Context Menu：右鍵複製所選程式，並轉為markdown

2023-03-19

我想做一個程式，因為每次都要手動把程式碼轉成Obsidian的Markdown格式，真的很麻煩，想要多選後右鍵有個特殊複製選項，點選後，把選擇的所有程式碼轉成markdown格式 例如有個 hellow.js 內容是 console.log("hello") 另外一個程式是hi.js 內容是 console.log("hi") 我框選這兩個程式，並且用我們自製的右鍵的特殊複製選項。 點下去後記到剪貼簿， 解貼簿內容如下:

```JS
console.log("hello")
```

```
console.log("hi")
```

這樣就複製所有資料跟內容了
成果如下。

`D:\DEV_Code\DEV_Python\Context Menu\convert_to_markdown.py`

```
import sys
import os
import mimetypes
import pyperclip
import tkinter.messagebox as messagebox
import json

# 定义扩展名与语言名称的对应表(一律小寫)
extension_mapping = {
    "py": "Python",
    "ts": "Typescript"
}

def convert_to_markdown(file_path):
    try:
        file_name = os.path.basename(file_path)
        file_extension = os.path.splitext(file_name)[1][1:] # 刪除開頭的句點
        lang = ""

        # 使用 mimetypes 库获取文件的 MIME 类型
        mime_type, encoding = mimetypes.guess_type(file_path)
        if mime_type:
            # 根据 MIME 类型获取语言名称
            ext_list = mimetypes.types_map.get(mime_type, "").split(",")
            if ext_list:
                lang = ext_list[0].upper()

        # 自定義附檔名
        lang = extension_mapping.get(file_extension.lower(), "")

        # 若依舊為空則抓現有副檔名
        if not lang:
```

```
            lang = file_extension

        with open(file_path, "r", encoding="utf-8") as file:
            content = file.read()
            markdown_output = f"`{file_path}`\n\n```{lang}:
{file_name}\n{content}\n```\n\n"

        return markdown_output

    except (UnicodeDecodeError, IsADirectoryError, FileNotFoundError):
        # 如果遇到 UnicodeDecodeError、IsADirectoryError 或 FileNotFoundError，彈
出警訊
        messagebox.showerror("Error", "Selected file is not a text file or does
not exist.")
        return ""

if __name__ == "__main__":
    file_path = sys.argv[1]
    markdown_output = convert_to_markdown(file_path)
    if markdown_output:
        pyperclip.copy(markdown_output)
```

為了實現這個功能，我們可以使用 Python 語言編寫一個程式。以下是一個可以達到您需求的 Python 程式範例：

步驟 1: 安裝必要的 Python 庫 在命令提示字元或終端機中執行以下命令，以安裝所需的庫文件：

```
pip install pyperclip
```

步驟 2: 建立一個 Python 檔案 (例如 `convert_to_markdown.py`) 將以下程式碼複製到新建的 Python 檔案中：

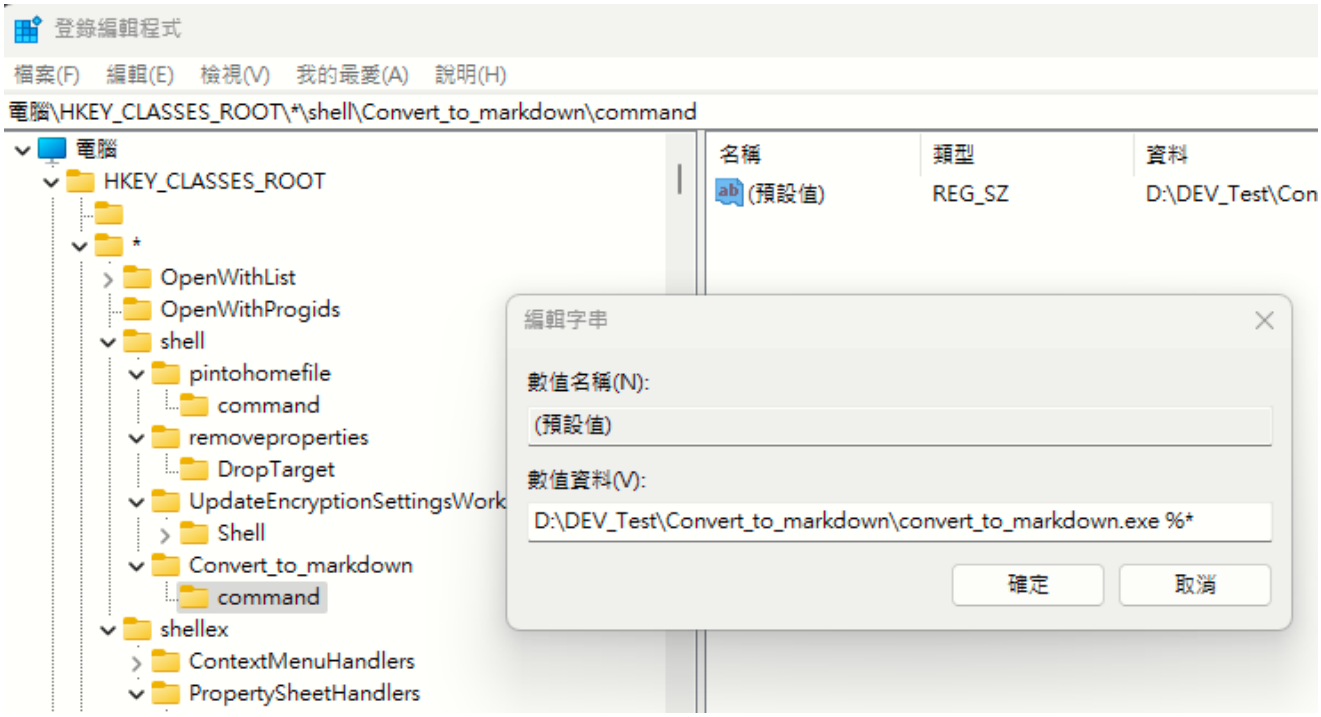
步驟 3:
`--noconsole` 在debug時候不要用

```
pyinstaller --onefile --noconsole convert_to_markdown.py
```

步驟 4: 在 Windows 系統上添加右键選項 為了將這個功能添加到右键選單，您需要編輯 Windows 註冊表。請按照以下步驟操作：

1. 打開 "註冊表編輯器"（按 Win + R 鍵，然後鍵入 `regedit`，按 Enter）。
2. 導航到 `HKEY_CLASSES_ROOT\*\shell`。
3. 在 `shell` 下，右键單擊並選擇 `新建 > 金鑰`，命名為 "轉換為Markdown"。
4. 在 "轉換為Markdown" 下，右鍵單擊並選擇 `新建 > 金鑰`，命名為 "command"。
5. 在右側的 "command"，雙擊 "預設" 值，將 "值數據" 設置為：`路徑\convert_to_markdown.exe %1`

例如
`D:\DEV_Test\Convert_to_markdown\convert_to_markdown.exe %1`



也能用HKEY_CLASSES_ROOT 金鑰
也就是.reg

`D:\DEV_Code\DEV_Python\Context Menu\conver_tomarkdown.reg`

```
Windows Registry Editor Version 5.00

[HKEY_CLASSES_ROOT\*\shell\Convert_to_markdown]
@="Convert to Markdown"

[HKEY_CLASSES_ROOT\*\shell\Convert_to_markdown\command]
@="\"D:\\DEV_Test\\Convert_to_markdown\\convert_to_markdown.exe\" \"%1\" \"%2\"
\"%3\" \"%4\" \"%5\" \"%6\" \"%7\" \"%8\" \"%9\""
```

不過我是用整合安裝包去組

意外花費很多時間的小插曲是 `%*`
理論上能多選後複製，但是我發現無論怎麼改都沒效果，所以只能放棄考量

此外嘗試採用整合成安裝包
Inno Setup

2023-03-20
發現python跑太慢
若走 Py cython 加速與應用C語言 會出現瑕疵
所以乾脆重寫 C++

`D:\DEV_Code\DEV_Cplus\Context Menu\convert_to_markdown\convert_to_markdown\convert_to_markdown.cpp`

```
#include <iostream>
#include <fstream>
#include <string>
#include <map>
#include <Windows.h>
#include <cassert>
#include <vector>

using namespace std;

map<string, string> extension_mapping = {
    { "py", "Python" },
    { "ts", "Typescript" }
};

string UTF8ToUnicode(const char* str)
{
    string result;
    WCHAR* strSrc;
    LPSTR szRes;

    //UT8轉unicode
    int i = MultiByteToWideChar(CP_UTF8, 0, str, -1, NULL, 0);
    strSrc = new WCHAR[i + 1];
    MultiByteToWideChar(CP_UTF8, 0, str, -1, strSrc, i);

    //WChar轉回Char
    i = WideCharToMultiByte(CP_ACP, 0, strSrc, -1, NULL, 0, NULL, NULL);
    szRes = new CHAR[i + 1];
    WideCharToMultiByte(CP_ACP, 0, strSrc, -1, szRes, i, NULL, NULL);

    result = szRes;
    delete[]strSrc;
    delete[]szRes;

    return result;
}

std::string readFile(std::string file)
{
    std::ifstream infile;
    infile.open(file.data());
    assert(infile.is_open());

    string s; string strAllLine;
    while (getline(infile, s)) {
        string line = UTF8ToUnicode(s.c_str()).c_str();
        strAllLine += line+"\n";
    }

    infile.close();
    return strAllLine;
}

string convert_to_markdown(string file_path) {
    try {
        string file_name = file_path.substr(file_path.find_last_of("\\/") + 1);
```

```
        string file_extension = file_name.substr(file_name.find_last_of(".") +
1);
        string lang = "";

        // 獲取文件 MIME 類型
        string mime_type = "";

        ifstream file(file_path);
        if (file) {
            char buffer[256];
            file.read(buffer, 256);
            streamsize bytesRead = file.gcount();
            mime_type = string(buffer, bytesRead);
        }
        file.close();

        // 依據MIM取得語言名稱
        size_t start = mime_type.find(": ");
        if (start != string::npos) {
            start += 2;
            size_t end = mime_type.find("\r\n", start);
            if (end != string::npos) {
                string mime_type_str = mime_type.substr(start, end - start);
                size_t pos = mime_type_str.find("/");
                if (pos != string::npos) {
                    lang = mime_type_str.substr(pos + 1);
                }
            }
        }

        // 自訂附檔名
        auto iter = extension_mapping.find(file_extension);
        if (iter != extension_mapping.end()) {
            lang = iter->second;
        }

        // 如果還是為空就用副檔名
        if (lang.empty()) {
            lang = file_extension;
        }

        // 文件內容
        string content = readFile(file_path);

        // 組合成markdown
        string markdown_output = "`" + file_path + "`\n\n```" + lang + ":" +
file_name + "\n" + content + "\n```\n\n";
        return markdown_output;
    }
    catch (exception& e) {
        return "";
    }
}

int main(int argc, char* argv[]) {

    // 抓資料並轉換
```

```
    string file_path = argv[1];
    string markdown_output = convert_to_markdown(file_path);

    //若不為空，處理後貼到剪貼簿
    if (!markdown_output.empty()) {
        if (OpenClipboard(NULL)) {
            EmptyClipboard();
            HGLOBAL clipbuffer;
            char* buffer;
            clipbuffer = GlobalAlloc(GMEM_DDESHARE, markdown_output.length() +
1);
            buffer = (char*)GlobalLock(clipbuffer);
            strcpy_s(buffer, markdown_output.length() + 1,
markdown_output.c_str());
            GlobalUnlock(clipbuffer);
            SetClipboardData(CF_TEXT, clipbuffer);
            CloseClipboard();
        }
    }
    return 0;
}
```

比較麻煩大概是無視窗建置，
不過這部分visual studio 2022方法卻很方便