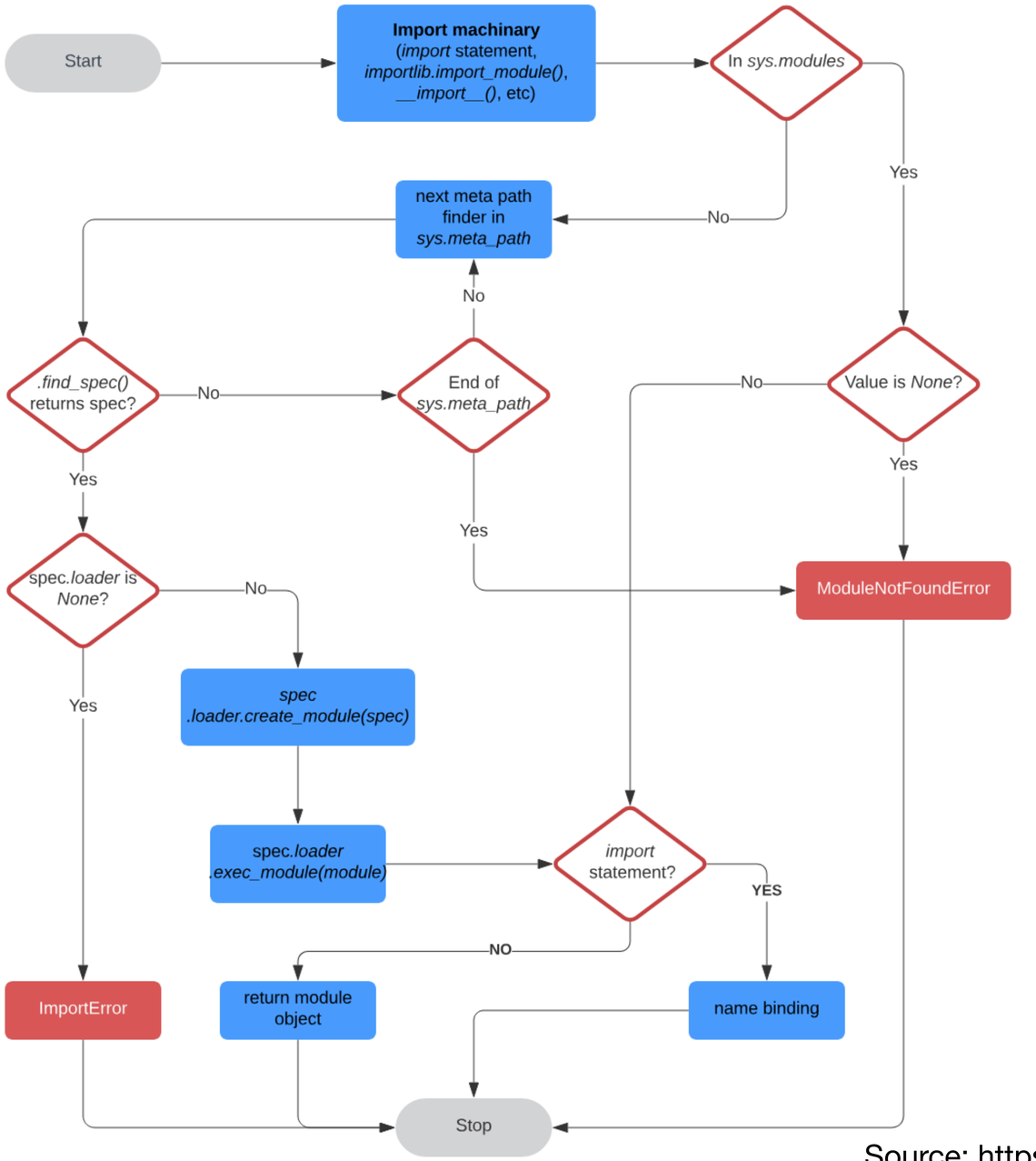
Lazy imports

PyCon Portugal 2026

Anže Pečar, 3 Sep 2026

imports

Python Import System



Source: https://github.com/pwwang/python-import-system

The problem

```
import time

time.sleep(3)

def my_func():
    pass
```

```
from previous_slide import my_func

...

if some_condition:
    my_func()
```

```
app on  lazy [$!?] via  desktop-linux is 📦 v1.0.0 via  v24.13.0 via 
v3.13.11 took 3s
❯ time ./manage.py check
```

```
app2 on  main [$!?⇣] via  desktop-linux is 📦 v1.0.0 via  v24.13.0 via
 v3.13.11 took 12s
❯ time ./manage.py check
```

python -X importtime

python -S -X importtime -c "import json" | tail -n 16

```
import time: self [us] | cumulative | imported package
import time:        55 |         55 |           itertools
import time:       334 |        334 |           keyword
import time:        56 |         56 |             _operator
import time:      1494 |       1550 |           operator
import time:      1124 |       1124 |           reprlib
import time:        34 |         34 |           _collections
import time:      5610 |       9200 |         collections
import time:        31 |         31 |         _functools
import time:      4307 |      13658 |       functools
import time:       866 |        866 |       copyreg
import time:      1884 |      36957 |     re
import time:        54 |         54 |       _json
import time:       675 |        728 |     json.scanner
import time:      1688 |      39372 |   json.decoder
import time:      1652 |       1652 |   json.encoder
import time:      1156 |      42179 | json
```

Inline imports

```
from pathlib import Path


def get_path():
    return Path(".")
```

```
def get_path():
    from pathlib import Path
    return Path(".")
```

Type annotations

```
from pathlib import Path


def get_path() -> Path:
    return Path(".")
```

```
def get_path() -> Path:
    from pathlib import Path
    return Path(".")
```

```python
from typing import TYPE_CHECKING

if TYPE_CHECKING:
    from pathlib import Path

def get_path() -> Path:
    from pathlib import Path
    return Path(".")
```

ENG-6097: Improve startup times with lazy imports #4986

Code

Merged aviator-app[bot] merged 11 commits into `main` from `apecar/eng-6097-lazy-import-google-imports-to-speed-up-startup-time` on Mar 16

Conversation 3 | Commits 11 | Checks 32 | Files changed 132 | +1,124 -546

anze3db commented on Mar 14 • edited

We have many heavy imports during startup which makes django commands and server restarts very slow.

`./manage.py check` Startup Time Benchmark

Each commit benchmarked with 5 runs, median taken.

Commit	Description	Median	Delta	vs Baseline
`origin/main`	**Baseline**	**9.45s**	—	—
`a8d58b7ba`	Make google imports lazy	4.60s	-4.85s	2.1x faster
`cf1aa8fbc`	Lazy snowflake	3.55s	-1.05s	2.7x faster
`671423d0c`	More lazyiness	2.91s	-0.64s	3.2x faster
`57f3d1b65`	Only import mypy_boto when type checking	2.62s	-0.29s	3.6x faster
`9849d3269`	Make datastructures lazy	2.48s	-0.14s	3.8x faster
`4c0bb2bb6`	Add lint rule for slow top level imports and remove more usages	2.34s	-0.14s	4.0x faster
`f8c7d0dae`	Lazy import svix, checkdmarc, openpyxl, pydantic_ai, user_agents	2.17s	-0.17s	4.4x faster

Total improvement: 9.45s → 2.17s (4.4x faster)

Reviewers
- hyperNURb ✓
- leetrout ✓
- Copilot Lite — Request

Attention
- leetrout
- hyperNURb
- anze3db

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestone
No milestone

TID253 - banned-module-level-imports

```
[tool.ruff.lint]
banned-module-level-imports = [
"snowflake", "google", "googleapiclient",
"kubernetes",
"mcp", "sigma"
]
```

Circular imports

```python
# app/a.py
from app.b import greet

def name():
    return "world"

# app/b.py
from app.a import name

def greet():
    print(f"hello {name()}")
```

```
$ python -c "import app.a"
Traceback (most recent call last):
  File "<string>", line 1, in <module>
    import app.a
  File "app/a.py", line 1, in <module>
    from app.b import greet
  File "app/b.py", line 1, in <module>
    from app.a import name
ImportError: cannot import name 'name' from
partially initialized module 'app.a'
(most likely due to a circular import) (app/a.py)
```

Lazy imports

PEP 690 – Lazy Imports

PEP 690 – Lazy Imports

[REJECTED]

PEP 810 – Explicit lazy imports

PEP 810 – Explicit lazy imports

[ACCEPTED]

```
import json
from json import dumps
```

```
lazy import json
lazy from json import dumps
```

```
__lazy_modules__ = ["json"]

import json
from json import dumps
```

`-X lazy_imports=<mode>`

- "normal" (or unset): Only explicitly marked lazy imports are lazy
- "all": All module-level imports (except in try blocks and import *) become potentially lazy
- "none": No imports are lazy, even those explicitly marked with lazy keyword

```
sys.set_lazy_imports_filter(func)
```

Inline imports

```
from pathlib import Path


def get_path():
    return Path(".")
```

```
lazy from pathlib import Path


def get_path():
    return Path(".")
```

Type annotations

```
from pathlib import Path


def get_path() -> Path:
    return Path(".")
```

```
lazy from pathlib import Path


def get_path() -> Path:
    return Path(".")
```

```python
from typing import TYPE_CHECKING

if TYPE_CHECKING:
    from pathlib import Path

def get_path() -> Path:
    from pathlib import Path
    return Path(".")
```

```
lazy from pathlib import Path


def get_path() -> Path:
    return Path(".")
```

Circular imports

```python
# app/a.py
from app.b import greet

def name():
    return "world"

# app/b.py
from app.a import name

def greet():
    print(f"hello {name()}")
```

```
$ python3.15 -X lazy_imports=all -c "import app.a"
hello world
```

```python
# app/a.py
from app.b import greet

def name():
    return "world"

# app/b.py
from app.a import name

def greet():
    print(f"hello {name()}")
```

```
# app/a.py
lazy from app.b import greet

def name():
    return "world"

# app/b.py
lazy from app.a import name

def greet():
    print(f"hello {name()}")
```

```
$ python -c "import app.a"
hello world
```

Gotchas

```python
# app.py
from flask import Flask
import yaml  # not installed

app = Flask(__name__)


@app.get("/export")
def export():
    return yaml.dump({"debug": True})
```

```
$ python -m flask run
Error: While importing 'app', an ImportError was raised:

  File "app.py", line 2, in <module>
    import yaml              # not installed
    ^^^^^^^^^^^
ModuleNotFoundError: No module named 'yaml'
```

```
$ python -X lazy_imports=all -m flask run
  * Running on http://127.0.0.1:5000
```

```
$ python -X lazy_imports=all -m flask run
  * Running on http://127.0.0.1:5000
[2026-09-03 01:00:22,588] ERROR in app: Exception on /export [GET]
Traceback (most recent call last):
  File "app.py", line 2, in <module>
    import yaml  # not installed
ImportError: lazy import of 'yaml' raised an exception during resolution

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "site-packages/flask/app.py", line 902, in dispatch_request
    return self.ensure_sync(self.view_functions[rule.endpoint])(**view_args)
  File "app.py", line 9, in export
    return yaml.dump({"debug": True})
           ^^^^
ModuleNotFoundError: No module named 'yaml'
127.0.0.1 - - [03/Sep/2026 01:00:22] "GET /export HTTP/1.1" 500 -
```

```python
# app.py
from flask import Flask
from googleapiclient.discovery import build
app = Flask(__name__)

@app.get("/files")
def files():
    build("drive", "v3")
    return "OK"
```

```
$ python -m flask --app app:app run
server ready after 3.2s

GET /files  200  0.002s
GET /files  200  0.002s
GET /files  200  0.001s
```

```
$ python -X lazy_imports=all -m flask --app app:app run
server ready after 0.2s

GET /files  200  3.022s  <- pays for the import
GET /files  200  0.002s
GET /files  200  0.001s
```

Questions?

braces

```
>>> from __future__ import braces
  File "<python-input-0>", line 1
    from __future__ import braces
                           ^^^^^^
SyntaxError: not a chance
```

```
lazy import json
print({k: v for k, v in globals().items() if not k.startswith("__")})

{'json': <lazy_import 'json'>}
```