ePYt: Automatic type-aware test input generator for python

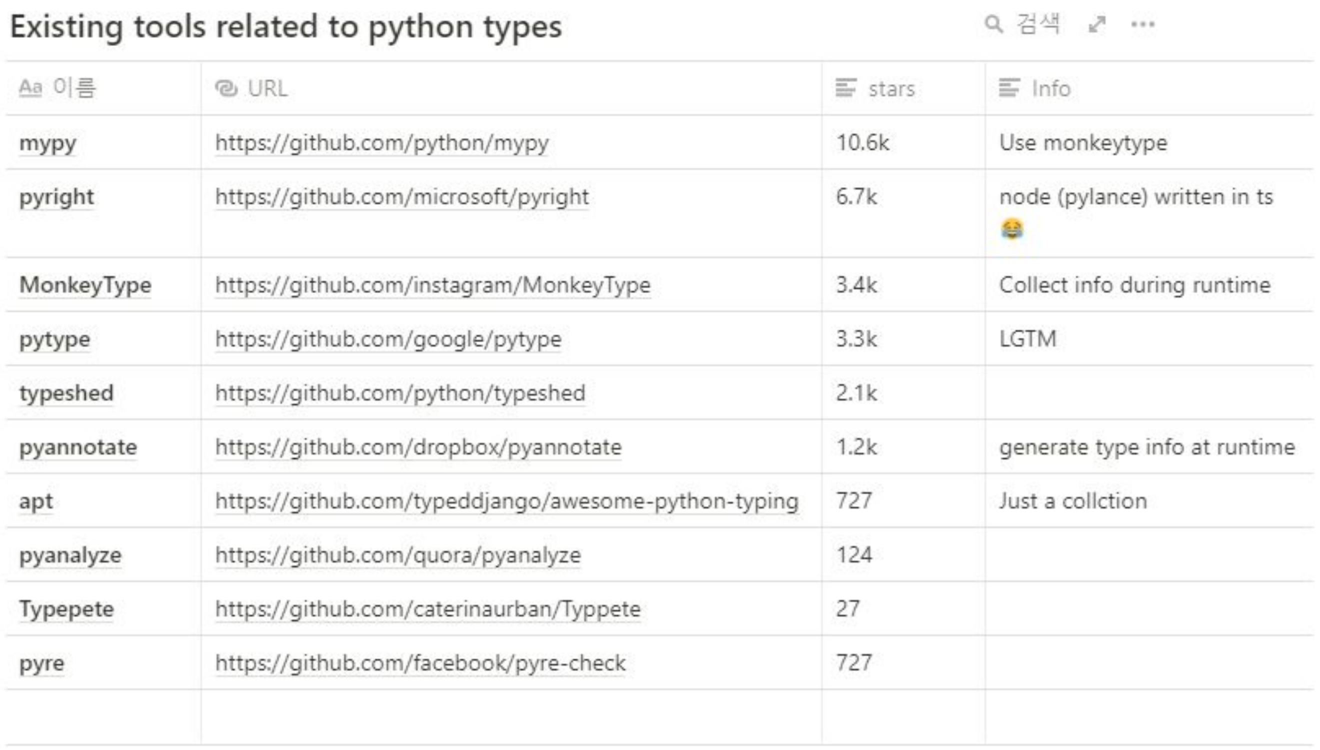
Team 2

Michael Tegegn, MyeongGeun Shin, Sihoon Lee, Yongwoo Lee

ePYt : GOAL

- Automatic type-aware test input generator for python
- Steps:
  - Gather type information using existing type inference tool
  - And, collect attributes variables use
  - Extend type inference functionality for structured input
  - Generate test cases based on inferred types

We searched a lot of tools

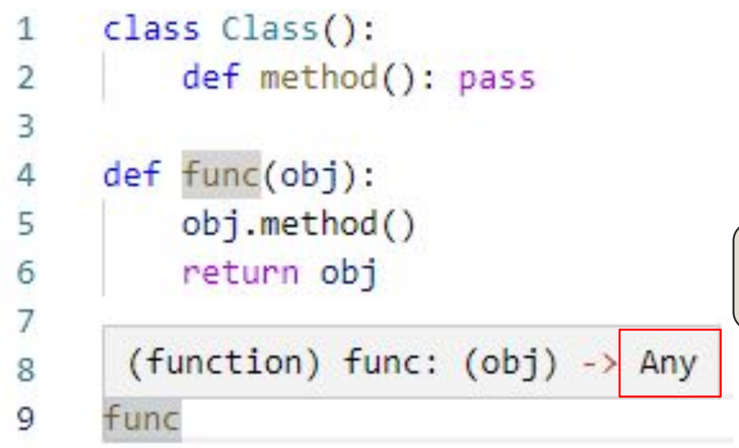


Existing tools related to python types

이름	URL	stars	Info
mypy	https://github.com/python/mypy	10.6k	Use monkeytype
pyright	https://github.com/microsoft/pyright	6.7k	node (pylance) written in ts 😂
MonkeyType	https://github.com/instagram/MonkeyType	3.4k	Collect info during runtime
pytype	https://github.com/google/pytype	3.3k	LGTM
typeshed	https://github.com/python/typeshed	2.1k	
pyannotate	https://github.com/dropbox/pyannotate	1.2k	generate type info at runtime
apt	https://github.com/typeddjango/awesome-python-typing	727	Just a collction
pyanalyze	https://github.com/quora/pyanalyze	124	
Typepete	https://github.com/caterinaurban/Typpete	27	
pyre	https://github.com/facebook/pyre-check	727	

Motivating example for ePYt

Pylance (Microsoft)



```
class Class():
    def method(): pass

def func(obj):
    obj.method()
    return obj

(function) func: (obj) -> Any
func
```

VS

ePYt

Gathered information

```
obj: hasAttr("method")
func: Callable[[obj], obj]
```

Inferred "func" type to:

```
Callable[[Class], Class]
```

Existing type inference tools

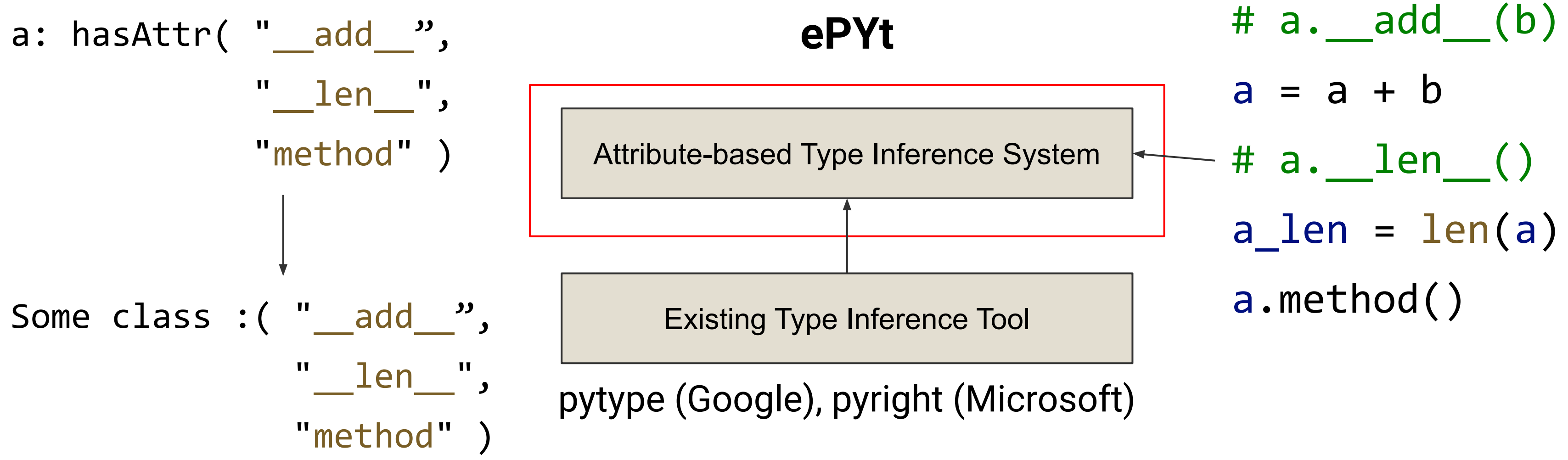
1. pyanalyze: https://github.com/quora/pyanalyze
    - Description:
        - Static analyzer
        - Programmatically detecting common mistakes in python code
        - Checking type annotations
        - Finding dead code
    - Limitation:
        - Documented but totally unimplemented type inference system for python expression

Existing type inference tools

2. Microsoft pyright: https://github.com/microsoft/pyright
   - Description:
     - Static type checker
     - Fast type checker more useful for large code bases
     - Inferring python symbol type based on value assignment, return type, expected type, …
   - Limitation:
     - Written in TypeScript. i.e. We cannot simply extend it

Attribute-based type inference system

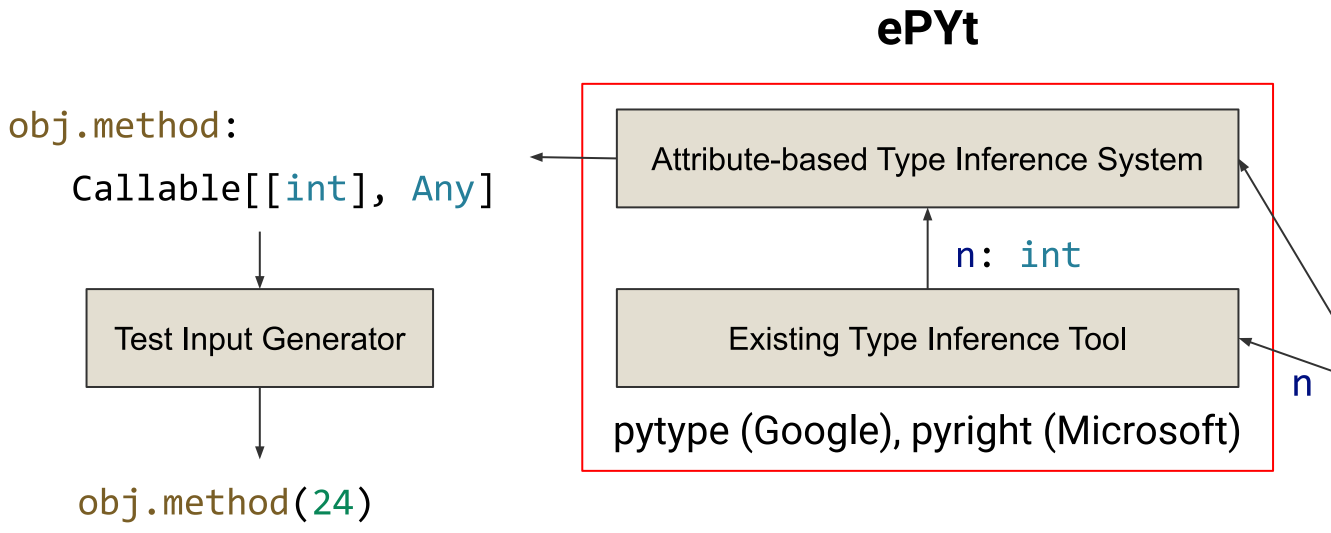
- ePYt infers based on attribute variable uses


a: hasAttr("__add__", "__len__", "method")
Some class :("__add__", "__len__", "method")
ePYt
Attribute-based Type Inference System
Existing Type Inference Tool
pytype (Google), pyright (Microsoft)

```python
# a.__add__(b)
a = a + b
# a.__len__()
a_len = len(a)
a.method()
```

Using existing type inference tool

- Existing type inference tool helps ePYt to infer type info

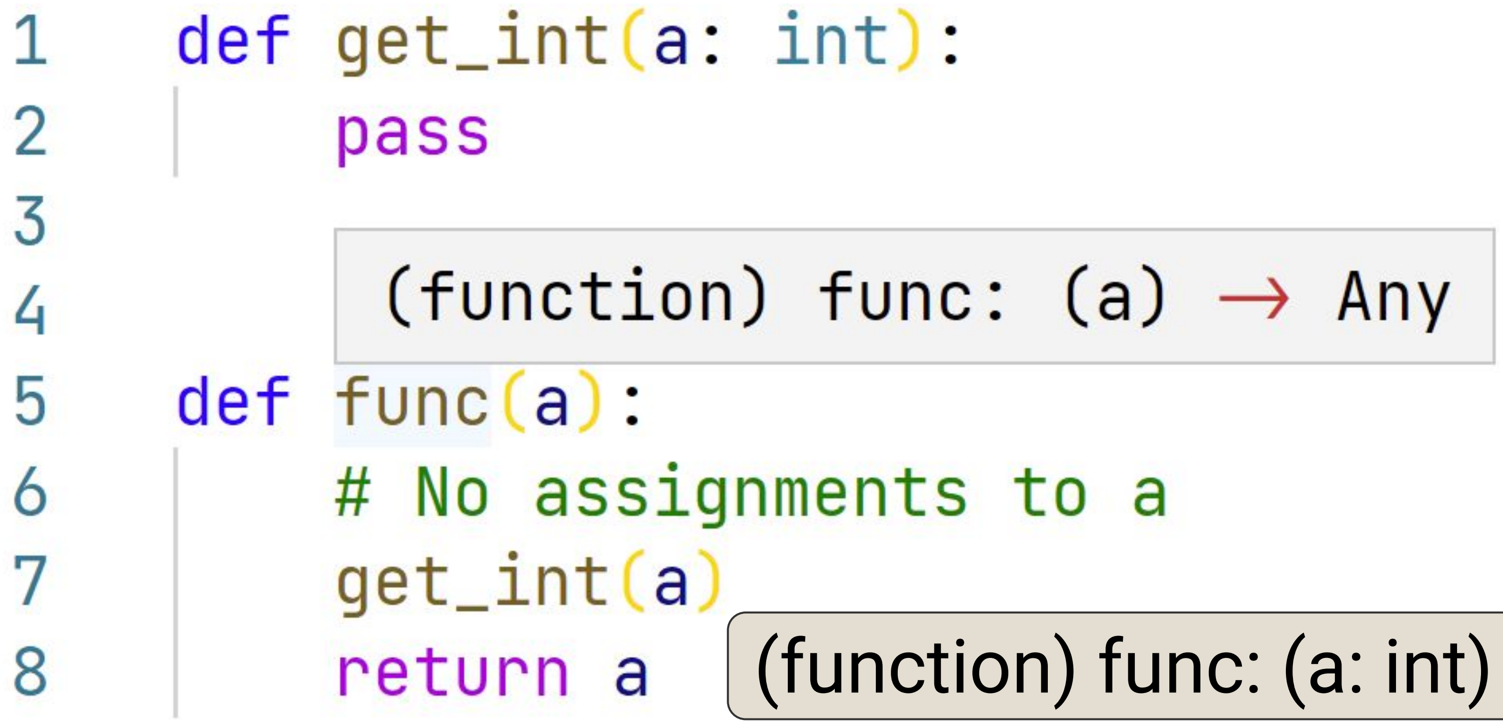

ePYt
obj.method:
Callable[[int], Any]
Attribute-based Type Inference System
n: int
Existing Type Inference Tool
pytype (Google), pyright (Microsoft)
Test Input Generator
obj.method(24)
n

```
class ClassA():
    def method(a: int):
        pass

class ClassB():
    def method(a: str):
        pass

n = 42
obj.method(n)
# ClassA or ClassB?
# A gets int so A
```

Another way to improve analysis



```python
def get_int(a: int):
    pass

def func(a):
    # No assignments to a
    get_int(a)
    return a
```

```
(function) func: (a) → Any
```

```
(function) func: (a: int) -> int
```

Current Status and Direction

- Github repo ready at https://github.com/luxroot/ePYt
- Use pytype as black box for basic type inference
- Build our own attribute-based type inference system
- Optionally implement type inference using back-propagation
- Use Evosuite to generate the tests

Questions?