

# 【Python】【\_\_init\_\_.py】，【\_\_all\_\_】說明

## Python 中的 `__init__.py` 與 `__all__` 用途

在 Python 中，`__init__.py` 是專門用來初始化套件的重要檔案，而 `__all__` 是用來控制模組或套件的對外公開接口。以下將針對這兩者的用途及使用方式進行完整說明。

## `__init__.py` 的用途

### 1. 標記目錄為 Python 套件

在 Python 3.3 之前，`__init__.py` 是標示一個目錄為套件的必要檔案。沒有這個檔案，目錄無法被視為套件。雖然 Python 3.3 之後允許隱式套件（目錄中沒有 `__init__.py` 也能被視為套件），但仍建議在目錄中保留 `__init__.py`，以明確表示該目錄是套件。

### 2. 套件初始化

`__init__.py` 的內容會在匯入套件時執行，因此你可以在這裡添加初始化代碼，例如：

- 設定全域變數或環境配置。
- 匯入套件內的模組或子模組，簡化使用者匯入的流程。
- 處理套件的必要初始化邏輯。

範例：

```
# 檔案結構
# mypackage/
# |— __init__.py
# |— module1.py
# |— module2.py

# __init__.py
print("Initializing mypackage...")
from .module1 import func1
from .module2 import func2

# 匯入時會執行初始化代碼
import mypackage
# 執行結果: "Initializing mypackage..."
```

### 3. 控制對外公開接口

`__init__.py` 可以用來控制使用者可以從套件匯入的功能，例如：

- 只公開指定的模組或函式。
- 隱藏內部的模組，避免使用者直接操作。

範例：

```
# __init__.py
from .module1 import func1
from .module2 import func2

__all__ = ['func1', 'func2']
```

使用者只需匯入套件即可：

```
from mypackage import *
```

```
func1() # 正常執行  
func2() # 正常執行
```

## `__all__` 的用途

`__all__` 是 Python 中的特殊變數，用於定義模組或套件的對外公開接口，特別是在使用 `from module import *` 時，`__all__` 可以控制哪些成員可以被匯入。

### 1. 限制公開的內容

`__all__` 是一個列表，用來列出所有允許匯入的成員。如果未定義 `__all__`，則預設會匯入所有不以下劃線 `_` 開頭的公開成員。

範例：

```
# example.py  
def func1():  
    return "This is func1"  
  
def func2():  
    return "This is func2"  
  
def _private_func():  
    return "This is a private function"  
  
__all__ = ['func1'] # 只允許 func1 被匯入
```

在其他程式中使用：

```
from example import *  
  
print(func1()) # 正常執行  
print(func2()) # NameError: name 'func2' is not defined
```

### 2. 提升可讀性

使用 `__all__` 可以明確標示模組或套件的公共接口，讓開發者清楚哪些功能是穩定且推薦使用的。

### 3. 結合套件使用

當一個套件有多個模組時，可以在 `__init__.py` 中使用 `__all__` 控制整個套件的公共接口。

範例：

```
# 檔案結構  
# mypackage/  
# |— __init__.py  
# |— module1.py  
# |— module2.py  
  
# module1.py  
def func1():  
    return "Function 1"  
  
# module2.py  
def func2():  
    return "Function 2"  
  
# __init__.py
```

```
from .module1 import func1
from .module2 import func2

__all__ = ['func1'] # 只公開 func1
```

在使用者端：

```
from mypackage import *

print(func1()) # 正常執行
print(func2()) # NameError: name 'func2' is not defined
```

## 總結

1. `__init__.py` 的用途：
  - 標記目錄為 Python 套件。
  - 執行套件初始化邏輯，例如匯入模組或設定全域變數。
  - 控制套件的對外接口，組織套件結構，簡化使用者的匯入流程。
2. `__all__` 的用途：
  - 控制 `from module import *` 時匯入的內容，避免暴露內部實現細節。
  - 提升程式碼的可讀性，讓開發者清楚哪些功能是公開的。
  - 結合 `__init__.py` 使用，控制整個套件的公共接口。

## 最佳實踐建議

- 總是為套件添加 `__init__.py`，即使在 Python 3.3 之後不強制要求。
- 明確定義 `__all__`，只公開穩定且推薦使用的功能，隱藏內部實現細節。
- 避免使用 `from module import *`，除非必要，以提升程式碼的可讀性和穩定性。

這樣可以讓你的模組和套件更加結構化、易於維護，並避免不必要的命名衝突。

🕒 修訂版本 #1

★ 由 treeman 建立於 23 🕒@🕒🕒 2025 11:40:39

✍ 由 treeman 更新於 23 🕒@🕒🕒 2025 11:47:26