https://stackoverflow.com/questions/9439480/from-import-vs-import

6 Answers

Sorted by: Highest score (default)

It depends on how you want to access the import when you refer to it.

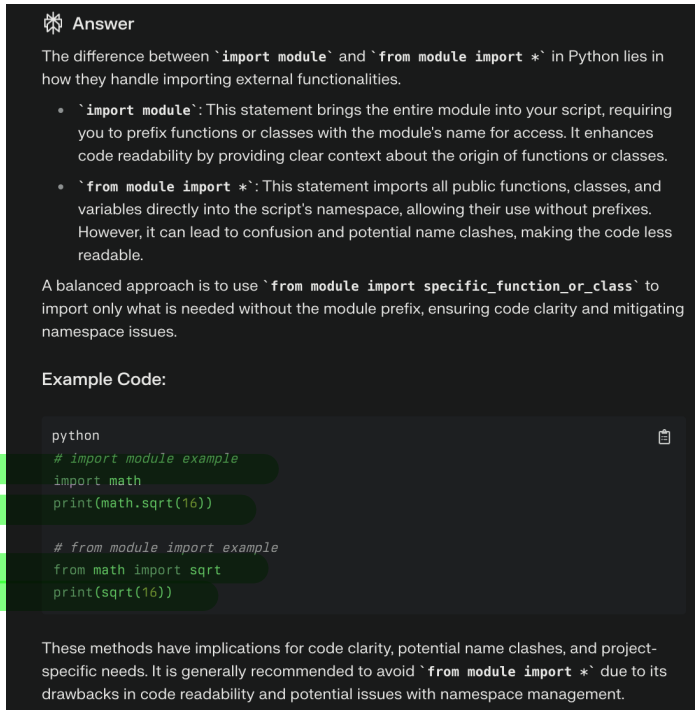
324

```
from urllib import request
# access request directly.
mine = request()

import urllib.request
# used as urllib.request
mine = urllib.request()
```

You can also alias things yourself when you import for simplicity or to avoid masking built ins:

```
from os import open as open_
# lets you use os.open without destroying the
# built in open() which returns file handles.
```



Answer

The difference between `import module` and `from module import *` in Python lies in how they handle importing external functionalities.

- `import module`: This statement brings the entire module into your script, requiring you to prefix functions or classes with the module's name for access. It enhances code readability by providing clear context about the origin of functions or classes.
- `from module import *`: This statement imports all public functions, classes, and variables directly into the script's namespace, allowing their use without prefixes. However, it can lead to confusion and potential name clashes, making the code less readable.

A balanced approach is to use `from module import specific_function_or_class` to import only what is needed without the module prefix, ensuring code clarity and mitigating namespace issues.

Example Code:

```python
# import module example
import math
print(math.sqrt(16))

# from module import example
from math import sqrt
print(sqrt(16))
```

These methods have implications for code clarity, potential name clashes, and project-specific needs. It is generally recommended to avoid `from module import *` due to its drawbacks in code readability and potential issues with namespace management.