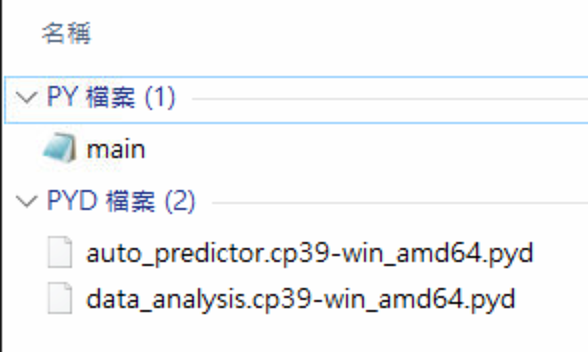
**nuitka**

1. 參數說明
2. 打包型式I -> 將一個檔案或是整個資料夾打包成.so檔案
3. 打包型式II -> pyd與exe
4. Q & A
5. 參數說明
   1. **--show-progress: 展示過程。**
   2. **--show-memory: 顯示記憶體的狀態。**
   3. **--standalone:** 放到其他的電腦就可直接執行，不用安裝python。產生的結果為一個**\*.dist**的資料夾，執行的時候需要有整個\*.dist的資料夾。
   4. **--onefile:** 打包成單一個exe。
   5. **--nofollow-imports:** 不編譯程式碼中所有的import入package或module，比如keras，numpy之類的。
   6. **--include-data-file:** 打包指定檔案，--include-data-file=<source>=<target>，EX: --include-data-file=./aria2c.exe=./aria2c.exe。
   7. **--include-data-dir:** 打包指定資料夾，--include-data-file=<source>=<target>
   8. **--windows-icon-from-ico=\*.ico**: 指定exe檔案的圖案。
   9. **--output-dir=out:** 指定編譯後的資料放到out資料夾裡面。
   10. **--remove-output:** 生成模組或 exe 之後，刪除生成目錄，預設關閉。
   11. **--mingw64:** 採用環境變數下的mingw64作為gcc的編譯。
   12. **--windows-disable-console:** exe執行時，不會出現cmd的視窗。
6. 打包型式I -> 將一個檔案或是整個資料夾打包成.so檔案
   1. 打包一個檔案，
      1. nuitka --module --mingw64 ./\*.py
   2. 打包整個資料夾，第一個test為打包後的結果，第二個test為想要打包的資料夾。
      1. EX: nuitka --module --include-package=test ../../test
   3. 如果打包的環境，沒有安裝某套件或是模組，指令需是如下:
      1. EX: nuitka --module --follow-import-to=**套件**或是**模組**名稱 --mingw64 ./auto\_predictor.py
7. 打包型式II -> pyd與exe

有三個檔案，**data\_analysis.py**、**auto\_predictor.py**、**main.py**，現今要將data\_analysis.py與auto\_predictor.py分別打包為pyd(動態連結檔)的形式，然後再將這2個pyd與main.py，也就是**“a.pyd + b.pyd + c.pyd + … …”** + **“main.py”**，一起打包成exe檔案**。**

* 1. pyd
     1. 指令，data\_analysis.py
        + 強迫用mingw64做gcc的編譯，沒指定輸出地方，EX: **nuitka --module --mingw64 ./data\_analysis.py**
        + 強迫用mingw64做gcc的編譯，有指定輸出地方，EX: **nuitka --module --output-dir=out --mingw64 ./data\_analysis.py**
        + 強迫用mingw64做gcc的編譯，有指定輸出地方，不from或import入package與module，EX: **nuitka --module --nofollow-imports --output-dir=out --mingw64 ./data\_analysis.py**
     2. 指令，auto\_predictor.py
        + 強迫用mingw64做gcc的編譯，沒指定輸出地方，EX: **nuitka --module --mingw64 ./auto\_predictor.py**
        + 強迫用mingw64做gcc的編譯，有指定輸出地方，EX: **nuitka --module --output-dir=out --mingw64 ./auto\_predictor.py**
        + 強迫用mingw64做gcc的編譯，有指定輸出地方，不from或import入package與module，EX: **nuitka --module --nofollow-imports --output-dir=out --mingw64 ./auto\_predictor.py**
  2. exe: \*.pyd + main.py: 放在一起，然後編譯
     1. 編譯時，資料內容如下圖，裡面有pyd的檔案，



* + 1. \*.pyd原始的python檔案裏頭，由於在編譯成exe時候，程式無法吃進去所from或import的套件與模組，並且在編譯成module時候，**使用--nofollow-imports的參數，確保乾淨**。
    2. main.py**裡面要有所需要的from或import的套件與模組**。
    3. 編譯的環境要有mian.py，所有要加入的package與module。
    4. 指令
       - 沒ico與指定輸出的地方，EX: **nuitka --standalone --onefile --mingw64 ./main.py**
       - 加入ico，輸出到out資料夾，會出現cmd視窗(能debug)，EX: **nuitka --standalone --onefile --windows-icon-from-ico=./icon\_02.ico --output-dir=out --mingw64 ./main.py**
       - 加入ico，輸出到out資料夾，不出現cmd視窗(不能debug)，EX: **nuitka --standalone --onefile --windows-icon-from-ico=./icon\_02.ico --output-dir=out --mingw64 --windows-disable-console ./main.py**
  1. exe: \*.pyd + main.py: 沒有放在一起編譯
     1. 單獨編譯main.py，成為exe。
     2. 要執行的時候，pyd與exe放在一起。
  2. **下面目前試沒有成功**

**exe: \*.pyd + main\_no\_import.py: 如果在編譯的時候，main.py沒有加入package與module(main\_no\_import.py)，並且單獨編譯main\_no\_import.py，做法如下:**

* + 1. **需要在編譯pyd時，使用--follow-import-to，把需要的package與module加入。**
    2. **單獨將main\_no\_import.py編譯，**
    3. **最後，exe與pyd放在一起執行即可。**

1. Q & A
   1. Q: --windows-disable-console 閃退問題?
      1. A: 刪除printf的部分。
   2. Q: 减少exe的size大小?
      1. A: 在 nuitka 使用--onefile命令打包時，**zstandard**模組可以減少size。
   3. 編譯完成的pyd，檔名可以更改?
      1. 可以更改，例如: 可從**data\_analysis.cp39-win\_amd64.pyd**變成**data\_analysis.pyd**。
   4. continue
2. continue

|  |  |  |
| --- | --- | --- |
|  | **1** |  |
|  | **2** |  |
|  | **3** |  |
|  | **4** |  |
|  | **5** |  |
|  | **6** |  |
|  | **7** |  |
|  | **8** |  |
|  | **9** |  |
|  | **10** |  |
|  | **11** |  |
|  | **13** |  |
|  | **13** |  |
|  | **14** |  |
|  | **15** |  |