Pyrefly

A Python typechecker

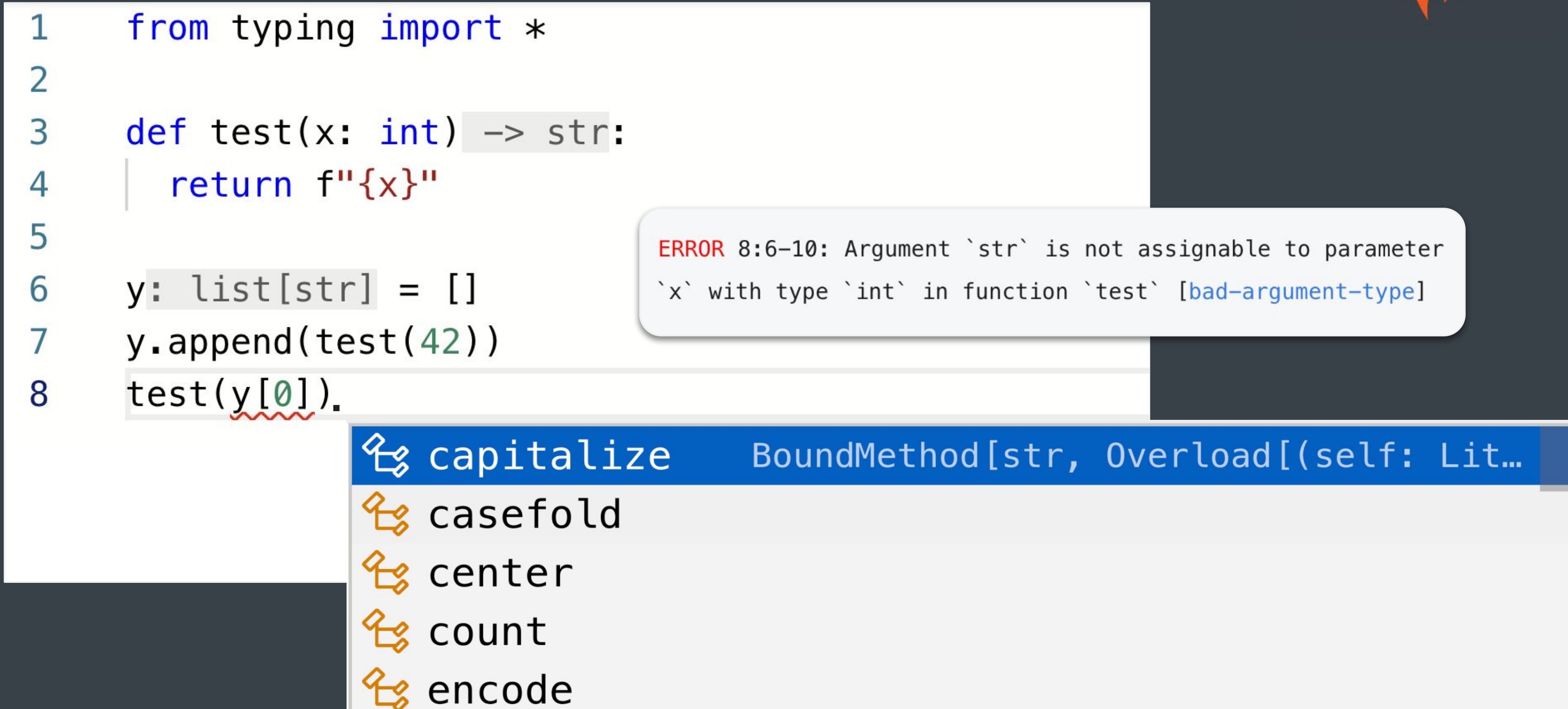
Neil Mitchell, Meta

What is Pyrefly?

- An open-source standards-compliant Python type checker
- An IDE/LSP provider
- Fast and parallel (written in Rust)
- The successor to Pyre

pyrefly.org

Sandbox (pyrefly.org/sandbox)



```
from typing import *

def test(x: int) -> str:
    return f"{x}"

y: list[str] = []
y.append(test(42))
test(y[0]).
```

```
ERROR 8:6-10: Argument `str` is not assignable to parameter
`x` with type `int` in function `test` [bad-argument-type]
```

```
capitalize    BoundMethod[str, Overload[(self: Lit…
casefold
center
count
encode
```

How to get it?

- Alpha version available now!
- **`pip install pyrefly && pyrefly init`**
- VS Code extension (search for “pyrefly”)

pyrefly.org

History of Pyrefly

- Meta develops Instagram which is a massive codebase of Python
- In 2017 we started work on Pyre
  - Descendent of Hack (PHP) and Flow (Javascript)
  - Written in OCaml
- Very useful! But…
  - Didn’t work on Windows
  - Parallelism was hard (multiprocess)
  - Performance a bit lacking
  - IDE was lackluster, switched to Pyright
  - Open source was never a focus

History of Pyrefly (2)

- August 2024 two of us started prototyping MiniPyre
  - 7 prototypes written, constraints, subset based, abstract interpretation…
  - Using Rust (cross platform and fast)
  - Hard bits first: generics, recursion, overloads, `import *`
- October 2024 it was working well, so we started ~~Pyre2~~ Pyrefly
  - Implement features, following the typing spec
  - Implement LSP
  - Package
  - Optimise
- May 2025 we released it at PyConUS 2025

Features 1/4 - Performance

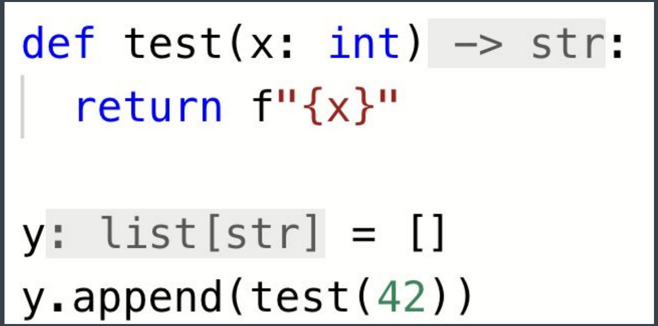
- Performance is a feature! 🚀
  - No trade off between safety and developer speed
  - Check on every keystroke - 1.8M lines/second*, 35x faster than Pyre (on Instagram), 14x faster than Mypy/Pyright (on Pytorch)**
- Fast as standard - Rust, memory representations
- Parallelism - at the module level, so larger projects go faster
- Incrementality - don’t invalidate too much, even with cyclic imports

Fight the *$O(n^2)$* monsters!

* On my Meta Linux dev machine, 166 cores, 228Gb RAM
** On a Macbook, 10 cores, 32Gb RAM

Features 2/4 - Inference

- I like types (my DNA is Haskell). Some people don’t.
- Pyrefly is designed to meet you where you are!
- Infer function return types
- Infer local types
- Infer container types



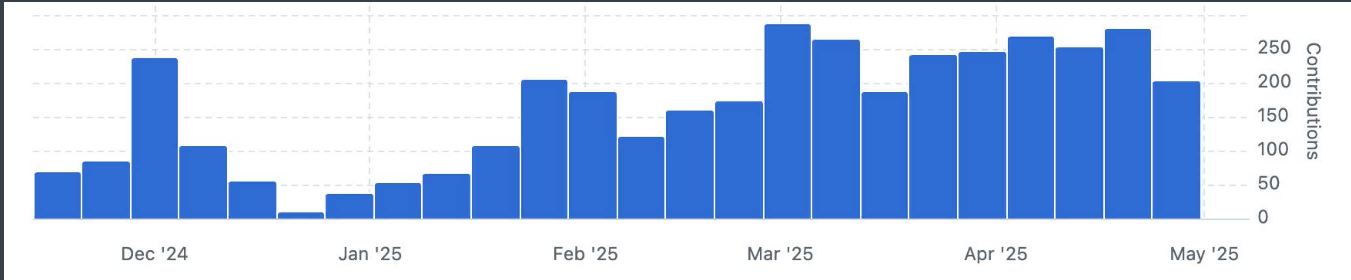
```
def test(x: int) -> str:
    return f"{x}"

y: list[str] = []
y.append(test(42))
```

Features 3/4 - IDE

- Designed as an IDE, that can also run on the command line
- In-memory transactional database to manage state
- VS Code extension, follows LSP (used on NeoVim too)
- Hover, goto-def, completions, find-refs, document symbols…
- Type inference: return types and container types
- Inlay hints - easily insert inferred types

Features 4/4 - Open Source

- We have gained much from open source!
  - Python itself
  - Python typing specification, plus existing checkers (Pyright, Mypy etc)
  - Ruff parser (really awesome - thanks!)
  - Open source Python projects, e.g. PyTorch
- MIT license, https://github.com/facebook/pyrefly
- Delighted to accept pull requests (5 last week), all issues are on issue tracker



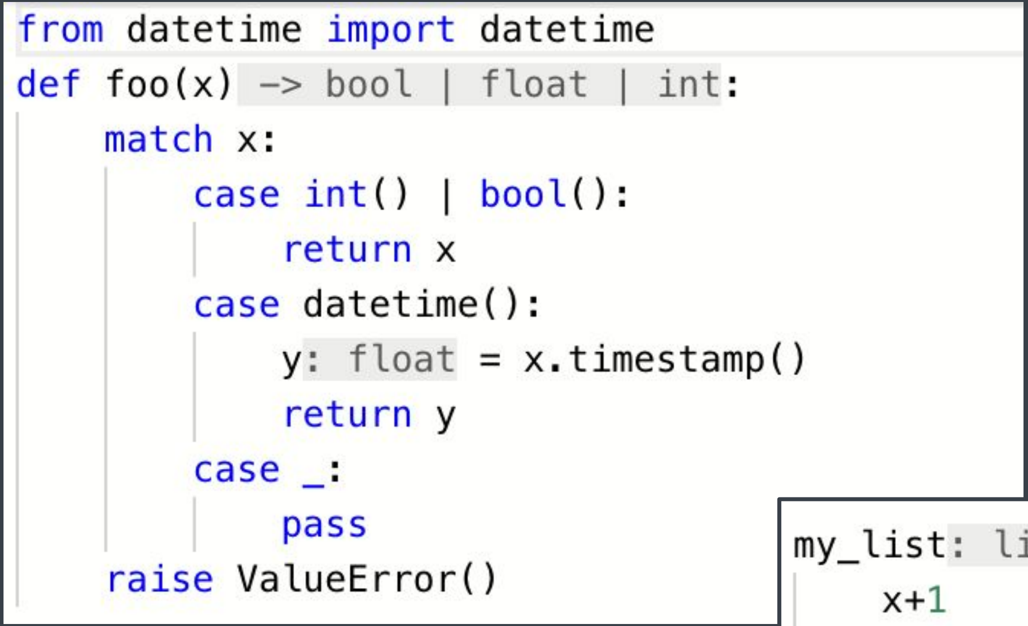
But Python is untyped?

- At runtime Python has types (**`str`** ≠ **`int`**)
- For developers, Python has types “the user identifier”
- If these don’t agree, your program will not have a good time

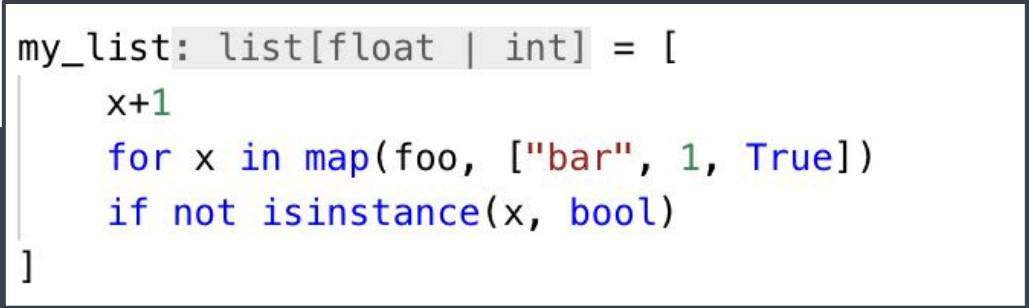
Python types (including annotations) let you connect between human types and interpreter types

Why types?

- Faster inner loop - run the code less
- Spot typos
- Make corner cases safer
- Understand the code better, documentation, goto-def
- Write code faster - auto-completion
- Machine checked documentation
- Refactor with peace of mind



```python
from datetime import datetime
def foo(x) -> bool | float | int:
    match x:
        case int() | bool():
            return x
        case datetime():
            y: float = x.timestamp()
            return y
        case _:
            pass
    raise ValueError()
```



```python
my_list: list[float | int] = [
    x+1
    for x in map(foo, ["bar", 1, True])
    if not isinstance(x, bool)
]
```

Why types?

- Reliability
- Productivity
- Understanding
- Where it makes sense!

```
final case class Kleisli[F[_], A, B](run: A => F[B]) {
  def map[C](f: B => C)(implicit F: Functor[F]): Kleisli[F, A, C] =
    Kleisli[F, A, C](a => F.map(run(a))(f))
}
```

Pyrefly design

3 phases! Each about 10x more expensive than the previous one

- Exports - what does each module export
  - Module **`foo`** exports **`builtins.str`** and **`MyClass`**
- Bindings - how do statements relate to each other
  - **`x`** on line 7 is defined at line 3
  - **`y`** is assigned to **`x.pop(4)`**
- Answers - how expressions/types relate
  - **`x`** is **`list[str]`**, **`4`** is **`Literal[4]`**, what is **`x.pop(4)`**

The journey of autocomplete

```
display(3.142).fraction.
```

```
from typing import *
from numbers import *
```

- Find the type of **`display(3.142).fraction`**
- First, find **`display`**
  - Might come from **`typing`** or **`numbers`**
  - Figure out the export table from each
  - Which might require a fixed-point of recursive **`*`** imports…

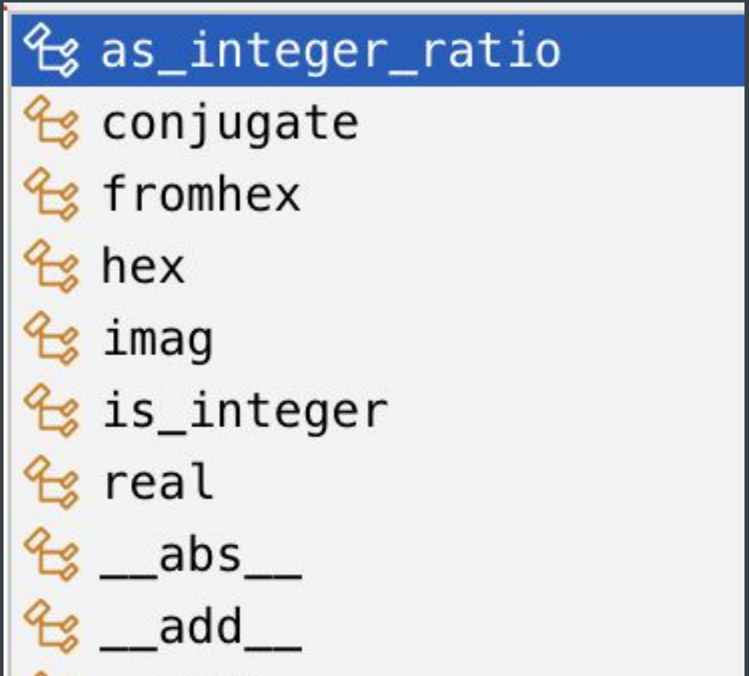
The journey of autocomplete (2)

```
@dataclass
class Number[T]:
    whole: T
    fraction: Final[T]


def display(x: float) -> Number[float]:
    whole = float(math.floor(x))
    fractional = x - whole
    return Number(whole, fractional)
```

- Interpret **`@dataclass`**
- Infer types for each variable
- Infer the return type
- Instantiate some generics
- Understand **`Final`**

The journey of autocomplete (3)

```
display(3.142).fraction.
```



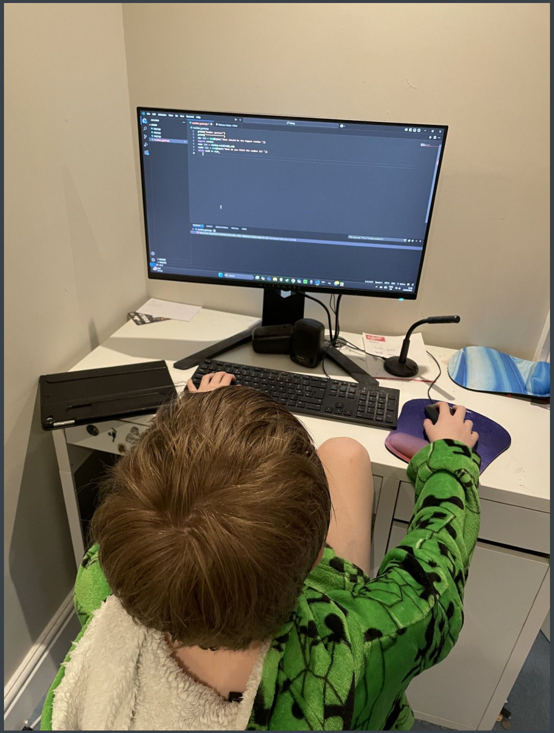
```
as_integer_ratio
conjugate
fromhex
hex
imag
is_integer
real
__abs__
__add__
```

```
class float:
    def __new__(cls, x = ...) -> Self
    @classmethod
    def fromhex(cls, x: str) -> Self
    @property
    def real(self) -> float
    def conjugate(self) -> float
    def __add__(self, float) -> float
```

- Now we know we have **`float`**
- Figure out what methods it has

Why not Pyrefly?

- It is an alpha - 25 known bugs, ∞ unknown
- You will be one of the first open-source users
- You will find bugs, most of which we will fix
- But you will get a sticker



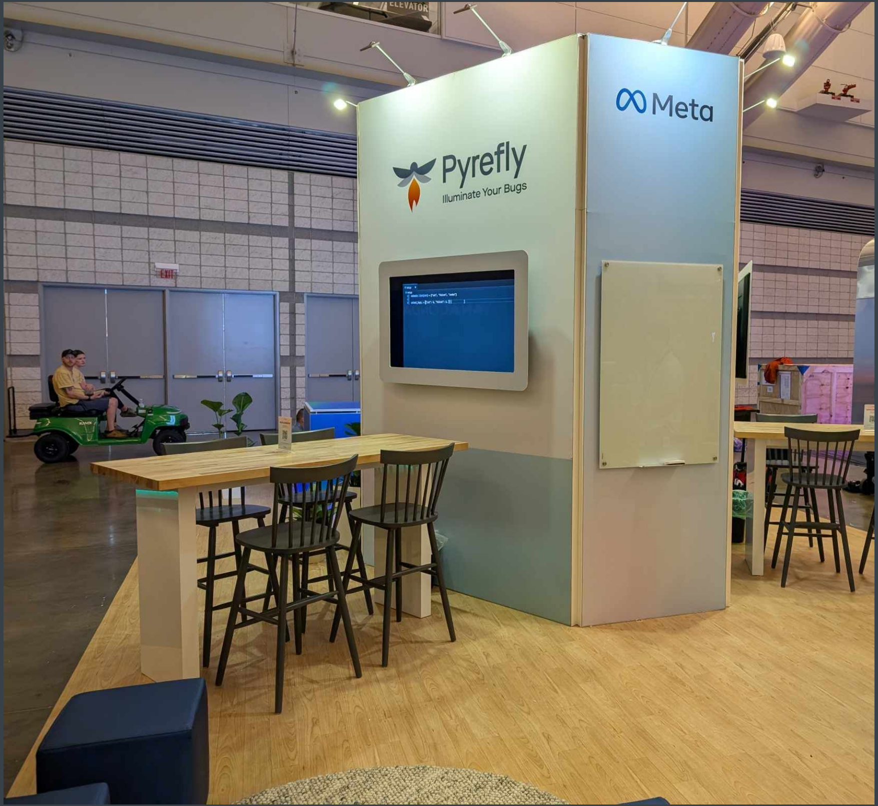
Pyrefly @ PyConUS 2025

Say hello!

We'll be here for the conference and the sprints.

Happy to help!

Pyrefly @ PyConUS 2025



Booth to the right of the main entrance

Look for the 'Meta Open Source'

Pyrefly @ PyConUS 2025

Typing summit! Tomorrow 2-6pm, room 319, all welcome! (no pre-registration)

- Introducing Pyrefly, Steven Troxler
- Preventing unwanted mutation with **`PyreReadOnly`**, Amritha Raghunath and Jia Chen
- Updates from the Typing Council, Rebecca Chen

Questions?

pyrefly.org