反轉蟒蛇

cheese_ge

2025/11/15

Slido

whoami

- 起司 / 起司哥 / cheese_ge
- 逢甲黑客社 12th 學術
- 2025 PUPC 銀牌獎
- 2025 NCPC rk.56
- 競程、Reverse
- DC: cheese_ge

Outline

- 蛤？逆向工程？
- 你看過蟒蛇嗎
- 蟒蛇反轉
- 現代問題需要現代手段

前置工具

蛤？逆向工程？

逆向工程簡介

逆向工程

- Reverse Engineering
- 即對一專案標產品進行逆向分析及研究，從而演繹並**得出該產品的處理流程、組織結構、功能效能規格**等設計要素，以製作出功能相近，但又不完全一樣的產品。

逆向工程

- 把程式變回 code 再看懂。

生活中的逆向工程

一份我做的吐司:D

生活中的逆向工程

熱壓機

生活中的逆向工程

生活中的逆向工程

可以幹嘛

- mod/plugin
- 找出程式漏洞
- 分析惡意軟體
- 遊戲外掛

mod/plugin

- Minecraft
  - bukkit
  - forge
- GTA
  - Multi Theft Auto

程式漏洞

- 理解程式背後運作方式
- 分析程式邏輯找出問題

惡意軟體分析

- 破壞方式
- 滲透方式
- 傳播方式
- 怎麼處理

遊戲外掛

- 修改記憶體
- 分析遊戲存檔方式
- 分析遊戲漏洞

你看過蟒蛇嗎

python 基礎知識

輸出

```
1 print('Hello World!')
2 print('today is 11/15')
```

變數

```
1 x = 2
2 y = 3
3 z = x + y
4 letter = 'A'
5 pi = 3.14
6
7 print(letter)
8 print("x + y = ", z)
```

```
1 z = x + y
2 z = x - y
3 z = x * y
4 z = x / y
5 z = x % y
```

字元

- ASCII table
- A = 65
- Z = 90
- a = 97
- z = 122

```
1 print(ord('A'))
```

列表

```
1 array = [0, 1, 2, 3, 4, 5]
2 print("Array:", array)
3 array.append(6)
4 array[0] = 10
5 print("Updated Array:", array)
```

條件判斷

```
1 if z > 4:
2     print("z is > 4")
3 elif z == 4:
4     print("z is == 4")
5 else:
6     print("z is < 4")
```

```
1 x = 1
2 y = 2
3 print(x > y)
4 print(x < y)
5 print(x >= y)
6 print(x <= y)
7 print(x == y)
8 print(x != y)
```

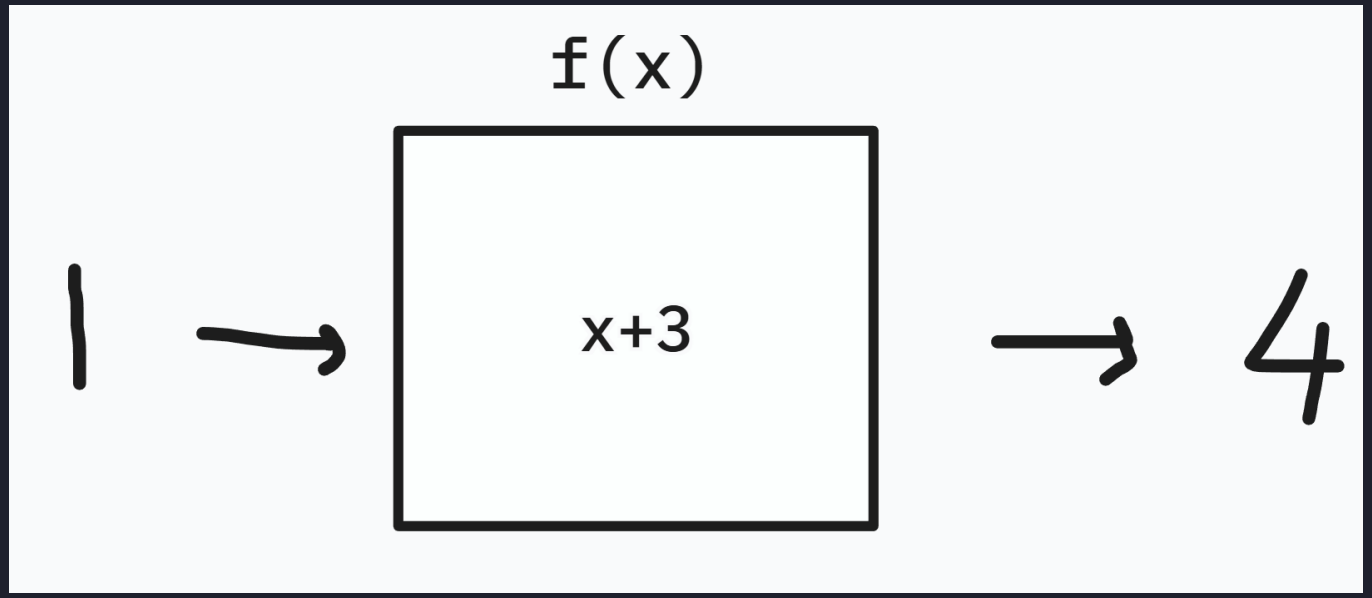
迴圈

```
1 array = [0, 1, 2, 3, 4, 5]
2 for i in range(0,3):
3     print("Loop iteration:", i)
4 for nunber in array:
5     print("Array element:", nunber)
```

函式

$$f(x) = x+3$$

函式



函式

```
1 print()
2
3 array = [0,1,2,3]
4 len(array)
5 ord('a')
6 chr(97)
```

stack

- 堆疊
- 可以從最上方放東西跟拿東西
- push/pop

第一關

- 輸入 21 輸出會是什麼

```
1 import hand
2 if hand.magic_number >= 20 or hand.magic_number <0:
3     print("no")
4 if (hand.magic_number*hand.magic_number)/7>25:
5     print("maybe")
6 if (hand.magic_number*hand.magic_number)%7==1:
7     print("yes")
```

第一關

- 輸出 maybe 跟 yes，請問 magic_number 是多少

```
1 import hand
2 if hand.magic_number >= 20 or hand.magic_number <0:
3     print("no")
4 if (hand.magic_number*hand.magic_number)/7>25:
5     print("maybe")
6 if (hand.magic_number*hand.magic_number)%7==1:
7     print("yes")
```

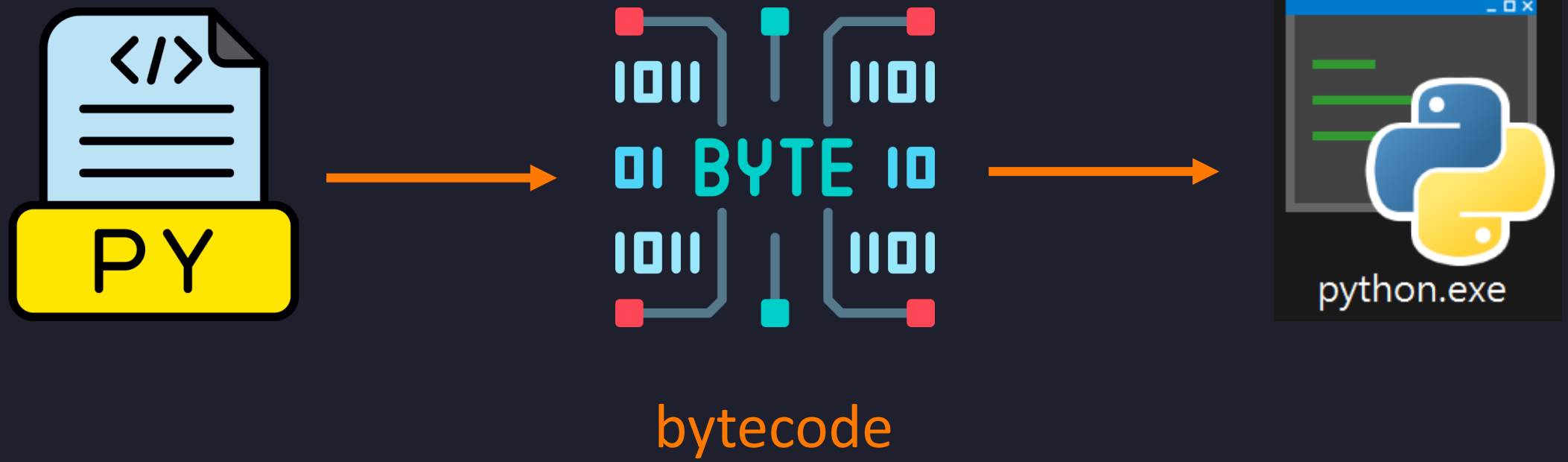
QA time

蟒蛇反轉

pyc,dis,bytecode

Python 怎麼執行

- 簡單來說...

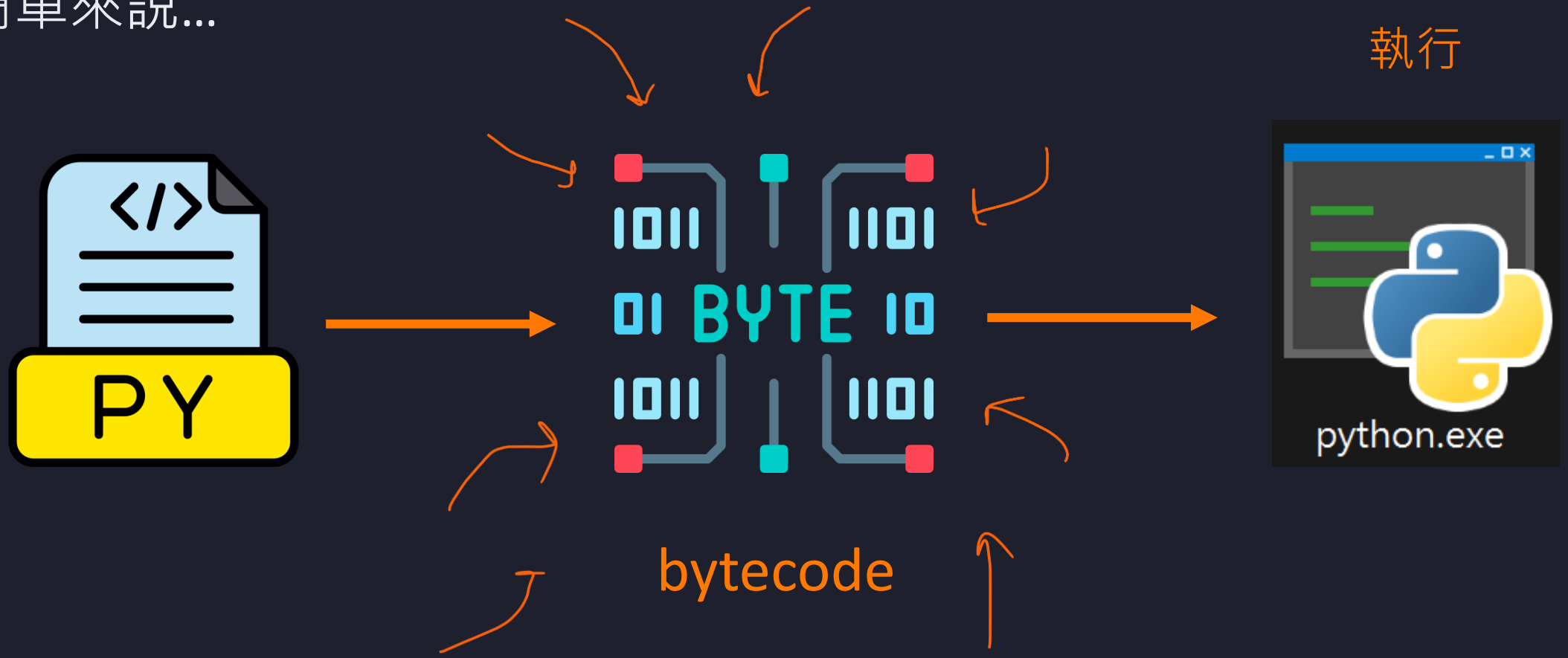


執行

bytecode

Python 怎麼執行

- 簡單來說...



bytecode 長怎樣

```
1    print("this is fine!")
2    me = "cheese_ge"
3    print("I'm ",me)
```

```
1           0 LOAD_NAME                0 (print)
            2 LOAD_CONST               0 ('this is fine!')
            4 CALL_FUNCTION            1
            6 POP_TOP

2           8 LOAD_CONST               1 ('cheese_ge')
           10 STORE_NAME               1 (me)

3          12 LOAD_NAME                0 (print)
           14 LOAD_CONST               2 ("I'm ")
           16 LOAD_NAME                1 (me)
           18 CALL_FUNCTION            2
           20 POP_TOP
           22 LOAD_CONST               3 (None)
           24 RETURN_VALUE
```

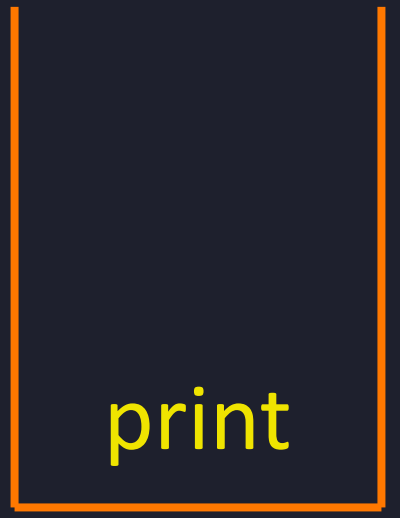
看程式的 bytecode

```
python -m dis 檔名.py
```

怎麼看 bytecode

Document

```
1           0 LOAD_NAME                0 (print)
            2 LOAD_CONST               0 ('this is fine!')
            4 CALL_FUNCTION            1
            6 POP_TOP
```



怎麼看 bytecode

push 'this is fine!'

```
1           0 LOAD_NAME                0 (print)
            2 LOAD_CONST               0 ('this is fine!')
            4 CALL_FUNCTION            1
            6 POP_TOP
```

怎麼看 bytecode

pop 1 個當參數 執行接下來 pop 的函式

```
1           0 LOAD_NAME                0 (print)
            2 LOAD_CONST               0 ('this is fine!')
            4 CALL_FUNCTION            1
            6 POP_TOP
```

怎麼看 bytecode

pop 1 個當參數 執行接下來 pop 的函式

```
1           0 LOAD_NAME                0 (print)
            2 LOAD_CONST               0 ('this is fine!')
            4 CALL_FUNCTION            1
            6 POP_TOP
```

_____('this is fine!')

怎麼看 bytecode

pop 1 個當參數 執行接下來 pop 的函式

```
1           0 LOAD_NAME                0 (print)
            2 LOAD_CONST               0 ('this is fine!')
            4 CALL_FUNCTION            1
            6 POP_TOP
```

print('this is fine!')

怎麼看 bytecode

還會 push 結果回去

```
1           0 LOAD_NAME                0 (print)
            2 LOAD_CONST               0 ('this is fine!')
            4 CALL_FUNCTION            1
            6 POP_TOP
```

怎麼看 bytecode

```
1           0 LOAD_NAME                0 (print)
            2 LOAD_CONST               0 ('this is fine!')
            4 CALL_FUNCTION            1
            6 POP_TOP
```

怎麼看 bytecode

行號　　指令位置　　指令參數

```
1           0 LOAD_NAME                0 (print)
            2 LOAD_CONST               0 ('this is fine!')
            4 CALL_FUNCTION            1
            6 POP_TOP
```

指令名稱　　解釋

看 bytecode #2

```
 1           0 LOAD_NAME                0 (range)
             2 LOAD_CONST               0 (10)
             4 CALL_FUNCTION            1
             6 GET_ITER
       >>    8 FOR_ITER                12 (to 22)
            10 STORE_NAME               1 (i)

 2          12 LOAD_NAME                2 (print)
            14 LOAD_NAME                1 (i)
            16 CALL_FUNCTION            1
            18 POP_TOP
            20 JUMP_ABSOLUTE            8
       >>   22 LOAD_CONST               1 (None)
            24 RETURN_VALUE
```

看 bytecode #2

```
1           0 LOAD_NAME                0 (range)
            2 LOAD_CONST               0 (10)
            4 CALL_FUNCTION            1
            6 GET_ITER
      >>    8 FOR_ITER                12 (to 22)
           10 STORE_NAME               1 (i)

2          12 LOAD_NAME                2 (print)
           14 LOAD_NAME                1 (i)
           16 CALL_FUNCTION            1
           18 POP_TOP
           20 JUMP_ABSOLUTE            8
      >>   22 LOAD_CONST               1 (None)
           24 RETURN_VALUE
```

range(10)

看 bytecode #2

迴圈 →

```
1           0 LOAD_NAME                0 (range)
            2 LOAD_CONST               0 (10)
            4 CALL_FUNCTION            1
            6 GET_ITER
      >>    8 FOR_ITER                12 (to 22)
           10 STORE_NAME               1 (i)

2          12 LOAD_NAME                2 (print)
           14 LOAD_NAME                1 (i)
           16 CALL_FUNCTION            1
           18 POP_TOP
           20 JUMP_ABSOLUTE            8
      >>   22 LOAD_CONST               1 (None)
           24 RETURN_VALUE
```

看 bytecode #2

for i in range(10):

```
1           0 LOAD_NAME                0 (range)
            2 LOAD_CONST               0 (10)
            4 CALL_FUNCTION            1
            6 GET_ITER
      >>    8 FOR_ITER                12 (to 22)
           10 STORE_NAME               1 (i)

2          12 LOAD_NAME                2 (print)
           14 LOAD_NAME                1 (i)
           16 CALL_FUNCTION            1
           18 POP_TOP
           20 JUMP_ABSOLUTE            8
      >>   22 LOAD_CONST               1 (None)
           24 RETURN_VALUE
```

看 bytecode #2

```
  1           0 LOAD_NAME                0 (range)
              2 LOAD_CONST               0 (10)
              4 CALL_FUNCTION            1
              6 GET_ITER
        >>    8 FOR_ITER                12 (to 22)
             10 STORE_NAME               1 (i)

  2          12 LOAD_NAME                2 (print)
             14 LOAD_NAME                1 (i)
             16 CALL_FUNCTION            1
             18 POP_TOP
             20 JUMP_ABSOLUTE            8
        >>   22 LOAD_CONST               1 (None)
             24 RETURN_VALUE
```

```
1 for i in range(10):
2   print(i)
```

一些指令功能

指令	行為
LOAD_XXXX	push 一個東西到 stack 裡
STORE_XXXX	pop 一個東西並存到指定位置
CALL_FUNCTION	pop n 個東西作為參數，再 pop 一個函式
POP_TOP	從 stack 移除一個東西
GET_ITER	pop 一個東西，push 一個 iterator
FOR_ITER	嘗試從 iterator 取出內容並 push 到 stack
BINARY_XXXX	pop 兩個做指定運算 再把結果 push
JUMP_XXXX	符合條件就跳到指定位置
COMPARE_OP	pop 兩個數字並比較
BUILD_LIST	創建一個 列表
LIST_EXTEND	pop 一個東西然後放到 stack 頂端的 list

小提醒

- bytecode 的實作方式隨時都有可能改變
- 3.6 , 3.10 , 3.11 都有更新

pyc?

python 在一些情況下會產生 .pyc 檔案
不同版本內容可能不一樣

- magic number
- 寫入時間
- 檔案大小
- bytecode

pyc?

- 可以用 python 執行同樣版本的 pyc
- 程式已經轉換成 bytecode 所以會快一點點
- 變成 exe 的 python 也會放 pyc

拿到 pyc

- 會出現在 __pycache__ 資料夾下

```
python -m py_compile 檔名.py
```

看 pyc 的 bytecode

- 創建一個 dispy.py 打以下內容

```python
import sys
from pathlib import Path
import marshal
import dis

data = Path(sys.argv[1]).read_bytes()
co = marshal.loads(data[16:])
dis.dis(co)
```

看 pyc 的 bytecode

```
python .\dispy.py 檔名.pyc
```

第二關

- 到 https://short.cheesege.com/ais3-2 下載題目
- 可以先嘗試把程式還原出來

題解時間!

現代問題需要現代手段

pylingual

Pylingual

- 機器學習
- 預測原本的程式碼
- **可能會錯或是逆不出來**
- pylingual.io

第三關

- https://short.cheesege.com/ais3-3
- 大家也可以把今天拿到的 pyc 丟進去看看

其他酷酷工具

- pycdc
  - 3.8 ~ 3.11
- uncompyle6
  - 2.4 ~ 3.8
- decompyle3
  - 3.7 ~ 3.8

QA

謝謝各位

追加題目

- https://short.cheesege.com/ais3-4

追加内容

- https://pyinstxtractor-web.netlify.app/
- pip install pyinstaller
- pyinstaller -F ./p4.py