Free-Threaded Python

Python Lisbon Meetup

Slides and examples

@anze3db, October 2, 2025

CPU bound vs I/O bound

CPU bound

The time it takes to complete a task is determined by the speed of the CPU.

I/O Bound

The time it takes to complete a task is determined by the period spent waiting for input/output.

👩‍🍳

🥄

🥘

🐍 Examples

Concurrency vs Parallelism

Concurrency

The ability of a system to handle multiple tasks at once, making progress on them by **switching** between them.

Parallelism

The ability of the system to execute multiple tasks truly **simultaneously**.

🐍 Examples

Concurrency

Async/Await

Parallelism

Processes

Concurrency

Async/Await

- Can only context switch on await
- No good for CPU Bound

Parallelism

Processes

- Uses more memory
- Not easy to share results between processes

Threads? 🧵

Concurrency

Async/Await

Threads

Parallelism

Processes

Why?!?

Global Interpreter Lock (GIL)

Is a mechanism in CPython that ensures only one thread executes Python bytecode at a time.

Single-threaded Performance

Without the GIL **reference counting*** becomes hard

*Python's method of managing memory

Reference Counting

```
my_lst = ["hello"]      # ref cnt of ["hello"] == 1
other_lst = my_lst      # ref cnt of ["hello"] == 2
del other_lst           # ref cnt of ["hello"] == 1
```

Race Conditions

Race condition with `cnt+=1`

Thread 1	Thread 2		Integer Value
			0
read value		📖	0
	read value	📖	0
increase value			0
	increase value		0
write value		✍️	**1**
	write value	✍️	**1**

Locks

```
import threading

lock = threading.Lock()
with lock:
    x += 1
```

1996 Greg Stein’s Free-Threading Patch

2008 Adam Olsen's python-safethread

2016 Larry Hastings' Gilectomy

2016 Trent Nelson's PyParallel

+ Jython, IronPython, PyPy: STM

Sam Gross' nogil

Sam Gross' ~~nogil~~ free-threaded Python

Biased locks + Immortalization + Deferred locks

https://peps.python.org/pep-0703/

Concurrency

Async/Await

~~Threads~~ $\xrightarrow[\textbf{Free-threaded Python}]{}$

Parallelism

Processes

Threads

🐍 Example

Phase 1

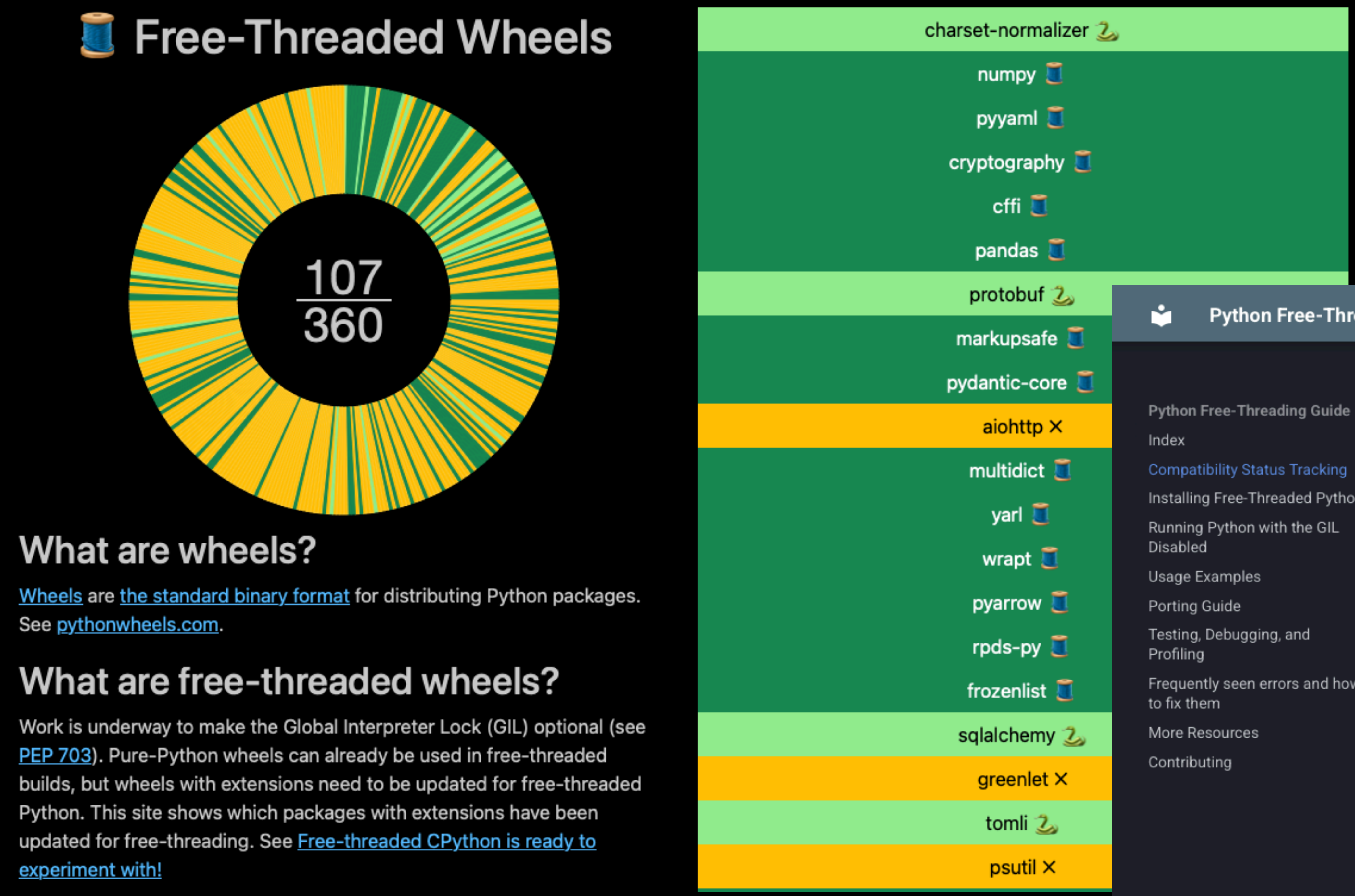
Free-threaded Python **build experimental** (3.13t)

Phase 2

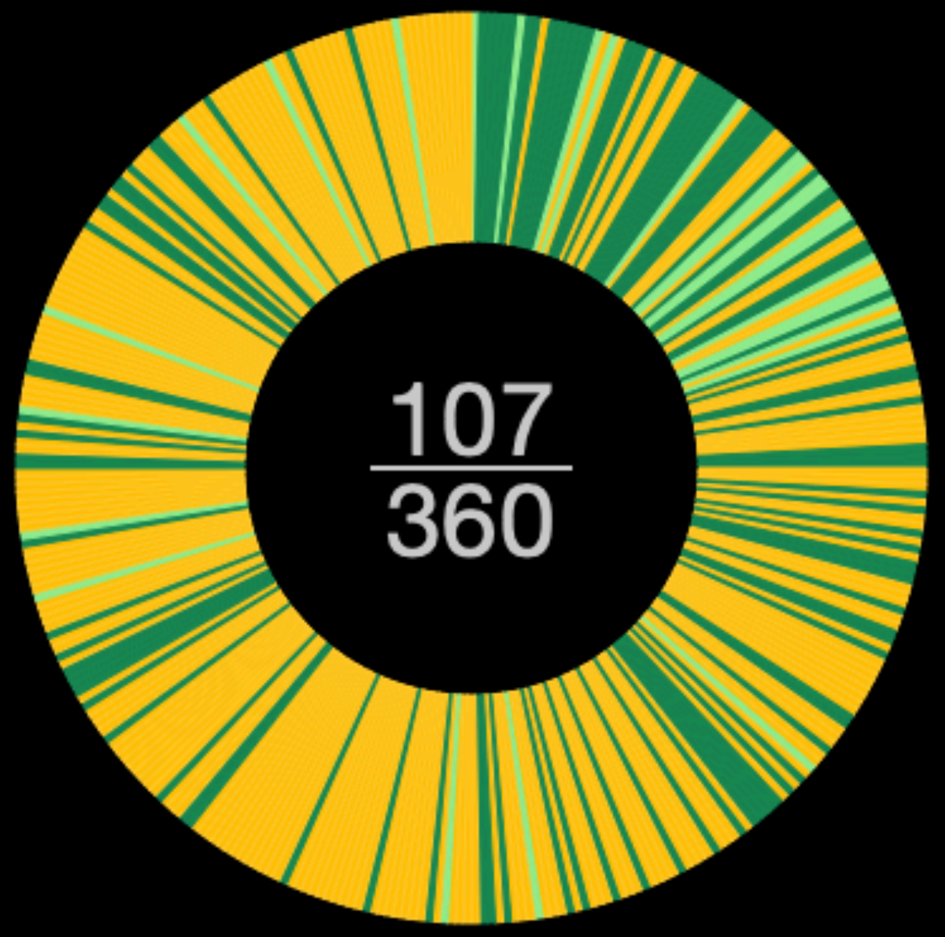
Free-threaded Python build **officially supported** (3.14t)

Phase 3

Free-threaded Python the **default** (3.??)



🧵 Free-Threaded Wheels



- charset-normalizer 🐍
- numpy 🧵
- pyyaml 🧵
- cryptography 🧵
- cffi 🧵
- pandas 🧵
- protobuf 🐍
- markupsafe 🧵
- pydantic-core 🧵
- aiohttp ✕
- multidict 🧵
- yarl 🧵
- wrapt 🧵
- pyarrow 🧵
- rpds-py 🧵
- frozenlist 🧵
- sqlalchemy 🐍
- greenlet ✕
- tomli 🐍
- psutil ✕

What are wheels?

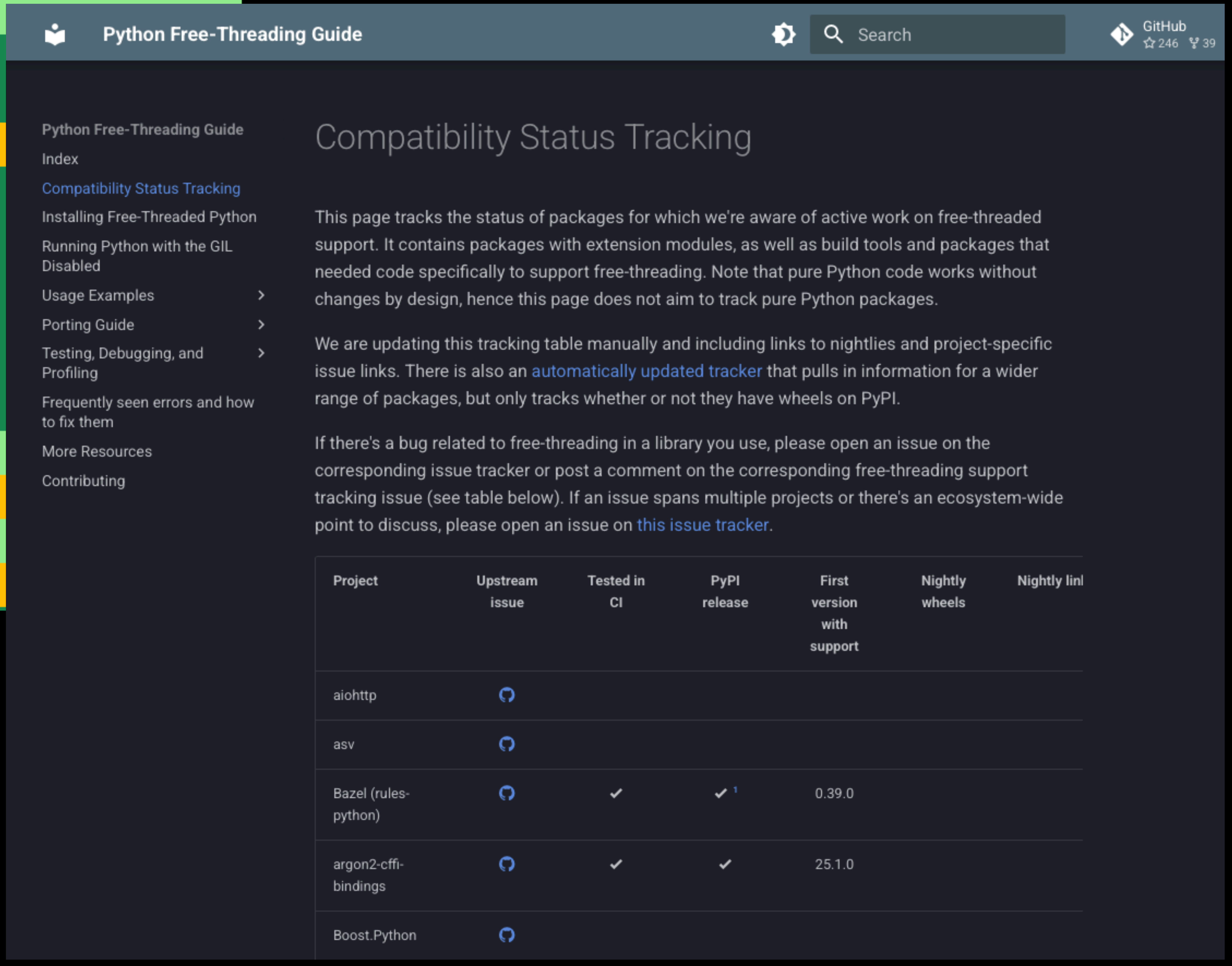
Wheels are the standard binary format for distributing Python packages. See pythonwheels.com.

What are free-threaded wheels?

Work is underway to make the Global Interpreter Lock (GIL) optional (see PEP 703). Pure-Python wheels can already be used in free-threaded builds, but wheels with extensions need to be updated for free-threaded Python. This site shows which packages with extensions have been updated for free-threading. See Free-threaded CPython is ready to experiment with!

https://hugovk.github.io/free-threaded-wheels/

https://py-free-threading.github.io/tracking/



Compatibility Status Tracking

This page tracks the status of packages for which we're aware of active work on free-threaded support. It contains packages with extension modules, as well as build tools and packages that needed code specifically to support free-threading. Note that pure Python code works without changes by design, hence this page does not aim to track pure Python packages.

We are updating this tracking table manually and including links to nightlies and project-specific issue links. There is also an automatically updated tracker that pulls in information for a wider range of packages, but only tracks whether or not they have wheels on PyPI.

If there's a bug related to free-threading in a library you use, please open an issue on the corresponding issue tracker or post a comment on the corresponding free-threading support tracking issue (see table below). If an issue spans multiple projects or there's an ecosystem-wide point to discuss, please open an issue on this issue tracker.

Project	Upstream issue	Tested in CI	PyPI release	First version with support	Nightly wheels	Nightly lin
aiohttp	●					
asv	●					
Bazel (rules-python)	●	✓	✓ [1]	0.39.0		
argon2-cffi-bindings	●	✓	✓	25.1.0		
Boost.Python	●					

Questions?

Slides and examples

@anze3db, October 2, 2025