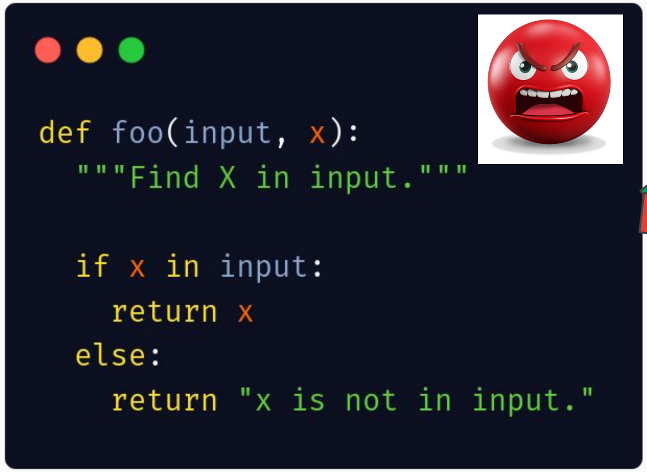
PyTifeX

An Empirical Study on Static Analyzer Toolsets to Reduce False Positives, False Negatives in Python Type Checkers

Benedek Kaibas, Allegheny College

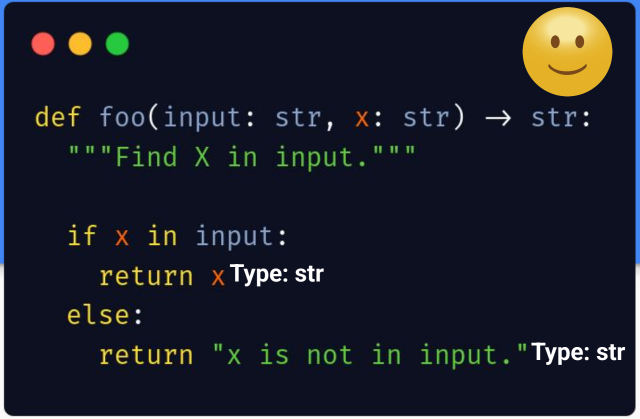
What Are Type Checkers?

Type Checking



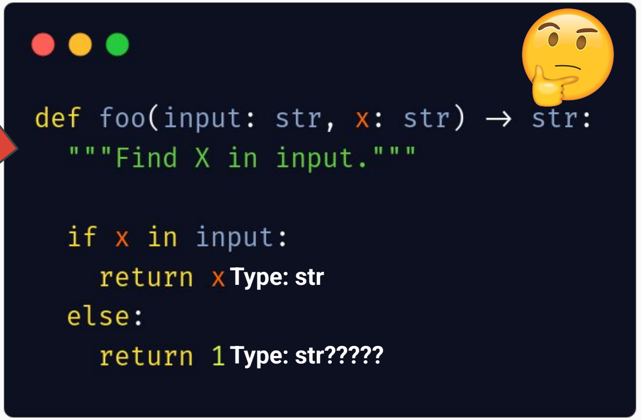
```
def foo(input, x):
  """Find X in input."""

  if x in input:
    return x
  else:
    return "x is not in input."
```



```
def foo(input: str, x: str) → str:
  """Find X in input."""

  if x in input:
    return x Type: str
  else:
    return "x is not in input." Type: str
```



```
def foo(input: str, x: str) → str:
  """Find X in input."""

  if x in input:
    return x Type: str
  else:
    return 1 Type: str?????
```

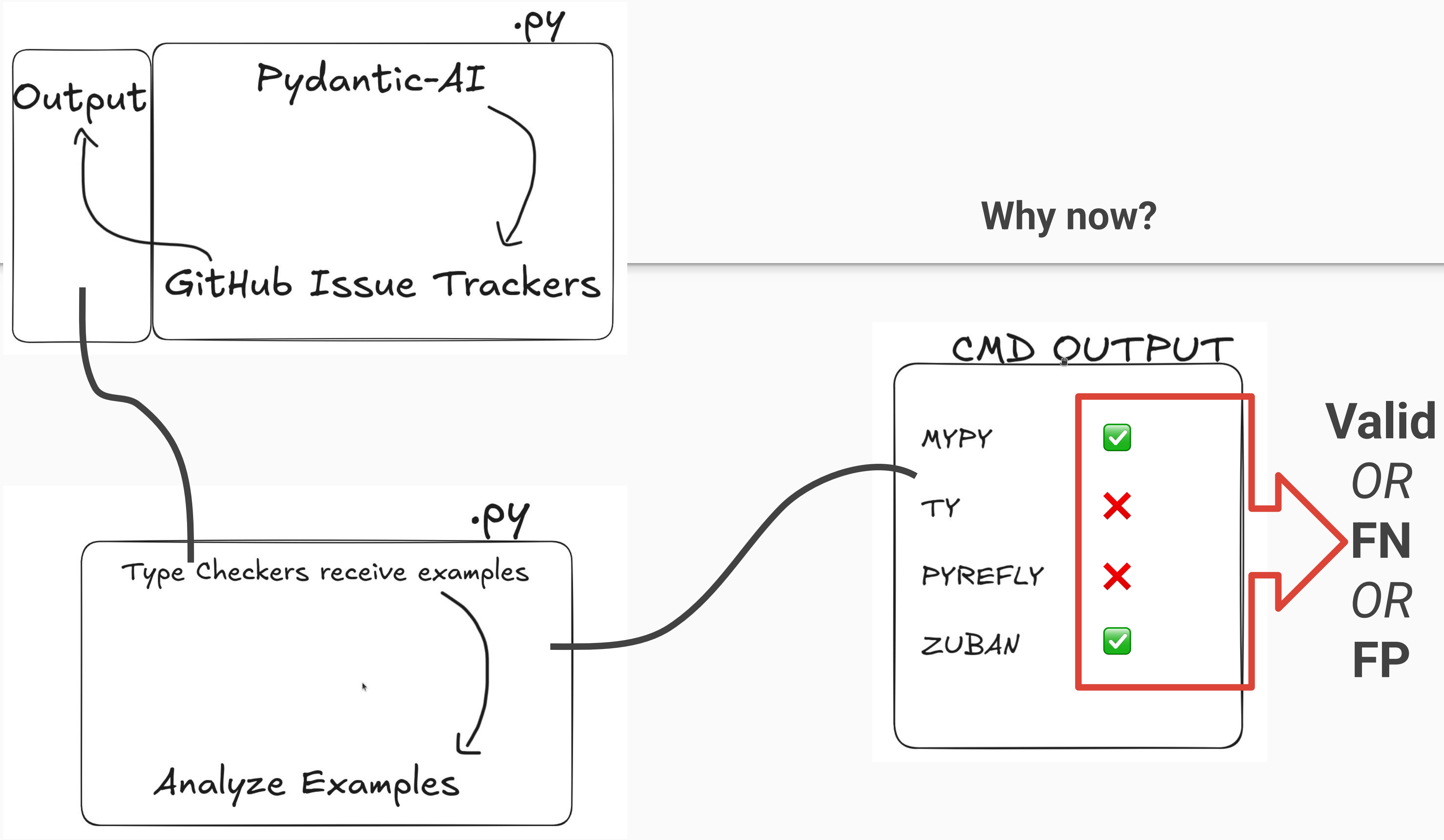
What Is PyTifeX?

Overview

1. Automated Code Generation
2. Interaction with LLMs
3. Analyze Type Checkers

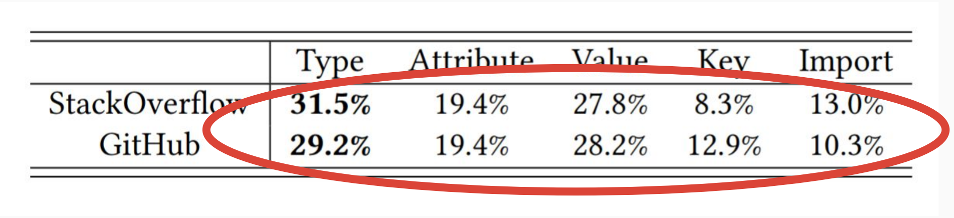
How Does The Tool Work?

Why now?


.py
Output
Pydantic-AI
GitHub Issue Trackers
.py
Type Checkers receive examples
Analyze Examples
CMD OUTPUT
MYPY ✓
TY ✗
PYREFLY ✗
ZUBAN ✓
Valid
OR
FN
OR
FP

What’s The Problem We Face?

Parts Of The Big Problem



	Type	Attribute	Value	Key	Import
StackOverflow	**31.5%**	19.4%	27.8%	8.3%	13.0%
GitHub	**29.2%**	19.4%	28.2%	12.9%	10.3%

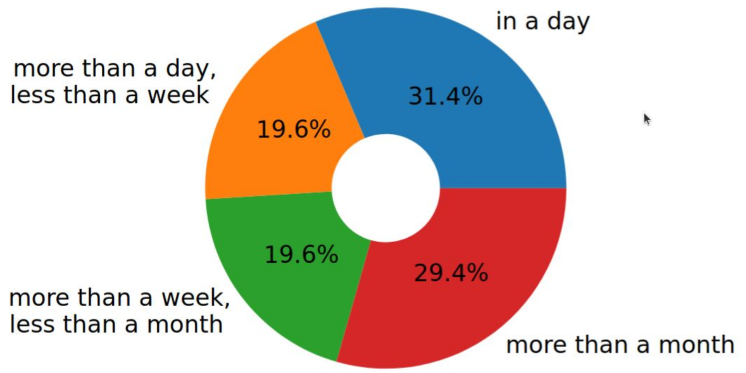
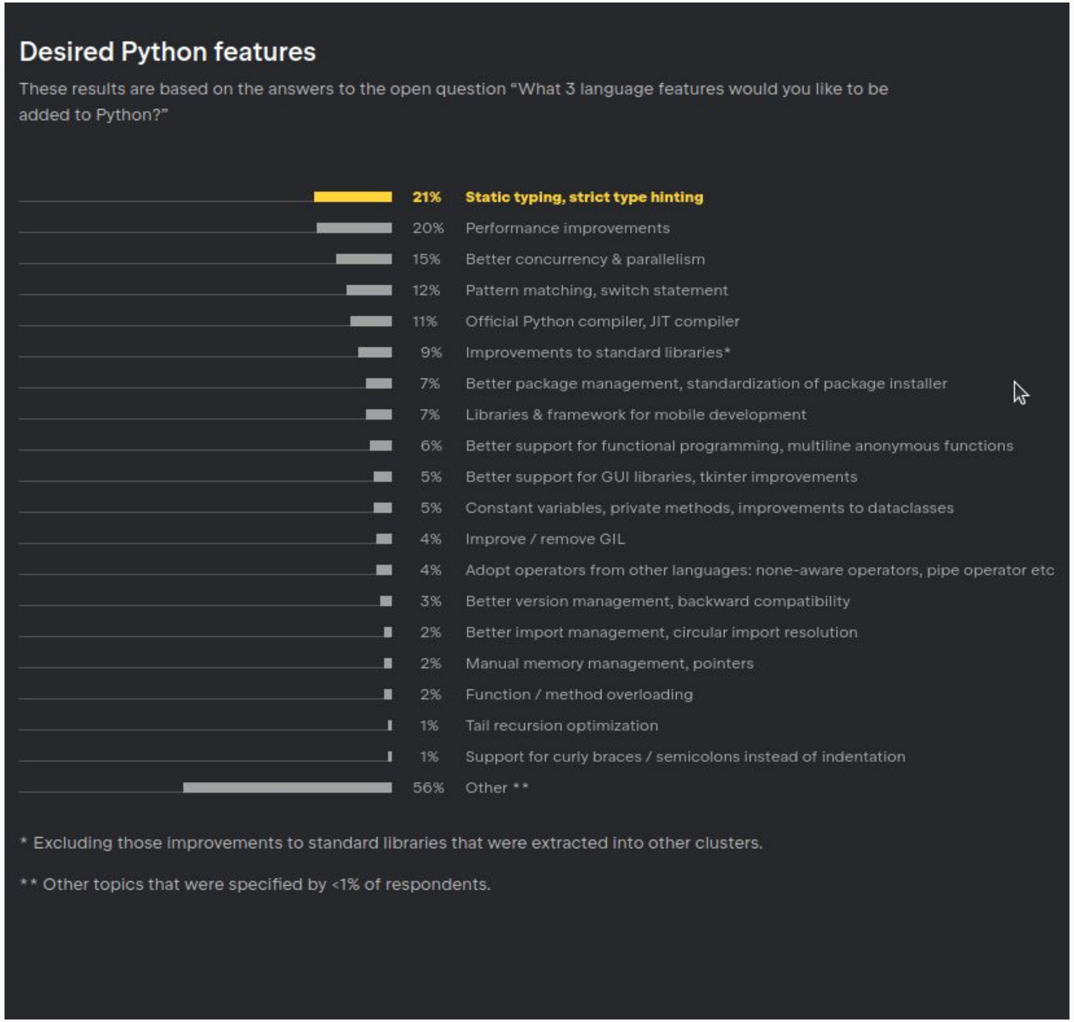


Figure 1: Statistics on the time period between reporting and patching a type error. For bugs in our benchmarks (Section 4), it took 82 days on average and about 30% took more than a month. The worst case took 1,277 days, more than three years.

What Do Developers Need?

JetBrains Survey

1. **Typing is important**
2. **Does my tool help?**



Desired Python features

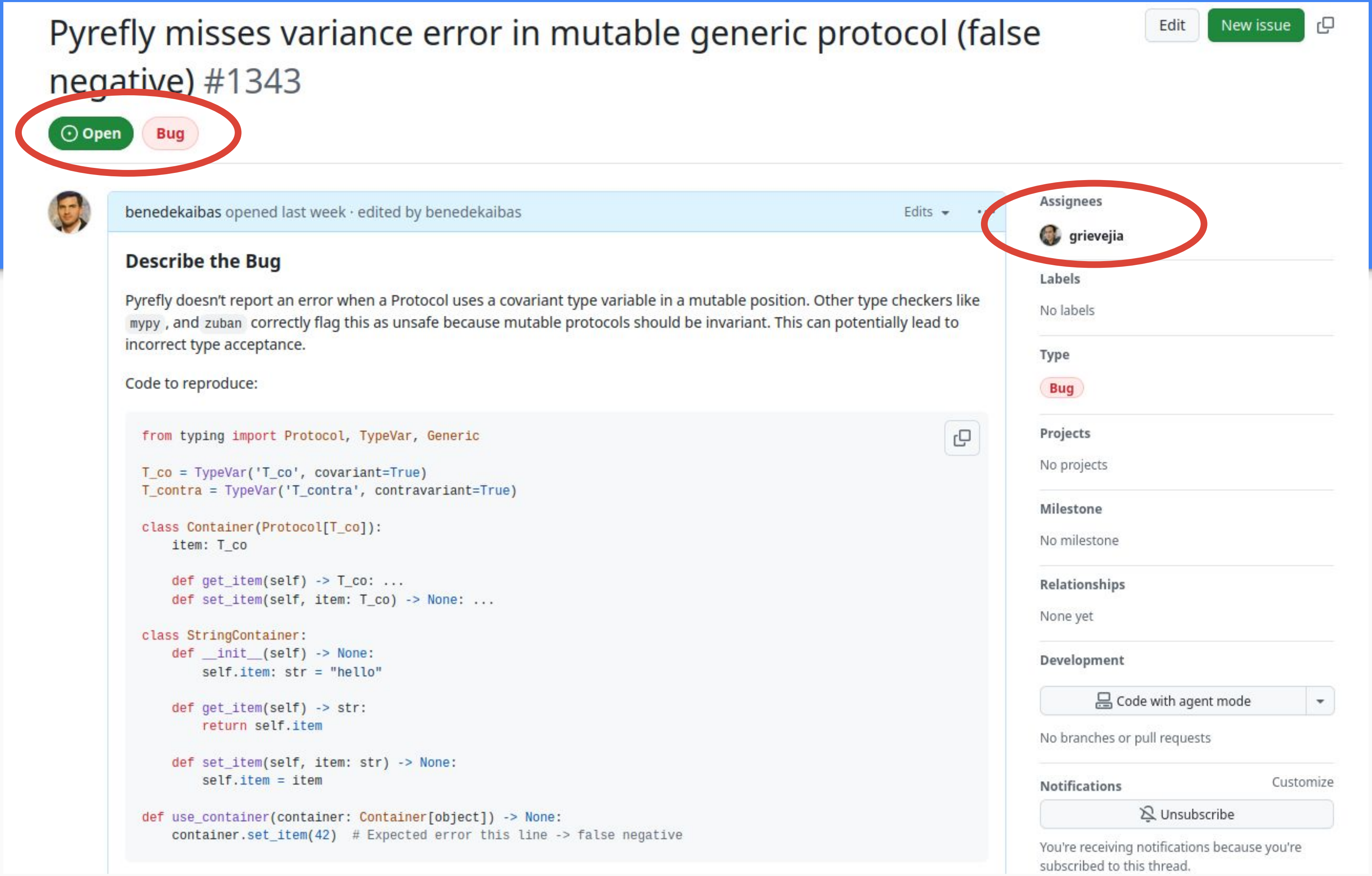
These results are based on the answers to the open question "What 3 language features would you like to be added to Python?"

21%	**Static typing, strict type hinting**
20%	Performance improvements
15%	Better concurrency & parallelism
12%	Pattern matching, switch statement
11%	Official Python compiler, JIT compiler
9%	Improvements to standard libraries*
7%	Better package management, standardization of package installer
7%	Libraries & framework for mobile development
6%	Better support for functional programming, multiline anonymous functions
5%	Better support for GUI libraries, tkinter improvements
5%	Constant variables, private methods, improvements to dataclasses
4%	Improve / remove GIL
4%	Adopt operators from other languages: none-aware operators, pipe operator etc
3%	Better version management, backward compatibility
2%	Better import management, circular import resolution
2%	Manual memory management, pointers
2%	Function / method overloading
1%	Tail recursion optimization
1%	Support for curly braces / semicolons instead of indentation
56%	Other **

* Excluding those improvements to standard libraries that were extracted into other clusters.

** Other topics that were specified by <1% of respondents.

Will PyTifeX Help?



Pyrefly misses variance error in mutable generic protocol (false negative) #1343

benedekaibas opened last week · edited by benedekaibas

Describe the Bug

Pyrefly doesn't report an error when a Protocol uses a covariant type variable in a mutable position. Other type checkers like `mypy`, and `zuban` correctly flag this as unsafe because mutable protocols should be invariant. This can potentially lead to incorrect type acceptance.

Code to reproduce:

```python
from typing import Protocol, TypeVar, Generic

T_co = TypeVar('T_co', covariant=True)
T_contra = TypeVar('T_contra', contravariant=True)

class Container(Protocol[T_co]):
    item: T_co

    def get_item(self) -> T_co: ...
    def set_item(self, item: T_co) -> None: ...

class StringContainer:
    def __init__(self) -> None:
        self.item: str = "hello"

    def get_item(self) -> str:
        return self.item

    def set_item(self, item: str) -> None:
        self.item = item

def use_container(container: Container[object]) -> None:
    container.set_item(42)  # Expected error this line -> false negative
```

Assignees
grievejia

Labels
No labels

Type
Bug

Projects
No projects

Milestone
No milestone

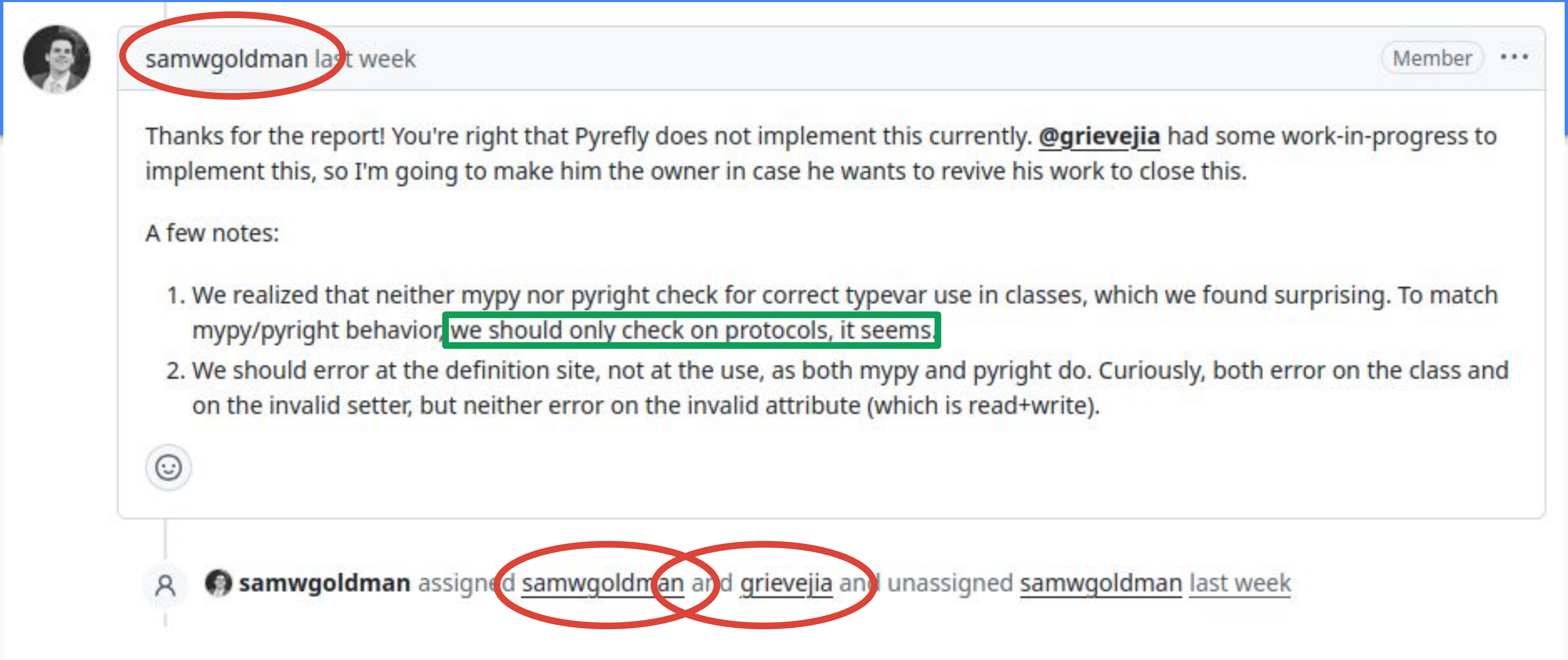
Relationships
None yet

Development

No branches or pull requests

Notifications

You're receiving notifications because you're subscribed to this thread.



samwgoldman last week

Member

Thanks for the report! You're right that Pyrefly does not implement this currently. **@grievejia** had some work-in-progress to implement this, so I'm going to make him the owner in case he wants to revive his work to close this.

A few notes:

1. We realized that neither mypy nor pyright check for correct typevar use in classes, which we found surprising. To match mypy/pyright behavior, we should only check on protocols, it seems.
2. We should error at the definition site, not at the use, as both mypy and pyright do. Curiously, both error on the class and on the invalid setter, but neither error on the invalid attribute (which is read+write).

samwgoldman assigned samwgoldman and grievejia and unassigned samwgoldman last week

Demo

DEMO I: What you will see?

1. Generating code examples

2. Feeding code examples into Type Checkers

3. Evaluate Type Checkers

DEMO II: Let’s make some BUGS!!!

DEMO III: What happened?

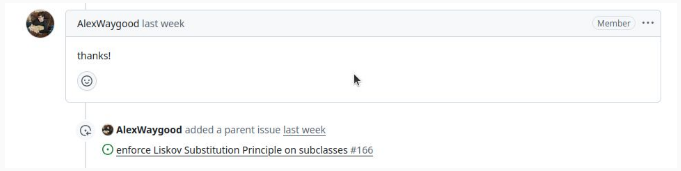
1. We created code examples
2. Forced Type Checkers to report either FN or FP
3. Analyzed output (Human and LLM)

Conclusion and Future Work

Is type checking still hard? → **Yes**

Does my tool help? →

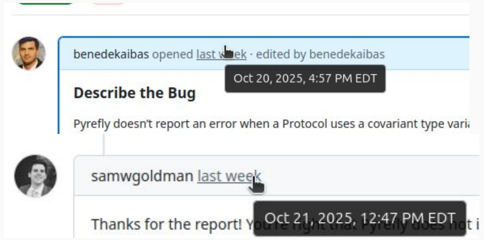
Do we reduce the time time? →


AlexWaygood last week Member

thanks!

AlexWaygood added a parent issue last week

enforce Liskov Substitution Principle on subclasses #166


benedekaibas opened last week · edited by benedekaibas

Oct 20, 2025, 4:57 PM EDT

Describe the Bug

Pyrefly doesn't report an error when a Protocol uses a covariant type vari

samwgoldman last week

Oct 21, 2025, 12:47 PM EDT

Thanks for the report! Y

1. Contribute with TC developer teams
2. Generate cluster of bugs
3. Be general!