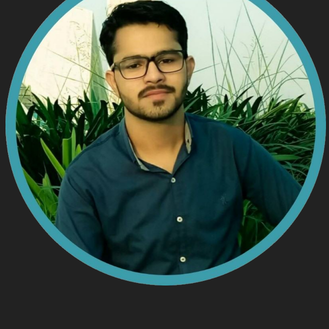
Mystery of

__init__.py



Ammar **Munir**

WHAT ?

The __init__.py file is used in Python to mark a directory as a package. A package is a way of organizing related modules into a single directory hierarchy. By including an __init__.py file in a directory, Python treats the directory as a package, allowing you to import modules from it.

BUT !

Without an __init__.py file, a directory is not considered a package in older versions of Python (before 3.3). However, starting from Python 3.3, implicit namespace packages were introduced, which allow directories to be recognized as packages without needing an __init__.py file.

Why we still use it ?

Ammar **Munir**

Here are the reasons why we still use

__init__.py in python3.3+

Initializing the Package

Controlling imports

Providing a convenient interface

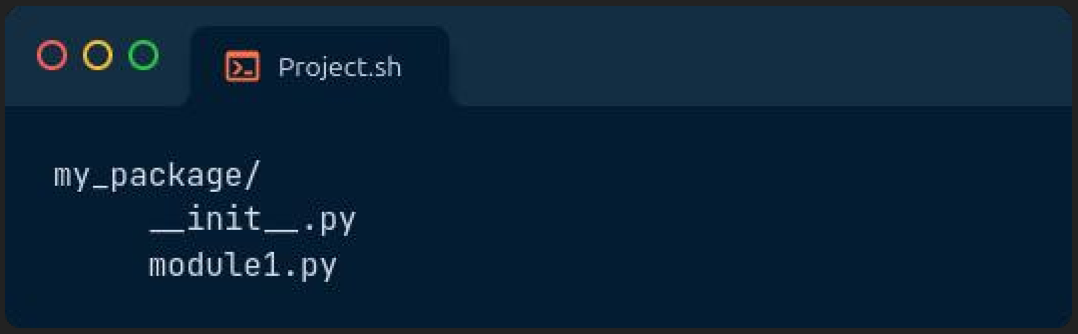
Ensuring compatibility and clarity

Ammar **Munir**

Initializing the Package

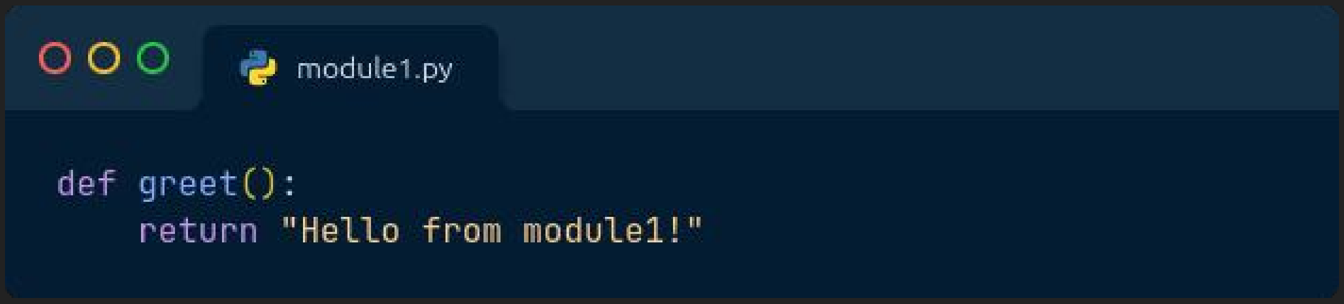
You can use __init__.py to initialize the package. This can include setting up package-wide variables, importing specific submodules, or executing package-level code.

Structure



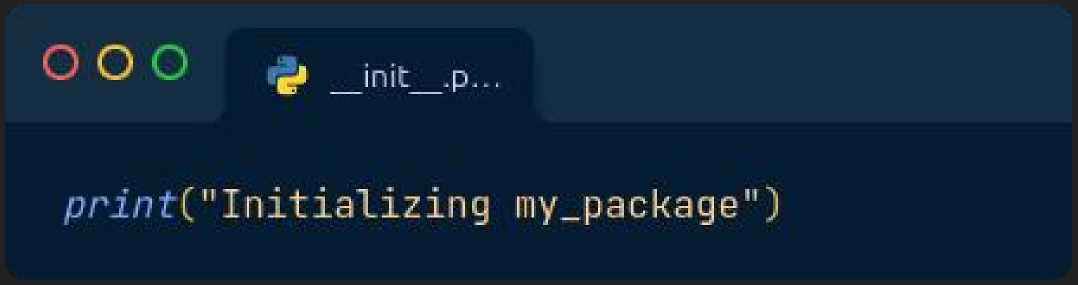
```
my_package/
    __init__.py
    module1.py
```

Module1



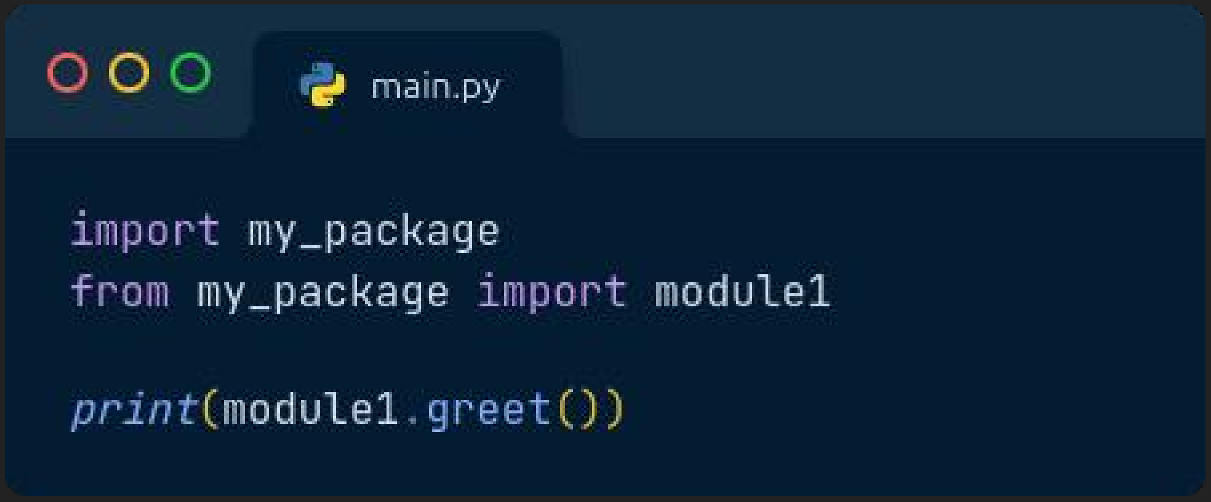
```python
def greet():
    return "Hello from module1!"
```

__init__.py



```python
print("Initializing my_package")
```

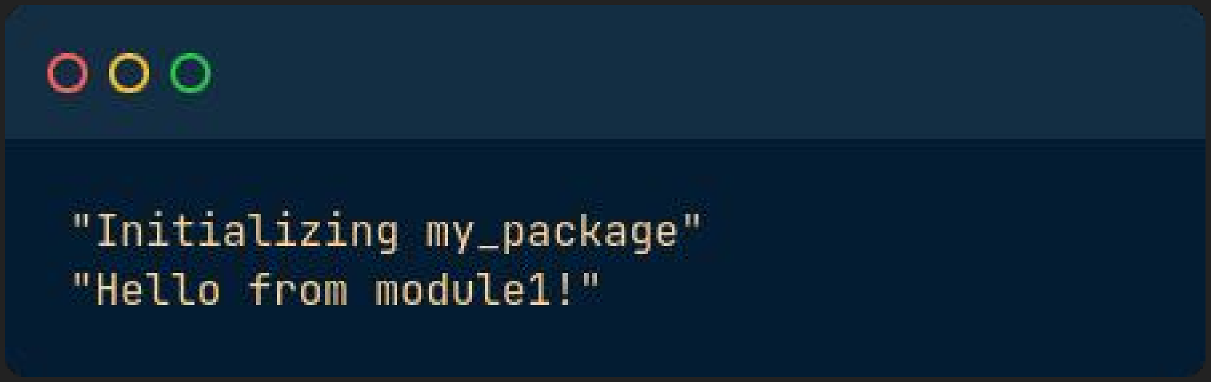
main.py



```python
import my_package
from my_package import module1

print(module1.greet())
```

output.py



```
"Initializing my_package"
"Hello from module1!"
```

Ammar **Munir**

Controlling imports

Structure

```
my_package/
    __init__.py
    module1.py
    module2.py
```

Module1

module1.py

```python
def func1():
    return "Function 1"
```

Module2

module2.py

```python
def func2():
    return "Function 2"
```

__init__.py

__init__.p...

```python
from .module1 import func1

__all__ = ['func1']
```

main.py

main.py

```python
from my_package import *

print(func1())  # Output: Function 1
# print(func2())  # This will raise an AttributeError
```

You can control what gets imported when you use a wildcard import (from my_package import *). By defining the __all__ list in __init__.py, you specify which modules or functions should be accessible.

Ammar **Munir**

Providing a convenient interface

You can provide a more convenient interface for your package by exposing selected functions or classes at the package level.

Structure

```
my_package/
    __init__.py
    module1.py
    module2.py
```

Module1

module1.py

```python
def add(a, b):
    return a + b
```

Module1

module2.py

```python
def uppercase(s):
    return s.upper()
```

__init__.py

__init__.p...

```python
from .module1 import add
from .module2 import uppercase
```

main.py

main.py

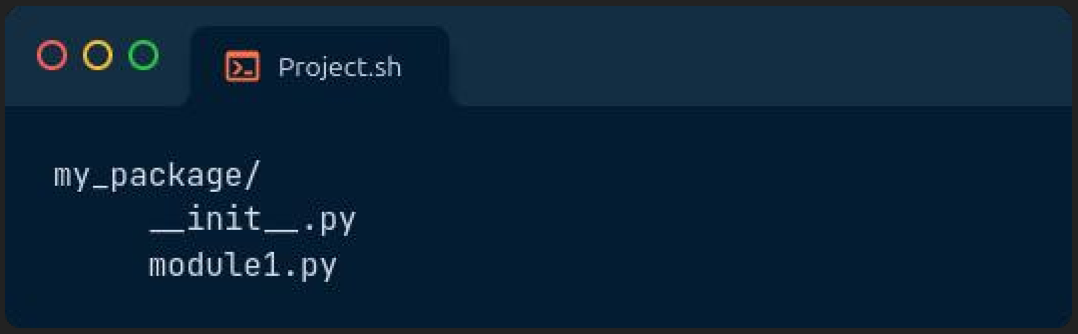
```python
from my_package import add, uppercase

print(add(3, 4))  # Output: 7
print(uppercase("hello"))  # Output: HELLO
```

Ammar **Munir**

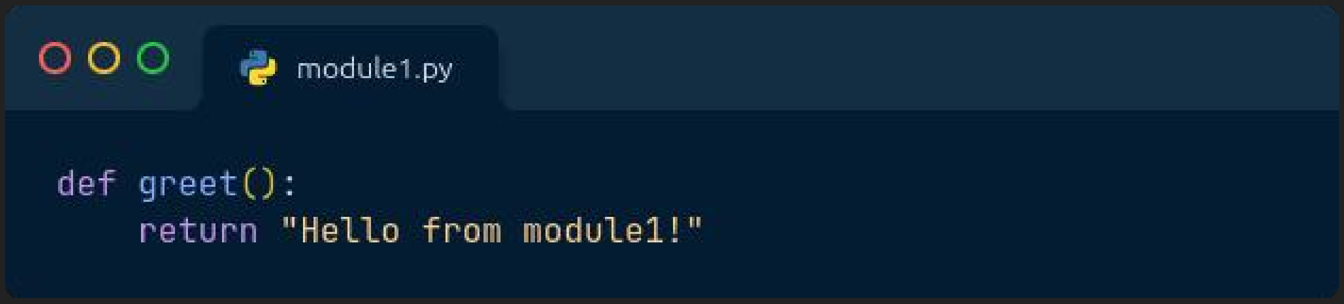
Ensuring compatibility and clarity

Structure

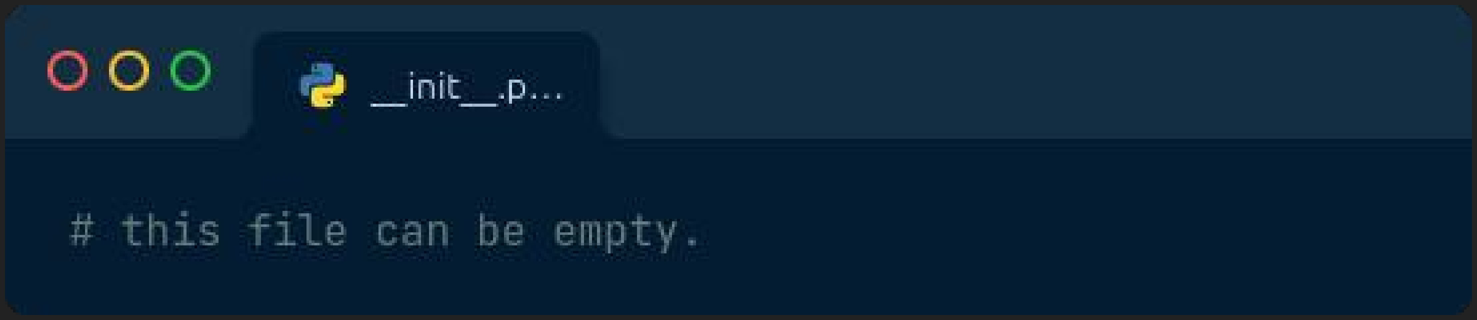

Project.sh

```
my_package/
    __init__.py
    module1.py
```

Module1

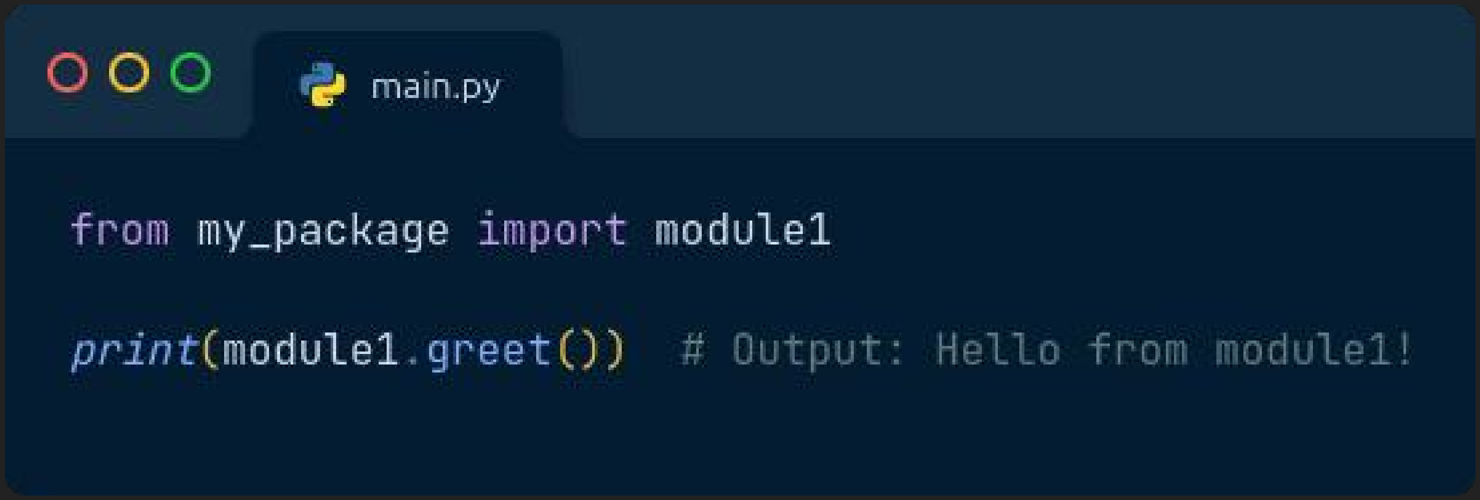

module1.py

```python
def greet():
    return "Hello from module1!"
```


__init__.p...

```python
# this file can be empty.
```

main.py


main.py

```python
from my_package import module1

print(module1.greet())  # Output: Hello from module1!
```

Using __init__.py makes it clear that the directory is intended to be a package, which can help with compatibility and understanding the package structure.

Ammar **Munir**

Was it helpful ?

Like Comment Share Save

Ammar **Munir**

Python | Django | Web Engineer