**HƯỚNG DẪN TẠO FILE EXE HOÀN TOÀN ĐỘC LẬP**

Dưới đây là hướng dẫn để tạo một file .exe hoàn toàn độc lập:

## Hướng dẫn tạo file .exe hoàn toàn độc lập

# Hướng dẫn tạo file .exe hoàn toàn độc lập

## Bước 1: Cài đặt các công cụ cần thiết

```powershell

# Cài đặt PyInstaller và các module cần thiết

pip install pyinstaller

pip install requests

pip install pillow

```

## Bước 2: Chuẩn bị file spec đặc biệt

# Tạo file với tên standalone\_launcher.spec với nội dung sau:

"""

# -\*- mode: python ; coding: utf-8 -\*-

block\_cipher = None

a = Analysis(

['fixed\_launcher.py'], # Tên file launcher của bạn

pathex=[],

binaries=[],

datas=[],

hiddenimports=['PIL', 'PIL.\_imagingtk', 'PIL.\_tkinter\_finder', 'tkinter', 'tkinter.filedialog'],

hookspath=[],

hooksconfig={},

runtime\_hooks=[],

excludes=[],

win\_no\_prefer\_redirects=False,

win\_private\_assemblies=False,

cipher=block\_cipher,

noarchive=False,

)

pyz = PYZ(a.pure, a.zipped\_data, cipher=block\_cipher)

exe = EXE(

pyz,

a.scripts,

a.binaries,

a.zipfiles,

a.datas,

[],

name='standalone\_calculator',

debug=False,

bootloader\_ignore\_signals=False,

strip=False,

upx=True,

upx\_exclude=[],

runtime\_tmpdir=None,

console=False, # Đặt True nếu muốn hiển thị console để debug

disable\_windowed\_traceback=False,

argv\_emulation=False,

target\_arch=None,

codesign\_identity=None,

entitlements\_file=None,

icon='calculator.ico', # Thêm đường dẫn tới file icon nếu có

)

"""

## Bước 3: Biên dịch với file .spec để tạo file .exe hoàn toàn độc lập

```powershell

# Chạy PyInstaller với file spec

pyinstaller standalone\_launcher.spec

```

## Bước 4: Cập nhật file launcher để hoạt động tốt trong môi trường đóng gói

# Đảm bảo rằng file launcher của bạn có mã nguồn như sau để tránh lỗi khi đóng gói:

```python

import os

import sys

import json

import tkinter as tk

from tkinter import messagebox

import subprocess

import time

import tempfile

try:

import requests

except ImportError:

# Trong trường hợp chạy từ exe đã đóng gói, module requests sẽ không được import trực tiếp

pass # PyInstaller sẽ xử lý việc này

# Hàm xác định đường dẫn cho thư mục ứng dụng

def get\_app\_folder():

if getattr(sys, 'frozen', False):

# Đang chạy từ file .exe tạo bởi PyInstaller

base\_dir = getattr(sys, '\_MEIPASS', os.path.dirname(sys.executable))

# Sử dụng thư mục gần file .exe để lưu trữ dữ liệu

data\_folder = os.path.join(os.path.dirname(sys.executable), "CalculatorData")

else:

# Đang chạy từ script Python

base\_dir = os.path.dirname(os.path.abspath(\_\_file\_\_))

data\_folder = os.path.join(os.path.expanduser("~"), "SimpleCalculator")

return data\_folder

# Tiếp tục các phần còn lại của file launcher như bình thường

# ...

# Khi tải máy tính từ GitHub, đảm bảo xử lý khi không có kết nối internet

def download\_calculator(url, destination):

try:

# Kiểm tra xem đang chạy từ exe đã đóng gói

if getattr(sys, 'frozen', False):

# Nếu là file exe đóng gói, sử dụng bản copy đã nhúng trong file exe

bundled\_calculator = os.path.join(getattr(sys, '\_MEIPASS', ''), "calculator.py")

if os.path.exists(bundled\_calculator):

with open(bundled\_calculator, 'r', encoding='utf-8') as src:

with open(destination, 'w', encoding='utf-8') as dest:

dest.write(src.read())

return True

# Nếu không có phiên bản nhúng hoặc không chạy từ exe, thử tải từ internet

import requests

# Tạo cửa sổ loading...

# ...

return True

except Exception as e:

# Trong trường hợp lỗi, sử dụng phiên bản mặc định

default\_calculator = """

import tkinter as tk

from tkinter import messagebox

class SimpleCalculator:

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

self.root = root

self.root.title("Máy Tính Đơn Giản")

self.root.geometry("300x400")

self.root.resizable(False, False)

self.root.configure(bg="#f0f0f0")

# Biến lưu giá trị hiển thị

self.current\_value = ""

# Tạo khung hiển thị kết quả

self.display\_frame = tk.Frame(root, bg="#f0f0f0")

self.display\_frame.pack(pady=10)

# Tạo ô hiển thị kết quả

self.display = tk.Entry(self.display\_frame, font=("Arial", 24), width=15,

bd=5, justify=tk.RIGHT)

self.display.pack()

# Tạo khung chứa các nút

self.buttons\_frame = tk.Frame(root, bg="#f0f0f0")

self.buttons\_frame.pack()

# Danh sách các nút

self.button\_list = [

"7", "8", "9", "/",

"4", "5", "6", "\*",

"1", "2", "3", "-",

"0", ".", "=", "+"

]

# Tạo nút Clear

self.clear\_button = tk.Button(self.buttons\_frame, text="C", font=("Arial", 18),

width=4, height=1, command=self.clear\_display)

self.clear\_button.grid(row=0, column=0, columnspan=4, padx=5, pady=5, sticky="we")

# Tạo các nút số và phép tính

self.create\_buttons()

def create\_buttons(self):

row\_val = 1

col\_val = 0

for button\_text in self.button\_list:

button = tk.Button(self.buttons\_frame, text=button\_text, font=("Arial", 18),

width=4, height=1, command=lambda x=button\_text: self.button\_click(x))

button.grid(row=row\_val, column=col\_val, padx=5, pady=5)

col\_val += 1

if col\_val > 3:

col\_val = 0

row\_val += 1

def button\_click(self, value):

if value == "=":

try:

# Tính toán kết quả

result = eval(self.current\_value)

self.current\_value = str(result)

self.display.delete(0, tk.END)

self.display.insert(0, self.current\_value)

except Exception as e:

messagebox.showerror("Lỗi", "Biểu thức không hợp lệ")

self.current\_value = ""

self.display.delete(0, tk.END)

else:

# Thêm giá trị vào biến hiện tại

self.current\_value += value

self.display.delete(0, tk.END)

self.display.insert(0, self.current\_value)

def clear\_display(self):

# Xóa hết giá trị

self.current\_value = ""

self.display.delete(0, tk.END)

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

root = tk.Tk()

app = SimpleCalculator(root)

root.mainloop()

"""

# Lưu mã nguồn mặc định vào file đích

with open(destination, 'w', encoding='utf-8') as f:

f.write(default\_calculator)

# Hiển thị thông báo cảnh báo nhưng vẫn tiếp tục

try:

show\_message(f"Không thể tải từ internet, sử dụng phiên bản mặc định.\nLỗi: {str(e)}", "Cảnh báo", True)

except:

pass

return True

```

## Bước 5: Thêm file calculator.py vào bộ cài đặt

# Cập nhật file spec để thêm calculator.py vào bundle:

"""

# Thêm dòng này vào phần datas trong file .spec:

datas=[('path\_to\_calculator.py', '.')]

"""

## Lưu ý quan trọng:

# 1. Đặt tên file launcher của bạn đúng trong file .spec

# 2. Thay 'path\_to\_calculator.py' bằng đường dẫn thực tế đến file máy tính

# 3. Thêm file icon nếu có, hoặc bỏ dòng đó đi

## Launcher hoàn toàn độc lập

import os

import sys

import json

import tkinter as tk

from tkinter import messagebox

import time

import tempfile

# Hàm xác định đường dẫn cho thư mục ứng dụng

def get\_app\_folder():

"""Trả về đường dẫn thư mục ứng dụng, hoạt động cả khi đóng gói với PyInstaller"""

if getattr(sys, 'frozen', False):

# Đang chạy từ file .exe tạo bởi PyInstaller

base\_dir = os.path.dirname(sys.executable)

# Lưu dữ liệu trong thư mục cạnh file .exe

data\_folder = os.path.join(base\_dir, "CalculatorData")

else:

# Đang chạy từ script Python

base\_dir = os.path.dirname(os.path.abspath(\_\_file\_\_))

data\_folder = os.path.join(os.path.expanduser("~"), "SimpleCalculator")

return data\_folder

# Đường dẫn cố định cho thư mục cài đặt

APP\_FOLDER = get\_app\_folder()

CONFIG\_FILE = os.path.join(APP\_FOLDER, "config.json")

DEFAULT\_CALCULATOR\_FILE = os.path.join(APP\_FOLDER, "calculator.py")

# Mã nguồn máy tính mặc định (sẽ được sử dụng nếu không tải được từ internet)

DEFAULT\_CALCULATOR\_CODE = """

import tkinter as tk

from tkinter import messagebox

class SimpleCalculator:

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

self.root = root

self.root.title("Máy Tính Đơn Giản")

self.root.geometry("300x400")

self.root.resizable(False, False)

self.root.configure(bg="#f0f0f0")

# Biến lưu giá trị hiển thị

self.current\_value = ""

# Tạo khung hiển thị kết quả

self.display\_frame = tk.Frame(root, bg="#f0f0f0")

self.display\_frame.pack(pady=10)

# Tạo ô hiển thị kết quả

self.display = tk.Entry(self.display\_frame, font=("Arial", 24), width=15,

bd=5, justify=tk.RIGHT)

self.display.pack()

# Tạo khung chứa các nút

self.buttons\_frame = tk.Frame(root, bg="#f0f0f0")

self.buttons\_frame.pack()

# Danh sách các nút

self.button\_list = [

"7", "8", "9", "/",

"4", "5", "6", "\*",

"1", "2", "3", "-",

"0", ".", "=", "+"

]

# Tạo nút Clear

self.clear\_button = tk.Button(self.buttons\_frame, text="C", font=("Arial", 18),

width=4, height=1, command=self.clear\_display)

self.clear\_button.grid(row=0, column=0, columnspan=4, padx=5, pady=5, sticky="we")

# Tạo các nút số và phép tính

self.create\_buttons()

def create\_buttons(self):

row\_val = 1

col\_val = 0

for button\_text in self.button\_list:

button = tk.Button(self.buttons\_frame, text=button\_text, font=("Arial", 18),

width=4, height=1, command=lambda x=button\_text: self.button\_click(x))

button.grid(row=row\_val, column=col\_val, padx=5, pady=5)

col\_val += 1

if col\_val > 3:

col\_val = 0

row\_val += 1

def button\_click(self, value):

if value == "=":

try:

# Tính toán kết quả

result = eval(self.current\_value)

self.current\_value = str(result)

self.display.delete(0, tk.END)

self.display.insert(0, self.current\_value)

except Exception as e:

messagebox.showerror("Lỗi", "Biểu thức không hợp lệ")

self.current\_value = ""

self.display.delete(0, tk.END)

else:

# Thêm giá trị vào biến hiện tại

self.current\_value += value

self.display.delete(0, tk.END)

self.display.insert(0, self.current\_value)

def clear\_display(self):

# Xóa hết giá trị

self.current\_value = ""

self.display.delete(0, tk.END)

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

root = tk.Tk()

app = SimpleCalculator(root)

root.mainloop()

"""

# File khóa

LOCK\_FILE = os.path.join(tempfile.gettempdir(), "calculator\_launcher.lock")

def is\_already\_running():

"""Kiểm tra xem chương trình đã đang chạy chưa"""

try:

# Nếu file khóa tồn tại và chứa nội dung

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

# Kiểm tra xem tiến trình đó còn sống không

with open(LOCK\_FILE, 'r') as f:

pid = f.read().strip()

if pid and pid.isdigit():

# Kiểm tra PID theo cách phù hợp với hệ điều hành

try:

if sys.platform == "win32":

# Windows

import ctypes

kernel32 = ctypes.windll.kernel32

handle = kernel32.OpenProcess(1, 0, int(pid))

if handle:

kernel32.CloseHandle(handle)

return True

else:

# Unix/Linux/Mac

os.kill(int(pid), 0) # Không gửi tín hiệu, chỉ kiểm tra

return True

except (OSError, ImportError):

# PID không tồn tại hoặc không thể import

pass

return False

except:

return False

def create\_lock\_file():

"""Tạo file khóa để đánh dấu chương trình đang chạy"""

try:

with open(LOCK\_FILE, 'w') as f:

f.write(str(os.getpid()))

except:

pass

def show\_message(message, title="Thông báo", error=False):

"""Hiển thị thông báo"""

root = tk.Tk()

root.withdraw()

if error:

messagebox.showerror(title, message)

else:

messagebox.showinfo(title, message)

root.destroy()

def clean\_up():

"""Xử lý khi ứng dụng thoát"""

# Xóa file khóa khi thoát

try:

os.remove(LOCK\_FILE)

except:

pass

def ensure\_app\_folder():

"""Đảm bảo thư mục ứng dụng tồn tại"""

if not os.path.exists(APP\_FOLDER):

os.makedirs(APP\_FOLDER)

def load\_config():

"""Tải cấu hình từ file"""

# Cấu hình mặc định

default\_config = {

"calculator\_path": DEFAULT\_CALCULATOR\_FILE,

"auto\_update": False,

"last\_updated": None

}

# Nếu file cấu hình không tồn tại, tạo file mới

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

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

json.dump(default\_config, f, indent=4)

return default\_config

# Tải cấu hình từ file

try:

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

config = json.load(f)

# Đảm bảo khóa calculator\_path luôn tồn tại

if "calculator\_path" not in config:

config["calculator\_path"] = DEFAULT\_CALCULATOR\_FILE

return config

except Exception as e:

# Nếu có lỗi, ghi log và trả về cấu hình mặc định

print(f"Lỗi khi đọc cấu hình: {str(e)}")

return default\_config

def save\_config(config):

"""Lưu cấu hình vào file"""

try:

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

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

except Exception as e:

print(f"Lỗi khi lưu cấu hình: {str(e)}")

def try\_download\_calculator(url, destination):

"""Thử tải máy tính từ URL"""

try:

# Kiểm tra xem đang chạy từ file đóng gói hay không

if getattr(sys, 'frozen', False):

# Nếu đang chạy từ file .exe, kiểm tra xem có file máy tính nhúng không

calculator\_resource = os.path.join(getattr(sys, '\_MEIPASS', ''), "calculator.py")

if os.path.exists(calculator\_resource):

# Nếu có file máy tính nhúng, sao chép vào thư mục đích

with open(calculator\_resource, 'r', encoding='utf-8') as f:

calculator\_code = f.read()

with open(destination, 'w', encoding='utf-8') as f:

f.write(calculator\_code)

return True

# Nếu không có file máy tính nhúng hoặc không chạy từ file .exe, thử tải từ internet

import requests

# Tạo cửa sổ loading

root = tk.Tk()

root.title("Đang tải...")

root.geometry("300x80")

root.resizable(False, False)

label = tk.Label(root, text="Đang tải máy tính từ GitHub...", font=("Arial", 10))

label.pack(pady=10)

progress = tk.Label(root, text="Vui lòng chờ...", font=("Arial", 8))

progress.pack(pady=5)

root.update()

# Tải file từ GitHub

response = requests.get(url, timeout=10)

# Kiểm tra trạng thái phản hồi

if response.status\_code != 200:

root.destroy()

raise Exception(f"Không thể tải file. Mã lỗi: {response.status\_code}")

# Lưu nội dung vào file

with open(destination, 'w', encoding='utf-8') as f:

f.write(response.text)

# Cập nhật thông báo và đóng cửa sổ

progress.config(text="Đã tải xong!")

root.after(1000, root.destroy)

root.mainloop()

return True

except Exception as e:

try:

if 'root' in locals():

root.destroy()

except:

pass

print(f"Lỗi khi tải máy tính: {str(e)}")

return False

def setup\_calculator():

"""Thiết lập máy tính lần đầu"""

ensure\_app\_folder()

config = load\_config()

# Đảm bảo đường dẫn file máy tính hợp lệ

calculator\_path = config.get("calculator\_path", DEFAULT\_CALCULATOR\_FILE)

# Kiểm tra xem file máy tính đã tồn tại chưa

if not os.path.exists(calculator\_path):

try:

# Thử tải máy tính từ GitHub

from\_github = False

try:

# Import thử module requests

import requests

from\_github = try\_download\_calculator("https://raw.githubusercontent.com/alfienguyenn/SVS/main/TestGUI.py", calculator\_path)

except ImportError:

from\_github = False

if not from\_github:

# Nếu không tải được từ GitHub, sử dụng mã nguồn mặc định

with open(calculator\_path, 'w', encoding='utf-8') as f:

f.write(DEFAULT\_CALCULATOR\_CODE)

except Exception as e:

# Nếu có lỗi, sử dụng mã nguồn mặc định

with open(calculator\_path, 'w', encoding='utf-8') as f:

f.write(DEFAULT\_CALCULATOR\_CODE)

# Cập nhật cấu hình

config["calculator\_path"] = calculator\_path # Đảm bảo đường dẫn được lưu

config["last\_updated"] = time.strftime("%Y-%m-%d %H:%M:%S")

save\_config(config)

return config

def run\_calculator\_directly():

"""Chạy máy tính trực tiếp trong process hiện tại"""

try:

config = setup\_calculator()

calculator\_path = config.get("calculator\_path", DEFAULT\_CALCULATOR\_FILE)

# Kiểm tra xem file máy tính có tồn tại không

if not os.path.exists(calculator\_path):

raise Exception(f"Không tìm thấy file máy tính tại: {calculator\_path}")

# Đọc code từ file

with open(calculator\_path, 'r', encoding='utf-8') as f:

calculator\_code = f.read()

# Thực thi code trực tiếp

exec(calculator\_code, globals())

return True

except Exception as e:

error\_msg = f"Lỗi khi chạy máy tính: {str(e)}"

show\_message(error\_msg, "Lỗi khởi chạy", True)

return False

def main():

"""Hàm chính của launcher"""

# Đăng ký hàm dọn dẹp khi thoát

import atexit

atexit.register(clean\_up)

try:

# Kiểm tra xem đã có instance đang chạy chưa

if is\_already\_running():

show\_message("Ứng dụng máy tính đã đang chạy.", "Thông báo")

sys.exit(0)

# Tạo file khóa

create\_lock\_file()

# Chạy máy tính trực tiếp

run\_calculator\_directly()

except Exception as e:

error\_msg = f"Lỗi không xác định: {str(e)}"

show\_message(error\_msg, "Lỗi", True)

sys.exit(1)

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

main()

## standalone.spec

# -\*- mode: python ; coding: utf-8 -\*-

block\_cipher = None

a = Analysis(

['fully\_integrated.py'], # Tên file Python của launcher tích hợp

pathex=[],

binaries=[],

datas=[],

hiddenimports=[

'PIL',

'PIL.\_imagingtk',

'PIL.\_tkinter\_finder',

'tkinter',

'tkinter.filedialog',

'requests'

],

hookspath=[],

hooksconfig={},

runtime\_hooks=[],

excludes=[],

win\_no\_prefer\_redirects=False,

win\_private\_assemblies=False,

cipher=block\_cipher,

noarchive=False,

)

# Thêm file calculator.py để nhúng vào bên trong EXE

a.datas += [('calculator.py', 'TestGUI.py', 'DATA')]

pyz = PYZ(a.pure, a.zipped\_data, cipher=block\_cipher)

exe = EXE(

pyz,

a.scripts,

a.binaries,

a.zipfiles,

a.datas,

[],

name='calculator\_standalone',

debug=False,

bootloader\_ignore\_signals=False,

strip=False,

upx=True,

upx\_exclude=[],

runtime\_tmpdir=None,

console=False, # Đặt thành True nếu muốn debug với console

disable\_windowed\_traceback=False,

argv\_emulation=False,

target\_arch=None,

codesign\_identity=None,

entitlements\_file=None,

)

## Hướng dẫn đóng gói thành file exe hoàn toàn độc lập

// Hướng dẫn đóng gói ứng dụng máy tính thành file .exe hoàn toàn độc lập

// BƯỚC 1: LƯU CÁC FILE

// Lưu file "fully\_integrated.py" (launcher tích hợp) và "standalone.spec" vào cùng thư mục với file TestGUI.py của bạn

// BƯỚC 2: CÀI ĐẶT CÁC CÔNG CỤ CẦN THIẾT

// Mở Command Prompt hoặc PowerShell và nhập các lệnh sau:

pip install pyinstaller

pip install requests

pip install pillow

// BƯỚC 3: BIÊN DỊCH ỨNG DỤNG

// Di chuyển đến thư mục chứa các file và chạy lệnh:

pyinstaller standalone.spec

// BƯỚC 4: KIỂM TRA VÀ ĐÓNG GÓI

// 1. File .exe sẽ được tạo trong thư mục "dist"

// 2. Thử chạy file .exe để kiểm tra xem nó có hoạt động không

// 3. Nếu muốn phân phối, chỉ cần sao chép file .exe

// GIẢI THÍCH VỀ CÁCH HOẠT ĐỘNG:

// 1. Launcher tích hợp sẽ:

// - Chạy máy tính trực tiếp trong cùng một process, không cần Python bên ngoài

// - Nhúng mã nguồn máy tính mặc định để sử dụng khi không có kết nối Internet

// - Có thể tự động cập nhật từ GitHub khi có kết nối mạng (tùy chọn)

// 2. File .spec đặc biệt sẽ:

// - Đảm bảo tất cả thư viện cần thiết được đóng gói

// - Nhúng file TestGUI.py vào bên trong .exe

// - Tắt console để tạo giao diện sạch sẽ

// LƯU Ý QUAN TRỌNG:

// - Không cần cài đặt Python trên máy tính đích

// - Tất cả module đã được tích hợp, bao gồm tkinter, PIL, requests

// - Lần đầu chạy, ứng dụng sẽ tạo thư mục "CalculatorData" bên cạnh file .exe

// - Để thay đổi mã nguồn máy tính, chỉ cần thay thế file trong thư mục "CalculatorData"

// XỬ LÝ SỰ CỐ:

// 1. Nếu gặp lỗi "Missing module", thêm module đó vào hiddenimports trong file .spec

// 2. Nếu muốn hiển thị console để debug, đặt console=True trong file .spec

// 3. Nếu file .exe quá lớn, có thể sử dụng UPX để nén (tùy chọn upx=True đã được bật)

## Decription

Tôi đã tạo một giải pháp hoàn chỉnh để tạo file .exe hoàn toàn độc lập, không cần cài đặt Python trên máy tính đích. Cách này sẽ hiệu quả hơn nhiều so với sử dụng PowerShell để cài đặt Python.

**Cách thức hoạt động:**

1. **Launcher tích hợp**: Tôi đã tạo một file launcher mới (fully\_integrated.py) có khả năng:
   * Chạy mã máy tính trực tiếp trong cùng một tiến trình
   * Nhúng sẵn mã máy tính mặc định (không cần tải từ internet)
   * Vẫn hỗ trợ cập nhật từ GitHub khi có kết nối mạng
2. **File spec đặc biệt**: File standalone.spec sẽ hướng dẫn PyInstaller:
   * Đóng gói mọi thư viện cần thiết (tkinter, PIL, requests)
   * Nhúng file TestGUI.py vào bên trong file .exe
   * Tạo file .exe hoàn toàn độc lập không cần Python

**Các bước thực hiện:**

1. **Lưu các file** vào cùng thư mục với TestGUI.py:
   * fully\_integrated.py (launcher tích hợp)
   * standalone.spec (file cấu hình PyInstaller)
2. **Cài đặt công cụ** (chỉ trên máy phát triển):

pip install pyinstaller requests pillow

1. **Biên dịch ứng dụng**:

pyinstaller standalone.spec

1. **Phân phối file .exe** trong thư mục dist

**Ưu điểm quan trọng:**

1. **Hoàn toàn độc lập**: Không cần Python hay bất kỳ thư viện nào trên máy đích
2. **Dung lượng nhỏ hơn**: So với việc đóng gói Python đầy đủ
3. **Dễ phân phối**: Chỉ cần một file .exe duy nhất
4. **Vẫn có thể cập nhật**: Người dùng có thể thay đổi file calculator.py trong thư mục CalculatorData

Đây là giải pháp tốt nhất cho trường hợp của bạn, giúp tránh các vấn đề khi cài đặt Python thủ công trên máy đích.