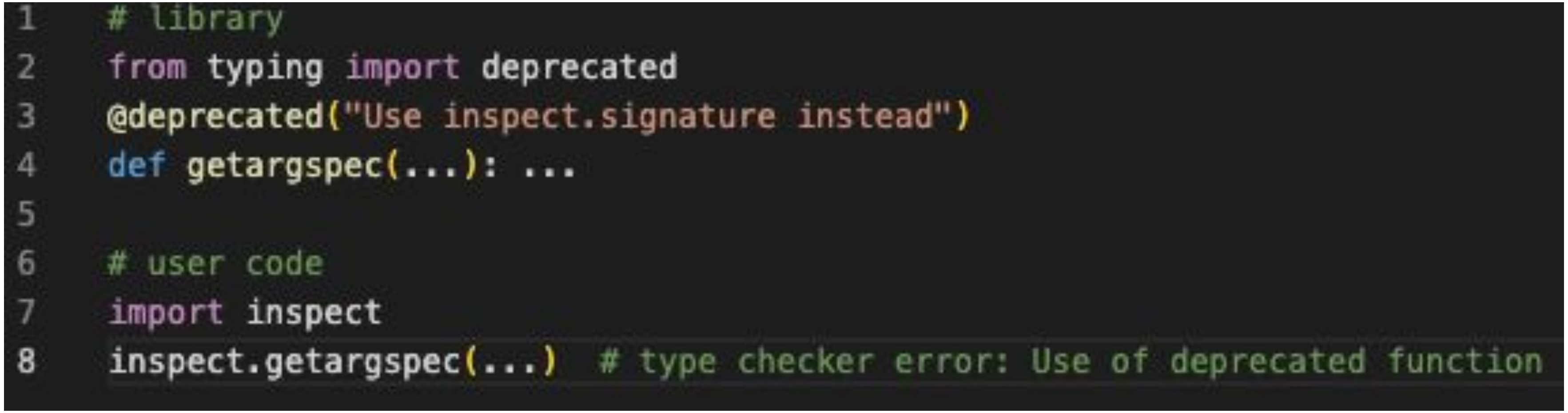
Typing support for deprecations and errors

Jelle Zijlstra

Quora

@typing.deprecated(...)

- Mark a class, function, or overload as deprecated
- Checkers should issue a diagnostic on usage

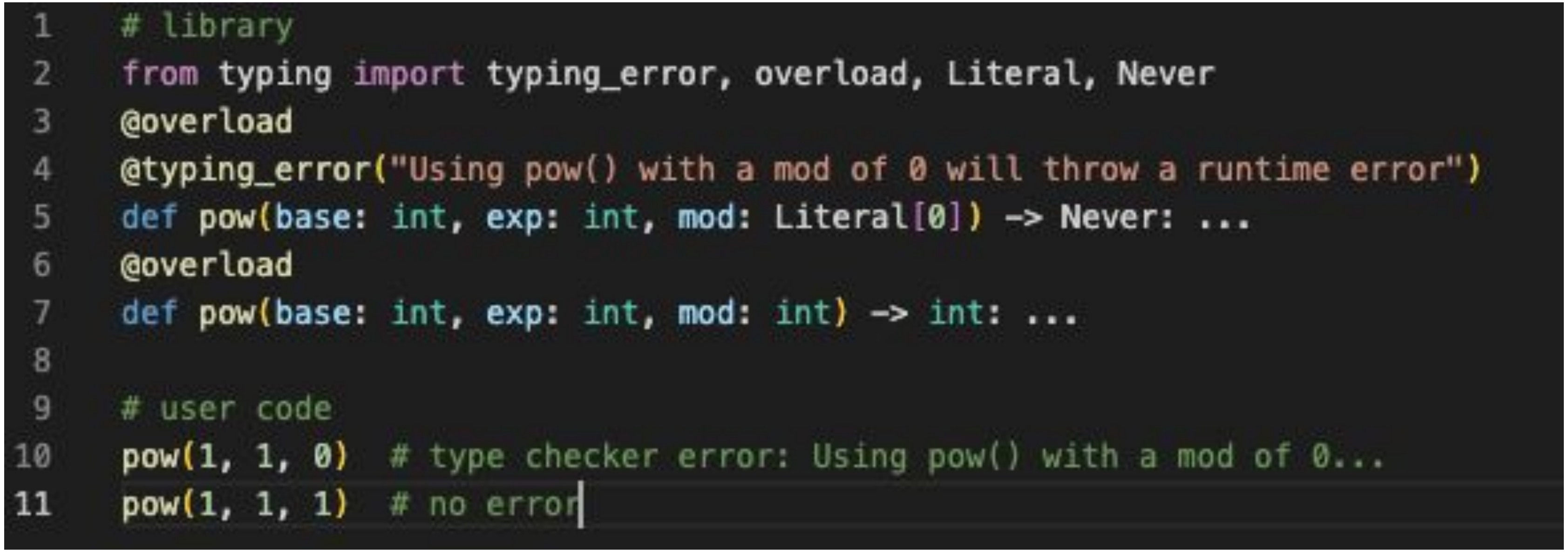


```python
# library
from typing import deprecated
@deprecated("Use inspect.signature instead")
def getargspec(...): ...

# user code
import inspect
inspect.getargspec(...)  # type checker error: Use of deprecated function
```

@typing.typing_error(...)

- Call to a marked function should produce an error



```python
# library
from typing import typing_error, overload, Literal, Never
@overload
@typing_error("Using pow() with a mod of 0 will throw a runtime error")
def pow(base: int, exp: int, mod: Literal[0]) -> Never: ...
@overload
def pow(base: int, exp: int, mod: int) -> int: ...

# user code
pow(1, 1, 0)  # type checker error: Using pow() with a mod of 0...
pow(1, 1, 1)  # no error
```

Use cases for @typing_error()

- https://github.com/python/typing/issues/1043
- Specific parameter combinations in overloads (e.g., open, pow)
- Methods that always throw
- Classes that cannot be constructed (put @typing_error() on __new__)

A wrinkle: overload resolution

- Given the `pow()` definition in the previous slide, what should `pow(1, 1, Any)` do?
- I would not want an error here
- But pyright's heuristic would pick the first overload, which has `@typing_error`.

Speculative ideas

- So far, presented a basic proposal for deprecation and error support
- Now, going into other potentially useful areas

Deprecations in CPython

- Looked at all deprecations in CPython main (150 total)
  - 74x whole function/method/class
  - 28x whole module
  - 9x parameter
  - 1x constant
  - 38x various complicated conditions

Can we cover more of these in the type system?

- A module-level `__deprecated__ = “This module is deprecated and will be removed in Python 3.13”`?
- param: Deprecated[SomeType, “This parameter is deprecated”]?

deprecated_transform?

- Third-party decorators may want to have some runtime effect (e.g., throw a warning) in addition to working like @typing.deprecated
- Idea: Add a PEP 681-style mechanism so that third-party decorators can have the same effect on type checkers as @typing.deprecated

Questions or discussion?

Bonus: Change what *args/**kwargs annotations mean?

- Thomas Wouters suggested that we explore changing what *args/**kwargs annotations mean
- Should we do this?