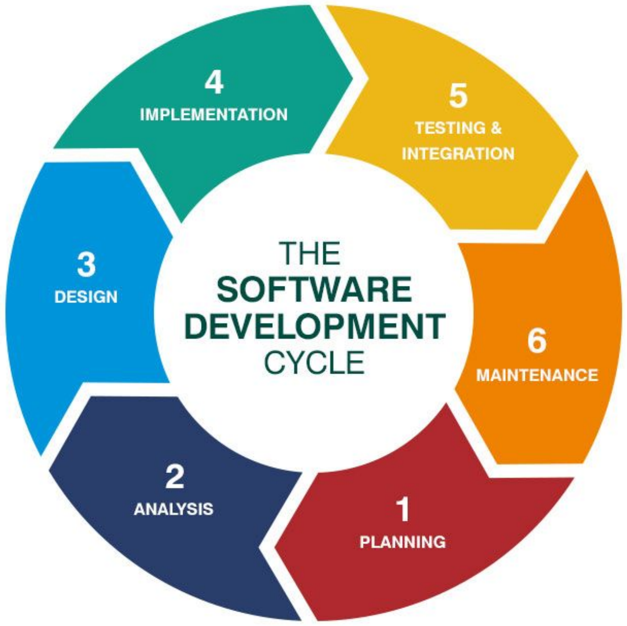
ePYt: Automatic type-aware test input generator for python

Team 2

Michael Tegegn, MyeongGeun Shin, Sihoon Lee, Yongwoo Lee

Introduction



Testing can enhance software quality.

However, making test suites by hand is costly and incomplete. 🤑

Automate test suite generatior!

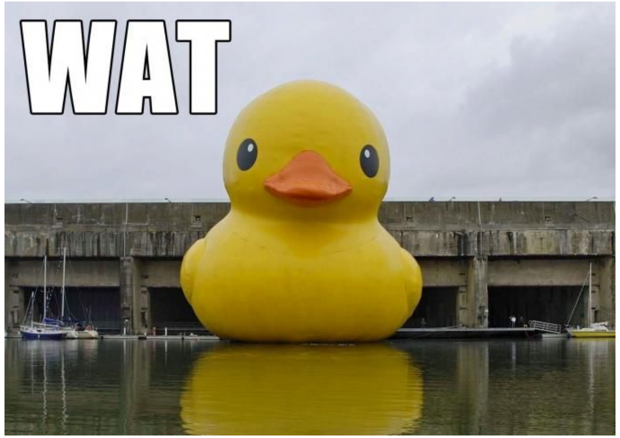
Problem

Java has automated test suite generator.

Python has test suite generator too.
But.. it isn't powerful as evosuite.

No type information !

Python's philosophy



Python is duck-typed language.

Types will be determined dynamically.

No one know the exact type of variable

… until execution

Problem - continued

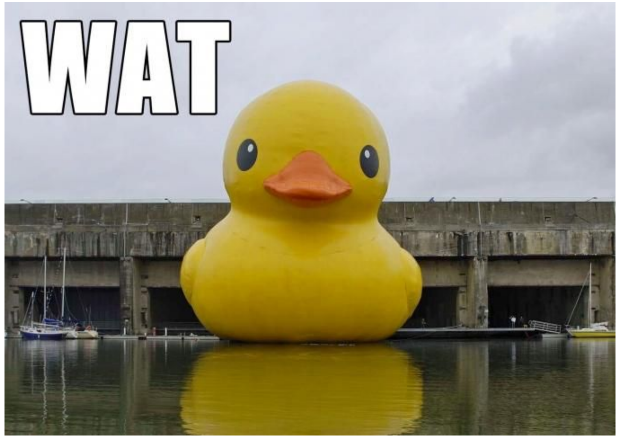
There are some static type checker but it can not infer the custom type(class).

```python
class MyType:
    def __init__(self) -> None:
        self.property = 1

def foo(bar):
    assert bar.property == 1
    bar.property += 1
```

Type of Any?

💡 Probably MyType!

Solution in a glance



```
class MyType:
    def __init__(self) -> None:
        self.property = 1

def foo(bar):
    assert bar.property == 1
    bar.property += 1
```

Type of Any?

💡 Probably MyType!

Collect class definitions → See argument's attribute usages → Infer type → Test input generation

Method Description - Control Flow Graph

For static analysis, we need control flow graph

So we implemented our control flow graph using AST module

FuncDef node

- Function definition
- Name
- Arguments

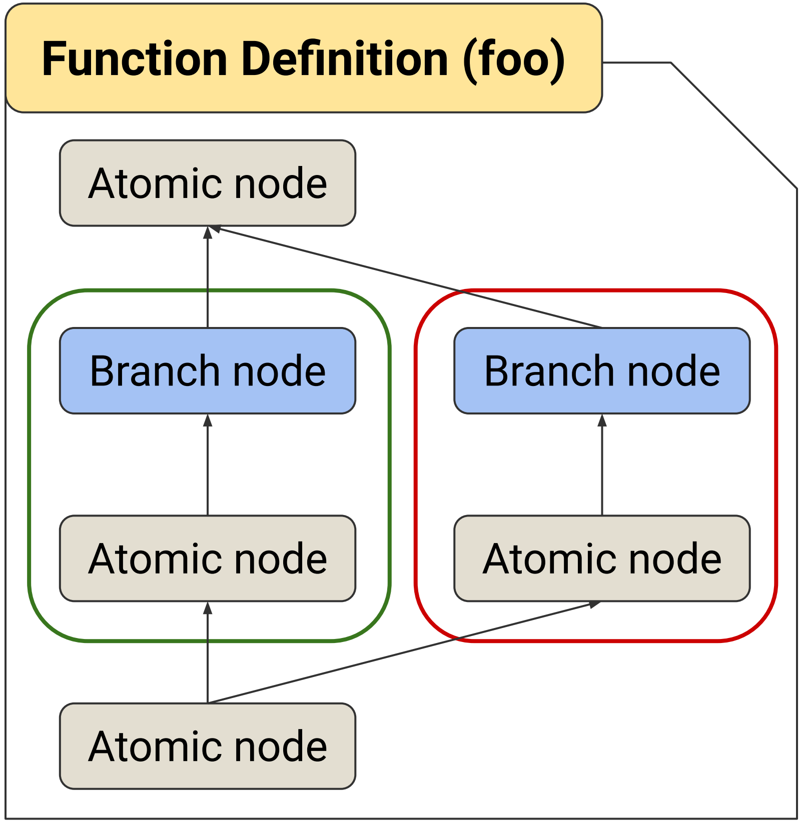
Branch node

- Separates flow
  - `if, for, while`
- Condition
- Whether taken

Atomic node

- All other instructions
- Instruction info

Building Control Flow Graph



```
def foo(bar):
    if bar == 0:
        print('Taken!')
    else:
        print('Not taken!')
```

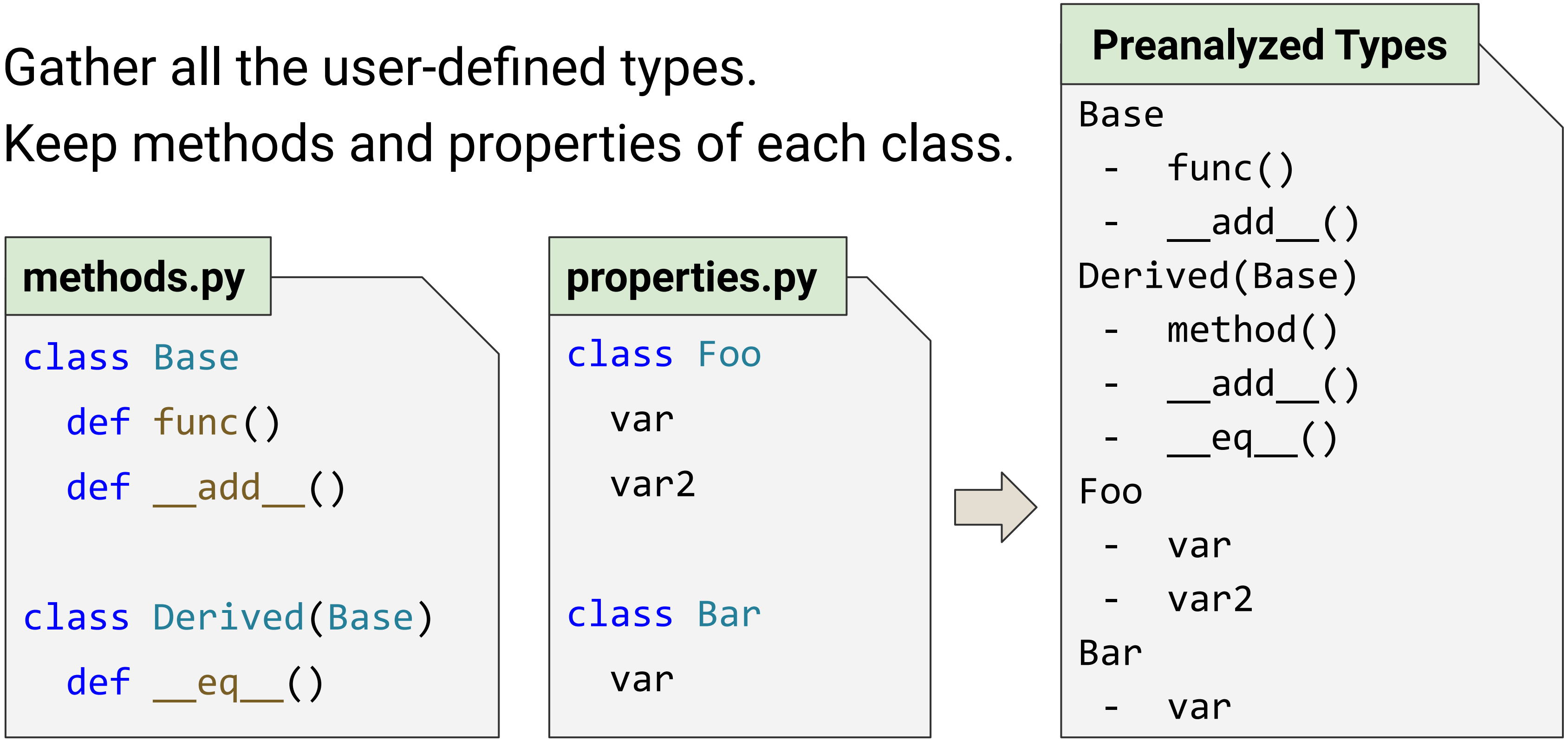
Why do we build CFG?

```
def func(x):
    x = abs(x)
    print(x + 2)
```

Method Description - PreAnalysis

Gather all the user-defined types.

Keep methods and properties of each class.



methods.py

```
class Base
  def func()
  def __add__()

class Derived(Base)
  def __eq__()
```

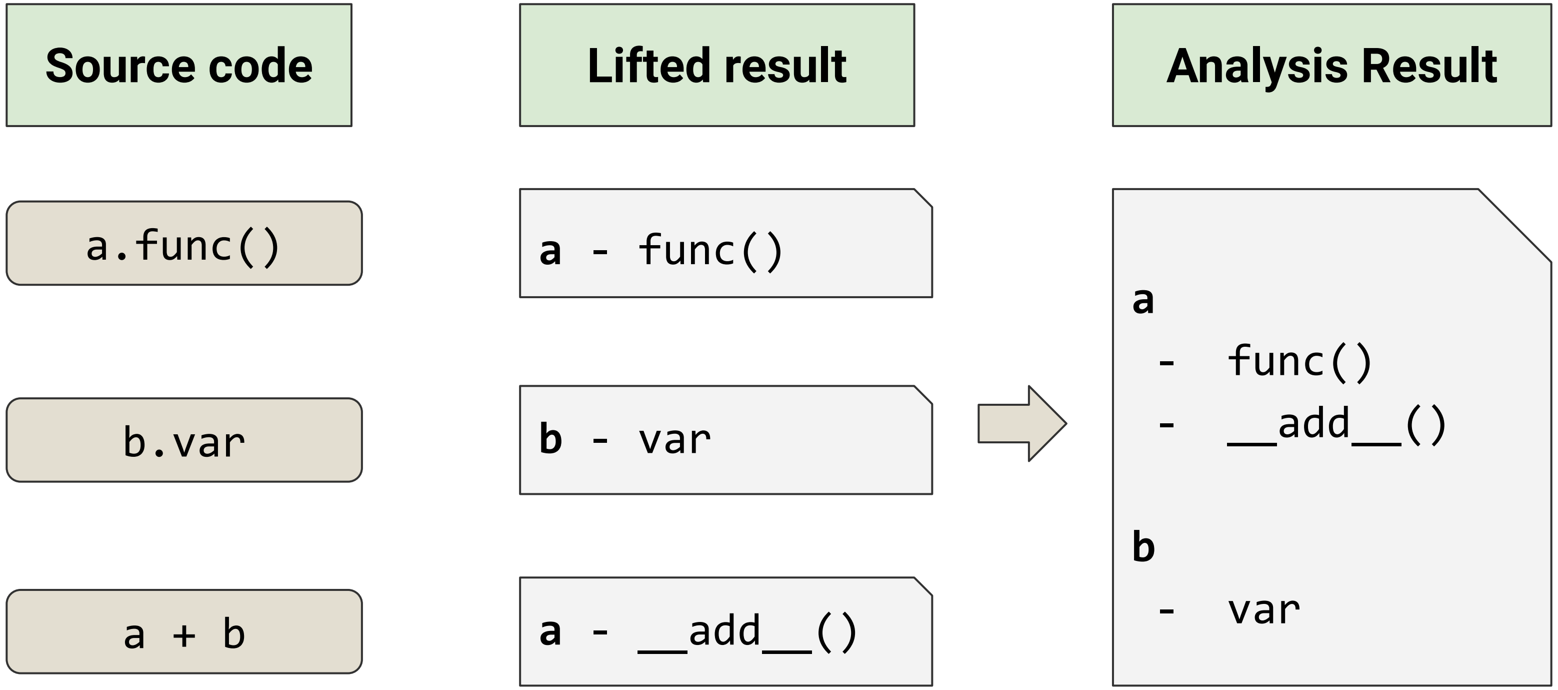
properties.py

```
class Foo
  var
  var2

class Bar
  var
```

Preanalyzed Types

```
Base
 -  func()
 -  __add__()
Derived(Base)
 -  method()
 -  __add__()
 -  __eq__()
Foo
 -  var
 -  var2
Bar
 -  var
```

Method Description - Analysis



Source code

```
a.func()
```

```
b.var
```

```
a + b
```

Lifted result

```
a - func()
```

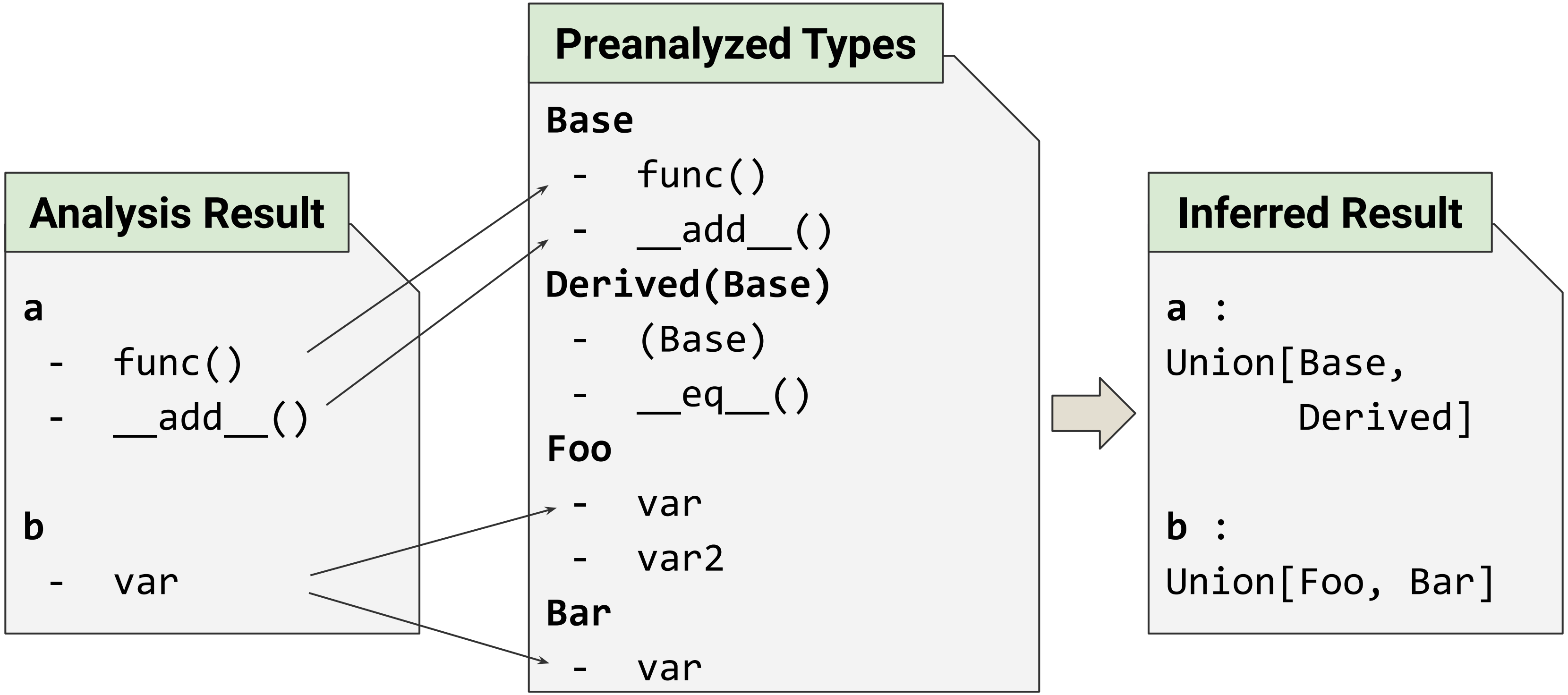
```
b - var
```

```
a - __add__()
```

Analysis Result

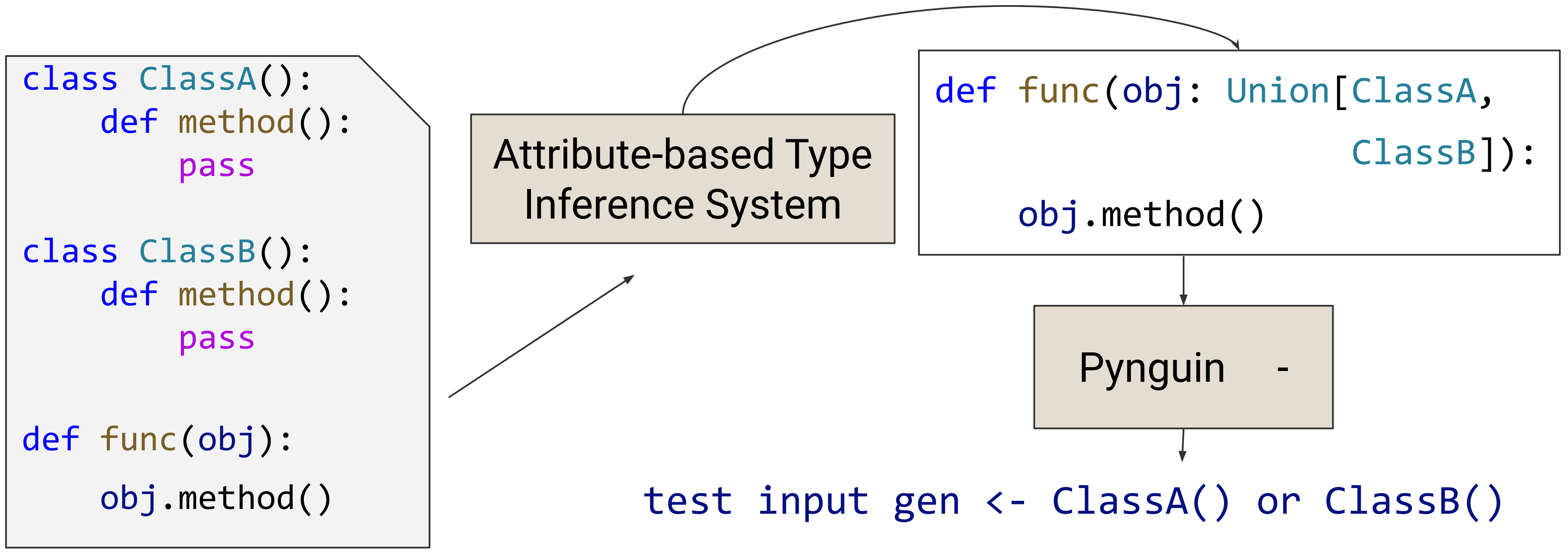
```
a
  -   func()
  -   __add__()

b
  -   var
```

Method Description - Type Infer



Generating Test Case

Pyngiun can generate test suites with the type-hinted source

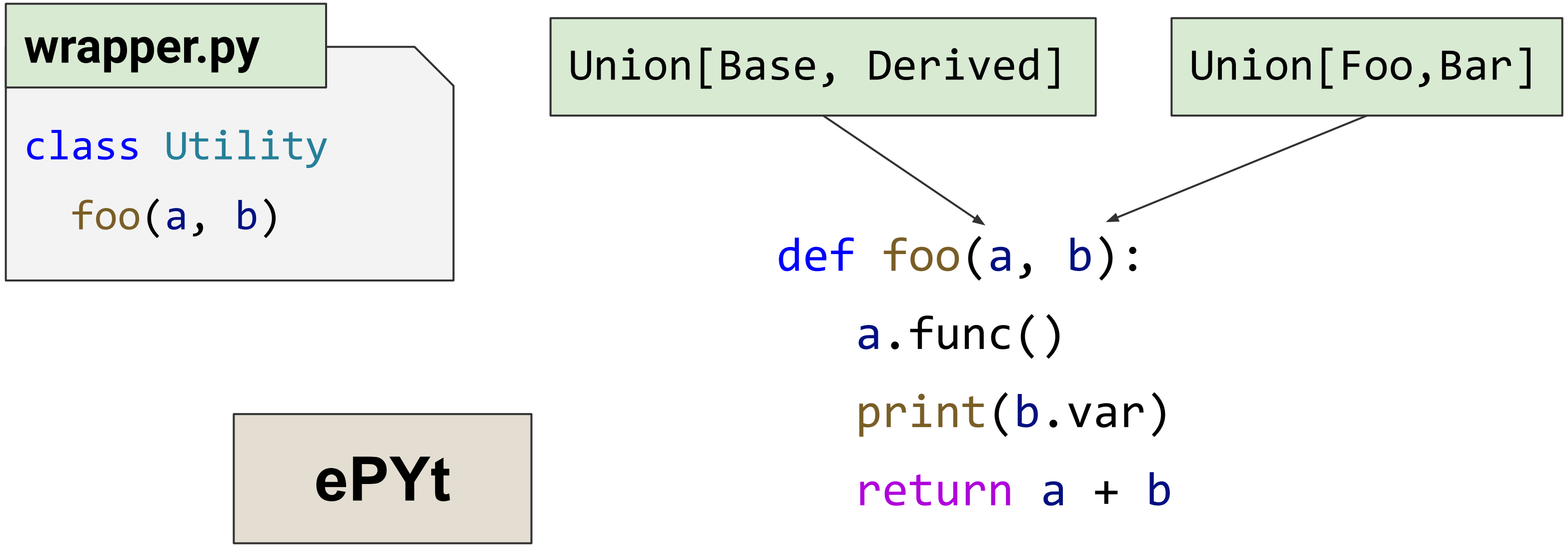


```
class ClassA():
    def method():
        pass

class ClassB():
    def method():
        pass


def func(obj):

    obj.method()
```

```
def func(obj: Union[ClassA,
                    ClassB]):

    obj.method()
```

```
test input gen <- ClassA() or ClassB()
```

Evaluation Metrics



```
class Utility
  foo(a, b)
```

```
def foo(a, b):
    a.func()
    print(b.var)
    return a + b
```

Evaluation Metrics

wrapper.py

```
class Utility
  foo(a, b)
```

```
def foo(a:Union [Base, Derived],
        b:Union [Foo, Bar]):
   a.func()
   print(b.var)
   return a + b
```

Pynguin

```
def test_01():
   var0 = Base()
   assert var0 is not None
   var1 = Foo()
   assert var1 is not None
   var2 = \
   wrapper.foo(var0, var1)
   assert var2 is None
```

Real world example - urllib

```
def http_error_auth_reqed(self, auth_header, host, req: Request,
headers: Union[Quoter, defaultdict, dict]):
    authreq = headers.get(auth_header, None)
    if self.retried > 5:
        raise HTTPError(req.full_url, 401, 'digest auth failed', headers, None)
    else:
        self.retried += 1
    if authreq:
        scheme = authreq.split()[0]
        if scheme.lower() == 'digest':
            return self.retry_http_digest_auth(req, authreq)
```

Real world example - urllib

```
def http_error_auth_reqed(self, auth_header, host, req: Request,
headers: Union[Quoter, defaultdict, dict]):
```

```
def http_error_auth_reqed(self, auth_header: str, host: str, req: Request,
headers: Mapping[str, str]) -> None: ...
```

Real signature from typeshed

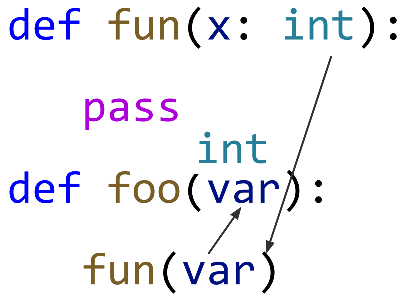
```
    else:
        self.retried += 1
    if authreq:
        scheme = authreq.split()[0]
        if scheme.lower() == 'digest':
            return self.retry_http_digest_auth(req, authreq)
```

Conclusion

- We present fully automated type-aware test input generator based on attributes
- To the best of our knowledge, this is the first study to infer the type of a variable using an attribute in Python
- This can be used in testing complex and large code like library code

Further work

- Currently, ePYt does not utilize function signature information
- In below code, we can infer `var` is int type if ePYt can utilize function signature information



```
def fun(x: int):
    pass

def foo(var):
    fun(var)
```

Questions?

Collect class definitions → See argument's attribute usages → Infer type → Test input generation

Lift python statement

Attribute-based Type Inference System

Union inferred types

Pynguin -

```
def http_error_auth_reqed(self, auth_header, host, req: Request,
headers: Union[Quoter, defaultdict, dict]):
```

Our result

```
def http_error_auth_reqed(self, auth_header: str, host: str, req: Request,
headers: Mapping[str, str]) -> None: ...
```

Real signature from typeshed