

Reactor

開始你學習Python的第一步



Map



个人介绍



Kinfey Lo – (盧建暉)

Microsoft Cloud Advocate

前微軟MVP、Xamarin MVP和微軟RD，擁有超過10年的雲解決、人工智能和流動應用經驗，為教育、金融和醫療提供應用。在微軟，為技術人員和不同行業宣講技術和相關應用場景。

Love Coding(Python , C# , TypeScript , Swift , Rust , Go)

專注於人工智能，雲原生，流動平台移動開發

Github : <https://github.com/kinfey>

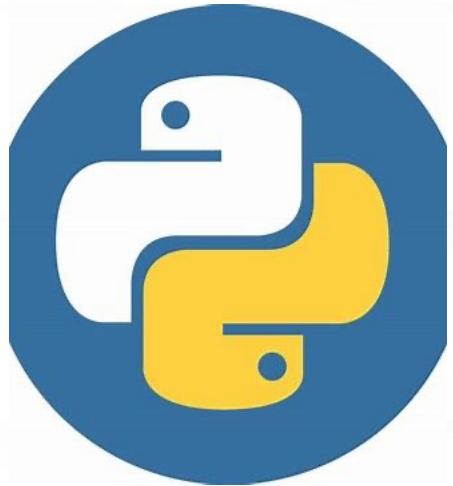
Email : kinfeylo@microsoft.com **Blog :** <https://dev.to/kinfey>

Twitter : @Ljh8304

Python 係?



Python 是什麼？



- **Python 是一種解釋型語言**

這意味著開發過程中沒有了編譯這個環節。類似於PHP和Perl語言。

- **Python 是互動式語言**

這意味著，您可以在一個 Python 提示符 >>> 後直接執行代碼。

- **Python 是物件導向語言**

這意味著Python支援物件導向的風格或代碼封裝在物件的程式設計技術。

- **Python 是初學者的語言**

Python 對初級程式師而言，是一種偉大的語言，它支援廣泛的應用程式開發，從簡單的文字處理到 WWW 流覽器再到遊戲。

Python 主要能做什麼？

網路爬蟲 (Web Spider)

從互聯網採集資料的程式腳本

資料分析 (Data Analysis)

收集、建模和分析資料以提取支援決策的見解的過程。

人工智慧 (Artificial Intelligence)

它是研究、開發用於模擬、延伸和擴展人的智慧的理論、方法、技術及應用系統的一門新的技術科學。

Python 如何安裝？

<https://www.python.org/>

Download the latest version for Mac OS X

[Download Python 3.9.6](#)

Looking for Python with a different OS? Python for [Windows](#),
[Linux/UNIX](#), [Mac OS X](#), [Other](#)

Want to help test development versions of Python? [Prereleases](#),
[Docker images](#)

Looking for Python 2.7? See below for specific releases



Python 是一個跨平臺語言，支援Windows/Linux/MacOS 安裝

Python 安裝的一些問題

• Python 版本選擇

Python版本有很多，現在存在於坊間的版本就有Python 2.x 和Python 3.x

• Python 多版本問題

建議使用 `pyenv` 或者 `conda` 去安裝管理Python的多版本問題，你可以依據不同專案切換你的Python版本

• Python 支持arm嗎？

Python支持arm的環境，你可以在IoT設備安裝Python環境，也可以安裝在最新的Apple Silicon設備上

關於 Python 的工具-pip和conda

• pip 介紹

pip 是一個安裝和管理 Python 包的工具，python安裝包的工具有easy_install, setuptools, pip, distribute。使用這些工具都能下載並安裝 flask / tensorflow / numpy 等協力廠商工具。, 而pip是easy_install的替代品。

• conda 介紹

Anaconda 是一個用於科學計算的 Python 發行版本，支持 Linux, Mac, Windows, 包含了眾多流行的科學計算、資料分析的 Python 包。其使用conda系統進行包管理。

• conda vs pip

1. conda 在安裝一些依賴 C、C++的 Python 庫時特別方便於流暢。比如：Numpy、Pandas、等一些資料庫驅動，直接通過命令就可以安裝，不需要額外自行編譯、安裝 C 庫。
2. pip 的優點在於包的豐富程度，Conda 找不到的包，在 pip 就可以找到。 (簡而言之，conda install 找不到的庫，就用 pip install 安裝)



MICROSOFT



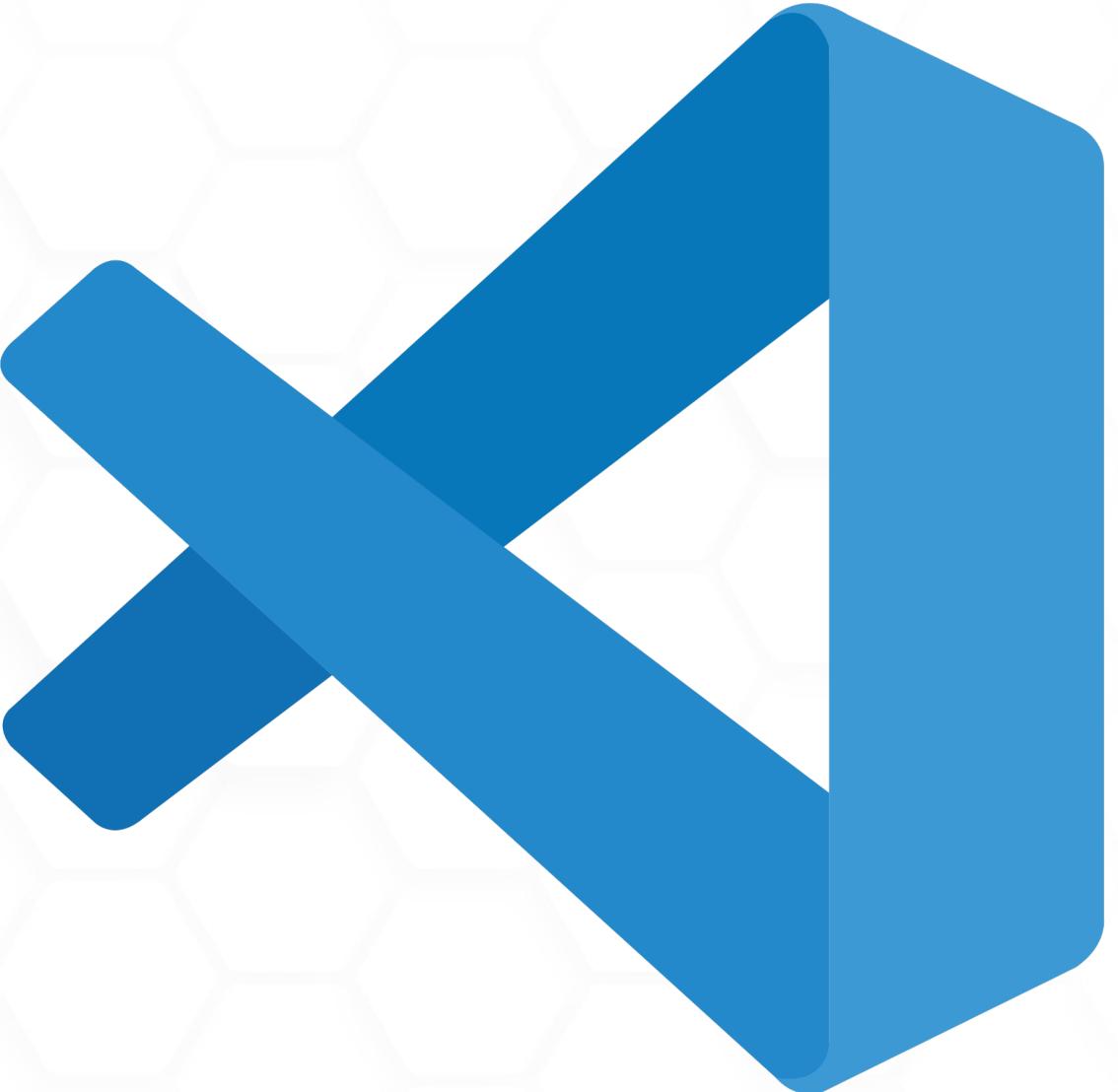
PYTHON

Visual Studio Code



Visual Studio Code

- a) 靜態代碼掃描（可以理解為代碼語法和格式錯誤提示，支援多種 linter）
- b) 智慧提示（自動補全，自動完成，包括了所在上下文的方法和變數）
- c) 自動縮進
- d) 代碼自動格式化
- e) 代碼重構（重命名，提取變數，提取方法，對import排序）
- f) 查看引用，代碼導航，查看簽名
- g) 完美的調試支持（通過SSH遠程調試，多執行緒，django，flask）
- h) 運行和調試單元測試
- i) 在python終端執行檔或代碼
- j) Snippets代碼片段





Python

ms-python.python

Microsoft

38,986,048

★★★★★

Repository

License

v2021.6.944021595

IntelliSense (Pylance), Linting, Debugging (multi-threaded, remote), Jupyter Notebooks, code formatting, refactoring, unit tests, and more.

Disable

Uninstall



This extension is enabled globally.

[Details](#) [Feature Contributions](#) [Changelog](#) [Extension Pack](#)

Python extension for Visual Studio Code

A [Visual Studio Code extension](#) with rich support for the [Python language](#) (for all [actively supported versions](#) of the language: >=3.6), including features such as IntelliSense (Pylance), linting, debugging, code navigation, code formatting, refactoring, variable explorer, test explorer, and more!

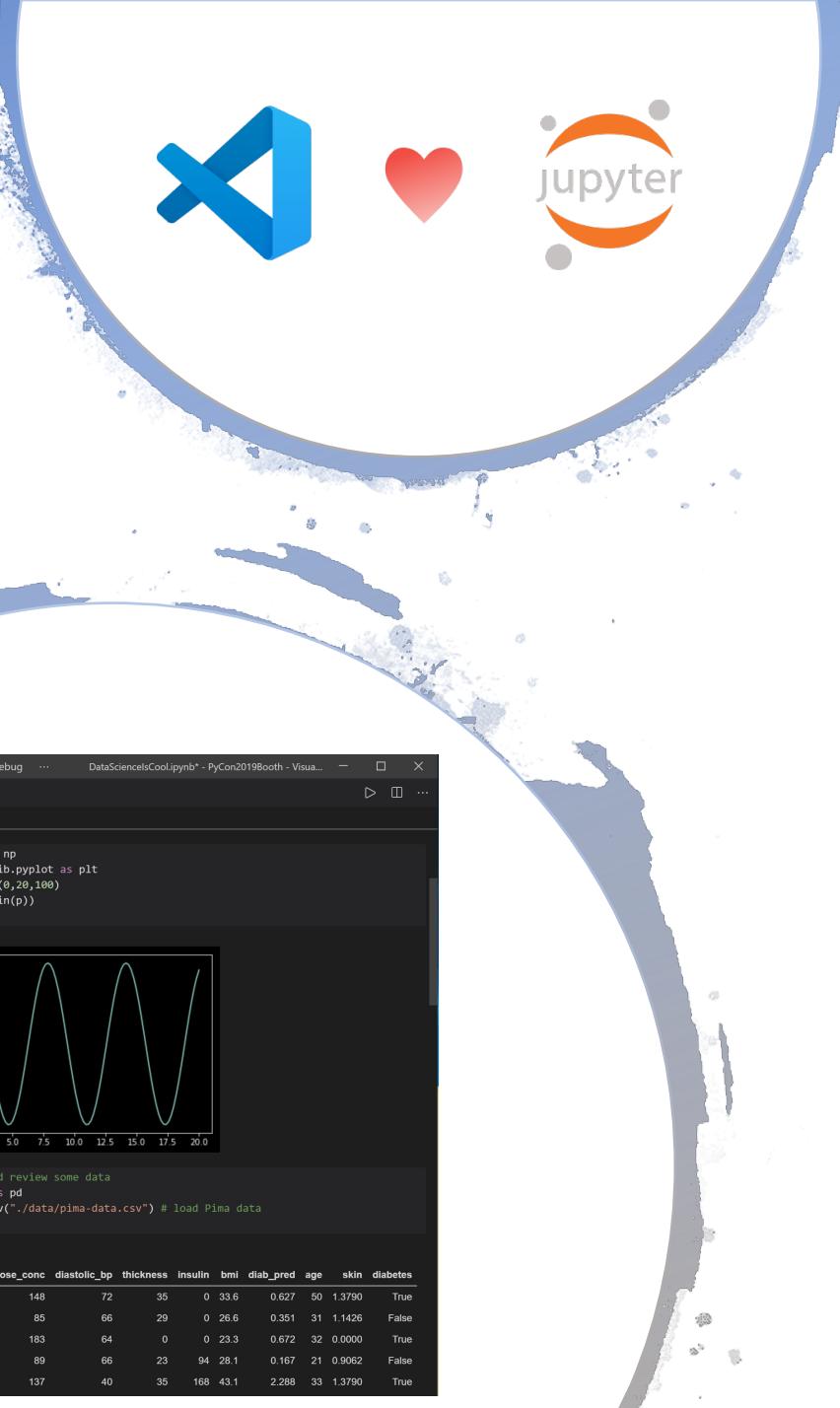
Installed extensions

The Python extension will automatically install the [Pylance](#) and [Jupyter](#) extensions to give you the best experience when working with Python files and Jupyter notebooks. However, Pylance is an optional dependency, meaning the Python extension will remain fully functional if it fails to be installed. You can also [uninstall](#) it at the expense of some features if you're using a different language server.

Extensions installed through the marketplace are subject to the [Marketplace Terms of Use](#).

Quick start

- [Step 1. Install a supported version of Python on your system](#) (note: that the system install of Python on macOS is not supported).
- [Step 2. Install the Python extension for Visual Studio Code](#).
- [Step 3. Open or create a Python file and start coding!](#)



```
[8] import numpy as np
> import matplotlib.pyplot as plt
p = np.linspace(0,20,100)
plt.plot(p,np.sin(p))
plt.show()

[4] # Let's load and review some data
> import pandas as pd
df = pd.read_csv("./data/pima-data.csv") # load Pima data
df.head(5)

  num_preg  glucose_conc  diastolic_bp  thickness  insulin  bmi  diag_pred  age  skin  diabetes
0       6          148           72         35       0  33.6   0.627    50  1.3790    True
1       1            85           66         29       0  26.6   0.351    31  1.1426   False
2       8           183           64           0       0  23.3   0.672    32  0.0000    True
3       1            89           66         23       94  28.1   0.167    21  0.9062   False
4       0           137           40         35      168  43.1   2.288    33  1.3790    True
```

Data science with Python in VS Code

- Jupyter Notebooks 和 VS Code整合
- VS Code 直接支援 ipynb 文檔
- Jupyter Notebooks 轉換Python 腳本
- 支援本地或遠端調試Jupyter伺服器
- Visualize data frames and plots
- 集成IPython 控制台
- 變數流覽器
- 智慧提示
- Cell單步調試

Python Language



Python 包, 庫, 模組

module: 就是.py檔，裡面定義了一些函數和變數，需要的時候就可以導入這些模組。

package: 在模組之上的概念，為了方便管理而將檔進行打包。包目錄下第一個檔便是 `_init_.py`，然後是一些模組檔和子目錄，假如子目錄中也有 `_init_.py`，那麼它就是這個包的子包了。

常見的包結構：

```
package_a
|__ __init__.py
|__ module_a1.py
|__ module_a2.py
```

library: 具有相關功能模組的集合。這也是Python的一大特色之一，即具有強大的標準庫、協力廠商庫以及自訂模組。

Python import

python裡有些內置函數，不需要導入模組，就可以直接使用，例如abs()

導入模組與包都是通過import來導入的，即import 模組名或者包名

有些包裡有內置函數，導入包名之後，可以直接通過包名.函數名來調用函數

在模組名或者包名過長時，為了後面的程式編寫，可以給其取別名，即 import 模組名/包名 as 別名

我們知道包是模組的合集，如果使用到了包裡的某些模組，一般情況下需要通過包名.模組名.函數名 來調用相應的函數，為了方便編寫程式，可以通過 from 包名 import 模組名 直接導入模組，這樣就不需要再寫包名了

如果包裡還有子包的話，可以通過 from 包名.子包名 import 模組名，導入模組，同樣為了方便程式編寫
總的來說如何使用一個包，與其自身的結構有很大的關係，具體使用還是需要參考具體的包結構。

Python 注釋

1、單行注釋（行注釋）

Python中使用#表示單行注釋。單行注釋可以作為單獨的一行放在被注釋代碼行之上，也可以放在語句或運算式之後。

```
# 單行注釋
```

2、多行注釋（塊注釋）

當注釋內容過多，導致一行無法顯示時，就可以使用多行注釋。Python中使用三個單引號或三個雙引號表示多行注釋。

```
"""
這是使用三個單引號的多行注釋
""
```

```
"""
這是使用三個雙引號的多行注釋
"""
```

Python 類型

Python 中的變數不需要聲明。每個變數在使用前都必須賦值，變數賦值以後該變數才會被創建。在 Python 中，變數就是變數，它沒有類型，我們所說的"類型"是變數所指的記憶體中物件的類型。

```
number = 100      # 整型變數  
money  = 1000.0    # 浮點型變數  
str    = 'reactor' # 字串
```

Python3 中有六個標準的資料類型：Number（數字）, String（字串）, List（列表）, Tuple（元組）, Set（集合）, Dictionary（字典）

Python3 的六個標準資料類型中：

不可變數據（3 個）：Number（數字）、String（字串）、Tuple（元組）；

可變數據（3 個）：List（列表）、Dictionary（字典）、Set（集合）。

Python List,tuple,Dictionary

列表(List)

用於存儲任意數目、任意類型的資料集合。

列表是內置可變序列，是包含多個元素的有序連續的記憶體空間。列表定義的標準語法格式：

language = ['Python','C++','.NET Core','Java'] 其中， Python, C++, .NET Core, Java 這些稱為：清單 language的元素。

清單中的元素可以各不相同，可以是任意類型。比如：**catalog = [88,2.0,'arsenal',True]**

元組(tuple)

清單屬於可變序列，可以任意修改清單中的元素。元組屬於不可變序列，不能修改元組中的元素。因此，元組沒有增加元素、修改元素、刪除元素相關的方法。

a = (10,20,30)或者a = 10,20,30

字典(Dictionary)

字典是“鍵值對”的無序可變序列，字典中的每個元素都是一個“鍵值對”，包含：“鍵物件”和“值物件”。可以通過“鍵物件”實現快速獲取、刪除、更新對應的“值對象”。清單中我們通過“下標數位”找到對應的物件。字典中通過“鍵物件”找到對應的“值對象”。“鍵”是任意的不可變資料，比如：整數、浮點數、字串、元組。但是：清單、字典、集合這些可變物件，不能作為“鍵”。並且“鍵”不可重複。“值”可以是任意的資料，並且可重複。

一個典型的字典的定義方式：**employee_info = {'name':'kinfey','age':'28', 'mvp':'AI'}**

Python List vs Dictionary

List優點：

1. 具有異構性，可以包含不同種類、任意類型的物件，可以嵌套清單
2. 具有有序性，清單裡裝的元素都是有順序的，可以按照位置序號獲取單個元素，也可以用分片的方法來進行多個連續元素的獲取
3. 清單裡裝的元素都是有順序的，可以按照位置序號獲取單個元素，也可以用分片的方法來進行多個連續元素的獲取

Dictionary優點：

字典是無序，通過鍵值對一一對應

Python class的定義

類把資料與功能綁定在一起。創建新類就是創建新的物件類型，從而創建該類型的新 實例 。類實例具有多種保持自身狀態的屬性。類實例還支援（由類定義的）修改自身狀態的方法。

```
class ClassName:  
    <statement-1>  
    ...  
    <statement-N>
```

```
class MyClass:  
    """A simple example class"""  
    i = 12345  
    def f(self):  
        return 'hello world'
```

```
class Complex:  
    def __init__(self, realpart, imagpart):  
        self.r = realpart  
        self.i = imagpart
```

```
x = Complex(3.0, -4.5)  
x.r, x.i (3.0, -4.5)
```

Python 類繼承和多重繼承

繼承

```
class DerivedClassName(BaseClassName):  
    <statement-1>  
    ...  
    <statement-N>
```

多重繼承

```
class DerivedClassName(Base1, Base2, Base3):  
    <statement-1>  
    ...  
    <statement-N>
```

Python 一些常用的library

```
import json

data = [ { 'a' : 1, 'b' : 2, 'c' : 3, 'd' : 4, 'e' : 5 } ]

data2 = json.dumps(data)
print(data2)

data = [ { 'a' : 1, 'b' : 2, 'c' : 3, 'd' : 4, 'e' : 5 } ]

data2 = json.dumps({ 'a': 'Runoob', 'b': 7}, sort_keys=True, indent=4, separators=(',', ': '))
print(data2)
```

```
import socket

# 導入 socket 模組
s = socket.socket()

# 創建 socket 對象
host = socket.gethostname()

# 獲取本地主機名稱 port = 12345

# 設置埠
s.bind((host, port))

# 綁定埠 s.listen(5)

# 等待用戶端連接
while True:
    c,addr = s.accept()

# 建立用戶端連接

print '連接位址: ', addr

c.send('歡迎訪問菜鳥教程! ')

c.close() # 關閉連接
```

Python 語法規範

The screenshot shows the left sidebar of the Google Python Style Guide. At the top, it says "Google 开源项目风格指南 latest". Below that is a search bar labeled "Search docs". Under "Google 开源项目风格指南 (中文版)", there are links for "C++ 风格指南 - 内容目录" and "Objective-C 风格指南 - 内容目录". The main menu has a section "Python 风格指南 - 内容目录" which is expanded, showing:

- 扉页
- 背景
- Python语言规范
 - Lint
 - 导入
 - 包
 - 异常
 - 全局变量
 - 嵌套/局部/内部类或函数
 - 列表推导(List Comprehensions)
 - 默认迭代器和操作符
 - 生成器
 - Lambda函数
 - 条件表达式
 - 默认参数值
 - 属性(properties)

At the bottom of the sidebar, there are "Read the Docs" and "v: latest" buttons.

Docs » Python 风格指南 - 内容目录 » Python语言规范

Edit on GitHub

Python语言规范

Lint

Tip

对你的代码运行pylint

定义:

pylint是一个在Python源代码中查找bug的工具. 对于C和C++这样的不那么动态的(译者注: 原文是less dynamic)语言, 这些bug通常由编译器来捕获. 由于Python的动态特性, 有些警告可能不对. 不过伪告警应该很少.

优点:

可以捕获容易忽视的错误, 例如输入错误, 使用未赋值的变量等.

缺点:

pylint不完美. 要利用其优势, 我们有时候需要:
a) 围绕着它来写代码
b) 抑制其告警
c) 改进它, 或者
d) 忽略它.

结论:

确保对你的代码运行pylint. 抑制不准确的警告, 以便能够将其他警告暴露出来.

你可以通过设置一个行注释来抑制警告. 例如:

```
dict = 'something awful' # Bad Idea... pylint: disable=redefined-builtin
```

<https://google.github.io/styleguide/pyguide.html>

實例三部曲



例子學習

01

控制台程式設計



02

Flask後端程式
設計

101010
010101
101010

03

AI人工智慧



考評一下



1. Python 可以通過什麼工具進行多版本管理(多選題)

- A. Conda
- B. pip
- C. Nuget
- D. wget

2. VSCode 的 Python外掛程式係米支持Jupyter Notebook(單選題)

- A. 是
- B. 否

3. 以下邊個語法係正確的? (多選題)

A. int num = 0

B. info = [a,1.0, 'hi']

C. num: Int =0

D. a = (1, 3)

4. Python 能做什麼(多選題)

- A. 後端開發
- B. 資料分析
- C. 人工智能
- D. 網路爬蟲

5. 以下邊個是Python Dictionary的描述是正確的

- A. “鍵值對”的無序可變序列
- B. “鍵值對”的有序不可變序列
- C. “鍵值對”的有序可變序列
- D. “鍵值對”的無序不可變序列

小結



Meetup link: [meetup.com/Microsoft-Reactor-London/](https://www.meetup.com/Microsoft-Reactor-London/)

MS Learn Module 推薦



<https://aka.ms/HKPythonLearn001>



Reactor

Thank You!

Q&A

m