Ініціюється процес автентифікації OAuth 2.0. PyDrive **неявно** використовує файл client\_secrets.json (який має знаходитися в тій самій директорії, що й скрипт для отримання облікових даних клієнта (client ID, client secret). Цей метод відкриває веб-браузер, де користувач повинен увійти до свого облікового запису Google та надати додатку дозвіл на доступ до Google Drive.

1. **Ініціалізація GoogleDrive:**

drive = GoogleDrive(gauth): Створюється екземпляр класу GoogleDrive, використовуючи автентифікований об'єкт gauth. Цей об'єкт drive надає методи для взаємодії з файлами на Google Drive.

1. **Ітерація по файлах та завантаження:**

successful\_uploads = 0: Ініціалізується лічильник успішно завантажених файлів.

Цикл for file\_path in files: перебирає кожен шлях до файлу зі списку files.

if not os.path.isfile(file\_path):: Перевіряється, чи є вказаний шлях файлом і чи він існує. Якщо ні, виводиться попередження, і файл пропускається.

file\_name = os.path.basename(file\_path): Отримується ім'я файлу з повного шляху.

file\_drive = drive.CreateFile({'title': file\_name, 'parents': [{'id': gdrive\_folder\_id, "kind": "drive#fileLink"}]}): Створюється об'єкт файлу на Google Drive. У словнику метаданих вказується:

'title': file\_name: Ім'я файлу, яке він матиме на Google Drive.

'parents': [{'id': gdrive\_folder\_id, "kind": "drive#fileLink"}]: Вказує батьківську папку для файлу. Використовується gdrive\_folder\_id, переданий у функцію. "kind": "drive#fileLink" є рекомендованим атрибутом для вказання батьківського зв'язку.

file\_drive.SetContentFile(file\_path): Встановлюється вміст файлу, який буде завантажено, шляхом вказання шляху до локального файлу.

file\_drive.Upload(): Виконується безпосереднє завантаження файлу на Google Drive.

Після успішного завантаження виводиться повідомлення, і лічильник successful\_uploads збільшується.

Внутрішній блок try...except Exception as e\_file: обробляє можливі помилки під час завантаження окремого файлу, виводячи повідомлення про помилку для конкретного файлу, але дозволяючи продовжити завантаження наступних файлів.

**Підсумок завантаження:** Після завершення циклу у консолі виводиться інформація про кількість успішно завантажених файлів.

**Обробка глобальних помилок:**

Зовнішній блок try...except обробляє більш загальні помилки, які можуть виникнути під час процесу автентифікації або ініціалізації GoogleDrive.

**FileNotFoundError:** Окремо обробляється помилка, якщо файл client\_secrets.json не знайдено, оскільки він є критично важливим для автентифікації.

**Exception as e:** Перехоплює будь-які інші непередбачені помилки під час процесу завантаження.

Ця функція забезпечує повний цикл завантаження файлів, включаючи автентифікацію, створення файлів у вказаній папці на Google Drive та обробку можливих помилок.

**3.3.4 Функція copy\_locally(files, destination)**

Функція copy\_locally відповідає за копіювання обраних файлів до вказаної папки на локальному диску користувача. Вона приймає два аргументи: files (список шляхів до файлів для копіювання) та destination (шлях до папки призначення).

def copy\_locally(files, destination):

if not destination:

print("Помилка: Локальна папка призначення не вказана.")

return

if not os.path.exists(destination):

try:

os.makedirs(destination)

print(f"Створено папку призначення: {destination}")

except Exception as e:

print(f"Помилка створення папки призначення '{destination}': {e}")

return

elif not os.path.isdir(destination):

print(f"Помилка: Вказаний шлях призначення '{destination}' не є папкою.")

return

successful\_copies = 0

try:

for file\_path in files:

if not os.path.isfile(file\_path):

print(f"Попередження: Файл '{os.path.basename(file\_path)}' не знайдено за шляхом '{file\_path}'. Пропускається.")

continue

file\_name = os.path.basename(file\_path)

dest\_path = os.path.join(destination, file\_name)

try:

shutil.copy2(file\_path, dest\_path)

print(f"Скопійовано на диск: {file\_name} -> {dest\_path}")

successful\_copies += 1

except Exception as e\_file:

print(f"Помилка копіювання файлу '{file\_name}' на диск: {e\_file}")

if successful\_copies > 0:

print(f"Успішно скопійовано {successful\_copies} файлів.")

else:

print("Жодного файлу не було скопійовано.")

except Exception as e:

print(f"Загальна помилка копіювання на диск: {e}")

**Логіка роботи функції:**

1. **Перевірка папки призначення:**

if not destination:: На початку функції перевіряється, чи було взагалі передано шлях до папки призначення. Якщо destination є порожнім рядком або None, виводиться повідомлення про помилку, і функція завершує свою роботу, оскільки копіювання без вказання місця призначення неможливе.

if not os.path.exists(destination):: Якщо шлях до папки призначення вказано, але сама папка не існує у файловій системі, програма намагається її створити. Для цього використовується функція os.makedirs(destination). os.makedirs має перевагу над os.mkdir, оскільки вона може створювати всі проміжні директорії у вказаному шляху, якщо вони також відсутні. Якщо під час створення папки виникає будь-яка помилка (наприклад, через недостатність прав доступу), ця помилка перехоплюється блоком except Exception as e, виводиться відповідне повідомлення, і функція завершує роботу.

elif not os.path.isdir(destination):: Якщо вказаний шлях destination існує, але не є директорією (наприклад, це шлях до файлу), копіювання в нього неможливе. У цьому випадку виводиться повідомлення про помилку, і функція також завершує роботу.

1. **Ітерація по файлах та копіювання:**

successful\_copies = 0: Ініціалізується змінна successful\_copies нулем. Ця змінна буде використовуватися як лічильник для підрахунку кількості файлів, які були успішно скопійовані.

Далі починається цикл for file\_path in files:, який послідовно перебирає кожен шлях до файлу, переданий у списку files.

if not os.path.isfile(file\_path):: Перед спробою копіювання для кожного файлу виконується перевірка, чи є вказаний file\_path дійсно файлом і чи він існує у файловій системі. Це важлива перевірка, щоб уникнути помилок, якщо у списку файлів опиниться шлях до директорії або до неіснуючого файлу. Якщо перевірка не проходить, виводиться попереджувальне повідомлення, і програма переходить до наступного файлу у списку за допомогою оператора continue.

file\_name = os.path.basename(file\_path): З повного шляху до файлу (file\_path) видобувається лише ім'я файлу разом з його розширенням. Це робиться за допомогою функції os.path.basename().

dest\_path = os.path.join(destination, file\_name): Формується повний шлях до місця, куди буде скопійовано файл. Це робиться шляхом об'єднання шляху до папки призначення (destination) та імені файлу (file\_name) за допомогою функції os.path.join(). Ця функція коректно обробляє роздільники шляхів для різних операційних систем.

shutil.copy2(file\_path, dest\_path): Виконується безпосереднє копіювання файлу. Для цього використовується функція shutil.copy2(). Перевага copy2 над shutil.copy полягає в тому, що copy2 намагається зберегти якомога більше метаданих оригінального файлу, включаючи час останньої модифікації та права доступу (наскільки це можливо для операційної системи).

Після успішного виконання операції копіювання (shutil.copy2) виводиться інформаційне повідомлення в консоль, яке вказує, який файл було скопійовано і куди. Лічильник successful\_copies збільшується на одиницю.

Внутрішній блок try...except Exception as e\_file: призначений для обробки можливих помилок, що можуть виникнути під час копіювання *окремого* файлу (наприклад, диск переповнений, немає прав на запис у папку призначення для конкретного файлу тощо). Якщо така помилка виникає, виводиться повідомлення про помилку для цього конкретного файлу, але програма не зупиняється, а продовжує спроби копіювання наступних файлів зі списку.

1. **Підсумок копіювання:**

Після завершення циклу обробки всіх файлів зі списку files перевіряється значення лічильника successful\_copies.

Якщо successful\_copies > 0, виводиться повідомлення про загальну кількість успішно скопійованих файлів.

Якщо successful\_copies залишився нулем (тобто жоден файл не вдалося скопіювати), виводиться відповідне повідомлення.

1. **Обробка глобальних помилок:**

Зовнішній блок try...except Exception as e: слугує для перехоплення будь-яких інших непередбачених помилок, що могли виникнути під час виконання функції copy\_locally загалом (наприклад, помилки, не пов'язані з копіюванням окремого файлу).

Таким чином, функція copy\_locally забезпечує надійний механізм копіювання файлів на локальний диск, включаючи створення папки призначення, перевірку файлів та обробку різноманітних помилок, що можуть виникнути в процесі.

## 3.3.5 Функція main()