ePYt: Automatic type-aware test input generator for python

Team 2

Michael Tegegn, MyeongGeun Shin, Sihoon Lee, Yongwoo Lee

Introduction: Test input generation

- It is a mature software testing paradigm
- Automates a large part of the testing process
- Developed tools exist for well known languages like Java, C
  - EvoSuite (Java), KLOVER (C), Pynguin(Python)...

Introduction: Black box testing

- Test input generation without knowledge of internal structure
- Generate valid tests?
  - Using type information on program inputs

ePYt

ePYt [ipait]

1. Reversed word ‘type’
2. And also includes ‘py’
3. Type-aware test input generator

Problem: Test input generation for python

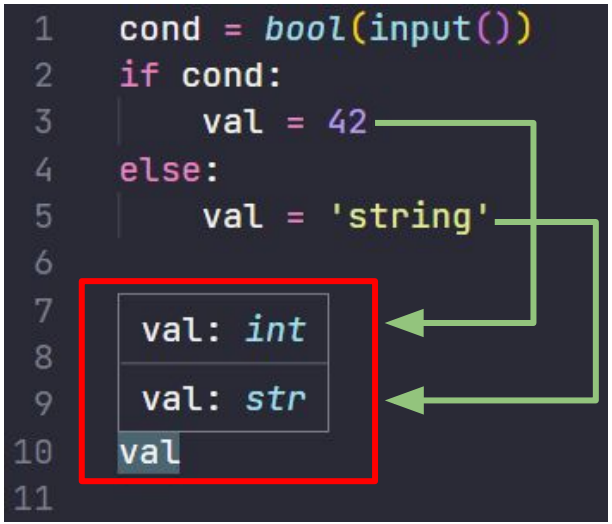
- Usually no type information in code
- Easy to provide type information of inputs?
  - Libraries with multiple functions?
    - Manually annotate every function?
- Need a way to automate the process

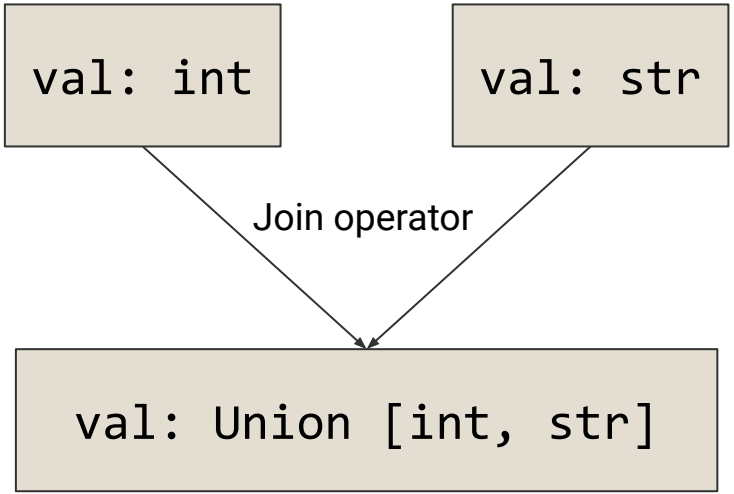
Two major difficulties

- Very few type hints
  - Get type information using static analysis
- Generate random/non-random input for complex type
  - Type-aware input generation

How to gain more type information?

Infer parameter, variable type using static analysis





Define Attr type

Default typing module doesn't care about attributes, methods.

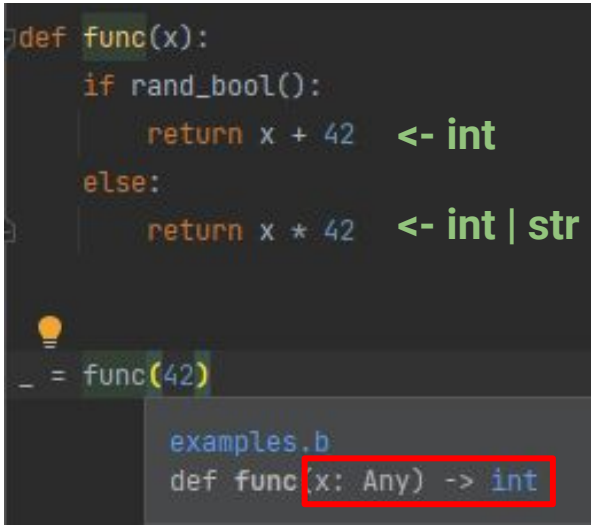
```
1    def func(x):
2        x.method()
3        return x + y
4
```

```
x: Any, func: Callable[[Any], Any]
x: Attr [method]
x: Attr [method, __add__]
```

```
func: Callable [[Attr [method, __add__]], Any]
```

Richer type leads to smarter inference

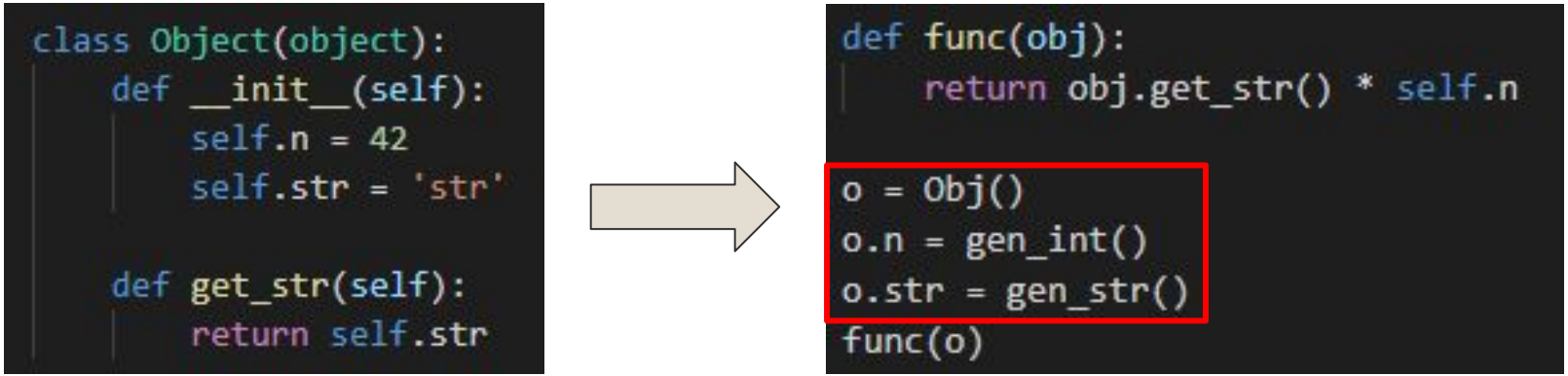
State-of-the-art IDEs are not quite smart
Richer type system can help out



1. `Any -> int`
2. `{int | str} -> {int | str}`
3. `Attr {__add__, __mul__} -> Any`

How to deal with complex type?

Parse initializers(`__init__`, `__enter__`, etc) and use them to generate input



Evaluation Strategy

Code coverage can be a good indicator

```
1   class URL:
2       def __init__(self, url):
3           self.url = url
4
5       def some_method(self, url):
6           if not isinstance(url, str):
7               raise Exception()
8           # Long code ~
9           # Long code ~
10
```

ePYt: our typed test generator

vs

Dumb test generator

Tests: [0, 1, 2, …, ‘a’, ‘b’, ‘c’, …, None, ‘abc’, …]

Conclusion

- Implement a tool that automatically generates **type-aware** test inputs
- Implement **smart type inference system**
- Generate test inputs for **complex types** (i.e. Non-primitive classes)

Questions?