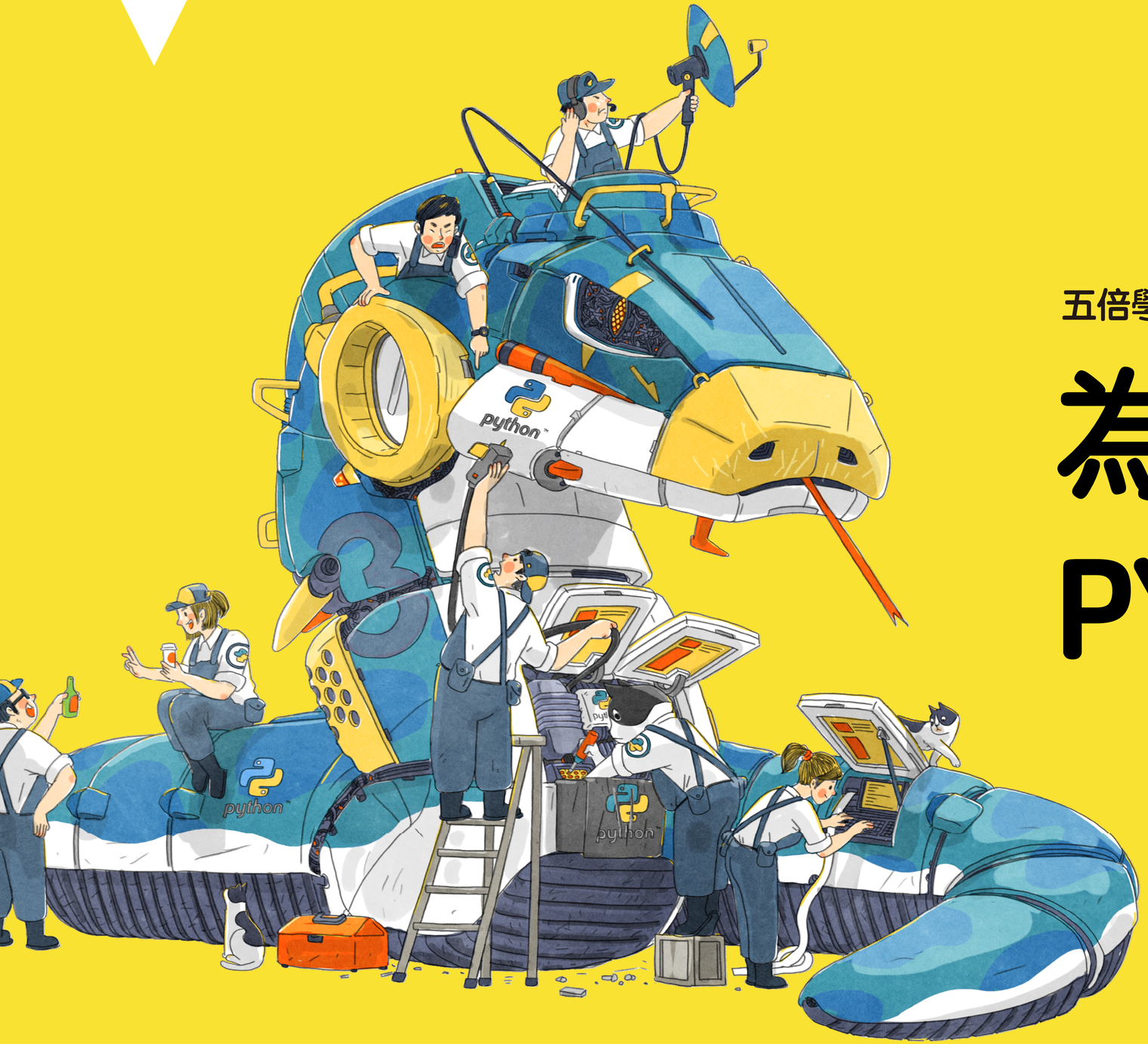


冷知識篇

五倍學院

為你自己學 PYTHON

高見龍

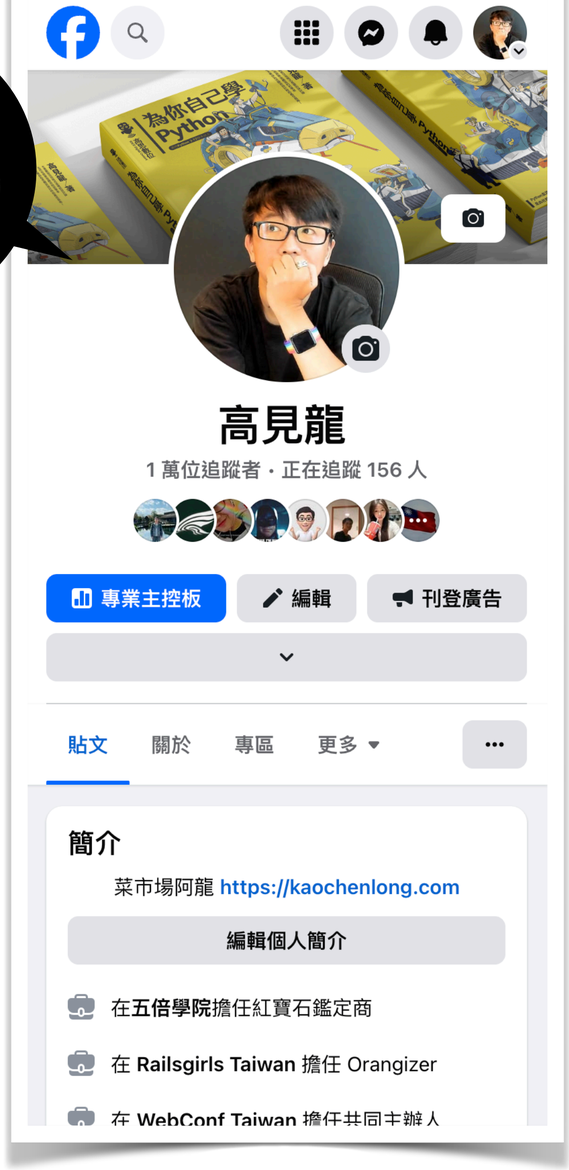
高思數位
KAOS Digital Publication

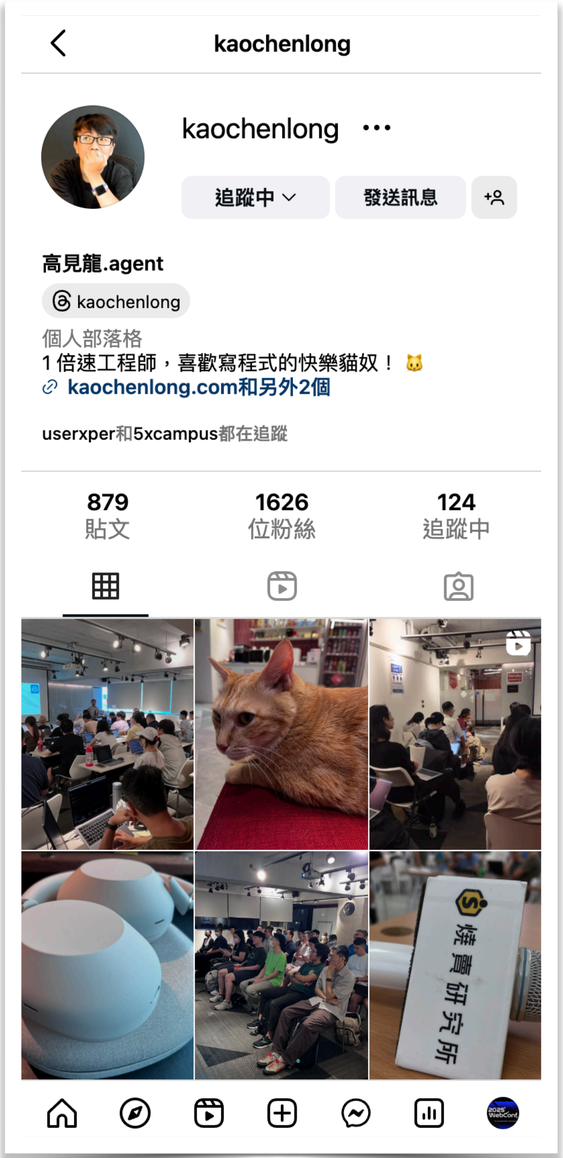
自我介紹

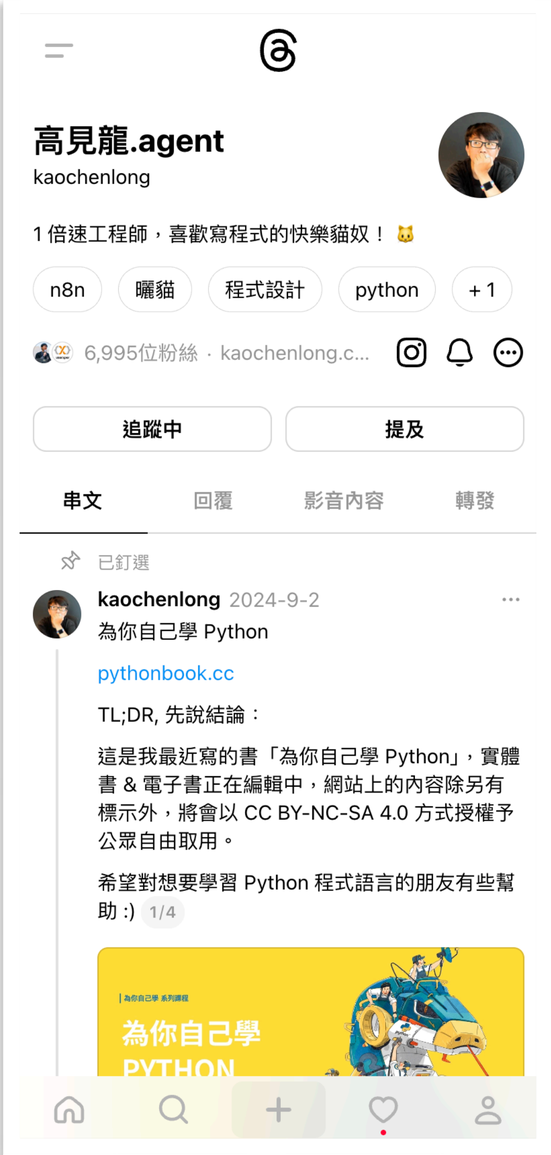
高見龍 @ 五倍學院

- 程式開發 ≒ 30 年
- 教學經驗 ≒ 16 年
- 出版：
  - 「為你自己學 Git」
  - 「為你自己學 Ruby on Rails」
  - 「為你自己學 Python」
- 是個喜歡打魔物獵人而且希望可以寫一輩子程式的電腦阿宅

高見龍


高見龍
1 萬位追蹤者・正在追蹤 156 人
專業主控板
編輯
刊登廣告
貼文
關於
專區
更多
簡介
菜市場阿龍 https://kaochenlong.com
編輯個人簡介
在五倍學院擔任紅寶石鑑定商
在 Railsgirls Taiwan 擔任 Orangizer
在 WebConf Taiwan 擔任共同主辦人


kaochenlong
kaochenlong
追蹤中
發送訊息
高見龍.agent
kaochenlong
個人部落格
1 倍速工程師，喜歡寫程式的快樂貓奴！
kaochenlong.com和另外2個
userxper和5xcampus都在追蹤
879
貼文
1626
位粉絲
124
追蹤中
燒賣研究所


高見龍.agent
kaochenlong
1 倍速工程師，喜歡寫程式的快樂貓奴！
n8n
曬貓
程式設計
python
+ 1
6,995位粉絲 · kaochenlong.c...
追蹤中
提及
串文
回覆
影音內容
轉發
已釘選
kaochenlong 2024-9-2
為你自己學 Python
pythonbook.cc
TL;DR, 先說結論：
這是我最近寫的書「為你自己學 Python」，實體書 & 電子書正在編輯中，網站上的內容除另有標示外，將會以 CC BY-NC-SA 4.0 方式授權予公眾自由取用。
希望對想要學習 Python 程式語言的朋友有些幫助 :)
1/4
為你自己學
PYTHON

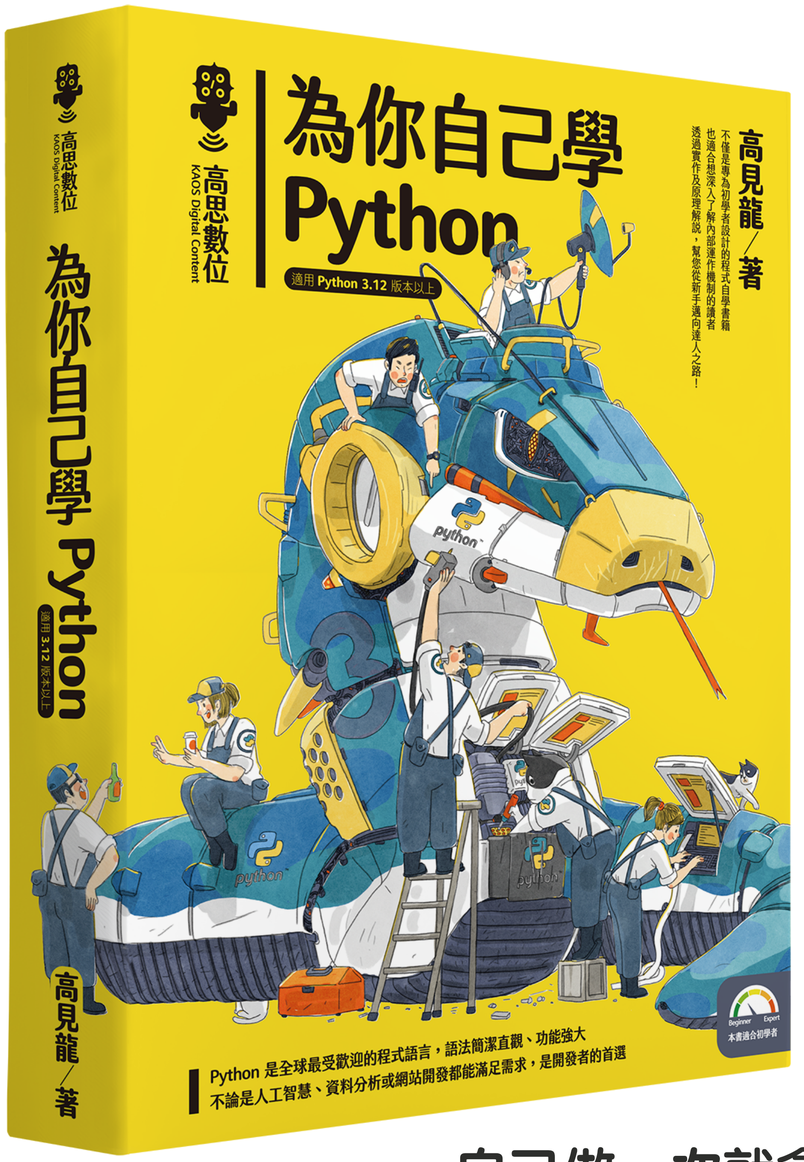
你寫 Python 嗎？

你是不是用別的語言在寫 Python？

重點是好玩！

(提示：可能還會有獎品)

關於這本書...



自己做一次就會知道在台灣寫電腦書不會賺錢...

CPython

Deep Dive CPython explores the internal mechanics of CPython, the widely used Python interpreter written in C. Starting with a practical guide on downloading and compiling the CPython source, this book is perfect for developers eager to understand Python's behaviour at a fundamental level.

The book takes readers from basic concepts to complex details with a systematic breakdown of core components. It covers everything from CPython's data structures like PyObject and PyTypeObject to object lifecycle management, giving insight into memory allocation and object reference counting. Each chapter illustrates CPython's architecture, such as Python's "everything is an object" philosophy, list handling, string manipulation, and dictionary operations. Readers will explore Python's REPL modifications, string internals, and custom type creation with practical examples, like crafting a "backflipping" PyKitty_Type. Detailed sections on Python's virtual machine operations, bytecode generation, and exception handling enrich readers' understanding of how Python code is parsed, compiled, and executed.

This book is a thorough guide for readers who want to go beyond basic Python use and understand how it works internally. Covering complex concepts like generators, iterators, descriptors, and metaclasses, this book equips readers with a thorough grasp of Python's performance optimization and design complexities.

What you will learn:

- How to download, compile, and modify CPython's source code
- Gain insight into fundamental structures like PyObject and PyTypeObject
- Understand Python's detailed handling of lists, strings, dictionaries, and the REPL environment
- What are bytecode generation, custom types, and the inner workings of Python's virtual machine

ISBN 979-8-8688-1768-7
55999
9 798868 817687

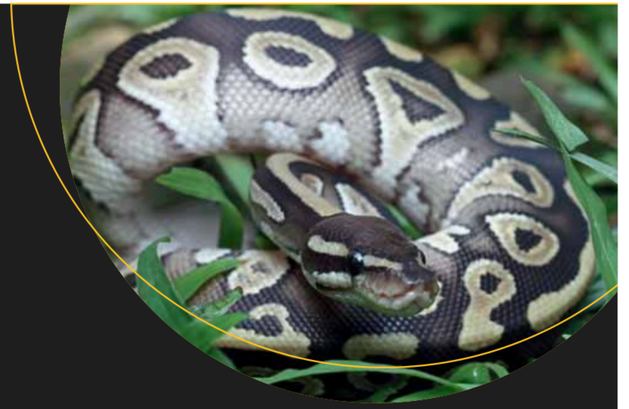
Shelve in:
Python

User Level:
Intermediate–Advanced

apress®
www.apress.com

Kao CPython

Apress®



CPython

A Complete Guide to CPython's Architecture and Performance

—

Chien-Lung Kao

Apress®

Q. 以下這幾種程式語言，誰最老？

1) Java

2) JavaScript

3) Python

4) Ruby

5) PHP

Q. 以下這幾種程式語言，誰最老？

1) Java (1995)

2) JavaScript (1995)

3) Python (1991)

4) Ruby (1995)

5) PHP (1995)

Q. 以下程式碼的執行結果為何？

```
def add_card(bk, card):
    bk.append(card)
    bk = ["奇犴", "西索", "尼飛彼多"]

book = ["小傑", "雷歐力"]
add_card(book, "酷拉皮卡")

print(book)   # 這會印出什麼？
```

1) ["小傑", "雷歐力", "酷拉皮卡"]

2) ["奇犴", "西索", "尼飛彼多"]

3) 別騙我了，這程式根本不會動

4) 我只想整套獵人卡片！

Q. 以下程式碼的執行結果為何？

```
def add_card(bk, card):
    bk.append(card)
    bk = ["奇犴", "西索", "尼飛彼多"]

book = ["小傑", "雷歐力"]
add_card(book, "酷拉皮卡")

print(book)   # 這會印出什麼？
```

1) ["小傑", "雷歐力", "酷拉皮卡"]

2) ["奇犴", "西索", "尼飛彼多"]

3) 別騙我了，這程式根本不會動

4) 我只想整套獵人卡片！

Q. 以下程式碼的執行結果為何？

```
x = 100

def add_one(x):
    x = x + 1
    return x

add_one(x)
print(x)  # 會印出什麼？
```

1) 1

2) 101

3) 100

4) 程式會出錯

Q. 以下程式碼的執行結果為何？

```
x = 100

def add_one(x):
    x = x + 1
    return x

add_one(x)
print(x)  # 會印出什麼？
```

1) 1

2) 101

3) 100

4) 程式會出錯

Python 的變數宣告？

Python 沒有像其他程式語言一樣有 var 或 let 的「宣告變數」關鍵字，當遇到 x = 1 的寫法的時候，Python 會從「作用域」來判斷是不是需要宣告一個新的變數 x，還是試著去找既有的變數 x 然後把它的值設定成 1。

參考資料：https://pythonbook.cc/chapters/basic/function#a--1

Q. 以下程式碼的執行結果為何？

```
i = 0

def add_i():
    i = i + 1

add_i()
print(i) # 這會印出什麼
```

1) 還是 0

2) 1

3) 2

4) 這程式會出錯

Q. 以下程式碼的執行結果為何？

```
i = 0

def add_i():
    i = i + 1

add_i()
print(i) # 這會印出什麼
```

1) 還是 0

2) 1

3) 2

4) 這程式會出錯

Q. 最後印出來的值，哪一個會跟其它三個不同？

```python
# 檔案：hello.py
a = 100
b = 200

def hey():
    return 300

res1 = 300
res2 = 100 + 200
res3 = a + b
res4 = hey()
```

```python
print(id(res1))
print(id(res2))
print(id(res3))
print(id(res4))
```

1) res1

2) res2

3) res3

4) res4

Q. 最後印出來的值，哪一個會跟其它三個不同？

```python
# 檔案：hello.py
a = 100
b = 200

def hey():
    return 300

res1 = 300
res2 = 100 + 200
res3 = a + b
res4 = hey()
```

```python
print(id(res1))
print(id(res2))
print(id(res3))
print(id(res4))
```

1) res1

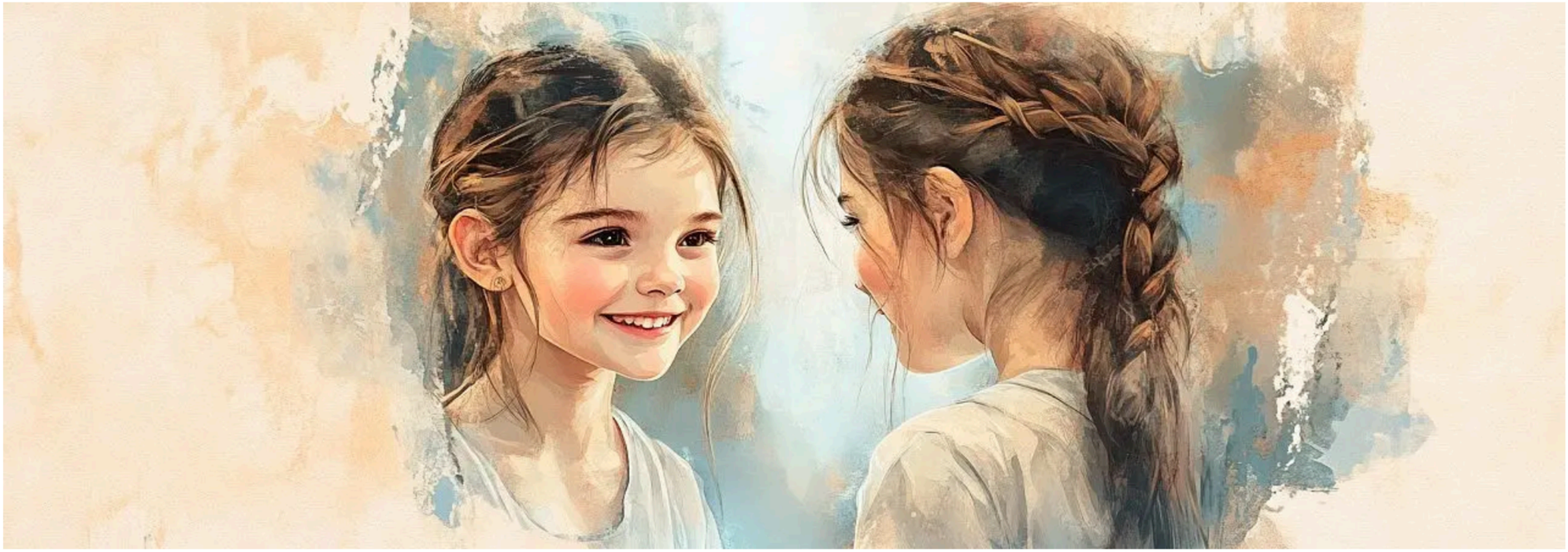
2) res2

3) res3

4) res4

你是數字，我是數字，我們不一樣！

2024.10.13



在 Python 裡的 `id()` 函數可以算出某個值或物件在 Python 世界裡的「編號」，如果有兩個值算出來的編號是一樣的，表示這兩個值不只相等，而且還是一顆物件。

參考資料：https://kaochenlong.com/2024/10/13/same-value-but-different-object-in-python.html

Q. 以下程式碼的執行結果為何？

```
a = 7
b = 11
c = a - b
print(c is -4) # 這會印出什麼？

c = c - 1
print(c is -5) # 這會印出什麼？

c = c - 1
print(c is -6) # 這會印出什麼？
```

1) True / True / True

2) True / True / False

3) False / False / True

4) False / False / False

Q. 以下程式碼的執行結果為何？

```
a = 7
b = 11
c = a - b
print(c is -4) # 這會印出什麼？

c = c - 1
print(c is -5) # 這會印出什麼？

c = c - 1
print(c is -6) # 這會印出什麼？
```

1) True / True / True

2) True / True / False

3) False / False / True

4) False / False / False

Python 裡的小整數

```
>>> a = 256
>>> b = 256
>>> a is b
True

>>> a = 257
>>> b = 257
>>> a is b
False
```

Q. 以下程式碼的執行結果為何？

```
a = float("nan")        # NaN
b = float("nan")        # NaN
print(a == a)           # NaN 不等於任何值，所以這裡會印出 True

print([a] == [a])       # 這裡會印出什麼？
print([a] == [b])       # 這裡會印出什麼？
```

1) True / True

2) True / False

3) False / True

4) False / False

Q. 以下程式碼的執行結果為何？

```
a = float("nan")        # NaN
b = float("nan")        # NaN
print(a == a)           # NaN 不等於任何值，所以這裡會印出 True

print([a] == [a])       # 這裡會印出什麼？
print([a] == [b])       # 這裡會印出什麼？
```

1) True / True

2) True / False

3) False / True

4) False / False

神奇的 NaN？

1. 在 Python 裡的 NaN 是一個浮點數，根據 IEEE 754 的定義，如果拿 NaN 跟任何值比較都會得到 False，包括它自己本身！
2. Python 的 List 的比較邏輯是先檢查元素「個數」是否相同。
3. 如果數量一樣，接下來比較每個元素是不是同一顆物件，也就是比較記憶體位置。
4. 原始碼：Objects/listobject.c（函數名稱 list_richcompare）

Q. 以下程式碼的執行結果為何？

```
華安 = (9, 5, 2, 7)
print(sorted(華安) == sorted(華安))          # 會印出什麼？
print(reversed(華安) == reversed(華安))      # 會印出什麼？
```

1) True / True

2) True / False

3) False / True

4) False / False

Q. 以下程式碼的執行結果為何？

```
華安 = (9, 5, 2, 7)
print(sorted(華安) == sorted(華安))          # 會印出什麼？
print(reversed(華安) == reversed(華安))      # 會印出什麼？
```

1) True / True

2) True / False

3) False / True

4) False / False

sorted vs reversed

1. **`sorted()`** 函數的回傳值是一個排序好的 List，也就是會得到 [2, 5, 7, 9]
2. **`reversed()`** 函數的回傳值是一個 Iterator，而兩次 reversed() 產生的 Iterator 是不一樣東西

Q. 以下程式碼的執行結果為何？

```
numbers = (i for i in range(3))

a = list(numbers)
b = list(numbers)

print(a + b)    # 會印出什麼？
```

1) [0, 1, 2]

2) [0, 1, 2, 0, 1, 2]

3) [ ]

4) 語法錯誤無法執行

Q. 以下程式碼的執行結果為何？

```
numbers = (i for i in range(3))

a = list(numbers)
b = list(numbers)

print(a + b)    # 會印出什麼？
```

1) [0, 1, 2]

2) [0, 1, 2, 0, 1, 2]

3) []

4) 語法錯誤無法執行

Python 的「產生器（Generator）」

1. `(i for i in range(3))` 不會產生一個串列，而是一個「產生器」。
2. 產生器不是一個具體的資料，而是可以每次跟它討一點東西。
3. 使用第一次 `list()` 的時候就會把全部內容拿光，所以 `a` 是 **`[0, 1, 2]`**
4. 但是 `b` 就沒東西可以拿了，所以 `b` 是 **`[]`**
5. `a + b` 就等於是 **`[0, 1, 2] + []`**，所以最後答案是 **`[0, 1, 2]`**

Q. 以下程式碼的執行結果為何？

```
print(type(1) == type(-1))                  # 會印出什麼？
print(1 ** 1 == 1 ** -1)                    # 會印出什麼？
print(type(1 ** 1) == type(1 ** -1))        # 會印出什麼？
```

1) True / True / True

2) True / True / False

3) False / True / False

4) 少騙我了，這程式根本不會動！

Q. 以下程式碼的執行結果為何？

```
print(type(1) == type(-1))                  # 會印出什麼？
print(1 ** 1 == 1 ** -1)                    # 會印出什麼？
print(type(1 ** 1) == type(1 ** -1))        # 會印出什麼？
```

1) True / True / True

2) True / True / False

3) False / True / False

4) 少騙我了，這程式根本不會動！

Python 的 N 次方

1. type(1) == type(-1) 都是整數，所以兩個都是 int
2. 1 ** 1 == 1 ** -1，數字 1 的 1 次方或 -1 次方都是 1？其實不太對，1 的 1 次方是 1 沒有錯，但 1 的 -1 次方就不是 1 了，是 1.0。
3. 在 Python 拿整數 1 跟浮點數 1.0 做等號的比較會得到 True
4. type(1 ** 1) == type(1 ** -1) 就不是了，這裡一個是 int，一個是 float。

Q. 以下這 4 種字串串接方式，哪一種的效能最差？

```python
# str1 使用 + 串接
str1 = "Hello " + "World"
```

```python
# str2 使用 += 串接
str2 = "Hello "
str2 += "World"
```

```python
# str3 使用 F 字串
a = "Hello"
b = "World"
str3 = f"{a} {b}"
```

```python
# str4 使用串列的 .join 方法
words = ["Hello", "World"]
str4 = " ".join(words)
```

1) str1

2) str2

3) str3

4) str4

Q. 以下這 4 種字串串接方式，哪種的效能最差？

```python
# str1 使用 + 串接
str1 = "Hello " + "World"
```

```python
# str2 使用 += 串接
str2 = "Hello "
str2 += "World"
```

```python
# str3 使用 F 字串
a = "Hello"
b = "World"
str3 = f"{a} {b}"
```

```python
# str4 使用串列的 .join 方法
words = ["Hello", "World"]
str4 = " ".join(words)
```

1) str1

2) str2

3) str3

4) str4

Python 字串組裝的效能比較

2024年10月17日

高見龍
五倍學院 負責人

在 Python 要組出一個 "Hello World" 字串有好幾種方法，有的看起來很簡單，但也可以寫的很囉嗦：

```
# 第一種
str1 = "Hello " + "World"

# 第二種
str2 = "Hello "
str2 += "World"

# 第三種
a = "Hello"
b = "World"
str3 = f"{a} {b}"

# 第四種
```

參考資料：https://pythonbook.cc/articles/string-concatenation-performance-in-python

Q. 以下程式碼的執行結果為何？

```
# 內建函數 isinstance 可用來判斷某個值是不是某種類型，例如：
# isinstance(9527, int)      #=> True
# isinstance("Kitty", int)   #=> False

isinstance(object, type) # 會印出什麼？
isinstance(type, object) # 會印出什麼？
```

1) True / True

2) True / False

3) False / True

4) False / False

Q. 以下程式碼的執行結果為何？

```
# 內建函數 isinstance 可用來判斷某個值是不是某種類型，例如：
# isinstance(9527, int)      #=> True
# isinstance("Kitty", int)   #=> False

isinstance(object, type) # 會印出什麼？
isinstance(type, object) # 會印出什麼？
```

1) True / True

2) True / False

3) False / True

4) False / False

爸爸的爸爸叫爺爺？

1. 在 Python 裡的所有類別，包括 type 類別本身，都是 type 類別生出來的，所以 isinstance(object, type) 會得到 True。
2. 在 Python 3 所有類別的最上層類別都是 object 類別，所以 isinstance(任何東西, object) 這句永遠都會成立。
3. 承上，在 Python 所有的東西都是物件，type 類別本身也是物件，所以 isinstance(type, object) 也會得到 True。

Q. 以下程式碼的執行結果為何？

```
# all([True, True, True])   => True
# all([True, False, True])  => False

a = []
b = [[]]
c = [[[]]]

print(all(a)) # 會印出什麼？
print(all(b)) # 會印出什麼？
print(all(c)) # 會印出什麼？
```

1) True / False / True

2) False / False / False

3) False / True / True

4) True / True / True

Q. 以下程式碼的執行結果為何？

```
# all([True, True, True])   => True
# all([True, False, True])  => False

a = []
b = [[]]
c = [[[]]]

print(all(a)) # 會印出什麼？
print(all(b)) # 會印出什麼？
print(all(c)) # 會印出什麼？
```

1) True / False / True

2) False / False / False

3) False / True / True

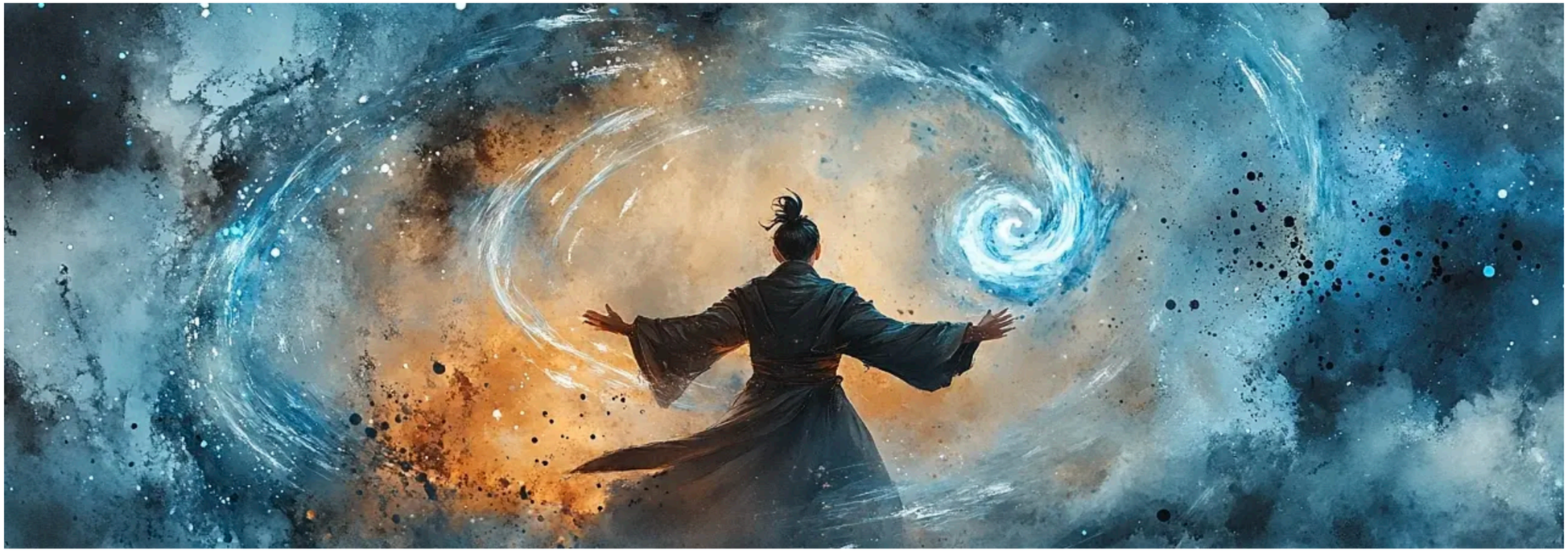
4) True / True / True

all 全部條件都滿足？

1. 在 Python 裡空串列 `[]` 會被當做 `False` 看待，所以 `all([[]])` 相當於 `all([False])`，答案是 `False`。
2. `[[]]` 是一個「包含空串列」的串列，所以它不是一個空的串列。有點像是箱子裡面裝了一個箱子，就算內層的箱子是空的，對外層的箱子來說它也是有裝東西的。所以 `all([[]])` 等於是 `all([True])` 的效果。
3. `all([])` 的話呢？

空虛的真

2024.10.09



來一個我自己覺得滿有趣的 Python 程式問答題。

在 Python 裡，`all()` 這個函數可以接一個串列，串列裡的每一顆元素都成立的時候就會回傳 `True`，反之只要有一顆不

參考資料：https://kaochenlong.com/2024/10/09/vacuous-truth.html

好玩嗎？好玩最重要！

工商服務

iThome 鐵人賽

主題：為你自己學 Gemini

Google
Gemini

五倍學院

iThome 鐵人賽

主題：為你自己學 n8n

五倍學院

Claude Code

AI Coding 實作工作坊（線上直播）

9/14（日） 13:00 ～ 17:00，共 4 小時

五倍學院