Parallel Python at last?
Subinterpreters & free-threading in practice

Vita Midori

Swiss Python Summit

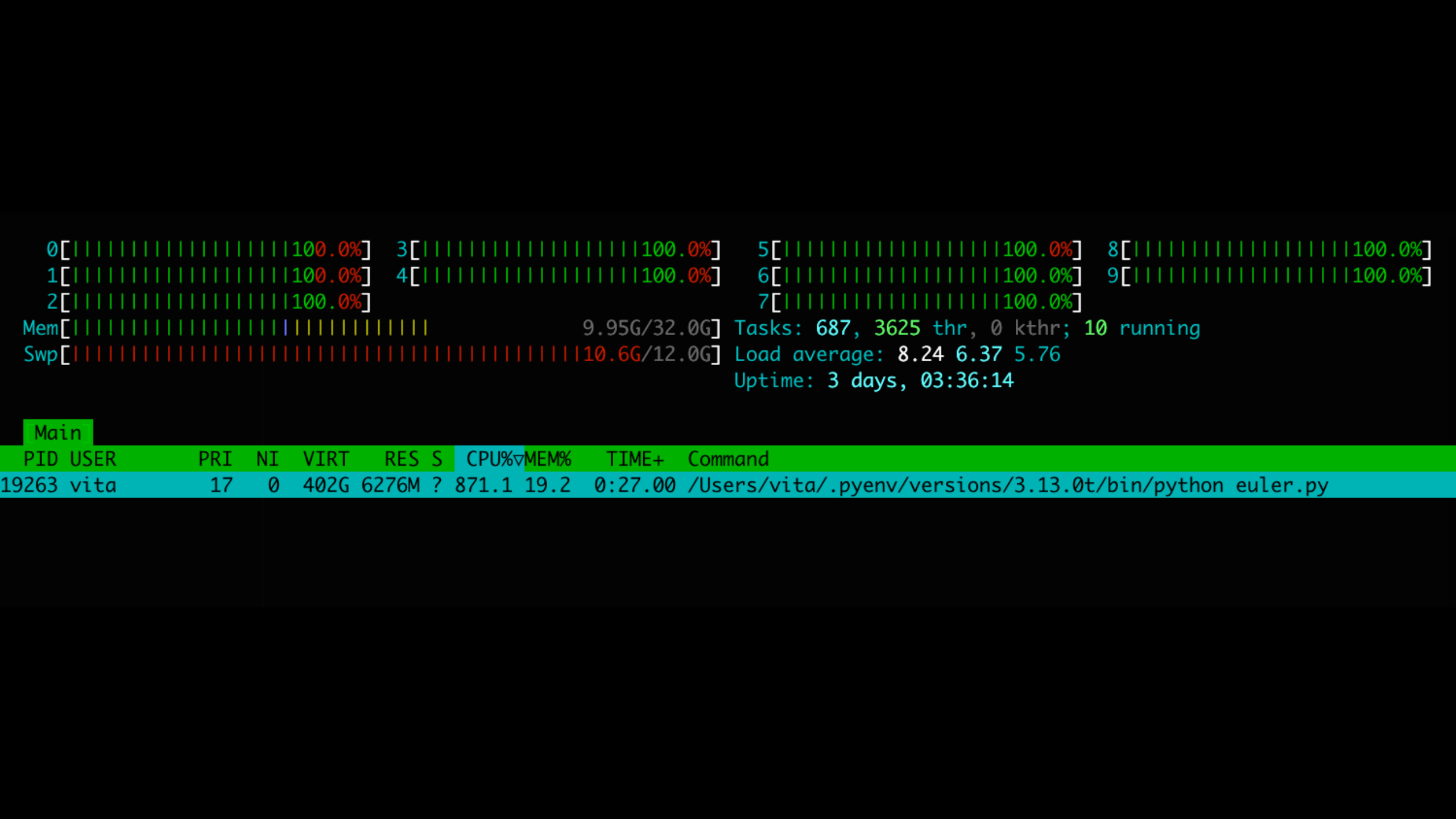
October 17, 2024

Hi, I am Vita 👋

Freelancer & consultant focusing on performance optimisation and distributed systems.

I also do machine learning at Demokratis.ch to make the Swiss Vernehmlassung process more accessible.

2.718281828459045235360287471352662497757247093699959574966967627
724076630353547594571382178525166427427466391932003059921817413596629043572900334295260595630738132328627943490763233829880753195251019011573834187930702154089149934884167509244761460668082648001684774118537423454424371075390777449920695517027618386062613313845830007520449338265602976067371132007093287091274437470472306969772093101416928368190255151086574637721112523897844250569536967707854499699679468644549059879316368923009879312773617821542499922957635148220826989519366803318252886939849646510582093923982948879332036250944311730123819706841614039701983767932068328237646480429531180232878250981945581530175671736133206981125099618188159304169035159888851934580727386673858942287922849989208680582574927961048419844436346324496848756023362482704197862320900216099023530436994184914631409343173814364054625315209618369088870701676839642437814059271456354906130310720851038375051011574770417189861068739696552126715468895703503540212340784981933432106


0[|||||||||||||||||100.0%]
1[|||||||||||||||||100.0%]
2[|||||||||||||||||100.0%]
3[|||||||||||||||||100.0%]
4[|||||||||||||||||100.0%]
5[|||||||||||||||||100.0%]
6[|||||||||||||||||100.0%]
7[|||||||||||||||||100.0%]
8[|||||||||||||||||100.0%]
9[|||||||||||||||||100.0%]
Mem[|||||||||||||||||||||||| 9.95G/32.0G]
Swp[|||||||||||||||||||||||||||||||||||||10.6G/12.0G]
Tasks: 687, 3625 thr, 0 kthr; 10 running
Load average: 8.24 6.37 5.76
Uptime: 3 days, 03:36:14
Main
PID USER PRI NI VIRT RES S CPU%▽MEM% TIME+ Command
19263 vita 17 0 402G 6276M ? 871.1 19.2 0:27.00 /Users/vita/.pyenv/versions/3.13.0t/bin/python euler.py

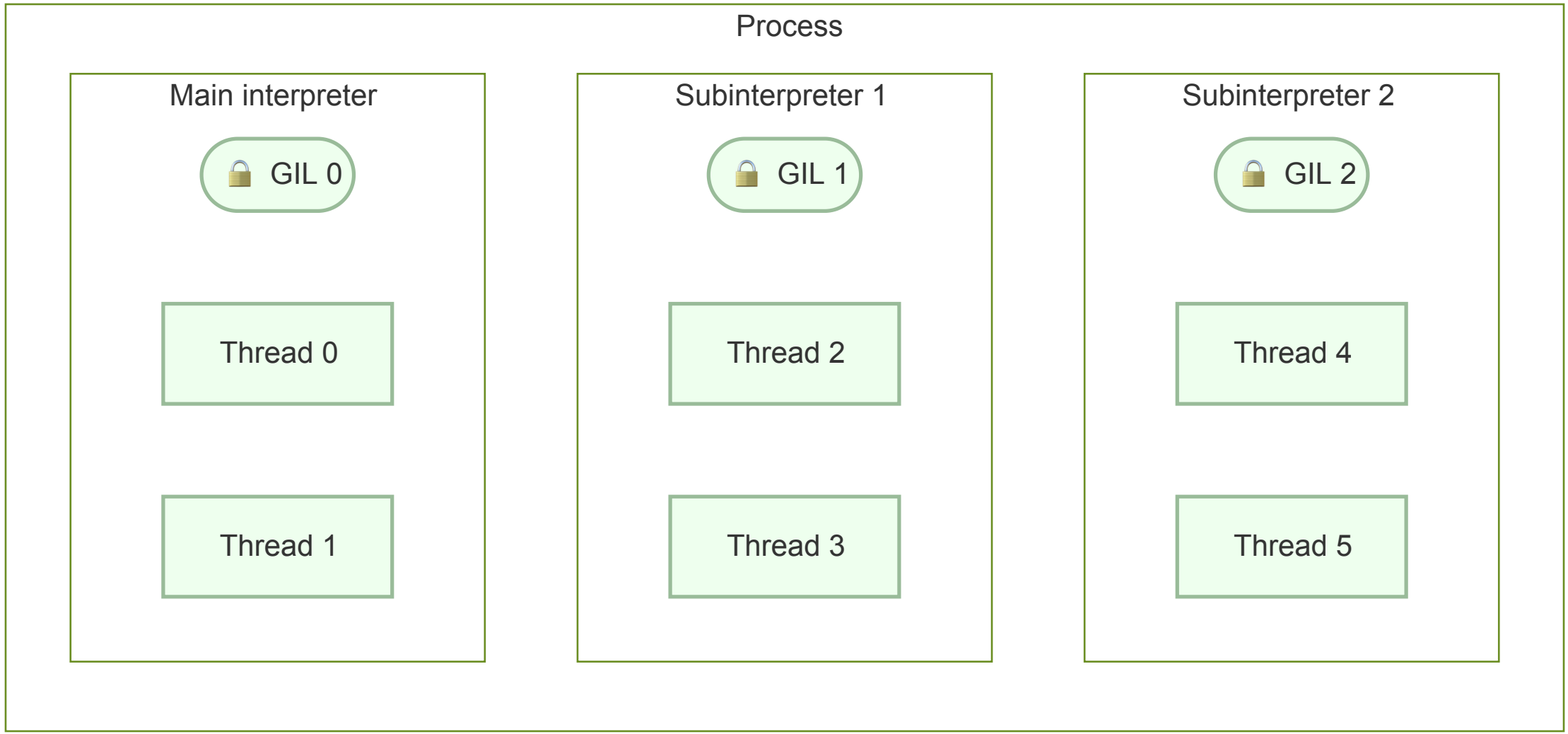
Here be dragons

- Subinterpreters and free-threading are a rapidly shifting landscape right now
- Excitement
- Momentum
- Confusion

A little bit about the GIL (Global Interpreter Lock)

- Python / CPython / C extensions
- Memory management for Python objects
- Rationale: synchronisation of *reference counting*
- Side effect: synchronisation of *almost all threads*

Subinterpreters


Process
Main interpreter
GIL 0
Thread 0
Thread 1
Subinterpreter 1
GIL 1
Thread 2
Thread 3
Subinterpreter 2
GIL 2
Thread 4
Thread 5

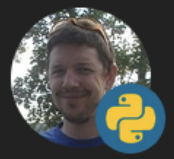
```
def my_function():
    print("Hello from a subinterpreter")

import _interpreters

subinterpreter = _interpreters.create()
_interpreters.exec(subinterpreter, my_function)
```

The confusing road to here

- 2022-2023, Python 3.12: PEP 684 – A Per-Interpreter GIL ✅
- 2017-2023: PEP 554 – Multiple Interpreters in the Stdlib ❌
- 2023-202_, Python 3.1_: PEP 734 – Multiple Interpreters in the Stdlib ⏳



Eric Snow CPython core developer 17h

rtobar:

I'm still a bit unclear about the expected future of the PEP.

Up until today, I was too. 🙂 I had a productive discussion with the Steering Council a few hours ago. The path forward is substantially more clear now.

https://discuss.python.org/t/expected-future-for-pep-734/64974

Officially unofficial

- PEP 734 is not yet accepted nor implemented in CPython
- However, you *can* already find *some* subinterpreter APIs in 3.13!

Cross-interpreter communication

- Queues and channels
- Some immutable data can be shared without copying
- Most things have to pickled – but it’s still faster than doing the same between processes

Real-life example

See the PyCon US talk Unlocking the Parallel Universe: Subinterpreters and Free-Threading in Python 3.13 by Anthony Shaw

github.com/tonybaloney/subinterpreter-web

tonybaloney.github.io/posts/sub-interpreter-web-workers.html

Sharing data like in the 1960s

See another PyCon US talk: Overcoming GIL with subinterpreters and immutability by Yury Selivanov

github.com/edgedb/memhive

Ways to use subinterpreters

- Isolated subinterpreters, talking via queues or channels
- Isolated subinterpreters, servicing different sockets/file descriptors
- ✨ Subinterpreters intelligently sharing data, taking advantage of the shared memory space

The catch: many extensions don’t work yet

```
ImportError('module _pydantic_core does not support loading in subinterpreters')
```

```
RuntimeError('CPU dispatcher tracer already initlized')
python3(83719,0x1f830bac0) malloc: *** error for object 0x104c4fd70: pointer being freed was not allocated
python3(83719,0x1f830bac0) malloc: *** set a breakpoint in malloc_error_break to debug
```

numpy#24755, cython#1715, PyO3#576, …

Free-threading Python

How is this possible?

2023-2024, Python 3.13: PEP 703 – Making the Global Interpreter Lock Optional in CPython

1. Reference counting
2. Memory management
3. Thread safety of containers (lists, dicts, ...)

Why did it take so long?

- Overhead around locking and reference counting
- ⚠️ The GIL-free Python 3.13 is slightly *slower* than normal 3.13 with the GIL

- C extension compatibility
- ⚠️ The GIL-free Python 3.13 *isn't yet compatible* with many C extensions

A tale of two Pythons

- Python 3.13 includes the GIL by default and nothing changes
- Python **3.13t** is a separate Python build with optional GIL

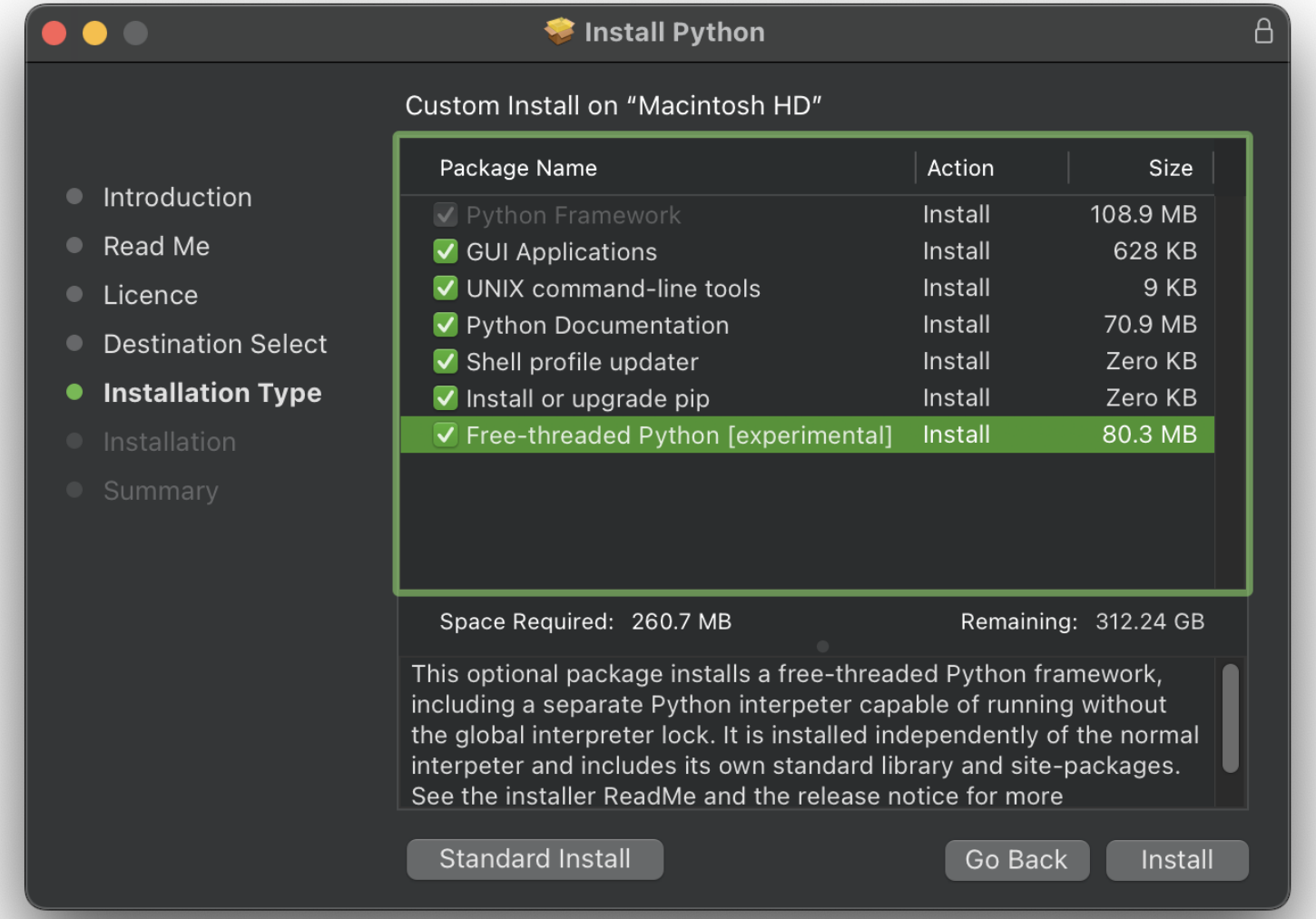
- The GIL removal is an experiment and may be abandoned in the future
- But the ideal future outcome is having just one, GIL-free Python

```
$ pyenv install 3.13t

$ pyenv local 3.13t
```

```
$ python -VV
Python 3.13.0 experimental free-threading build (main, Oct 11 2024, 14:20:47) [GCC 12.2.0]
```

```
$ uv venv --python 3.13t
```


Install Python
Introduction
Read Me
Licence
Destination Select
Installation Type
Installation
Summary
Custom Install on “Macintosh HD”
Package Name
Action
Size
Python Framework
Install
108.9 MB
GUI Applications
Install
628 KB
UNIX command-line tools
Install
9 KB
Python Documentation
Install
70.9 MB
Shell profile updater
Install
Zero KB
Install or upgrade pip
Install
Zero KB
Free-threaded Python [experimental]
Install
80.3 MB
Space Required: 260.7 MB
Remaining: 312.24 GB
This optional package installs a free-threaded Python framework, including a separate Python interpeter capable of running without the global interpreter lock. It is installed independently of the normal interpeter and includes its own standard library and site-packages. See the installer ReadMe and the release notice for more
Standard Install
Go Back
Install

The catch: many extensions don’t install or they still require the GIL

- Special builds needed for the free-threading Python ABI
- GIL is re-enabled when importing an incompatible extension

Package compatibility tracker: Python 3.13 free-threading and subinterpreters

Package	Test date	Tested version	Free threading		Subinterpreters	
			Installs	Imports without the GIL	Installs	Imports in a subinterpreter
boto3	2024-10-11	1.35.38	yes	yes	yes	yes
urllib3	2024-10-11	2.2.3	yes	yes	yes	yes
botocore	2024-10-11	1.35.38	yes	yes	yes	yes
requests	2024-10-11	2.32.3	yes	yes	yes	yes
setuptools	2024-10-11	75.1.0	yes	yes	yes	yes
certifi	2024-10-11	2024.8.30	yes	yes	yes	yes
idna	2024-10-11	3.10	yes	yes	yes	yes
charset-normalizer	2024-10-11	3.4.0	yes	yes	yes	no
typing-extensions	2024-10-11	4.12.2	yes	yes	yes	yes

https://parallel.python.tips

Common obstacles for subinterpreters and free-threading

- Global state and thread safety
- The GIL has allowed us to cut corners for many years
- True parallelism is hard

The future looks bright

Viable paths forward + strong motivation + momentum + excitement = ❤️

The future looks bright *and parallel*

excitement❤️ Viable paths motivation strong + =momentum forward + +

Thank you!

Parallel Python tracker & these slides — parallel.python.tips

Feedback — hi@vitamidori.com