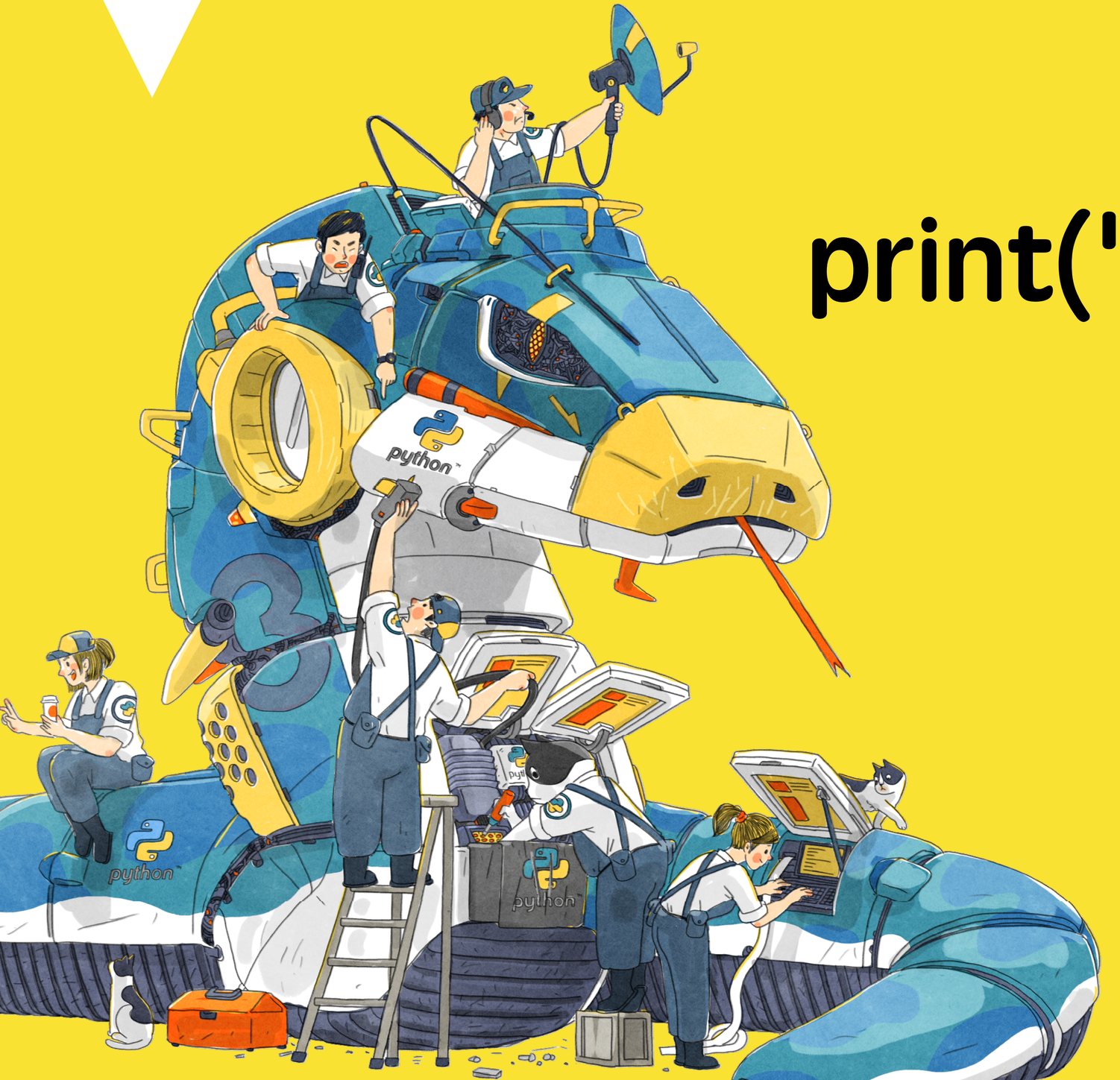
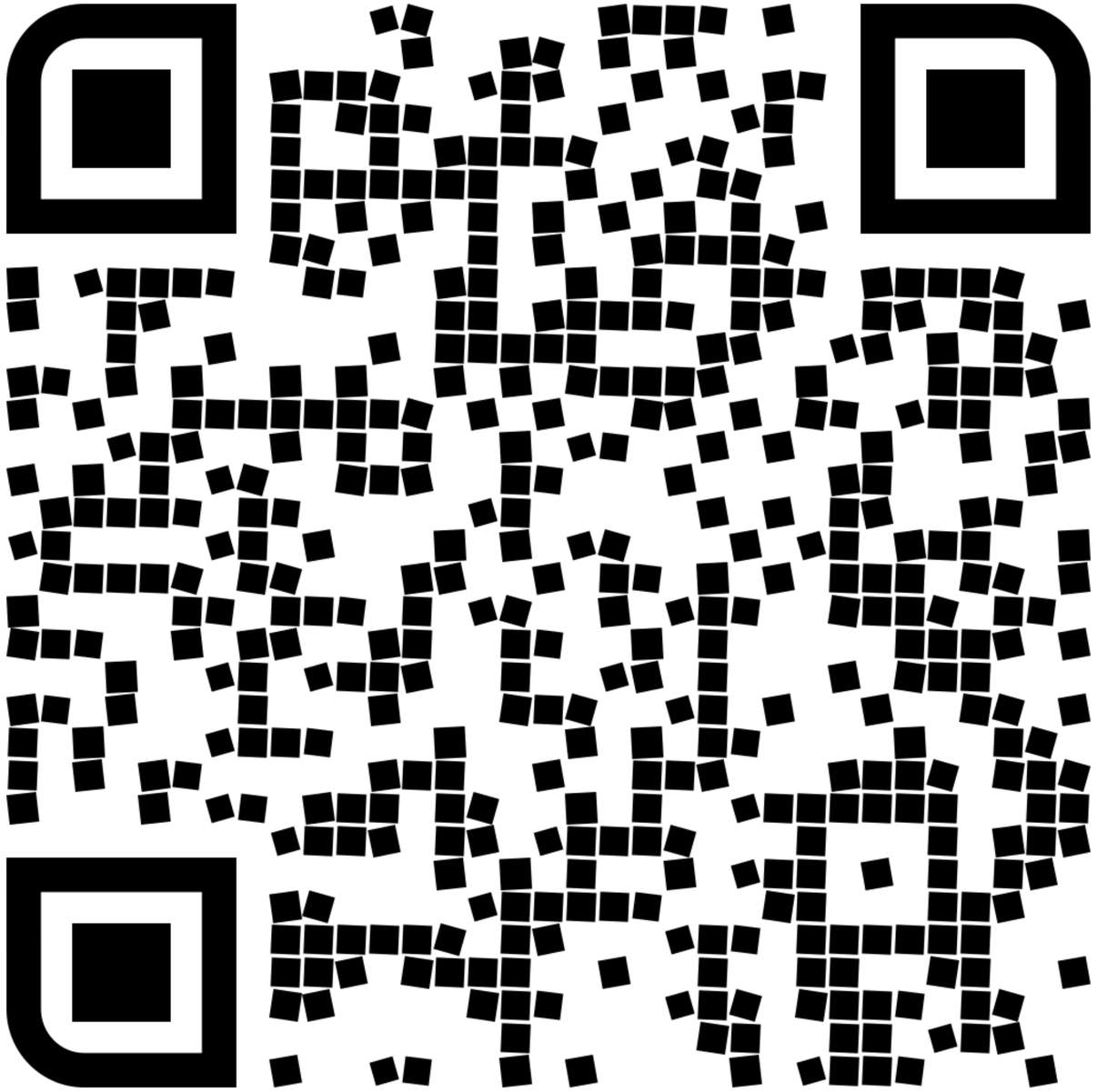



# print("Hello, World")

高見龍



# 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


高思數位
KAOS Digital Content
為你自己學 Python
適用 Python 3.12 版本以上
高見龍／著
不僅是專為初學者設計的程式自學書籍
也適合想深入了解內部運作機制的讀者
透過實作及原理解說，幫您從新手邁向達人之路！
Python 是全球最受歡迎的程式語言，語法簡潔直觀、功能強大
不論是人工智慧、資料分析或網站開發都能滿足需求，是開發者的首選
Beginner
Expert
本書適合初學者

## 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®

since 1972

```
print("Hello World")
```

# Python 程式怎麼運作的？

Plain Text

→ Tokenization

→ AST (Abstract Syntax Tree)

→ ByteCode

→ Virtual Machine

# Tokenization

```
a = 1 + 2
```

```
NAME (a)
EQUAL (=)
NUMBER (1)
PLUS (+)
NUMBER (2)
```

```c
int _PyTokenizer_Get(struct tok_state *tok, struct token *token) {
  int result = tok_get(tok, token);
  if (tok->decoding_erred) {
    result = ERRORTOKEN;
    tok->done = E_DECODE;
  }
  return result;
}
```

檔名：Parser/tokenizer.c

# Grammar/Tokens

```
ENDMARKER
NAME
NUMBER
STRING
NEWLINE
INDENT
DEDENT

LPAR                    '('
RPAR                    ')'
LSQB                    '['
RSQB                    ']'
COLON                   ':'
COMMA                   ','
SEMI                    ';'
PLUS                    '+'
MINUS                   '-'
// ... 略
```

檔名：Grammar/Tokens

```
print("Hello, World")
```

```
print("Hello, Kitty")
```

# dis → ByteCode

```python
def greeting(name):
    return f"Hello, {name}!"

print(greeting("Kitty"))
```

檔名：hello.py

```
python -m dis hello.py
```

```
  0           0 RESUME                   0

  1           2 LOAD_CONST               0 (<code object greeting at 0x102b1f5d0, file "hello.py", line 1>)
              4 MAKE_FUNCTION            0
              6 STORE_NAME               0 (greeting)

  4           8 PUSH_NULL
             10 LOAD_NAME                1 (print)
             12 PUSH_NULL
             14 LOAD_NAME                0 (greeting)
             16 LOAD_CONST               1 ('Kitty')
             18 CALL                     1
             26 CALL                     1
             34 POP_TOP
             36 RETURN_CONST             2 (None)

Disassembly of <code object greeting at 0x102b1f5d0, file "hello.py", line 1>:
  1           0 RESUME                   0

  2           2 LOAD_CONST               1 ('Hello, ')
              4 LOAD_FAST                0 (name)
              6 FORMAT_VALUE             0
              8 LOAD_CONST               2 ('!')
             10 BUILD_STRING             3
             12 RETURN_VALUE
```

行號

# ByteCode

# 原始碼追追追！

# Why?

# 我覺得很好玩！

# 看不懂 C 語言怎麼辦？

version → 3.12.6

# GDB / LLDB

```
٩(´Ꙋ`*)و🍟🧋❯ lldb ./python.exe hello.py
(lldb) target create "./python.exe"
Current executable set to '/Users/kaochenlong/projects/
sources/python/cpython/python.exe' (arm64).
(lldb) settings set -- target.run-args  "hello.py"
(lldb) breakpoint set --name main
Breakpoint 1: 13 locations.
(lldb)
```

中斷點

```
٩(´Ω`*)و🍟🧋❯ (lldb) run
Process 80499 launched: '/Users/kaochenlong/projects/sources/python/
cpython/python.exe' (arm64)
Process 80499 stopped
* thread #1, queue = 'com.apple.main-thread', stop reason = breakpoint
1.1
    frame #0: 0x0000000100003d1c python.exe`main(argc=2,
argv=0x000000016fdfe758) at python.c:15:12 [opt]
   12   int
   13   main(int argc, char **argv)
   14   {
-> 15       return Py_BytesMain(argc, argv);
   16   }
   17   #endif
Target 0: (python.exe) stopped.
warning: python.exe was compiled with optimization - stepping may behave
oddly; variables may not be available.
(lldb)
```

進入點

# Py_BytesMain() → 入口

```c
int
Py_BytesMain(int argc, char **argv)
{
    _PyArgv args = {
        .argc = argc,
        .use_bytes_argv = 1,
        .bytes_argv = argv,
        .wchar_argv = NULL};
    return pymain_main(&args);
}
```

檔名：Modules/main.c

```c
static int
pymain_main(_PyArgv *args)
{
    PyStatus status = pymain_init(args);
    if (_PyStatus_IS_EXIT(status)) {
        pymain_free();
        return status.exitcode;
    }
    if (_PyStatus_EXCEPTION(status)) {
        pymain_exit_error(status);
    }

    return Py_RunMain();
}
```

檔名：Modules/main.c

```c
int
Py_RunMain(void)
{
    int exitcode = 0;

    pymain_run_python(&exitcode);

    if (Py_FinalizeEx() < 0) {
        /* Value unlikely to be confused with a non-error exit status or
           other special meaning */
        exitcode = 120;
    }

    pymain_free();

    if (_PyRuntime.signals.unhandled_keyboard_interrupt) {
        exitcode = exit_sigint();
    }

    return exitcode;
}
```

檔名：Modules/main.c

```c
static void
pymain_run_python(int *exitcode)
{
    // ... 略 ...

    if (config->run_command) {
        *exitcode = pymain_run_command(config->run_command);
    }
    else if (config->run_module) {
        *exitcode = pymain_run_module(config->run_module, 1);
    }
    else if (main_importer_path != NULL) {
        *exitcode = pymain_run_module(L"__main__", 0);
    }
    else if (config->run_filename != NULL) {
        *exitcode = pymain_run_file(config);
    }
    else{
        *exitcode = pymain_run_stdin(config);
    }

    // ... 略 ...
}
```

python -c "print('hello')"

python -m module_name

python hello.py

檔名：Modules/main.c

```c
static int
pymain_run_file(const PyConfig *config)
{
    // ... 略 ...

    int res = pymain_run_file_obj(program_name, filename,
                                  config->skip_source_first_line);
    Py_DECREF(filename);
    Py_DECREF(program_name);
    return res;
}
```

檔名：Modules/main.c

```c
static int
pymain_run_file_obj(PyObject *program_name, PyObject *filename,
                    int skip_source_first_line)
{
    // ... 略 ...
    FILE *fp = _Py_fopen_obj(filename, "rb");

    // ... 略 ...

    PyCompilerFlags cf = _PyCompilerFlags_INIT;
    int run = _PyRun_AnyFileObject(fp, filename, 1, &cf);
    return (run != 0);
}
```

檔名：Modules/main.c

```c
int
_PyRun_AnyFileObject(FILE *fp, PyObject *filename, int closeit,
                     PyCompilerFlags *flags)
{
    // ... 略 ...
    int res;
    if (_Py_FdIsInteractive(fp, filename)) {
        res = _PyRun_InteractiveLoopObject(fp, filename, flags);
        if (closeit) {
            fclose(fp);
        }
    }
    else {
        res = _PyRun_SimpleFileObject(fp, filename, closeit, flags);
    }

    // ... 略 ...
}
```

檔名：Modules/main.c

```c
int
_PyRun_SimpleFileObject(FILE *fp, PyObject *filename, int closeit,
                        PyCompilerFlags *f
{
    // ... 略 ...
    m = PyImport_AddModule("__main__");

    // ... 略 ...
    int pyc = maybe_pyc_file(fp, filename, closeit);

    // ... 略 ...
    if (pyc) {
        FILE *pyc_fp;
        // ... 略 ...

        v = run_pyc_file(pyc_fp, d, d, flags);
    } else {
        // ... 略 ...
        v = pyrun_file(fp, filename, Py_file_input, d, d,
                       closeit, flags);
    }
    // ... 略 ...
}
```

__main__

?

檔名：Python/pythonrun.c

```c
static PyObject *
pyrun_file(FILE *fp, PyObject *filename, int start, PyObject *globals,
           PyObject *locals, int closeit, PyCompilerFlags *flags)
{
    // ... 略 ...
    PyArena *arena = _PyArena_New();

    // ... 略 ...
    mod_ty mod;
    mod = _PyParser_ASTFromFile(fp, filename, NULL, start, NULL, NULL,
                                flags, NULL, arena);

    // ... 略 ...
}
```

檔名：Python/pythonrun.c

Plain Text

→ Tokenization

→ AST (Abstract Syntax Tree)

→ ByteCode

→ Virtual Machine

```c
static PyObject *
run_mod(mod_ty mod, PyObject *filename, PyObject *globals, PyObject *locals,
            PyCompilerFlags *flags, PyArena *arena)
{
    PyThreadState *tstate = _PyThreadState_GET();
    PyCodeObject *co = _PyAST_Compile(mod, filename, flags, -1, arena);

    // ... 略 ...

    PyObject *v = run_eval_code_obj(tstate, co, globals, locals);
    Py_DECREF(co);
    return v;
}
```

檔名：Python/pythonrun.c

# Code Object ?

# ByteCode 就儲存在 Code Object 裡

# .pyc 檔？

hello.py → hello.cpython-312.pyc

# 把 ByteCode 快取成 .pyc 檔案

Plain Text

→ Tokenization

→ AST (Abstract Syntax Tree)

→ ByteCode

→ Virtual Machine

# 啟動！

```
static PyObject *
run_eval_code_obj(PyThreadState *tstate, PyCodeObject *co, PyObject *globals,
PyObject *locals)
{
    PyObject *v;
    _PyRuntime.signals.unhandled_keyboard_interrupt = 0;

    // ... 略 ...

    v = PyEval_EvalCode((PyObject*)co, globals, locals);
    if (!v && _PyErr_Occurred(tstate) == PyExc_KeyboardInterrupt) {
        _PyRuntime.signals.unhandled_keyboard_interrupt = 1;
    }
    return v;
}
```

檔名：Python/pythonrun.c

Plain Text

→ Tokenization

→ AST (Abstract Syntax Tree)

→ ByteCode

→ Virtual Machine

# 關於 .pyc 檔

# 不一定需要 .py，有 .pyc 就能動了！

# ByteCode

# maybe_pyc_file()
# 可能有？在有跟沒有之間？

```c
int
_PyRun_SimpleFileObject(FILE *fp, PyObject *filename, int closeit,
                        PyCompilerFlags *flags)
{
    // ... 略 ...
    m = PyImport_AddModule("__main__");

    // ... 略 ...
    int pyc = maybe_pyc_file(fp, filename, closeit);

    // ... 略 ...
    if (pyc) {
        FILE *pyc_fp;
        // ... 略 ...

        v = run_pyc_file(pyc_fp, d, d, flags);
    } else {
        // ... 略 ...
        v = pyrun_file(fp, filename, Py_file_input, d, d,
                       closeit, flags);
    }
    // ... 略 ...
}
```

?

檔名：Python/pythonrun.c

```c
static int
maybe_pyc_file(FILE *fp, PyObject *filename, int closeit)
{
    PyObject *ext = PyUnicode_FromString(".pyc");
    if (ext == NULL) {
        return -1;
    }

    // ... 略 ...

    /* Read only two bytes of the magic. If the file was opened in
       text mode, the bytes 3 and 4 of the magic (\r\n) might not
       be read as they are on disk. */
    unsigned int halfmagic = PyImport_GetMagicNumber() & 0xFFFF;
    unsigned char buf[2];

    // ... 略 ...
}
```

檔名：Python/pythonrun.c

```
long
PyImport_GetMagicNumber(void)
{
    long res;
    PyInterpreterState *interp = _PyInterpreterState_GET();
    PyObject *external, *pyc_magic;

    external = PyObject_GetAttrString(IMPORTLIB(interp), "_bootstrap_external");
    if (external == NULL)
        return -1;
    pyc_magic = PyObject_GetAttrString(external, "_RAW_MAGIC_NUMBER");
    Py_DECREF(external);
    if (pyc_magic == NULL)
        return -1;
    res = PyLong_AsLong(pyc_magic);
    Py_DECREF(pyc_magic);
    return res;
}
```

檔名：Python/import.c

```
MAGIC_NUMBER = (3531).to_bytes(2, 'little') + b'\r\n'
_RAW_MAGIC_NUMBER = int.from_bytes(MAGIC_NUMBER, 'little')
```

檔名：Lib/importlib/_bootstrap_external.py

3531 ?

# 魔術數字？Magic！

```
# Known values:
#  Python 1.5:   20121
#  Python 1.5.1: 20121
#     Python 1.5.2: 20121
#     Python 1.6:   50428
#     Python 2.0:   50823
#     ... 略 ...
#     Python 2.7a0  62211 (introduce MAP_ADD and SET_ADD)
#     Python 3000:   3000
#                    3010 (removed UNARY_CONVERT)
#     ... 略 ...
#     Python 3.12b1 3530 (Shrink the LOAD_SUPER_ATTR caches)
#     Python 3.12b1 3531 (Add PEP 695 changes)

#     Python 3.13 will start with 3550
```

檔名：Lib/importlib/_bootstrap_external.py

# 每個版本都有一個對應的「魔術數字」

```
# Python 3.11.9
>>> from importlib._bootstrap_external import MAGIC_NUMBER
>>> MAGIC_NUMBER
b'\xa7\r\r\n'

# Python 3.12.6
>>> from importlib._bootstrap_external import MAGIC_NUMBER
>>> MAGIC_NUMBER
b'\xcb\r\r\n'
```

# 不同版本 Python 編譯出來的 Bytecode 是不相容的

```
٩(´ᗜ`*)و🍟🧋❯ python __pycache__/hello.cpython-312.pyc
RuntimeError: Bad magic number in .pyc file
```

# Mess and terrible hacks ?

```
/* Mess:  In case of -x, the stream is NOT at its start now,
   and ungetc() was used to push back the first newline,
   which makes the current stream position formally undefined,
   and a x-platform nightmare.
   Unfortunately, we have no direct way to know whether -x
   was specified.  So we use a terrible hack:  if the current
   stream position is not 0, we assume -x was specified, and
   give up.  Bug 132850 on SourceForge spells out the
   hopelessness of trying anything else (fseek and ftell
   don't work predictably x-platform for text-mode files).
*/
```

檔名：Python/pythonrun.c

# Windows、Linux、Mac 換行方式不同

# python -x

# 解開 .pyc！

```python
with open("__pycache__/hello.cpython-312.pyc", "rb") as f:
    print(f.read())
```

檔名：pyc_demo.py

```
٩(´Ω`*)و🍟🧋❯ ./python.exe pyc_demo.py
b'\xcb\r\r\n\x00\x00\x00\x00\xce\xac\xbchK\x00\x00\x00\xe3\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x05\x00\x00\x00\x00\x00\x00\x00\xf3&\x00\x00\
x00\x97\x00d\x00\x84\x00Z\x00\x02\x00e\x01\x02\x00e\x00d\x01\xab\x01\x00\x00
\x00\x00\x00\x00\xab\x01\x00\x00\x00\x00\x00\x00\x01\x00y\x02)\x03c\x01\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x03\x00\x00\x00\x03\x00\x00\x00\xf3\
x0e\x00\x00\x00\x97\x00d\x01|\x00\x9b\x00d\x02\x9d\x03S\x00)\x03Nz\x07Hello,
\xfa\x01!\xa9\x00)\x01\xda\x04names\x01\x00\x00\x00
\xfa\x08hello.py\xda\x08greetingr\x07\x00\x00\x00\x01\x00\x00\x00s\x11\x00\x
00\x00\x80\x00\xd8\r\x14\x90T\x90F\x98!
\xd0\x0b\x1c\xd0\x04\x1c\xf3\x00\x00\x00\x00\xda\x05KittyN)
\x02r\x07\x00\x00\x00\xda\x05printr\x04\x00\x00\x00r\x08\x00\x00\x00r\x06\x0
0\x00\x00\xfa\x08<module>r\x0b\x00\x00\x00\x01\x00\x00\x00s\x19\x00\x00\x00\
xf0\x03\x01\x01\x01\xf2\x02\x01\x01\x1d\xf1\x06\x00\x01\x06\x81h\x88w\xd3\x0
6\x17\xd5\x00\x18r\x08\x00\x00\x00'
```

```python
import marshal

with open("__pycache__/hello.cpython-312.pyc", "rb") as f:
    f.read(16)

    content = marshal.load(f)
    code_object = content.co_code

    print(type(content))
    print(code_object)
    print(list(code_object))
```

魔術數字

檔名：pyc_demo2.py

```
٩(´Ω`*)و🍟🧋❯ ./python.exe pyc_demo2.py
<class 'code'>

b'\x97\x00d\x00\x84\x00Z\x00\x02\x00e\x01\x02\x00e\x00d\x01\xab\
x01\x00\x00\x00\x00\x00\x00\xab\x01\x00\x00\x00\x00\x00\x00\x01\
x00y\x02'

[151, 0, 100, 0, 132, 0, 90, 0, 2, 0, 101, 1, 2, 0, 101, 0, 100,
1, 171, 1, 0, 0, 0, 0, 0, 0, 171, 1, 0, 0, 0, 0, 0, 0, 1, 0,
121, 2]
```

```
#define CACHE                                   0
#define POP_TOP                                 1
#define PUSH_NULL                               2
#define INTERPRETER_EXIT                        3
#define END_FOR                                 4
// ... 略 ...
#define LOAD_CONST                            100
#define LOAD_NAME                             101
#define BUILD_TUPLE                           102
#define BUILD_LIST                            103
// ... 略 ...
#define COPY_FREE_VARS                        149
#define YIELD_VALUE                           150
#define RESUME                                151
// ... 略 ...
```

檔名：Include/opcode.h

```
>>> import dis
>>> ops = [151, 0, 100, 0, 132, 0, 90, 0, 2, 0, 101, 1, 2, 0, 101, 0, 100,
1, 171, 1, 0, 0, 0, 0, 0, 0, 171, 1, 0, 0, 0, 0, 0, 0, 1, 0, 121, 2]
>>> for op in ops:
...   print(dis.opname[op])
...
RESUME
CACHE
LOAD_CONST
CACHE
MAKE_FUNCTION
CACHE
STORE_NAMECACHE
CACHE
POP_TOP
CACHE
RETURN_CONST
PUSH_NULL
```

# .pyc 裡面的 ByteCode = 一連串的 opcode 組成的 ByteArray

# 小結

# 覺得好玩嗎？好玩很重要！

# 還有更多...

# 為你自己讀 CPython 原始碼



今天的內容

# （非）工商服務

# iThome 鐵人賽

## 主題：為你自己學 Gemini

Google
Gemini

# iThome 鐵人賽

## 主題：為你自己學 n8n

五倍學院

# 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