Backup Utility

📁 IMPORTS AND DEPENDENCIES

```
import os
import sys
import shutil
import datetime
import threading
import tkinter as tk
from tkinter import filedialog, messagebox, ttk
import pyzipper
import ttkbootstrap as ttkb
from ttkbootstrap.constants import *
from google.oauth2.credentials import Credentials
from google_auth_oauthlib.flow import InstalledAppFlow
from googleapiclient.discovery import build
from googleapiclient.http import MediaFileUpload
```

- `os, sys, shutil`: Used for file system and OS-level operations.
- `datetime`: For generating timestamped filenames.
- `threading`: Allows backup to run without freezing the GUI.
- `tkinter`: For GUI elements.
- `filedialog`, `messagebox`, `ttk`: For dialog boxes and widgets.
- `pyzipper`: For encrypted ZIP file creation.
- `ttkbootstrap`: Stylish version of Tkinter widgets with theming.
- `google.*`: Used to handle OAuth and communicate with Google Drive.

```
try:
    from tkinterdnd2 import DND_FILES, TkinterDnD
except ImportError:
    messagebox.showerror("Missing Package", "Install 'tkinterdnd2' via pip:\npip install tkinterdnd2")
    exit()
```

- Tries to import drag-and-drop functionality (`tkinterdnd2`). If missing, shows error and exits.

🔐 GOOGLE DRIVE CONSTANTS

```
SCOPES = ['https://www.googleapis.com/auth/drive.file']
CREDENTIALS_FILE = 'credentials.json'
```

```
TOKEN_FILE = 'token.json'
```

- Defines the Google Drive API permissions scope and paths to the credentials/token files.

🛠️ RESOURCE HANDLER FOR .EXE COMPATIBILITY

```
def resource_path(relative_path):
    if hasattr(sys, '_MEIPASS'):
        return os.path.join(sys._MEIPASS, relative_path)
    return os.path.abspath(relative_path)
```

- Ensures icons/resources are found even when bundled with PyInstaller (`_MEIPASS` is a special temp dir used by PyInstaller).

🧠 MAIN APPLICATION CLASS

```
class BackupApp:
    def __init__(self, root):
        ...
```

- `__init__` : Constructor method. Sets up the app state, theme, and calls `setup_ui()` to build the interface.

```
self.theme = "darkly"
self.upload_to_drive_var = tk.BooleanVar(value=False)
```

- Initializes the default theme and checkbox state for Google Drive upload.

🎨 UI SETUP

```
def setup_ui(self):
```

- Sets the window title, icon, size, and adds all widgets (labels, entries, buttons, progress bar).

```
self.root.iconbitmap(resource_path("cloud.ico"))
```

- Sets app icon using the resource handler.

```
self.source_entry.drop_target_register(DND_FILES)
self.source_entry.dnd_bind('<<Drop>>', self.handle_drop_source)
```

- Enables drag-and-drop for the source folder entry.

🌓 THEME TOGGLER

```
def toggle_theme(self):
    self.theme = "flatly" if self.theme == "darkly" else "darkly"
    self.style.theme_use(self.theme)
```

- Toggles between light (`flatly`) and dark (`darkly`) themes.

📁 BROWSING AND DRAG-DROP

```
def handle_drop_source(self, event):
```

- Handles drag-and-drop file path and sets it in the source entry.

```
def browse_source(self):
```

- Opens folder browser and sets the source path.

```
def browse_destination(self):
```

- Same as above but for destination.

🧵 START BACKUP IN THREAD

```
def start_backup(self):
    threading.Thread(target=self.backup_files).start()
```

- Starts the backup in a separate thread to prevent UI freezing.

🗜 ZIP AND BACKUP LOGIC

```
def backup_files(self):
    ...
```

- Gets all form input: source, destination, file types, password.

```
include_types = self.file_types_entry.get().split(',')
```

- Converts comma-separated extensions into a list.

```
timestamp = datetime.datetime.now().strftime("%Y%m%d%H%M%S")
zip_path = os.path.join(dest, f"backup_{timestamp}.zip")
```

- Creates a timestamped ZIP filename.

```
for root, _, files in os.walk(source):
    for file in files:
        if not include_types or any(file.endswith(ext.strip()) for ext in
include_types):
            ...
```

- Recursively walks through files and filters by extension.

```
with pyzipper.AESZipFile(zip_path, 'w', compression=pyzipper.ZIP_DEFLATED) as zf:
    if password:
        zf.setpassword(password.encode())
        zf.setencryption(pyzipper.WZ_AES, nbits=256)
```

- Creates encrypted ZIP if password is given.

```
self.progress.config(maximum=len(files_to_backup))
...
self.progress.step()
self.root.update_idletasks()
```

- Updates the progress bar as files are added to the ZIP.

```
if self.upload_to_drive_var.get():
    self.upload_to_drive(zip_path)
```

- Uploads to Google Drive if the checkbox is selected.

☁ UPLOAD TO GOOGLE DRIVE

```python
def upload_to_drive(self, filepath):
```

- Loads saved Google token or initiates OAuth flow.

```python
creds = Credentials.from_authorized_user_file(TOKEN_FILE, SCOPES)
...
service = build('drive', 'v3', credentials=creds)
file_metadata = {'name': os.path.basename(filepath)}
media = MediaFileUpload(filepath, resumable=True)
service.files().create(...).execute()
```

- Uploads the file using the Drive v3 API.

🚀 STARTING THE APPLICATION

```python
if __name__ == '__main__':
    root = TkinterDnD.Tk()
    ttkb.Style("darkly")
    app = BackupApp(root)
    root.mainloop()
```

- Entry point of the program.
- Initializes the drag-and-drop enabled root window, sets default theme, and launches the app.

Future Updates:

- Logging to `.txt`
- Restore functionality
- Tray icon/minimize-to-tray
- Notification system
- Export/import settings