PyCon Greece
29-30 August 2025
Technopolis City of Athens

Consistent importing

Jan Bielecki

Intro

Jan Bielecki

Senior Full Stack Developer at Ørsted with 6 years of professional experience in software development. Interested in science, digitalization, electrification, sports, music and comedy.

Scan for full presentation

consistent_importing_all_slides.pdf

Ørsted

Ørsted is a Danish multinational power company and a global leader in renewable energy. The company is committed to sustainability and aims to create a world that runs entirely on green energy.[1]

Scan for full presentation

consistent_importing_all_slides.pdf

Agenda

I. How importing in Python works?
II. How to import in a consistent way?
III. How internal dependencies impact a codebase architecture?
IV. How we can enforce the selected importing strategy in CI?

Consistent importing

Re-exporting entities through __init__.py

```python
# model/geometry/node.py
class Node: ...

# model/geometry/__init__.py
from .node import Node, function_a
from .edge import Edge, function_b

# model/__init__.py
from .geometry import Node, Edge, function_a, function_b
from .other_module import SomeOtherClass

# gui/__init__.py
from model import Node, Edge, function_a, function_b, SomeOtherClass
```

✓ Pros:

- Encapsulation
- Single entry point
- Short imports

✗ Cons:

- Maintenance overhead
- IDE ambiguous suggestions
- Hard to automate consistency

Consistency reduces decision fatigue

"the more decisions you make throughout the day, the worse you are at making them"[2]

Python importing without consistency

How importing in Python works?

Python module

```
(base) jan@jan-Inspiron-3543:~/consistent_importing$ python
Python 3.12.8 [GCC 11.2.0] on linux
>>> def hello_world() -> None:
...     print("Hello world!")
...
>>> hello_world()
Hello world!
>>> exit()
(base) jan@jan-Inspiron-3543:~/consistent_importing$ python
Python 3.12.8 [GCC 11.2.0] on linux
>>> hello_world()
Traceback (most recent call last):
  File "", line 1, in
NameError: name 'hello_world' is not defined
>>>
```

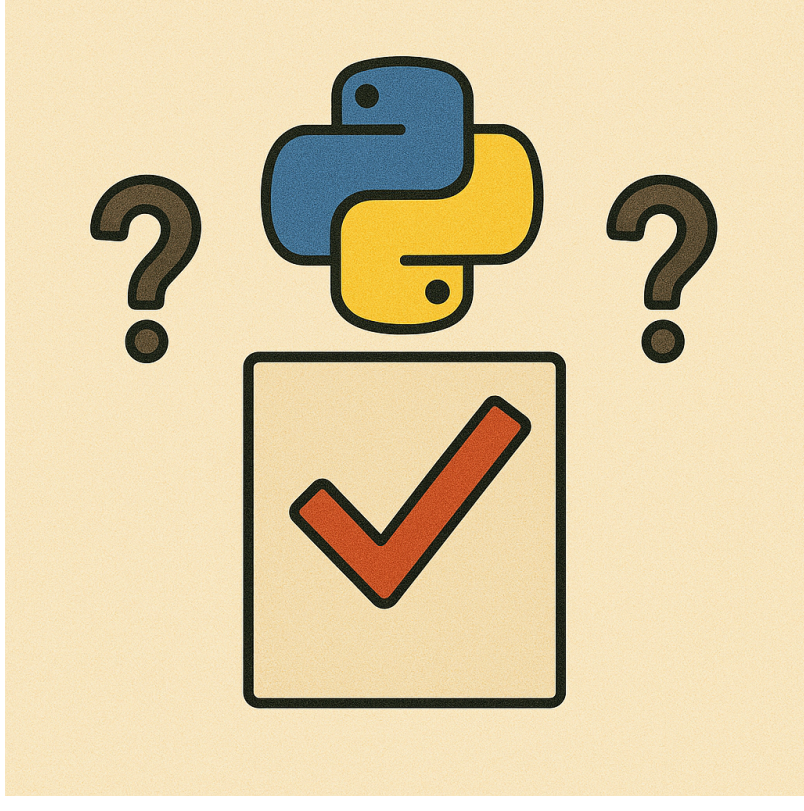
1. Python module

```python
# my_module.py
def hello_world() -> None:
  print("Hello world!")

print("Hello from my_module!")
```

```
(base) jan@jan-Inspiron-3543:~/consistent_importing$ python
Python 3.12.8 [GCC 11.2.0] on linux
>>> import my_module
Hello from my_module!
>>> dir(my_module)
[..., '__name__', '__package__', '__spec__', 'hello_world']
>>> my_module.hello_world()
Hello world!
>>>
```

II. What is the best importing strategy?



0. Typing

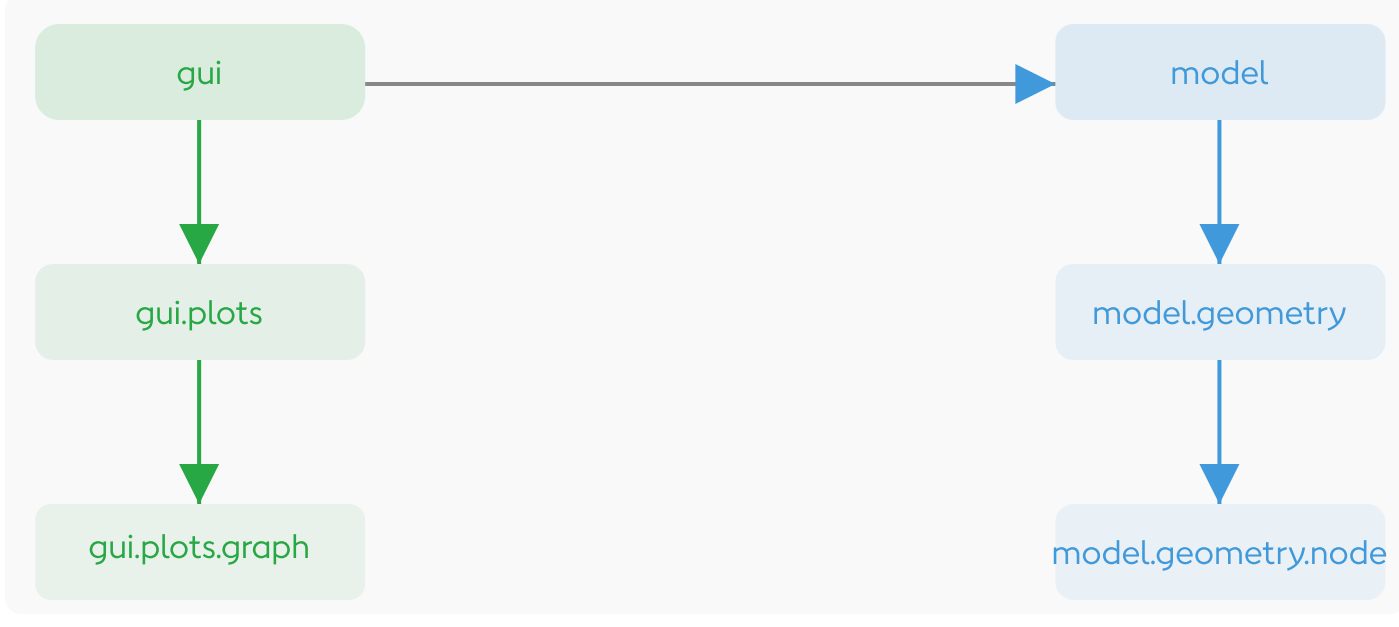
TODO text here ...

III. How internal dependencies impact a codebase architecture?

Python circular imports -> bug or a feature?

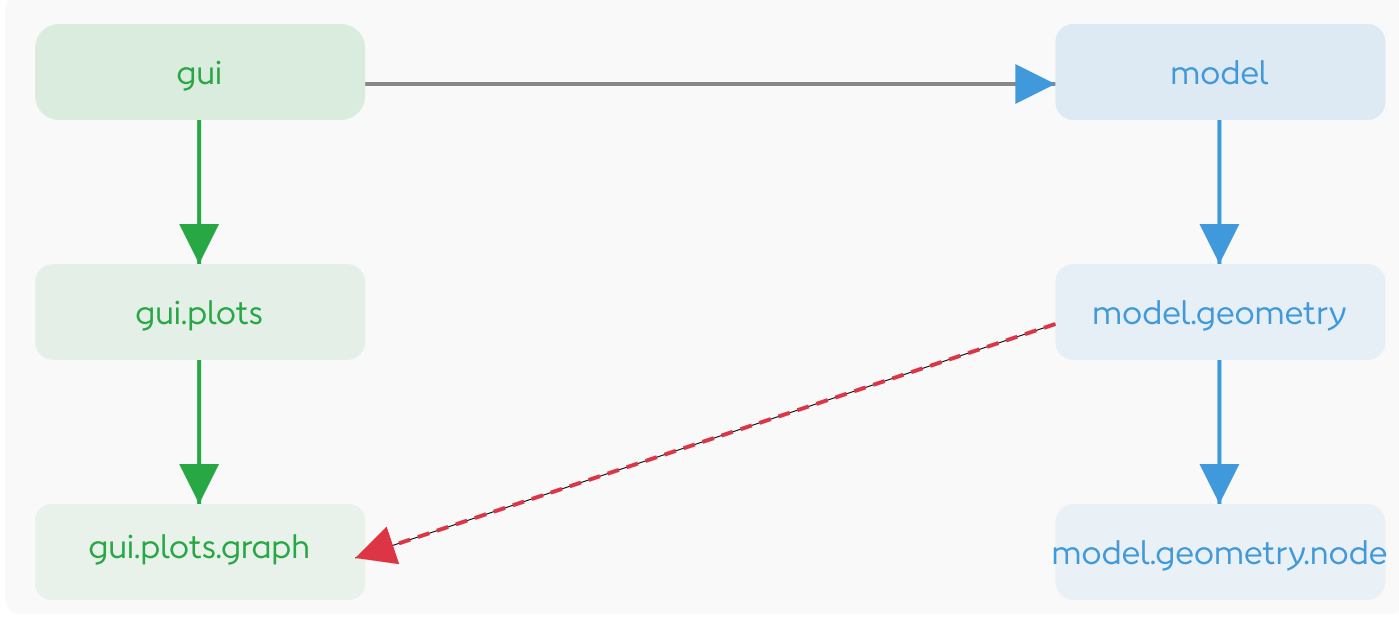
Circular imports can be both a bug and a feature: they often signal a design issue, but sometimes are used intentionally for plugin systems or late binding. In most cases, refactoring to avoid them leads to clearer architecture.

Circular dependencies between packages



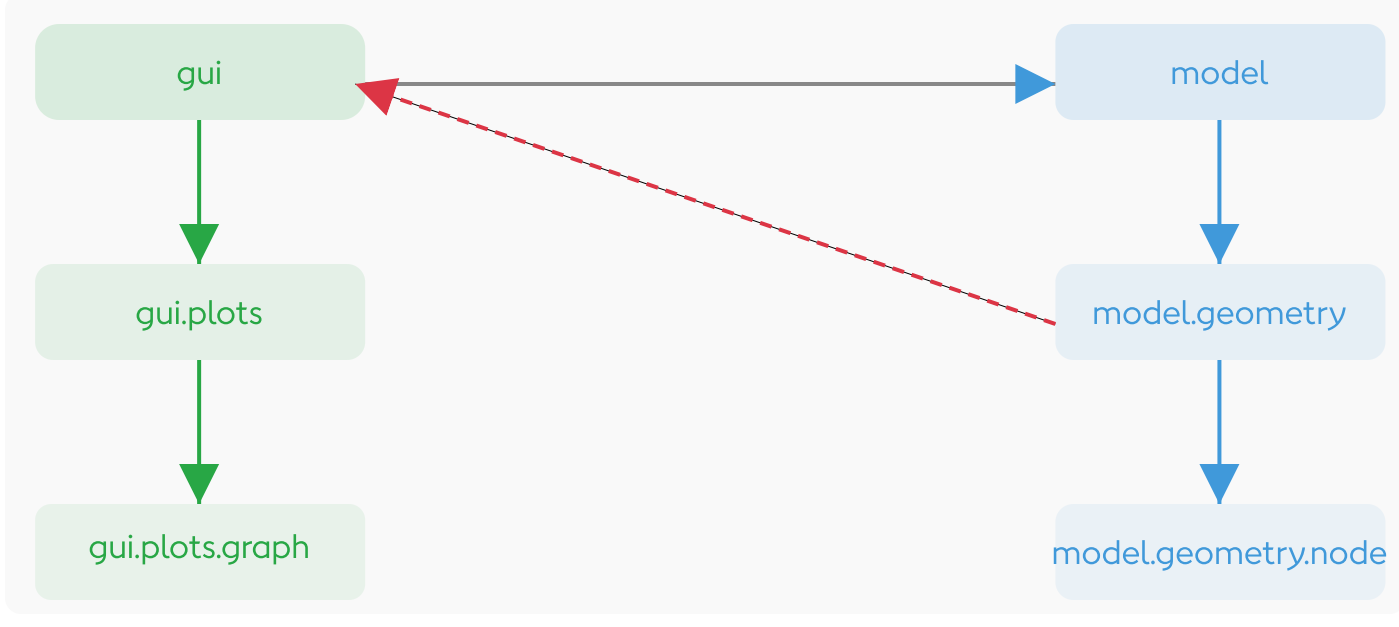
Packages "model" and "gui" re-exporting everything through __init__.py.
Package "gui" imports from "model".

Circular dependencies between packages



Attempting to add a dependency from "model" to "gui" directly from the the module "gui.plots.graph".

Circular dependencies between packages



Attempting to add a dependency from "model" to "gui", using re-exported entity, creates a circular dependency.

Circular dependencies between packages

```python
# model/geometry/node.py
class Node: ...

# model/geometry/__init__.py
from .node import Node

# model/__init__.py
from .geometry import Node

# gui/__init__.py
from model import Node

# model/geometry/__init__.py
from gui import Graph
```

```
File "gui/__init__.py", line 1, in
    from model import Node
  File "model/__init__.py", line 1, in
    from .geometry import Node
  File "model/geometry/__init__.py", line 3, in
    from gui import Graph
    ^^^^^^^^^^^^^^^^^^^^
ImportError: cannot import name 'Graph' from partially initialized module 'gui'
(most likely due to a circular import) (/gui/__init__.py)
```

Acyclic dependencies principle (ADP)

IV. How can we enforce the selected strategy in CI?

1. Import Linter

TODO text here ...

Bibliography

1. Photo by Lester Hsu, https://orsted.com/en/media/news/2022/04/20220421515811
2. https://www.todaywellspent.com/blogs/articles/why-do-successful-people-wear-the-same-outfits-every-day