What’s new in Python 3.11

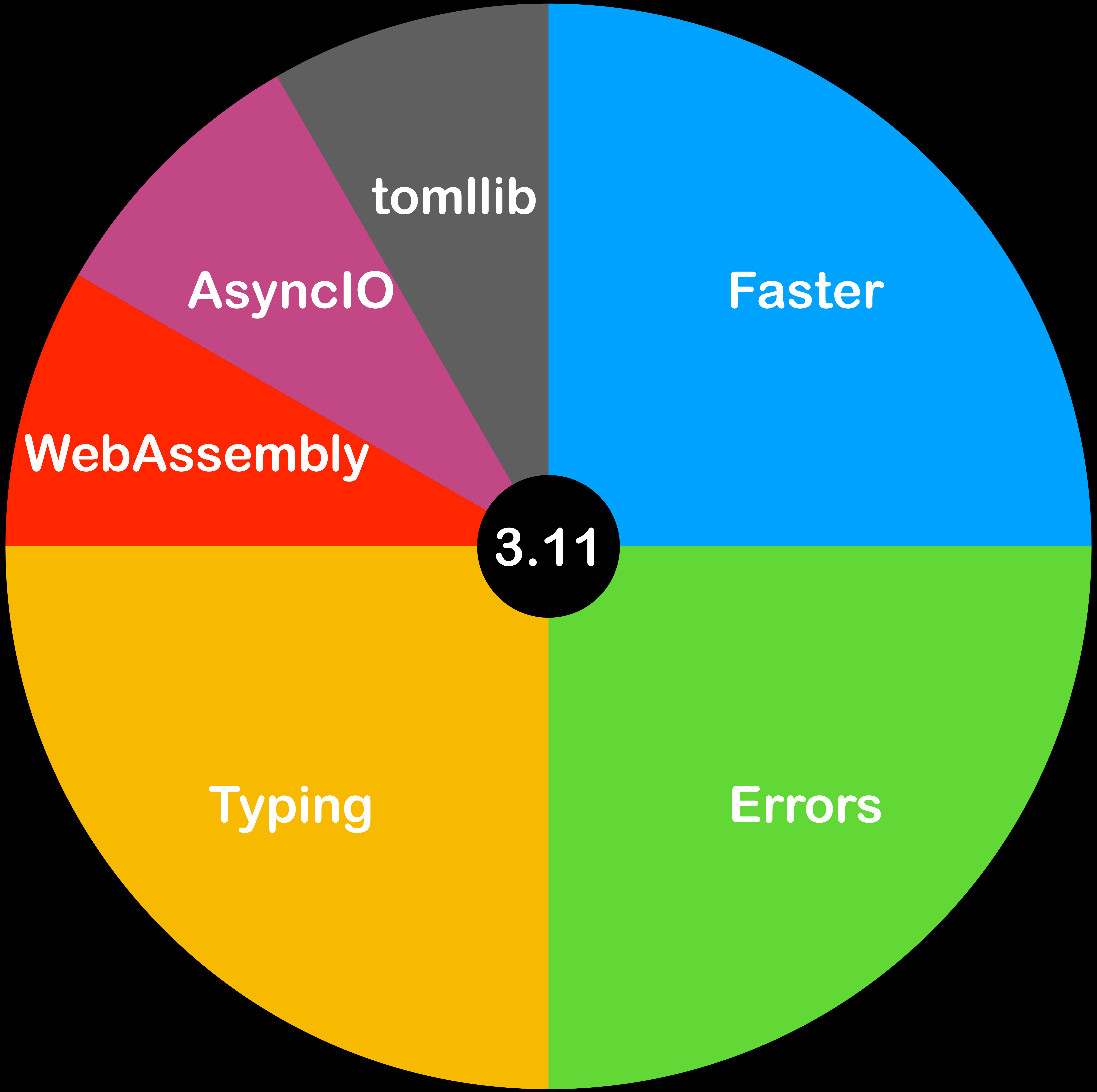
Henry Schreiner - PyHEP 2022

⚡

https://iscinumpy.dev/post/python-311

Final

release candidate available now!



Coming Oct 3, 2022!

Faster CPython Project

5%-60% faster - 25% on average!

3.11

Zero cost exceptions (if not thrown)

10% faster re & atomic grouping, possessive qualifiers

10-15% faster startup

Faster function calls

C-style formatting sometimes as fast as f-string

Less memory for string keys in dicts

Specializing adaptive interpreter

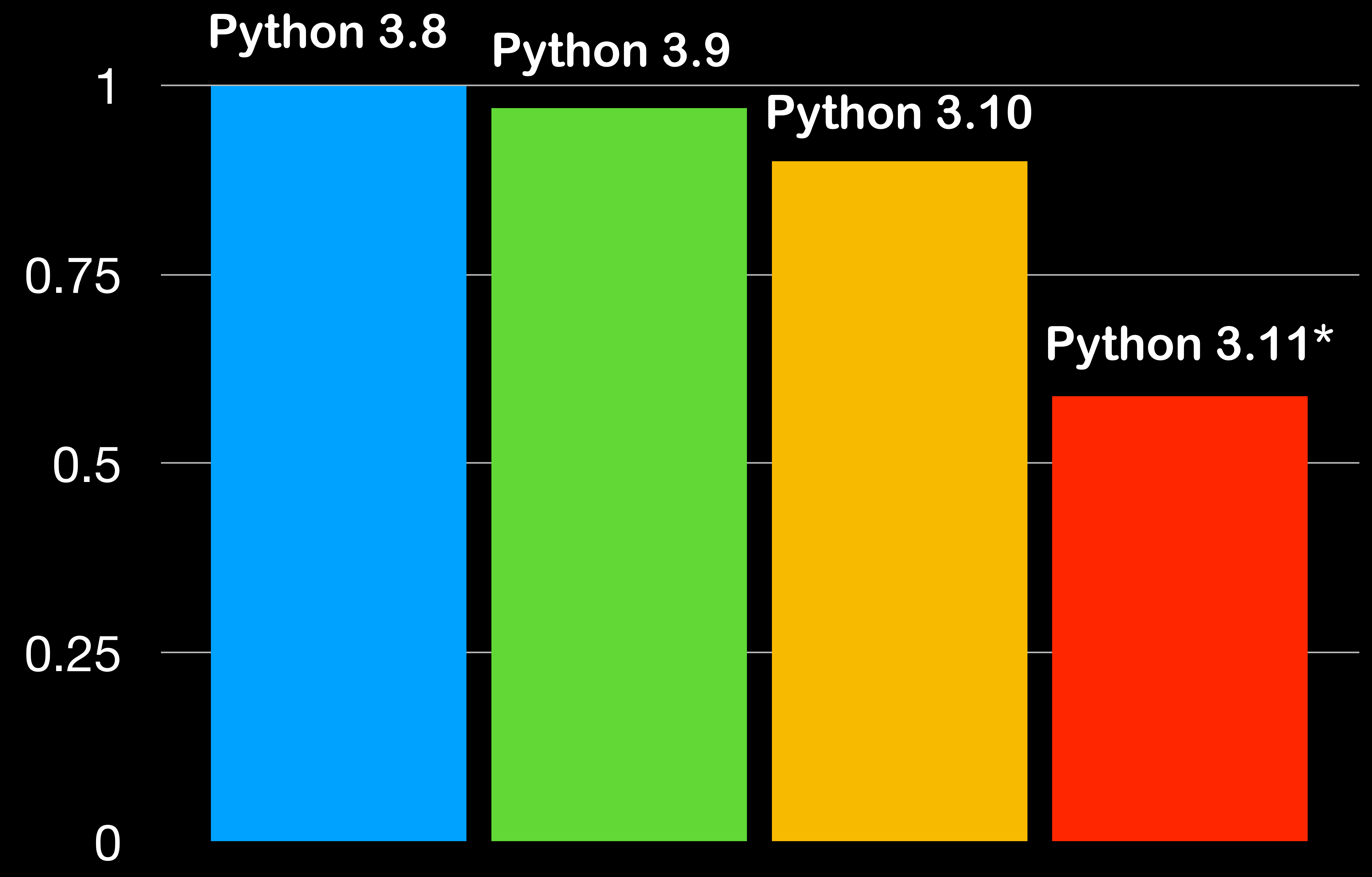
And more!

Future

Major focus for the next several releases

Simple JIT planned eventually

The main driver behind C API changes



Raytracing example, from speed.python.org
*: Python master Sep 8, 2022
(may include some 3.12 speedups)

Better Error messages

Before 3.11

```
>>> obj = {"a": {"b": None}}
>>> obj["a"]["b"]["c"]["d"]
Traceback (most recent call last):
  File "tmp.py", line 3, in <module>
    obj["a"]["b"]["c"]["d"]
TypeError: 'NoneType' object is not subscriptable
```

Better Error messages

After 3.11

```
>>> obj = {"a": {"b": None}}
>>> obj["a"]["b"]["c"]["d"]
Traceback (most recent call last):
  File "tmp.py", line 3, in <module>
    obj["a"]["b"]["c"]["d"]
    ~~~~~~~~~~~~~^^^^^
TypeError: 'NoneType' object is not subscriptable
```

Exception notes

```
try:
    import skbuild
except ModuleNotFoundError as err:
    err.add_note("Please pip or conda install 'scikit-build', or upgrade pip.")
    raise
```

```
Traceback (most recent call last):
  File "tmp.py", line 2, in <module>
    import skbuild
    ^^^^^^^^^^^^^^
ModuleNotFoundError: No module named 'skbuild'
Please pip or conda install 'scikit-build', or upgrade pip.
```

Note!

Also

`sys.exception()`

Exception groups

```python
try:
    raise ExceptionGroup(
        "multiexcept",
        [TypeError(1), KeyError("two")],
    )
except* TypeError as err:
    print("Got", *(repr(r) for r in err.exceptions))
except* KeyError as err:
    print("Got", *(repr(r) for r in err.exceptions))
```

↓

```
Got TypeError(1)
Got KeyError('two')
```

Backported for 3.7+ as
`pip install exceptiongroups`
(minus the new syntax)

Already used by cattrs!

```
  +-+---------------- 1 ----------------
    | Exception Group Traceback (most recent call last):
    |   File "<cattrs generated structure scikit_build_core.file_api.model.index.Index>", line 15, in structure_Index
    |     res['reply'] = __c_structure_reply(o['reply'], __c_type_reply)
    |   File "<cattrs generated structure scikit_build_core.file_api.model.index.Reply>", line 24, in structure_Reply
    |     if errors: raise __c_cve('While structuring Reply', errors, __cl)
    | cattrs.errors.ClassValidationError: While structuring Reply (2 sub-exceptions)
    | Structuring class Index @ attribute reply
    +-+---------------- 1 ----------------
      | Traceback (most recent call last):
      |   File "<cattrs generated structure scikit_build_core.file_api.model.index.Reply>", line 10, in structure_Reply
      |     res['cache_v2'] = __c_structure_cache_v2(o['cache_v2'], __c_type_cache_v2)
      | KeyError: 'cache_v2'
      | Structuring class Reply @ attribute cache_v2  ← Note!
      +---------------- 2 ----------------
      | Traceback (most recent call last):
      |   File "<cattrs generated structure scikit_build_core.file_api.model.index.Reply>", line 15, in structure_Reply
      |     res['cmakefiles_v1'] = __c_structure_cmakefiles_v1(o['cmakefiles_v1'], __c_type_cmakefiles_v1)
      | KeyError: 'cmakefiles_v1'
      | Structuring class Reply @ attribute cmakefiles_v1  ← Note!
```

All conversion errors are collected and shown at once!

(This example is using the backport and Python 3.7-3.10)

AsyncIO: TaskGroups

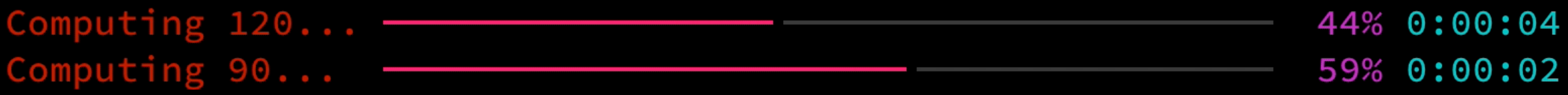
Enabled by ExceptionGroups!

```python
from rich.progress import Progress
import asyncio

async def lots_of_work(n: int, progress: Progress) -> None:
    for i in progress.track(range(n), description=f"[red]Computing {n}..."):
        await asyncio.sleep(0.05)

async def main():
    with Progress() as progress:
        async with asyncio.TaskGroup() as g:
            g.create_task(lots_of_work(120, progress))
            g.create_task(lots_of_work(90, progress))

asyncio.run(main())
```



```
Computing 120...                                              44% 0:00:04
Computing 90...                                               59% 0:00:02
```

Static Typing

Variadic Generics

Great for ND arrays

`Array[float, width, height]`

LiteralString

SQL Injection protection

f-strings propagate literalness

Self type

Works for class methods too!

`def unit(self) -> Self:`

Exhaustiveness checking

Perfect for pattern matching

`typing.assert_never(x)`

And more!

Dataclass transforms

(Not)Required for TypedDict

`typing.reveal_type()`

Generic TypedDict/NamedTuple

Any is subclassable

`typing.assert_type()`

Tomllib / WebAssembly

Reading TOML files in the stdlib!
Identical to tomli
Still use tomli-w to write TOML

Run Python in your browser!
Currently support level tier 3
Tier 2 (binaries) planned for 3.12!

Will also make Pyodide -> PyScript easier to support
(and the webassembly forge project, etc)

Other things

`contextlib.chdir()`

`PYTHONSAFEPATH`

`operator.call()`

functools.single_dispatch supports Unions

* unpacking inside for

More Python 2 compat removals

PyBuffer added to Limited API / Stable ABI

New deprecations of rarely used modules

Support

Python is ABI stable RC 1

cibuildwheel optional support in betas

Full default support started RC 1

NumPy has already released binaries
(due to now using cibuildwheel)

SciPy might release binaries in a (few) week(s)
(due to moving to cibuildwheel)

AwkwardArray has shipped binaries
(due to cibuildwheel + numpy)

Matplotlib is shipping binaries soon
(due to using cibuildwheel)

GitHub Actions has been shipping dev versions
We are often testing on 3.11-dev!

Many of these libraries have never shipped wheels before a Python release!

More of the ecosystem will be ready on release day than ever before!

The future

3.12 will remove distutils

Avoid direct usage of distutils
Consider a modern unaffected builder, like Hatchling
Watch the scikit-build project for compiling!

3.15 will make unicode default!

Always specify an encoding with any open function
You can use “native” encoding in 3.11+ (but, why?)

```
open(…, encoding="utf-8")
Path(…).read_txt(encoding="utf-8")
```