Вывести типы из Python: проблемы анализа Python-кода

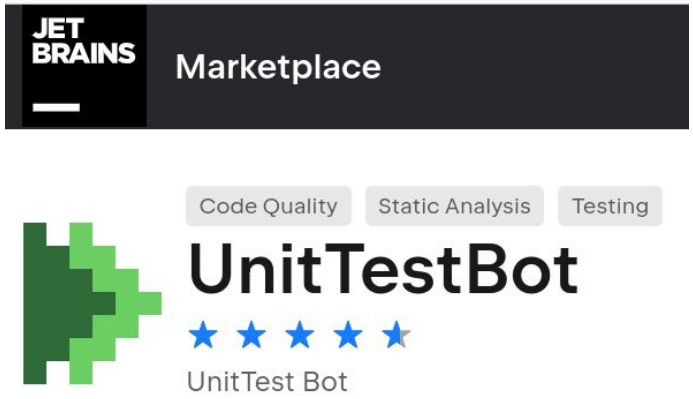
Вячеслав Тамарин
Екатерина Точилина

Чем мы занимаемся

Разработка **удобного** инструмента для:

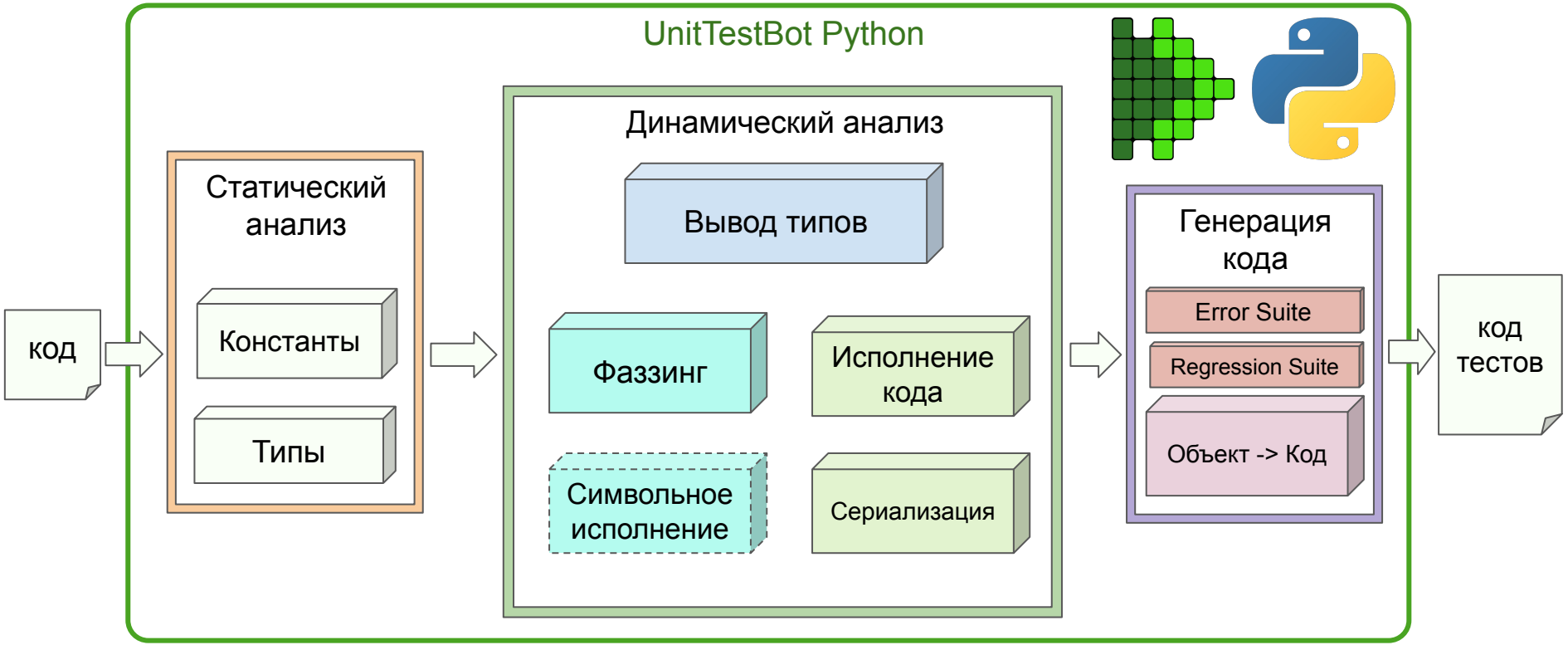
- поиска ошибок в коде
- генерации юнит-тестов

github.com/UnitTestBot



www.utbot.org/

Из чего состоит UnitTestBot



```python
@dataclasses.dataclass
class Panda:
    weight: int

    def __lt__(self, other):
        return self.weight < other.weight


def merge_sort(pandas: list[Panda]):
    if len(pandas) == 1:
        return pandas
    mid = len(pandas) // 2
    sorted_left = merge_sort(pandas[:mid])
    sorted_right = merge_sort(pandas[mid:])
    return list(heapq.merge(sorted_left, sorted_right))
```

```python
@dataclasses.dataclass
class Panda:
    weight: int

    def __lt__(self, other):
        return self.weight < other.weight


def merge_sort(pandas: list[Panda]):
    if len(pandas) == 1:
        return pandas
    mid = len(pandas) // 2
    sorted_left = merge_sort(pandas[:mid])
    sorted_right = merge_sort(pandas[mid:])
    return list(heapq.merge(sorted_left, sorted_right))
```

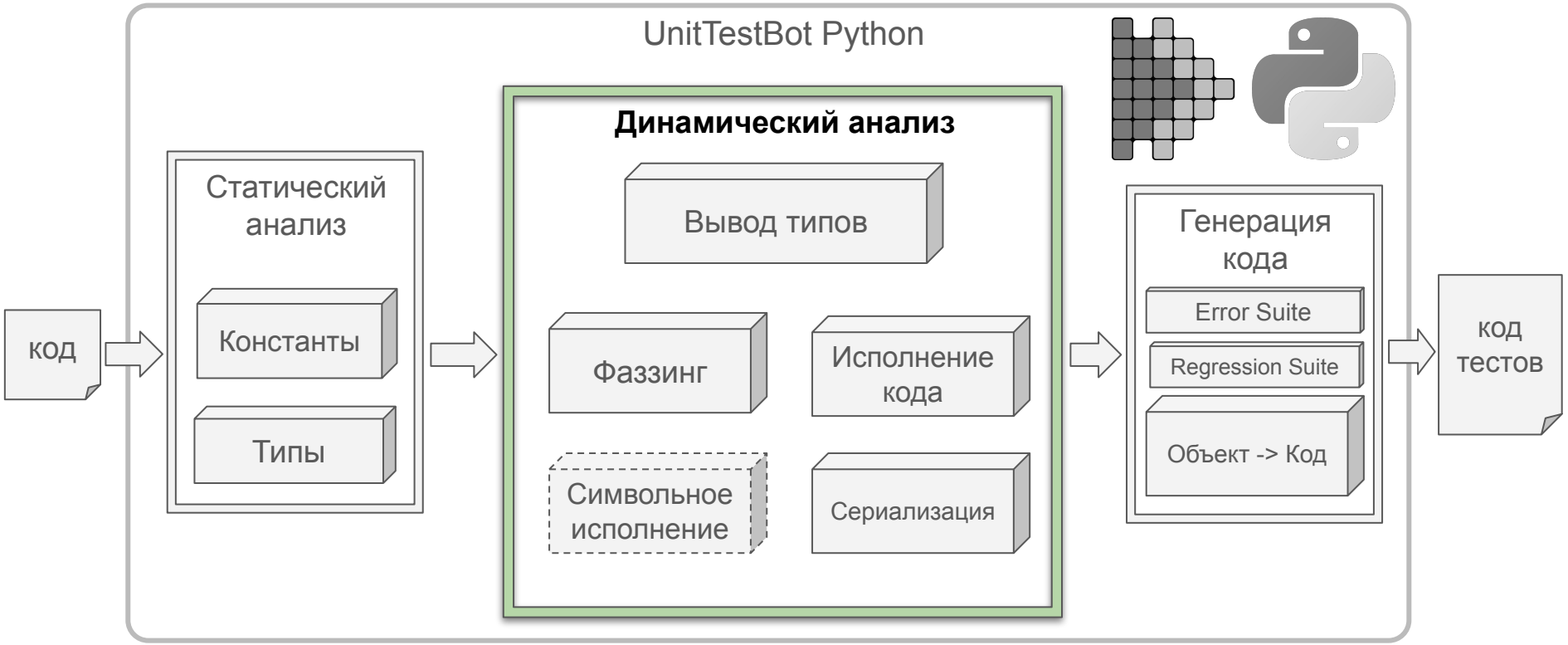
```python
class TestTopLevelFunctions(unittest.TestCase):
    # region Test suites for executable main.merge_sort

    # region FUZZER

    def test_merge_sort(self):
        """
        pandas = builtins.list[main.Panda]
        """
        panda = main.Panda(0)
        panda.weight = -1
        panda1 = main.Panda(-170141183460469231731687303715884105728)
        panda1.weight = 170141183460469231731687303715884105727
        panda2 = main.Panda(-1)
        panda2.weight = 1

        actual = main.merge_sort([panda, panda1, panda2])

        self.assertEqual([panda, panda1, panda2], actual)

    def test_merge_sort1(self):
        """
        pandas = builtins.list[main.Panda]
        """
        panda = main.Panda(2)
        panda.weight = 131072

        actual = main.merge_sort([panda])

        self.assertEqual([panda], actual)

    def test_merge_sort_with_exception(self):
        """
        pandas = builtins.list[main.Panda]
        """
        with self.assertRaises(RecursionError):
            main.merge_sort([])
    # endregion

    # endregion
```

```python
def merge_sort(pandas: list[Panda]):
    if len(pandas) == 1:
        return pandas
    mid = len(pandas) // 2
    sorted_left = merge_sort(pandas[:mid])
    sorted_right = merge_sort(pandas[mid:])
    return list(
        heapq.merge(
            sorted_left,
            sorted_right
        )
    )
```

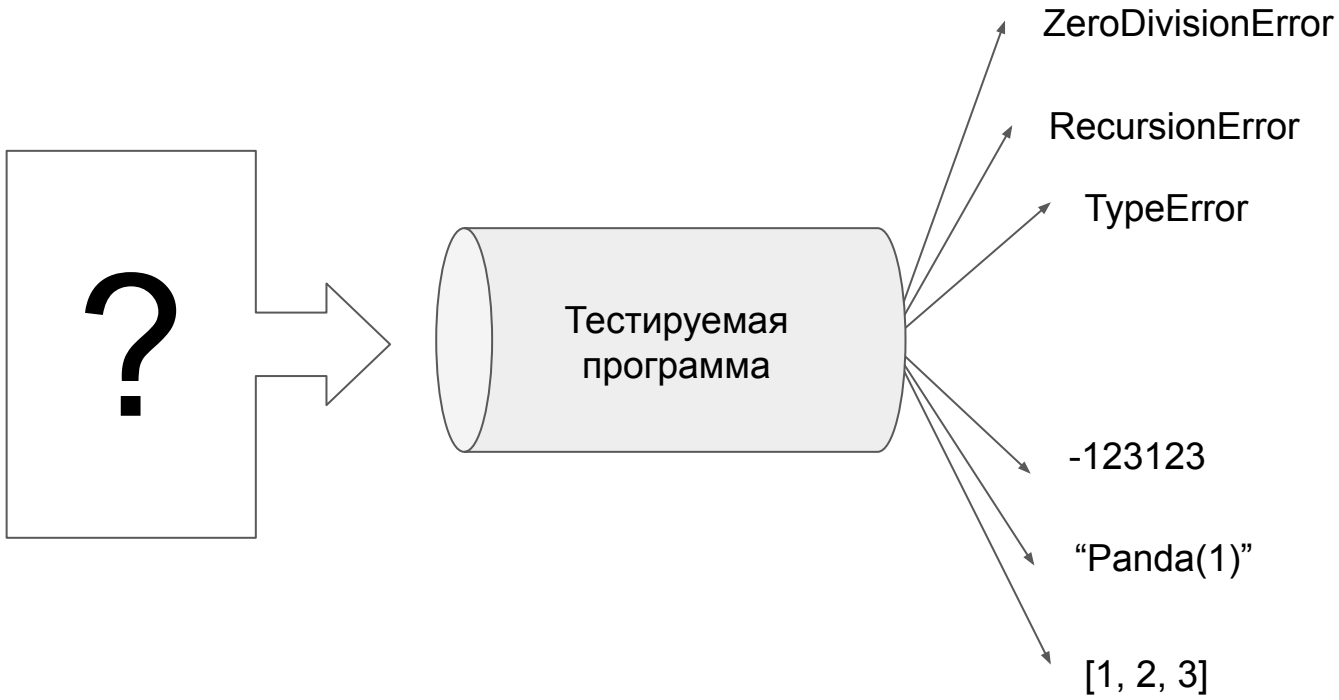
```python
class TestTopLevelFunctions(unittest.TestCase):
    # region Test suites for executable main.merge_sort

    # region FUZZER

    def test_merge_sort(self):
        """
        pandas = builtins.list[main.Panda]
        """
        panda = main.Panda(0)
        panda.weight = -1
        panda1 = main.Panda(-170141183460469231731687303715884105728)
        panda1.weight = 170141183460469231731687303715884105727
        panda2 = main.Panda(-1)
        panda2.weight = 1

        actual = main.merge_sort([panda, panda1, panda2])

        self.assertEqual([panda, panda1, panda2], actual)

    def test_merge_sort1(self):
        """
        pandas = builtins.list[main.Panda]
        """
        panda = main.Panda(2)
        panda.weight = 131072

        actual = main.merge_sort([panda])

        self.assertEqual([panda], actual)

    def test_merge_sort_with_exception(self):
        """
        pandas = builtins.list[main.Panda]
        """
        with self.assertRaises(RecursionError):
            main.merge_sort([])
    # endregion

    # endregion
```

```python
def merge_sort(pandas: list[Panda]):
    if len(pandas) == 1:
        return pandas
    mid = len(pandas) // 2
    sorted_left = merge_sort(pandas[:mid])
    sorted_right = merge_sort(pandas[mid:])
    return list(
        heapq.merge(
            sorted_left,
            sorted_right
        )
    )
```

Введение: обзор существующих решений



Более широкая задача



Error Suite

Regression Suite

Fuzzing

Atheris

diligence-fuzzing

Concolic Execution

CrossHair

Property based tesing

Hypothesis

Как измерять качество анализа?

Fuzzing

Atheris

diligence-fuzzing

Concolic Execution

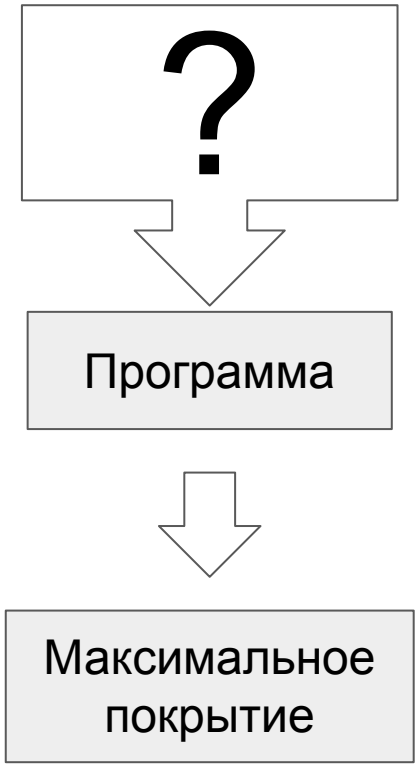
CrossHair

Property based tesing

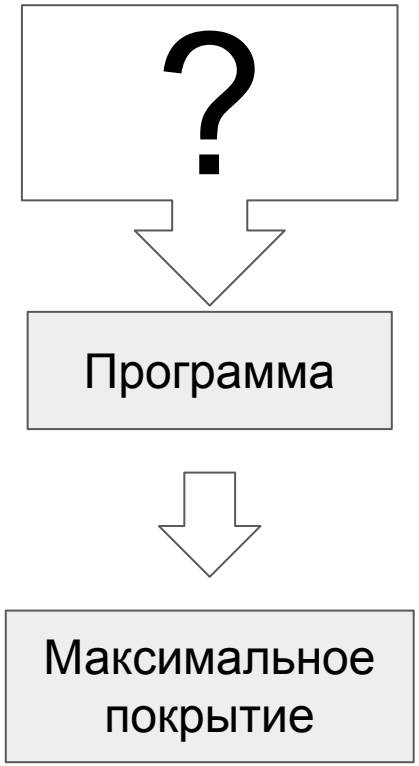
Hypothesis

```python
def f(x: int, y: int, z: int):
    if x + y > 100:
        return 0
    y += 10 ** 9
    if 0 < x + z + 1 < 100 and y > 0:
        return 1
    elif x + 3 < -2 - z and x < y:
        return 2
    elif x * 100 % 7 == 0 and z + y % 100 == 0:
        return 3
    elif x % 155 == 0 and x + y - z < 0:
        return 4
    elif z == 15789 and y + x > 10 ** 9:
        return 5
    elif x + y + z == -10 ** 9 and x != 0 and z == 2598:
        return 6
    else:
        return 7
```

Более узкая, но формальная задача



Более узкая, но формальная задача



Алгоритмически неразрешимо

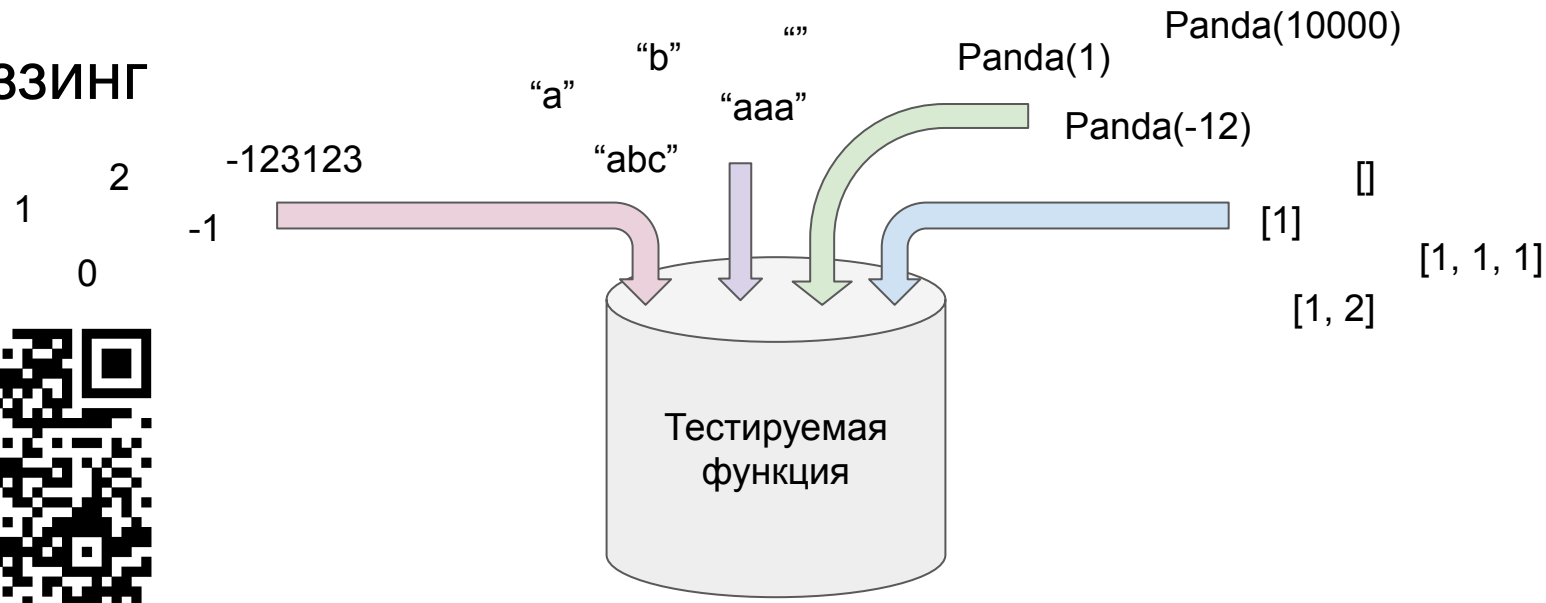
(выводится из теоремы Райса)

Фаззинг

Фаззинг

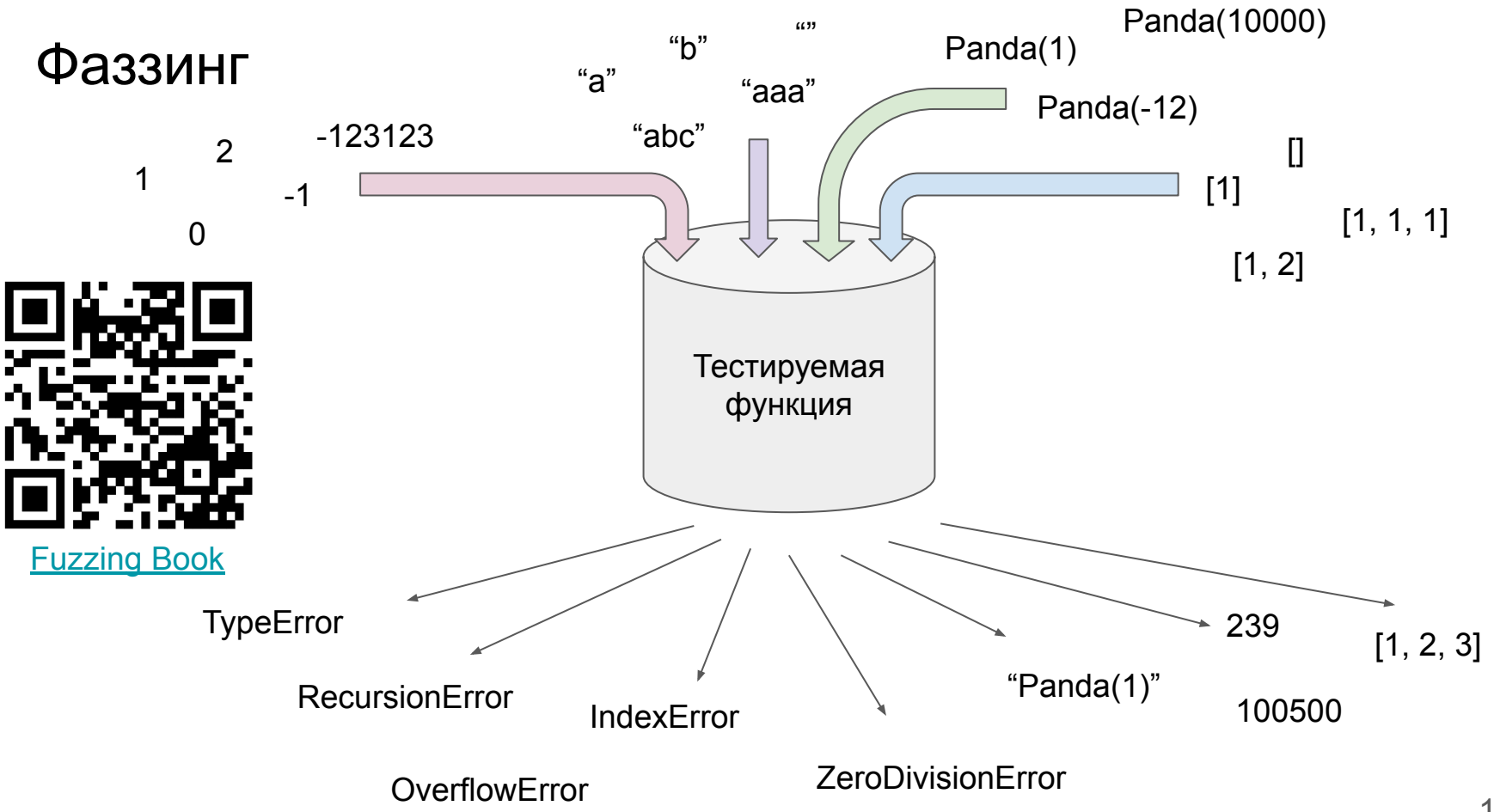
Fuzzing Book

Фаззинг



Fuzzing Book

Фаззинг



Fuzzing Book

Использование Atheris

```
def merge_sort(pandas):
    if len(pandas) == 1:
        return pandas
    mid = len(pandas) // 2
    sorted_left = merge_sort(pandas[:mid])
    sorted_right = merge_sort(pandas[mid:])
    return list(
        heapq.merge(
            sorted_left,
            sorted_right
        )
    )
```

Atheris GitHub

Использование Atheris

```
def merge_sort(pandas):
    if len(pandas) == 1:
        return pandas
    mid = len(pandas) // 2
    sorted_left = merge_sort(pandas[:mid])
    sorted_right = merge_sort(pandas[mid:])
    return list(
        heapq.merge(
            sorted_left,
            sorted_right
        )
    )
```

Atheris GitHub

Использование Atheris

```python
def merge_sort(pandas):
    if len(pandas) == 1:
        return pandas
    mid = len(pandas) // 2
    sorted_left = merge_sort(pandas[:mid])
    sorted_right = merge_sort(pandas[mid:])
    return list(
        heapq.merge(
            sorted_left,
            sorted_right
        )
    )
```

```python
import atheris

with atheris.instrument_imports():
    from main import merge_sort
    import sys

def TestOneInput(data: bytes):
    merge_sort(data)

atheris.Setup(sys.argv, TestOneInput)
atheris.Fuzz()
```

Atheris GitHub

Использование Atheris

Atheris GitHub

```
 === Uncaught Python exception: ===
RecursionError: maximum recursion depth exceeded in comparison
Traceback (most recent call last):
  File "/home/tochilinak/Documents/projects/utbot/test_project/atheris_usage.py", line 8, in TestOneInput
    merge_sort(data)
  File "/home/tochilinak/Documents/projects/utbot/test_project/main.py", line 221, in merge_sort
    merge_sort(items[:mid]),
  File "/home/tochilinak/Documents/projects/utbot/test_project/main.py", line 221, in merge_sort
    merge_sort(items[:mid]),
  File "/home/tochilinak/Documents/projects/utbot/test_project/main.py", line 221, in merge_sort
    merge_sort(items[:mid]),
  [Previous line repeated 989 more times]
  File "/home/tochilinak/Documents/projects/utbot/test_project/main.py", line 220, in merge_sort
    heapq.merge(
  File "/home/tochilinak/Documents/projects/utbot/test_project/main.py", line 215, in merge_sort
    def merge_sort(items):
RecursionError: maximum recursion depth exceeded in comparison

==24186== ERROR: libFuzzer: fuzz target exited
SUMMARY: libFuzzer: fuzz target exited
MS: 0 ; base unit: 0000000000000000000000000000000000000000
0xa,
\012
artifact_prefix='./'; Test unit written to ./crash-adc83b19e793491b1c6ea0fd8b46cd9f32e592fc
Base64: Cg==
```

```python
import atheris

with atheris.instrument_imports():
    from main import merge_sort
    import sys

def TestOneInput(data: bytes):
    merge_sort(data)

atheris.Setup(sys.argv, TestOneInput)
atheris.Fuzz()
```

Использование Atheris

Atheris GitHub

```
 === Uncaught Python exception: ===
RecursionError: maximum recursion depth exceeded in comparison
Traceback (most recent call last):
  File "/home/tochilinak/Documents/projects/utbot/test_project/atheris_usage.py", line 8, in TestOneInput
    merge_sort(data)
  File "/home/tochilinak/Documents/projects/utbot/test_project/main.py", line 221, in merge_sort
    merge_sort(items[:mid]),
  File "/home/tochilinak/Documents/projects/utbot/test_project/main.py", line 221, in merge_sort
    merge_sort(items[:mid]),
  File "/home/tochilinak/Documents/projects/utbot/test_project/main.py", line 221, in merge_sort
    merge_sort(items[:mid]),
  [Previous line repeated 989 more times]
  File "/home/tochilinak/Documents/projects/utbot/test_project/main.py", line 220, in merge_sort
    heapq.merge(
  File "/home/tochilinak/Documents/projects/utbot/test_project/main.py", line 215, in merge_sort
    def merge_sort(items):
RecursionError: maximum recursion depth exceeded in comparison

==24186== ERROR: libFuzzer: fuzz target exited
SUMMARY: libFuzzer: fuzz target exited
MS: 0 ; base unit: 0000000000000000000000000000000000000000
0xa,
\012
artifact_prefix='./'; Test unit written to ./crash-adc83b19e793491b1c6ea0fd8b46cd9f32e592fc
Base64: Cg==
```

```python
import atheris

with atheris.instrument_imports():
    from main import merge_sort
    import sys

def TestOneInput(data: bytes):
    merge_sort(data)

atheris.Setup(sys.argv, TestOneInput)
atheris.Fuzz()
```

Использование Atheris

Atheris GitHub

```
 === Uncaught Python exception: ===
RecursionError: maximum recursion depth exceeded in comparison
Traceback (most recent call last):
  File "/home/tochilinak/Documents/projects/utbot/test_project/atheris_usage.py", line 8, in TestOneInput
    merge_sort(data)
  File "/home/tochilinak/Documents/projects/utbot/test_project/main.py", line 221, in merge_sort
    merge_sort(items[:mid]),
  File "/home/tochilinak/Documents/projects/utbot/test_project/main.py", line 221, in merge_sort
    merge_sort(items[:mid]),
  File "/home/tochilinak/Documents/projects/utbot/test_project/main.py", line 221, in merge_sort
    merge_sort(items[:mid]),
  [Previous line repeated 989 more times]
  File "/home/tochilinak/Documents/projects/utbot/test_project/main.py", line 220, in merge_sort
    heapq.merge(
  File "/home/tochilinak/Documents/projects/utbot/test_project/main.py", line 215, in merge_sort
    def merge_sort(items):
RecursionError: maximum recursion depth exceeded in comparison

==24186== ERROR: libFuzzer: fuzz target exited
SUMMARY: libFuzzer: fuzz target exited
MS: 0 ; base unit: 0000000000000000000000000000000000000000
0xa,
\012
artifact_prefix='./'; Test unit written to ./crash-adc83b19e793491b1c6ea0fd8b46cd9f32e592fc
Base64: Cg==
```

```python
import atheris

with atheris.instrument_imports():
    from main import merge_sort
    import sys

def TestOneInput(data: bytes):
    merge_sort(data)

atheris.Setup(sys.argv, TestOneInput)
atheris.Fuzz()
```

Как предоставлять информацию об ошибках?

- сохранить входные данные
- воспроизвести ошибку

```
[Panda(1), Panda(2)]
[]
```

```
def merge_sort(pandas: list[Panda]):
    if len(pandas) == 1:
        return pandas
    mid = len(pandas) // 2
    sorted_left = merge_sort(pandas[:mid])
    sorted_right = merge_sort(pandas[mid:])
    return list(
        heapq.merge(
            sorted_left,
            sorted_right
        )
    )
```

```
    merge_sort(items[:mid]),
E   RecursionError: maximum recursion depth exceeded
!!! Recursion detected (same locals & position)
```

Существующие инструменты: генерация тестов

Python

Существующие инструменты: генерация тестов

Python

Java

UNITTEST BOT

EVOSUITE

Randoop

Automatic unit test generation for Java

Kex

Использование Pynguin

Pynguin GitHub

```
def merge_sort(pandas: list[Panda]):
    if len(pandas) == 1:
        return pandas
    mid = len(pandas) // 2
    sorted_left = merge_sort(pandas[:mid])
    sorted_right = merge_sort(pandas[mid:])
    return list(
        heapq.merge(
            sorted_left,
            sorted_right
        )
    )
```

Использование Pynguin

```python
def merge_sort(pandas: list[Panda]):
    if len(pandas) == 1:
        return pandas
    mid = len(pandas) // 2
    sorted_left = merge_sort(pandas[:mid])
    sorted_right = merge_sort(pandas[mid:])
    return list(
        heapq.merge(
            sorted_left,
            sorted_right
        )
    )
```

```python
import pytest
import main as module_0
import inspect as module_1


@pytest.mark.xfail(strict=True)
def test_case_1():
    list_0 = []
    module_0.merge_sort(list_0)
```

Pynguin GitHub

Использование Pynguin

```python
def merge_sort(pandas: list[Panda]):
    if len(pandas) == 1:
        return pandas
    mid = len(pandas) // 2
    sorted_left = merge_sort(pandas[:mid])
    sorted_right = merge_sort(pandas[mid:])
    return list(
        heapq.merge(
            sorted_left,
            sorted_right
        )
    )
```

```python
import pytest
import main as module_0
import inspect as module_1


@pytest.mark.xfail(strict=True)
def test_case_0():
    none_type_0 = None
    panda_0 = module_0.Panda(none_type_0)
    var_0 = module_1.ismemberdescriptor(none_type_0)
    var_1 = panda_0.__repr__()
    var_2 = var_0.__repr__()
    var_3 = module_0.merge_sort(var_2)
    var_0.__contains__(var_3)


@pytest.mark.xfail(strict=True)
def test_case_1():
    list_0 = []
    module_0.merge_sort(list_0)
```

Pynguin GitHub

Использование Pynguin

```python
def merge_sort(pandas: list[Panda]):
    if len(pandas) == 1:
        return pandas
    mid = len(pandas) // 2
    sorted_left = merge_sort(pandas[:mid])
    sorted_right = merge_sort(pandas[mid:])
    return list(
        heapq.merge(
            sorted_left,
            sorted_right
        )
    )
```

```python
import pytest
import main as module_0
import inspect as module_1


@pytest.mark.xfail(strict=True)
def test_case_0():
    none_type_0 = None
    panda_0 = module_0.Panda(none_type_0)
    var_0 = module_1.ismemberdescriptor(none_type_0)
    var_1 = panda_0.__repr__()
    var_2 = var_0.__repr__()
    var_3 = module_0.merge_sort(var_2)
    var_0.__contains__(var_3)


@pytest.mark.xfail(strict=True)
def test_case_1():
    list_0 = []
    module_0.merge_sort(list_0)
```

Pynguin GitHub

Тестирование на основе свойств (Property-based testing)

PBT

```
def encode(text):
    ...

def decode(encoded_text):
    ...
```

What is PBT?

PBT

```python
def encode(text):
    ...

def decode(encoded_text):
    ...

@given(data=...)
def test_encoder(data: str):
    assert data == decode(encode(data))
```

What is PBT?

Использование Hypothesis

```
def merge_sort(pandas: list[Panda]):
    if len(pandas) == 1:
        return pandas
    mid = len(pandas) // 2
    sorted_left = merge_sort(pandas[:mid])
    sorted_right = merge_sort(pandas[mid:])
    return list(
        heapq.merge(
            sorted_left,
            sorted_right
        )
    )
```

Hypothesis github

Использование Hypothesis

Hypothesis github

```python
def merge_sort(pandas: list[Panda]):
    if len(pandas) == 1:
        return pandas
    mid = len(pandas) // 2
    sorted_left = merge_sort(pandas[:mid])
    sorted_right = merge_sort(pandas[mid:])
    return list(
        heapq.merge(
            sorted_left,
            sorted_right
        )
    )

@given(pandas=...)
def test_merge_sort(pandas: list[Panda]):
     merge_sort(pandas)
```

Использование Hypothesis

Hypothesis github

```python
def merge_sort(pandas: list[Panda]):
    if len(pandas) == 1:
        return pandas
    mid = len(pandas) // 2
    sorted_left = merge_sort(pandas[:mid])
    sorted_right = merge_sort(pandas[mid:])
    return list(
        heapq.merge(
            sorted_left,
            sorted_right
        )
    )

@given(pandas=...)
def test_merge_sort(pandas: list[Panda]):
     merge_sort(pandas)
```

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
main.py:328: in test_merge_sort
    merge_sort(pandas)
main.py:316: in merge_sort
    sorted_left = merge_sort(pandas[:mid])
main.py:316: in merge_sort
    sorted_left = merge_sort(pandas[:mid])
E   RecursionError: maximum recursion depth exceeded
E   Falsifying example: test_merge_sort(
E       pandas=[],
E   )
!!! Recursion detected (same locals & position)
```

Использование Hypothesis

```
def f(x: int):
    if x == 1234567:
        return 1 / 0
    if x == 12345678:
        return x + 1
    return x
```

Hypothesis github

Использование Hypothesis

Hypothesis github

```
def f(x: int):
    if x == 1234567:
        return 1 / 0
    if x == 12345678:
        return x + 1
    return x


@given(x=...)
@settings(verbosity=Verbosity.verbose)
def test_f(x: int):
    assert f(x) == x
```

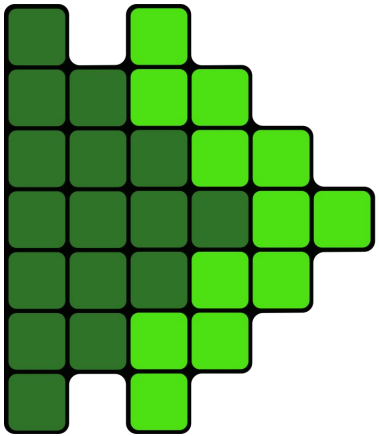
Использование Hypothesis

```python
def f(x: int):
    if x == 1234567:
        return 1 / 0
    if x == 12345678:
        return x + 1
    return x


@given(x=...)
@settings(verbosity=Verbosity.verbose)
def test_f(x: int):
    assert f(x) == x
```

```
Trying example: test_f(
    x=72,
)
Trying example: test_f(
    x=-385078233,
)
Trying example: test_f(
    x=-18167,
)
Trying example: test_f(
    x=-70,
)
Trying example: test_f(
    x=-97,
)
Trying example: test_f(
    x=2,
)
Trying example: test_f(
    x=-460058550279713970673953977144951708895,
)
Trying example: test_f(
    x=-112,
)
```

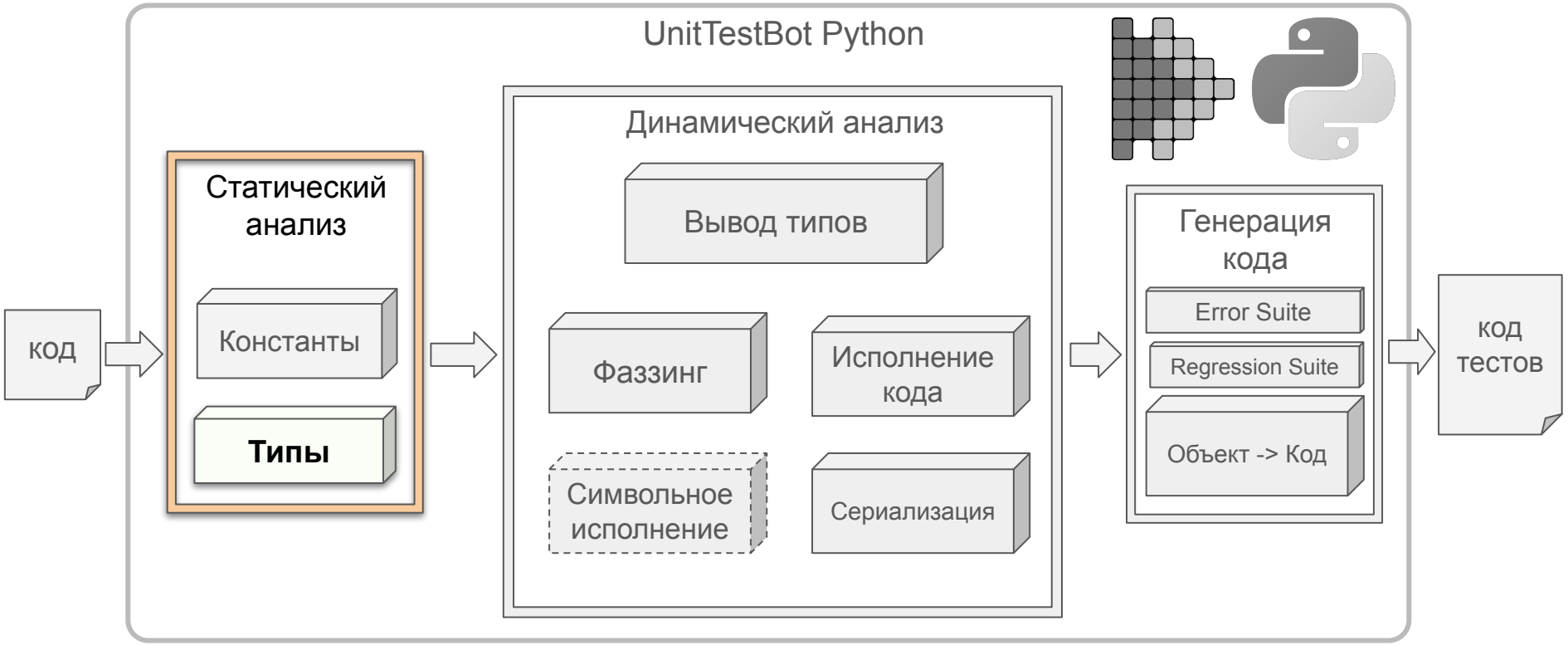
Hypothesis github

План основной части доклада



- Статическая типизация в Python
- Генерация входных значений
  - Фаззинг
  - Вывод типов
  - Символьное исполнение
- Кодогенерация
- Демо

Сбор информации о типах



Напоминание: статическая типизация в Python

Документация по typing

PEP 483

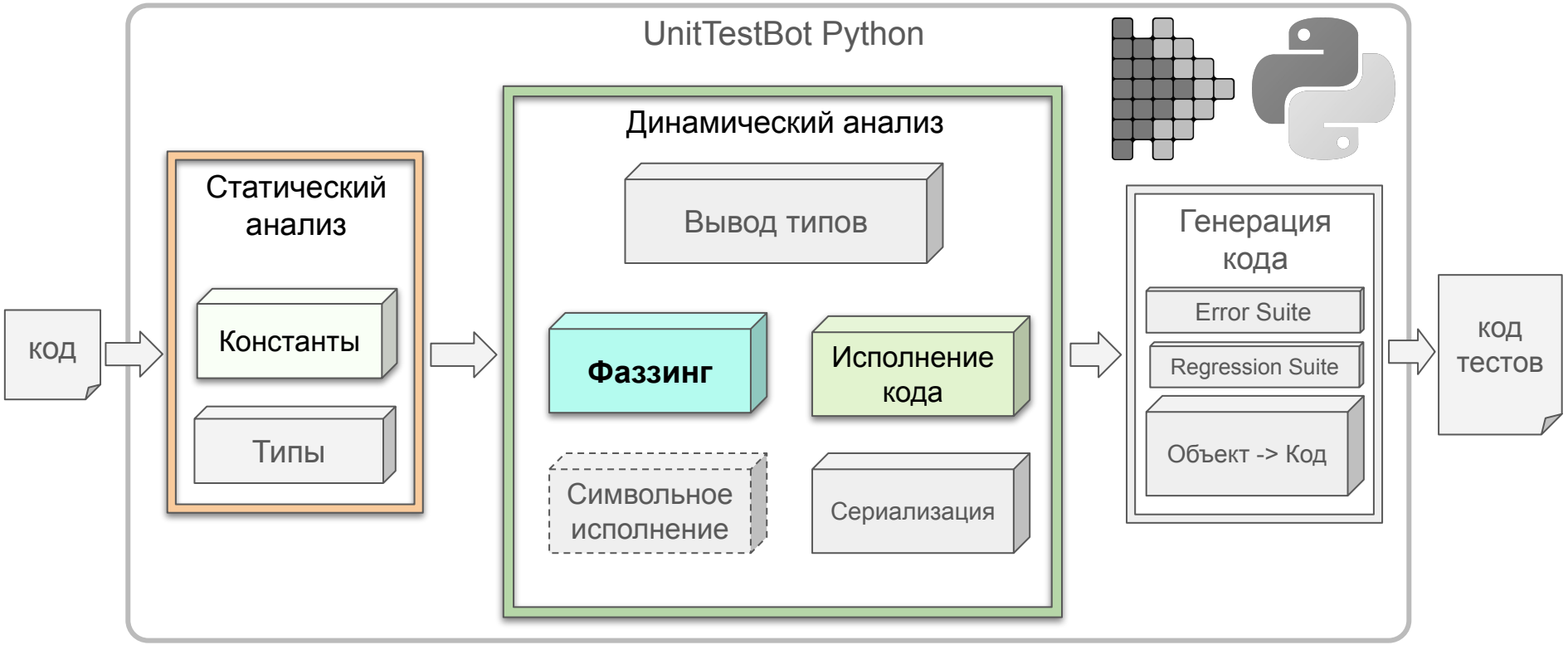
PEP 484

```python
import typing as tp

class Cat:
    speak = "Meow"

class Dog:
    speak = "Woof"

class SupportsSpeak(tp.Protocol):
    speak: str

def do_speaking(animal: SupportsSpeak):
    print(animal.speak)

do_speaking(Cat())
do_speaking(Dog())
```

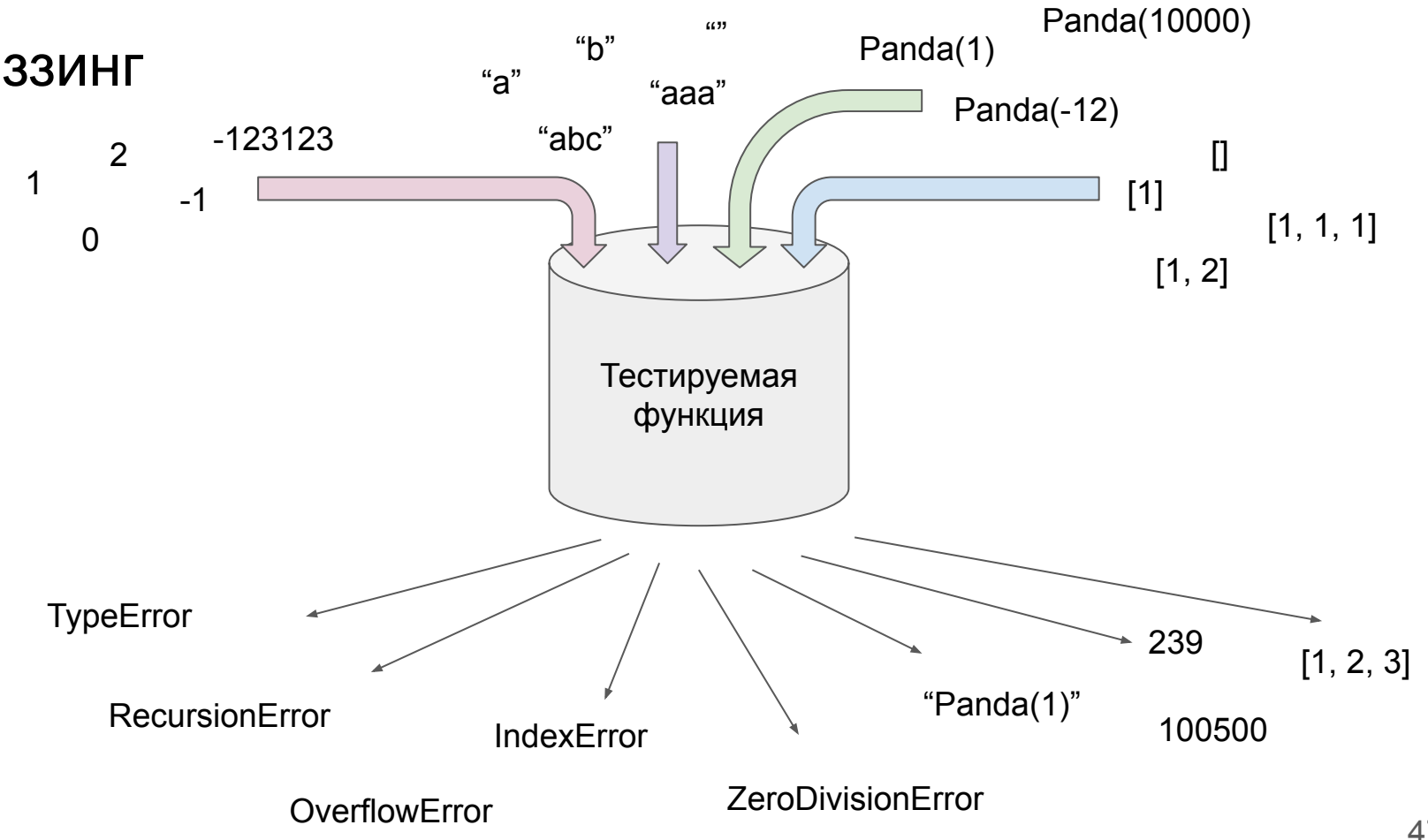
Хранение информации о типах

- Много примитивов (Union, Callable, Literal, …)
- Типовые переменные, подстановки
- Циклические зависимости
- Учет наследования и MRO

Фаззинг


UnitTestBot Python
код
Статический анализ
Константы
Типы
Динамический анализ
Вывод типов
Фаззинг
Исполнение кода
Символьное исполнение
Сериализация
Генерация кода
Error Suite
Regression Suite
Объект -> Код
код тестов

Фаззинг


Panda(10000)
“”
“b”
Panda(1)
“a”
“aaa”
Panda(-12)
-123123
“abc”
[]
2
1
[1]
-1
[1, 1, 1]
0
[1, 2]
Тестируемая функция
TypeError
239
[1, 2, 3]
RecursionError
“Panda(1)”
IndexError
100500
OverflowError
ZeroDivisionError

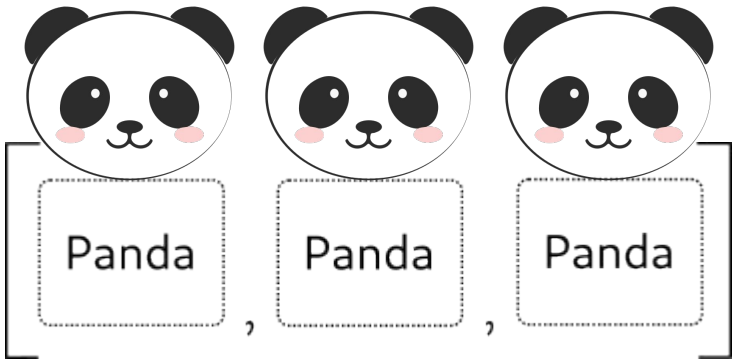
Как генерировать объекты?

```
class Panda:
    def __init__(self, weight: int):
        self.weight = weight

    def __lt__(self, other):
        return self.weight < other.weight


def merge_sort(pandas: list[Panda]):
    if len(pandas) == 1:
        return pandas
    mid = len(pandas) // 2
    sorted_left = merge_sort(pandas[:mid])
    sorted_right = merge_sort(pandas[mid:])
    return list(
        heapq.merge(
            sorted_left,
            sorted_right
        )
    )
```

Как генерировать объекты?

- Коллекции `list[Panda]`
  - list, dict, set, tuple

```
[Panda(0), Panda(1), Panda(2)]
[Panda(123984723984287361O2), Panda(-1)]
[Panda(99999999)]
[]
```




```python
class Panda:
    def __init__(self, weight: int):
        self.weight = weight

    def __lt__(self, other):
        return self.weight < other.weight


def merge_sort(pandas: list[Panda]):
    if len(pandas) == 1:
        return pandas
    mid = len(pandas) // 2
    sorted_left = merge_sort(pandas[:mid])
    sorted_right = merge_sort(pandas[mid:])
    return list(
        heapq.merge(
            sorted_left,
            sorted_right
        )
    )
```

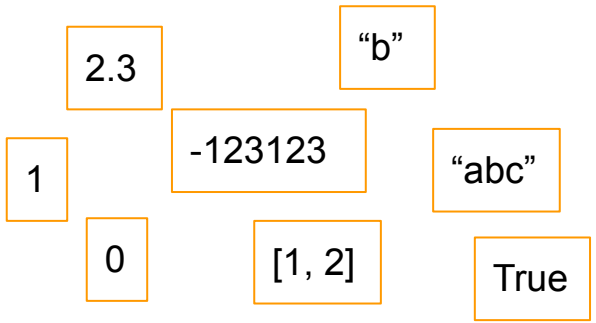
Как генерировать объекты?

- Объекты `Panda`
  - `__init__`
  - `__new__`
  - поле `weight`

```python
class Panda:
    def __init__(self, weight: int):
        self.weight = weight

    def __lt__(self, other):
        return self.weight < other.weight


def merge_sort(pandas: list[Panda]):
    if len(pandas) == 1:
        return pandas
    mid = len(pandas) // 2
    sorted_left = merge_sort(pandas[:mid])
    sorted_right = merge_sort(pandas[mid:])
    return list(
        heapq.merge(
            sorted_left,
            sorted_right
        )
    )
```

Как генерировать объекты?

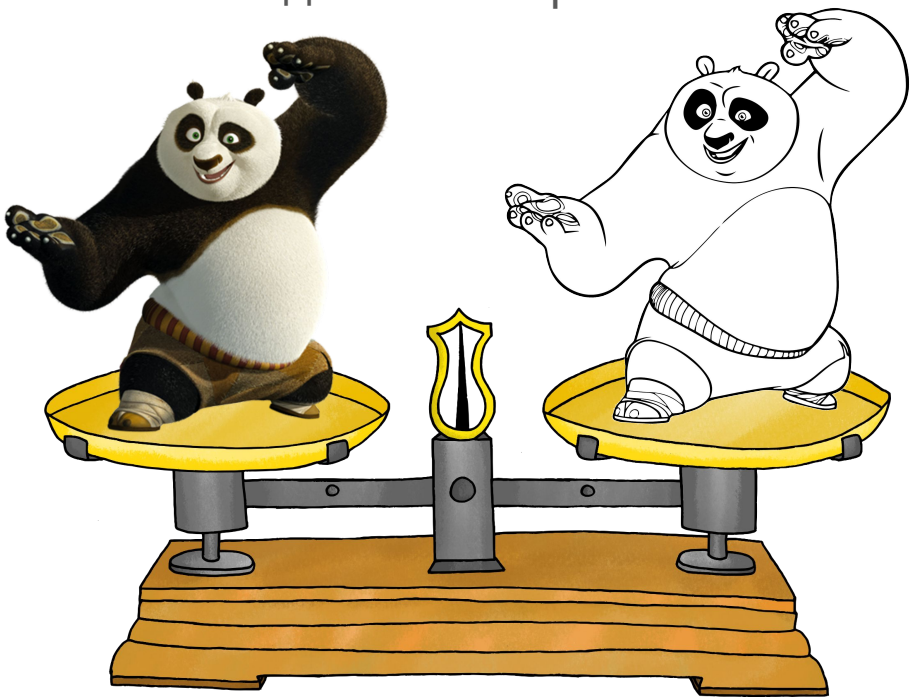
- Примитивы `weight: int`
  - `int`
  - `float`
  - `str`
  - `bool`
- Константы




```python
class Panda:
    def __init__(self, weight: int):
        self.weight = weight

    def __lt__(self, other):
        return self.weight < other.weight


def merge_sort(pandas: list[Panda]):
    if len(pandas) == 1:
        return pandas
    mid = len(pandas) // 2
    sorted_left = merge_sort(pandas[:mid])
    sorted_right = merge_sort(pandas[mid:])
    return list(
        heapq.merge(
            sorted_left,
            sorted_right
        )
    )
```

Как генерировать объекты?

- Наследование и протоколы

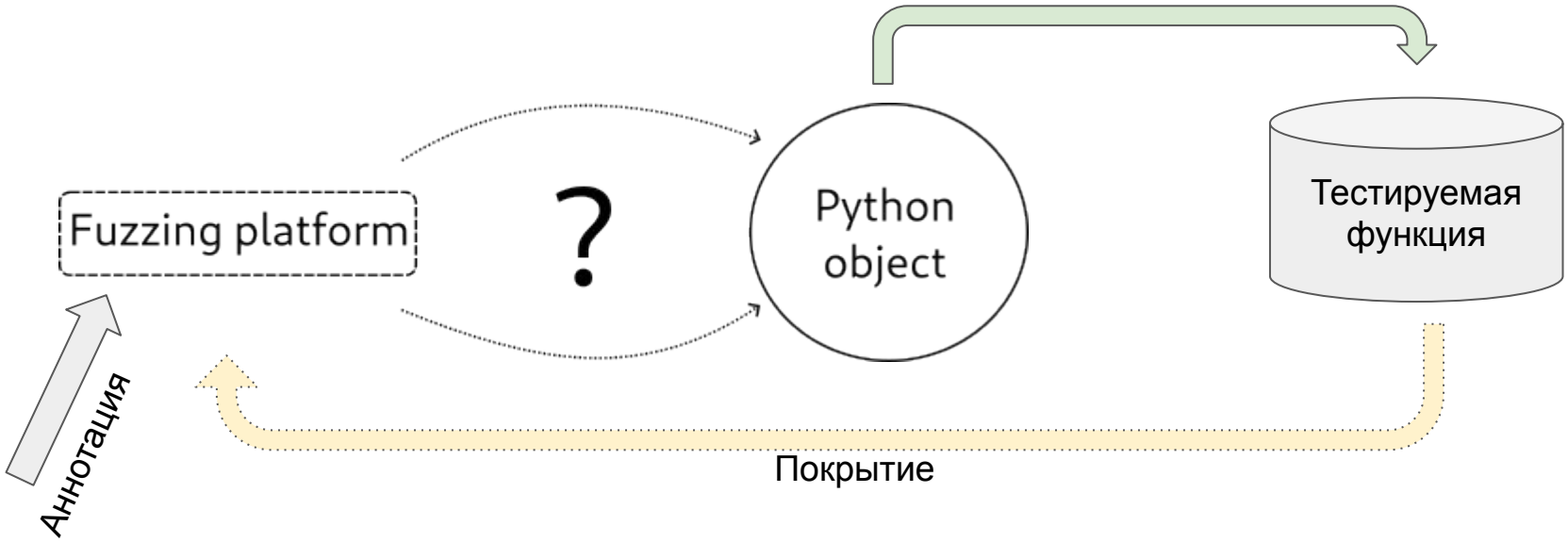


```python
class SupportsWeight(typing.Protocol):
    weight: int
```

```python
class Panda:
    def __init__(self, weight: int):
        self.weight = weight

    def __lt__(self, other):
        return self.weight < other.weight

def merge_sort(pandas: list[SupportsWeight]):
    if len(pandas) == 1:
        return pandas
    mid = len(pandas) // 2
    sorted_left = merge_sort(pandas[:mid])
    sorted_right = merge_sort(pandas[mid:])
    return list(
        heapq.merge(
            sorted_left,
            sorted_right
        )
    )
```

Как генерировать объекты?

Fuzzing Book



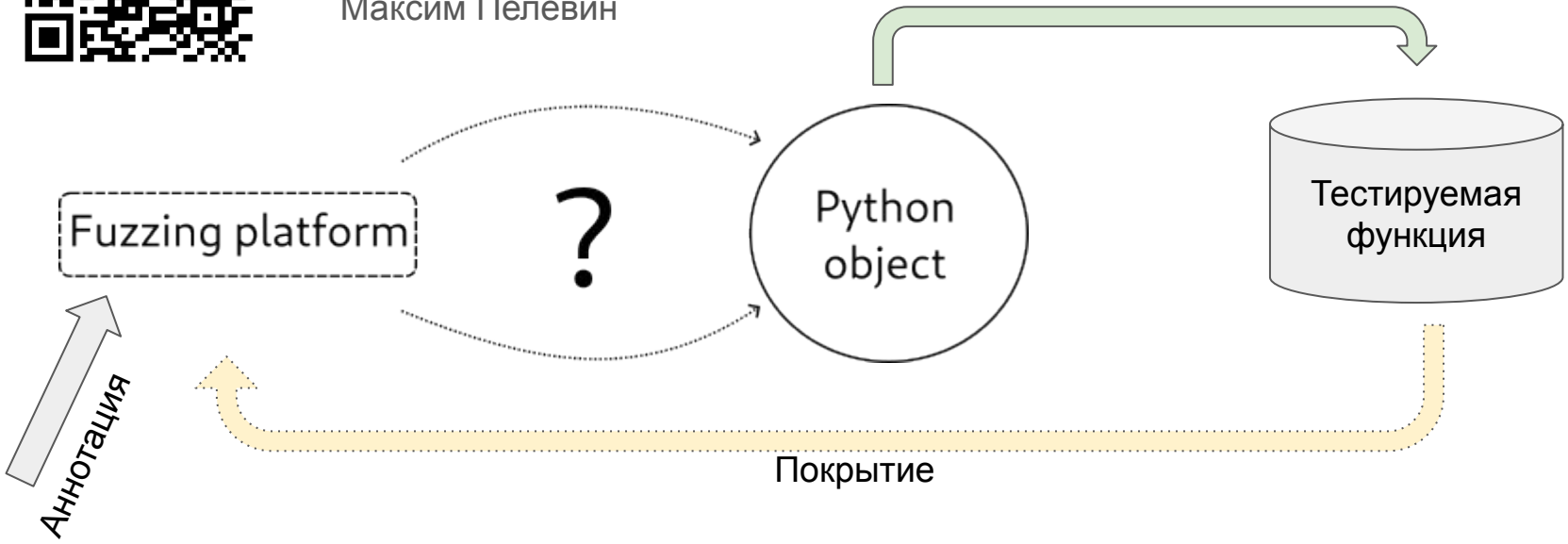
Как генерировать объекты?

Joker<?>

Знакомство с фаззерами

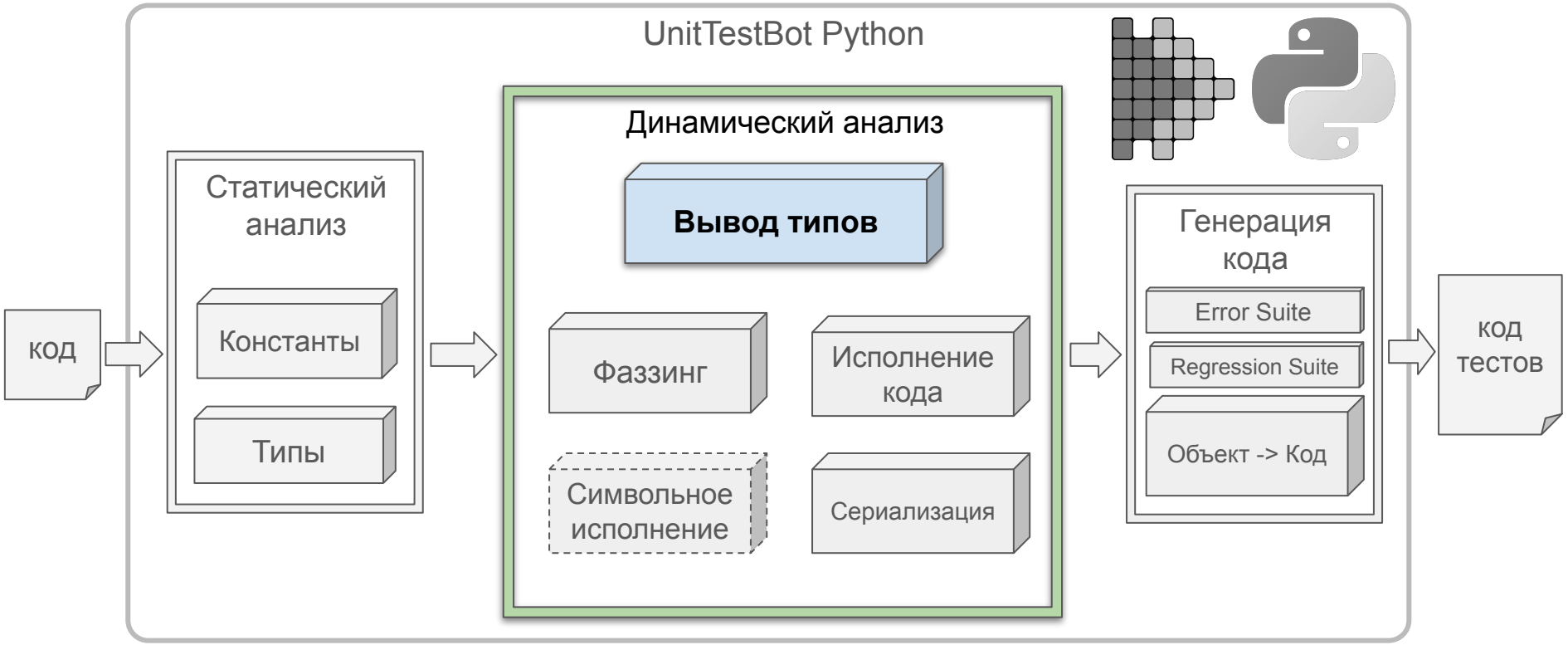
Максим Пелевин

Fuzzing Book



А если аннотаций типов нет?

А если аннотаций типов нет?



А если аннотаций типов нет?

- Использования функций
  - Динамически: MonkeyType, PyAnnotate
  - Статически: mypy.dmypy suggest (experimental)

MonkeyType

PyAnnotate

А если аннотаций типов нет?

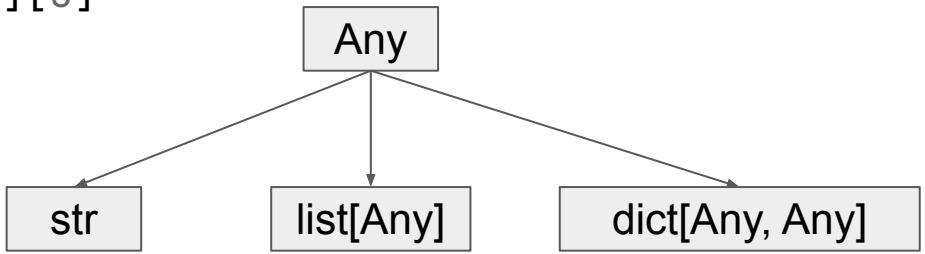
- Использования функций
  - Динамически: MonkeyType, PyAnnotate
  - Статически: mypy.dmypy suggest (experimental)
- Формальные подходы: сбор типовых ограничений, унификация

А если аннотаций типов нет?

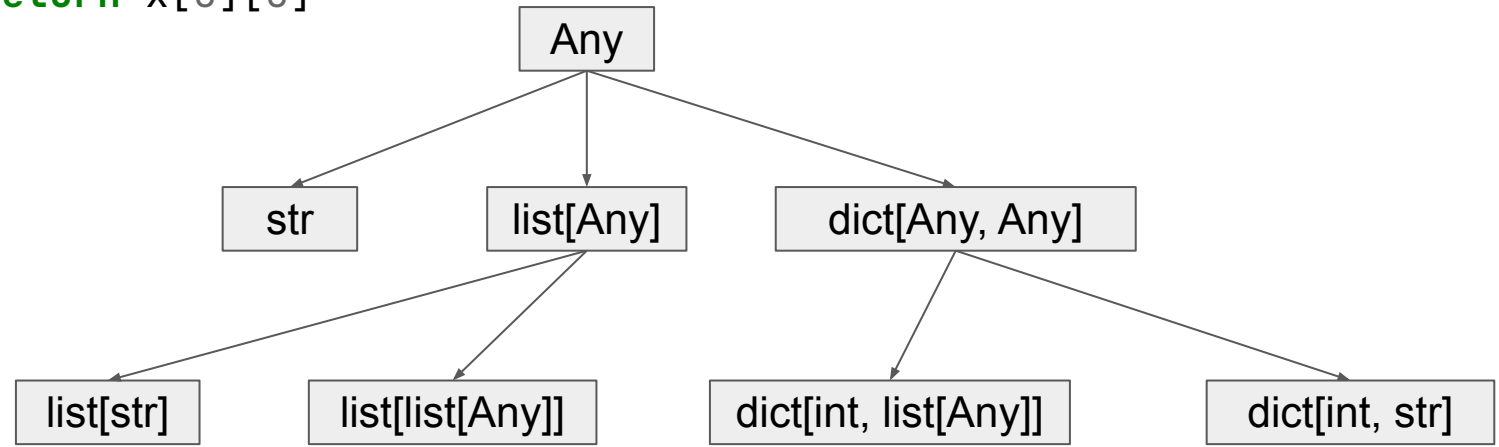
- Использования функций
  - Динамически: MonkeyType, PyAnnotate
  - Статически: mypy.dmypy suggest (experimental)
- Формальные подходы: сбор типовых ограничений, унификация
- Машинное обучение, LLM

```
def f(x):
    return x[0][0]
```

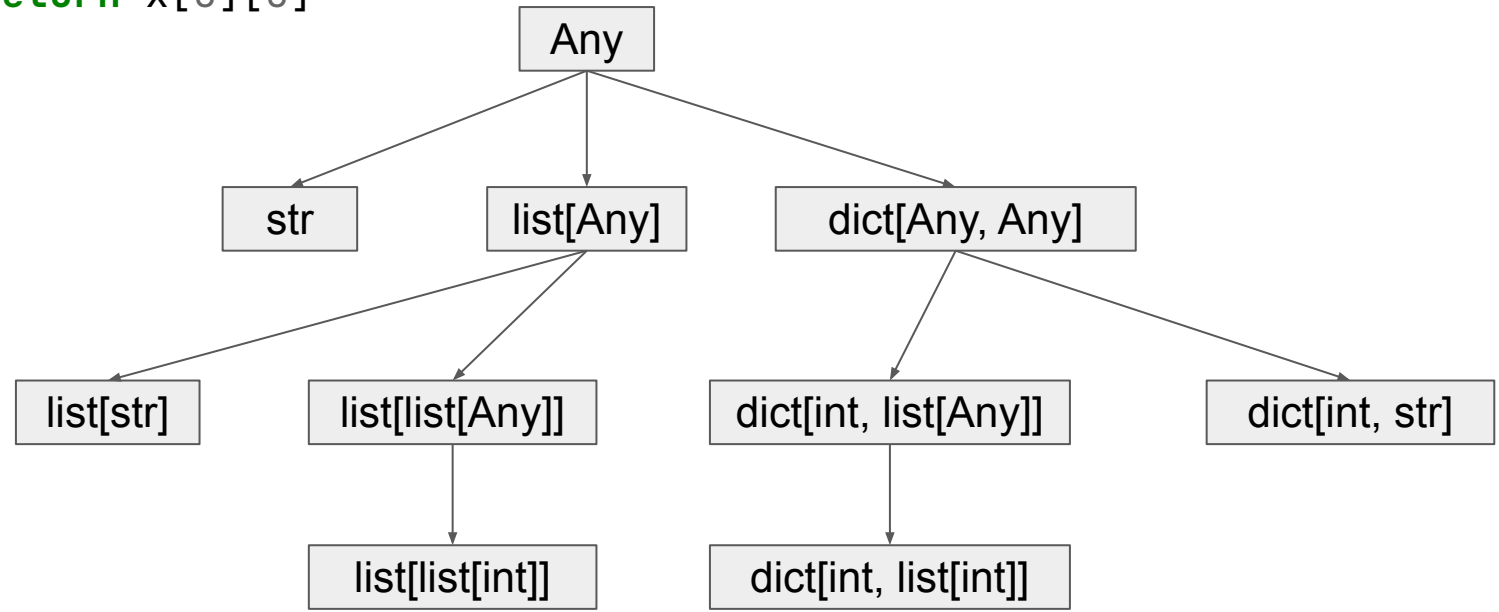
```
def f(x):
    return x[0][0]
```



```
def f(x):
    return x[0][0]
```



```python
def f(x):
    return x[0][0]
```



Как приоритизировать типы?

```
int  str  bool  list[int]  float  list[list[int]]  dict[int, str]
```

Как приоритизировать типы?

```
int  str  bool  list[int]  float  list[list[int]]  dict[int, str]
```

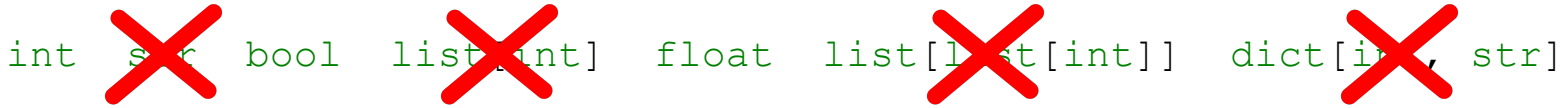
```
def f(x):
    return x + 1
```

Как приоритизировать типы?

```
int  str  bool  list[int]  float  list[list[int]]  dict[int, str]
```

```
def f(x):
    return x + 1


class SupportsAdd(tp.Protocol):
    def __add__(self, other: int): ...
```

Как приоритизировать типы?




```python
def f(x):
    return x + 1
```

```python
class SupportsAdd(tp.Protocol):
    def __add__(self, other: int): ...
```

Как приоритизировать типы?

```
int  str  bool  list[int]  float  list[list[int]]  dict[int, str]
```

```
def f(x):
    return x[0][0]
```

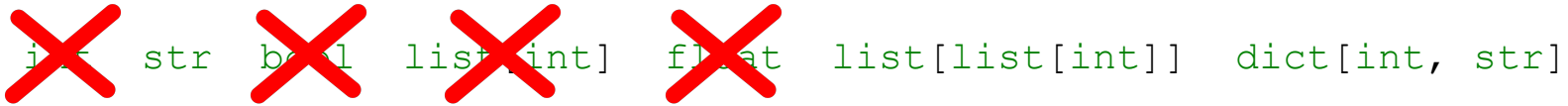
Как приоритизировать типы?

```
int  str  bool  list[int]  float  list[list[int]]  dict[int, str]
```

```
def f(x):
    return x[0][0]
```

```
class SupportsSubscript(tp.Protocol):
    def __getitem__(self, index: int): ...

class SupportsDoubleSubscript(tp.Protocol):
    def __getitem__(self, index: int) → SupportsSubscript: ...
```

Как приоритизировать типы?




```python
def f(x):
    return x[0][0]
```

```python
class SupportsSubscript(tp.Protocol):
    def __getitem__(self, index: int): ...

class SupportsDoubleSubscript(tp.Protocol):
    def __getitem__(self, index: int) → SupportsSubscript: ...
```

Как приоритизировать типы?

```
int  str  bool  list[int]  float  list[list[int]]  dict[int, str]
```

```
def f(x):
    heapq.heapify(x)
```

Как приоритизировать типы?

```
int  str  bool  list[int]  float  list[list[int]]  dict[int, str]
```

```
def f(x):
    heapq.heapify(x)
```

```
def heapify(__heap: list[Any]) → None: ...
```

typeshed

Как приоритизировать типы?

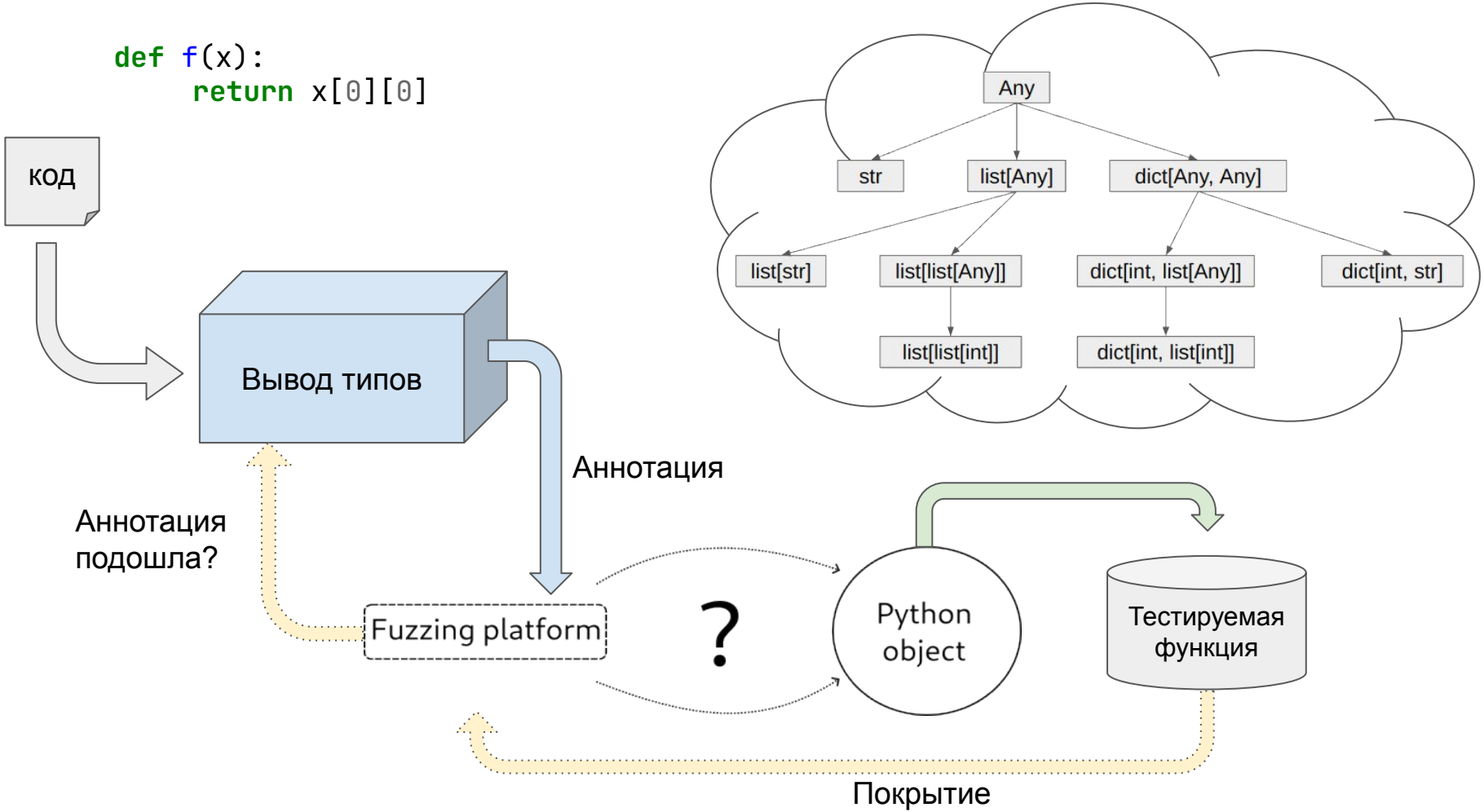
~~int~~ ~~str~~ ~~bool~~ `list[int]` ~~float~~ `list[list[int]]` `dict[int, str]`

```
def f(x):
    heapq.heapify(x)
```

```
def heapify(__heap: list[Any]) → None: ...
```

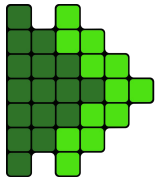
typeshed

```python
def f(x):
    return x[0][0]
```



```python
def matrix(data: list[list[int]], target: int):
    n = len(data)
    assert n >= 5 and target < 0
    for line in data:
        assert len(line) == n
        for elem in line:
            assert elem == target

    # some smart work
    return data
```

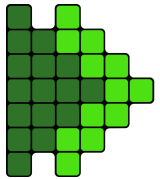
```python
def matrix(data: list[list[int]], target: int):
    n = len(data)
    assert n >= 5 and target < 0
    for line in data:
        assert len(line) == n
        for elem in line:
            assert elem == target

    # some smart work
    return data
```



```python
def matrix(
        data: list[list[int]],
        target: int
    ):
    n = len(data)
    assert n >= 5 and target < 0
    for line in data:
        assert len(line) == n
        for elem in line:
            assert elem == target

    # some smart work
    return data
```

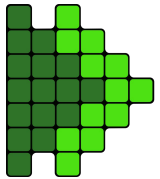
```python
def test_matrix_with_exception(self):
    """
    data = builtins.list[builtins.list[builtins.int]]
    target = -1 (mutated from max)
    """
    # This test fails because
    # function [matrix.matrix] produces [AssertionError]
    matrix.matrix(
        [
            [6, 4, -170141183460469231731687303715884105728],
            [6, 6, 6],
            [5, 5, 1],
        ],
        -1
    )
```

```python
def matrix(
        data: list[list[int]],
        target: int
    ):
    n = len(data)
    assert n >= 5 and target < 0
    for line in data:
        assert len(line) == n
        for elem in line:
            assert elem == target

    # some smart work
    return data
```



```python
def test_matrix_with_exception1(self):
    """
    data = builtins.list[builtins.list[builtins.int]]
    target = -170141183460469231731687303578445152255
    """
    # This test fails because
    # function [matrix.matrix] produces [AssertionError]
    matrix.matrix(
        [
            [4, 4, 4],
            [1],
            [170141183460469231731687303715884105727, 4, 0, -1, -1],
            [1, 4, 1, 5, 5],
            [5, 170141183460469231731687303715884105727, 6, 6, 4],
        ],
        -170141183460469231731687303578445152255,
    )
```

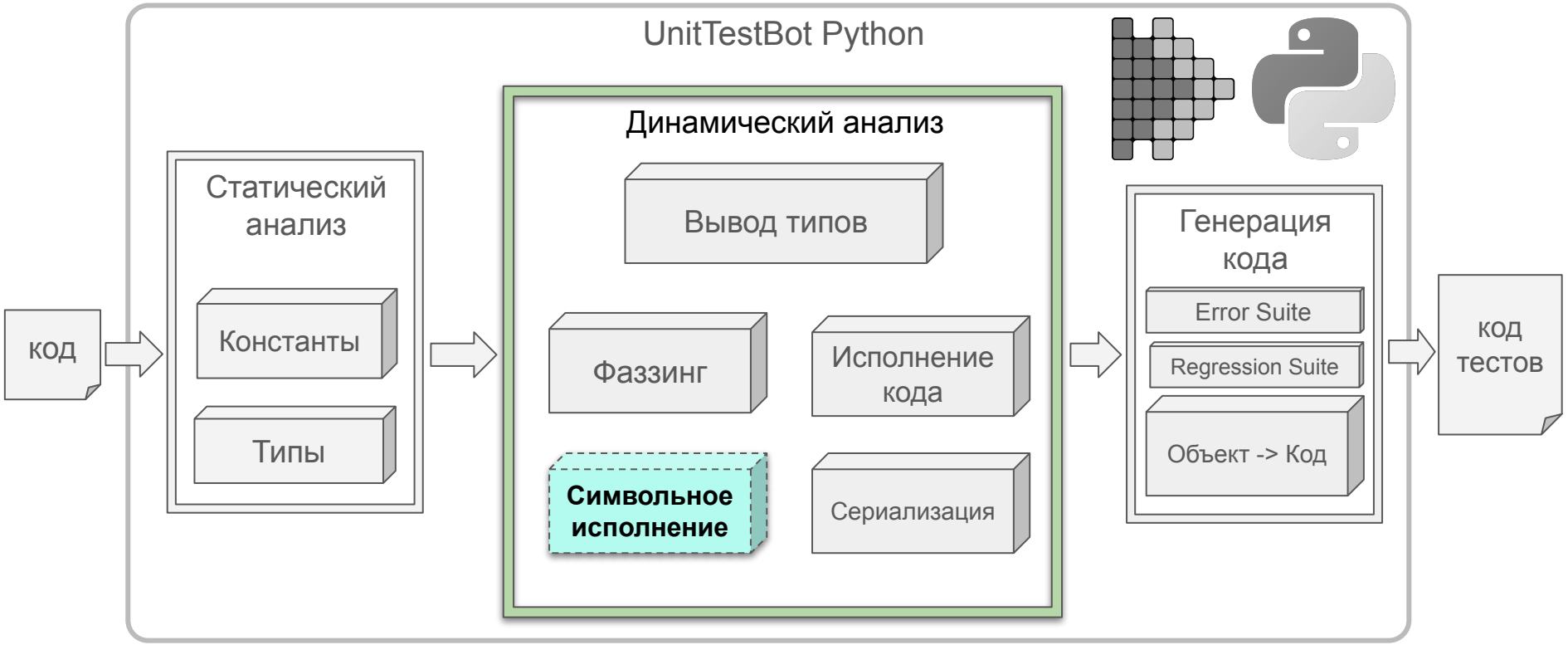
Фаззинг не поможет!



```python
def matrix(
        data: list[list[int]],
        target: int
    ):
    n = len(data)
    assert n >= 5 and target < 0
    for line in data:
        assert len(line) == n
        for elem in line:
            assert elem == target

    # some smart work
    return data
```

```python
def test_matrix_with_exception2(self):
    """
    data = builtins.list[builtins.list[builtins.int]]
    target = min
    """
    # This test fails because
    # function [matrix.matrix] produces [AssertionError]
    matrix.matrix(
        [
            [5, 1, 4, 4, 4],
            [5, 1, 4, 4, 4],
            [5, 1, 4, 4, 4],
            [5, 1, 4, 4, 4],
            [5, 1, 4, 4, 4],
        ],
        -170141183460469231731687303715884105728
    )
```

Символьное исполнение



```python
def function(a: int, b: int):
    x = 0
    if a > 0:
        x = 10
    if 15 < b + x < 20:
        return x + 1
    return x + 2
```

```python
def function(a: int, b: int):
    x = 0
    if a > 0:
        x = 10
    if 15 < b + x < 20:
        return x + 1
    return x + 2
```

$$a := \alpha \quad b := \beta$$

```python
def function(a: int, b: int):
    x = 0
    if a > 0:
        x = 10
    if 15 < b + x < 20:
        return x + 1
    return x + 2
```

$$a := \alpha \quad b := \beta$$

$$x := 0$$

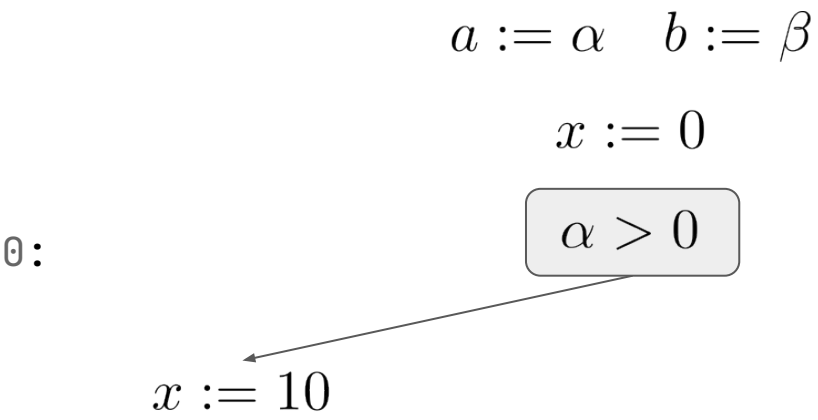
```
def function(a: int, b: int):
    x = 0
    if a > 0:
        x = 10
    if 15 < b + x < 20:
        return x + 1
    return x + 2
```

$$a := \alpha \quad b := \beta$$

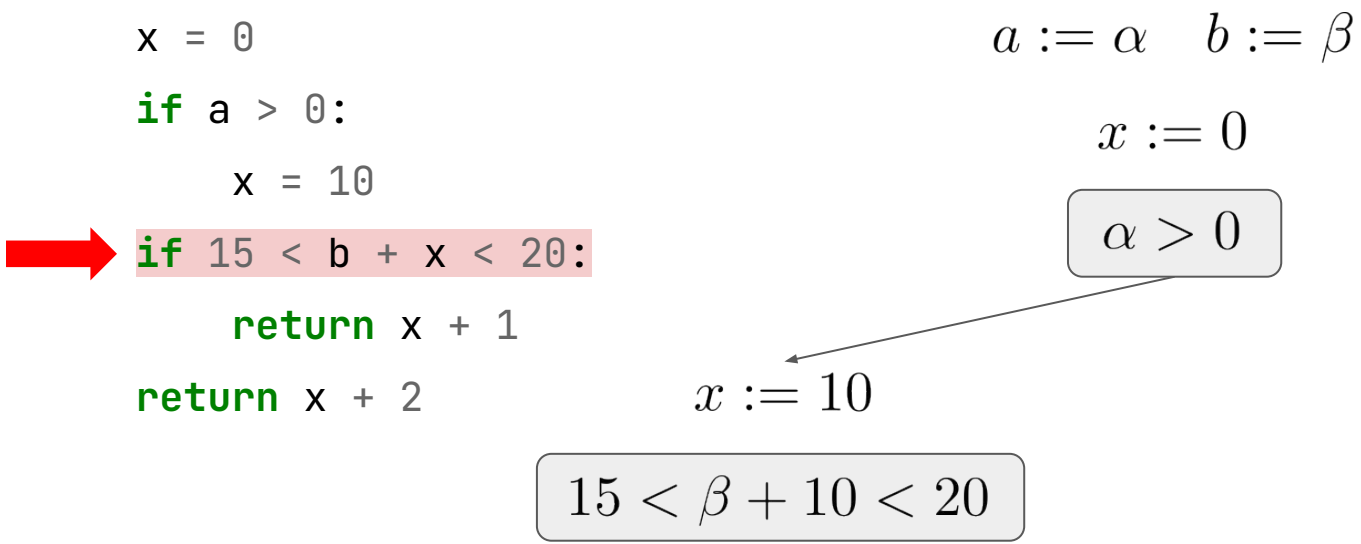
$$x := 0$$

$$\alpha > 0$$

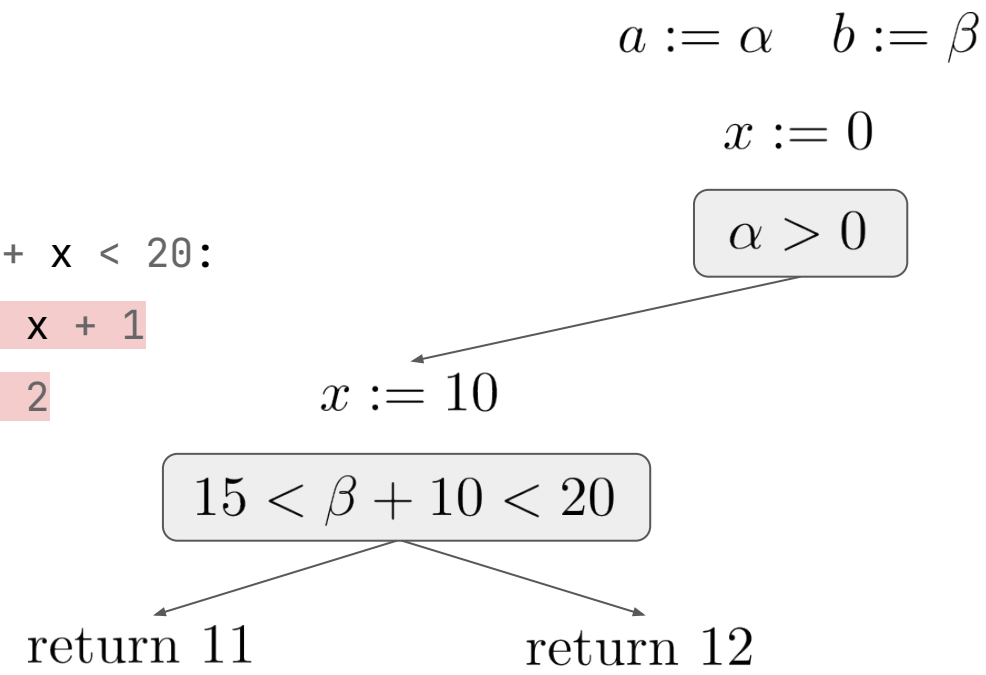
```python
def function(a: int, b: int):
    x = 0
    if a > 0:
        x = 10
    if 15 < b + x < 20:
        return x + 1
    return x + 2
```



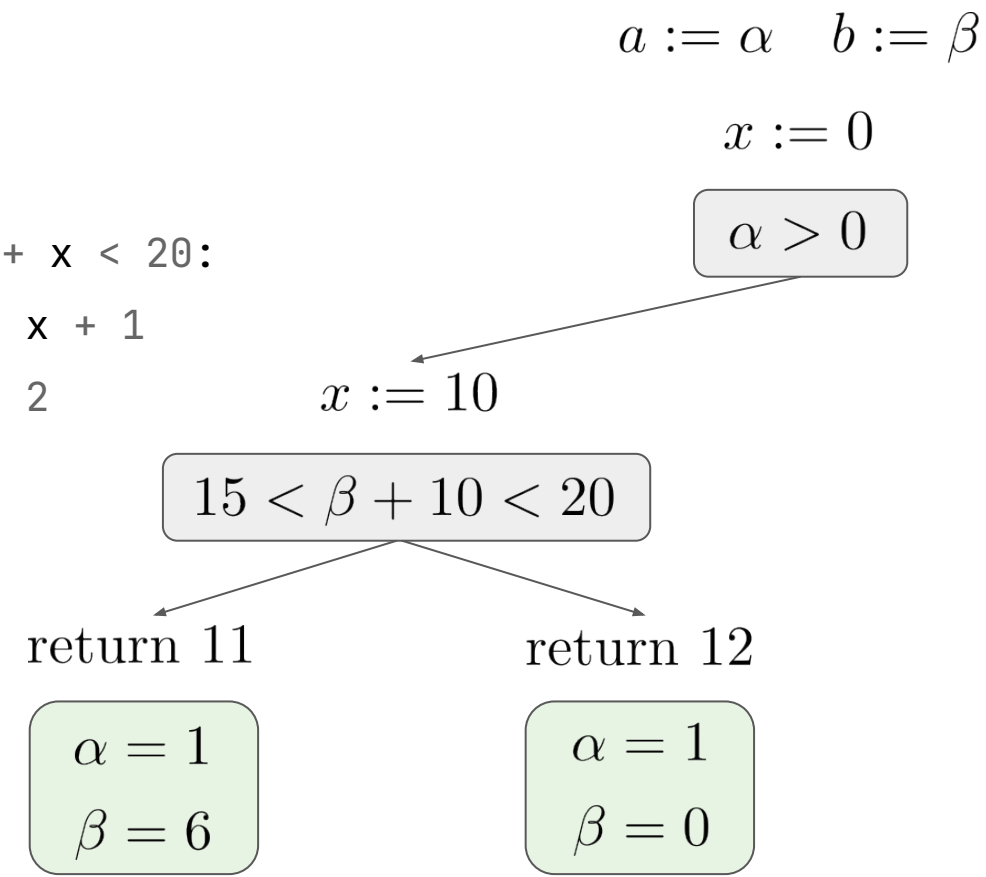
```
def function(a: int, b: int):
    x = 0
    if a > 0:
        x = 10
    if 15 < b + x < 20:
        return x + 1
    return x + 2
```



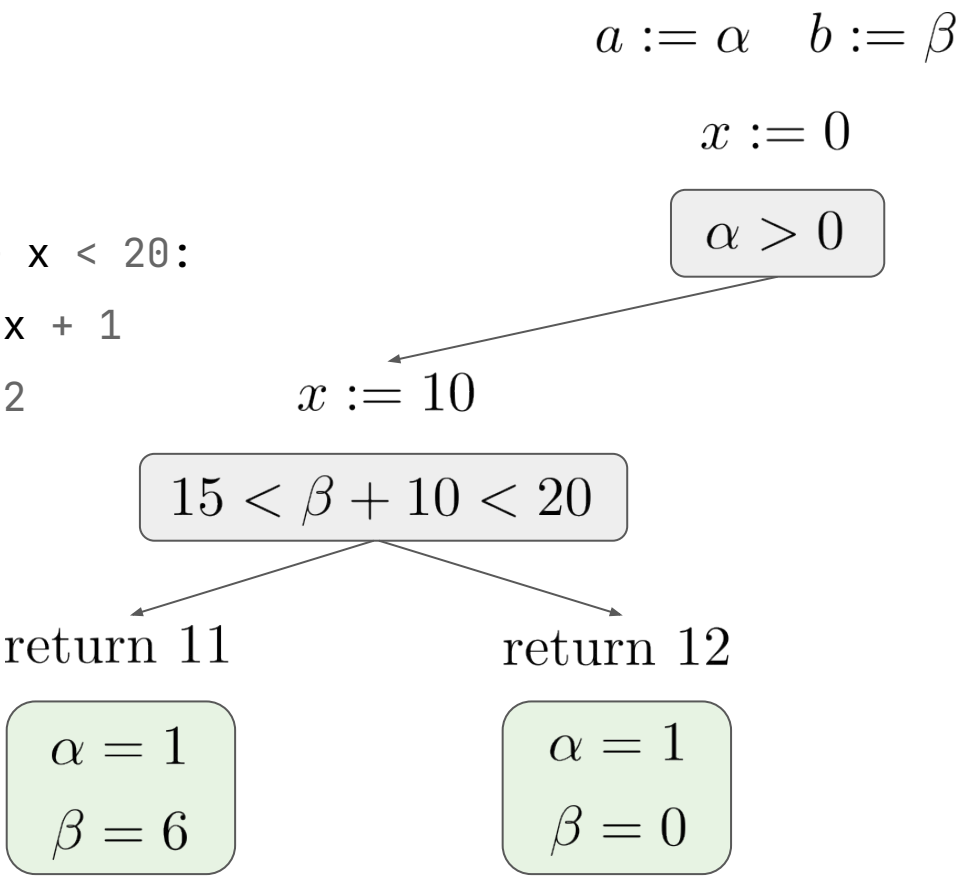
```python
def function(a: int, b: int):
    x = 0
    if a > 0:
        x = 10
    if 15 < b + x < 20:
        return x + 1
    return x + 2
```



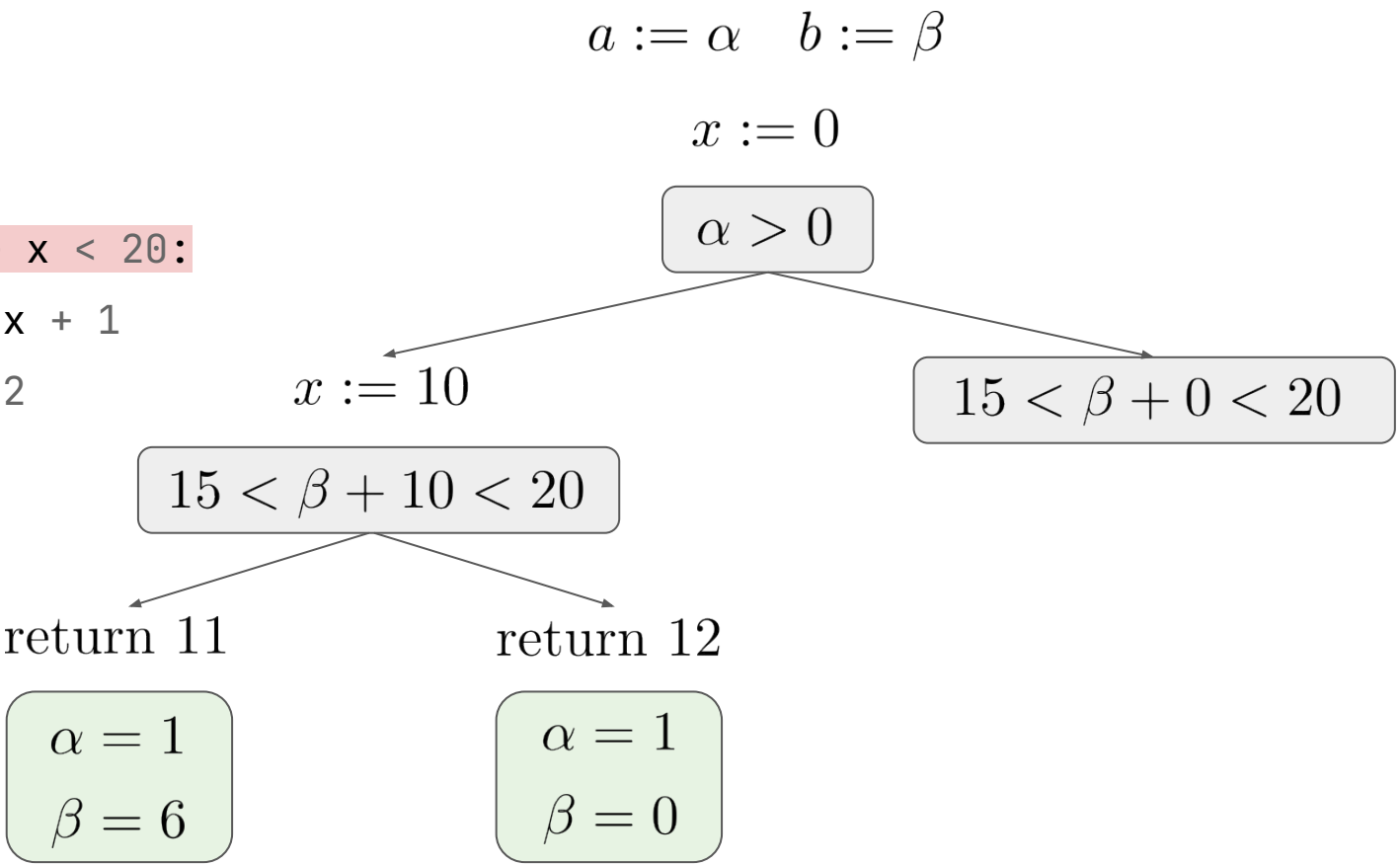
```python
def function(a: int, b: int):
    x = 0
    if a > 0:
        x = 10
    if 15 < b + x < 20:
        return x + 1
    return x + 2
```



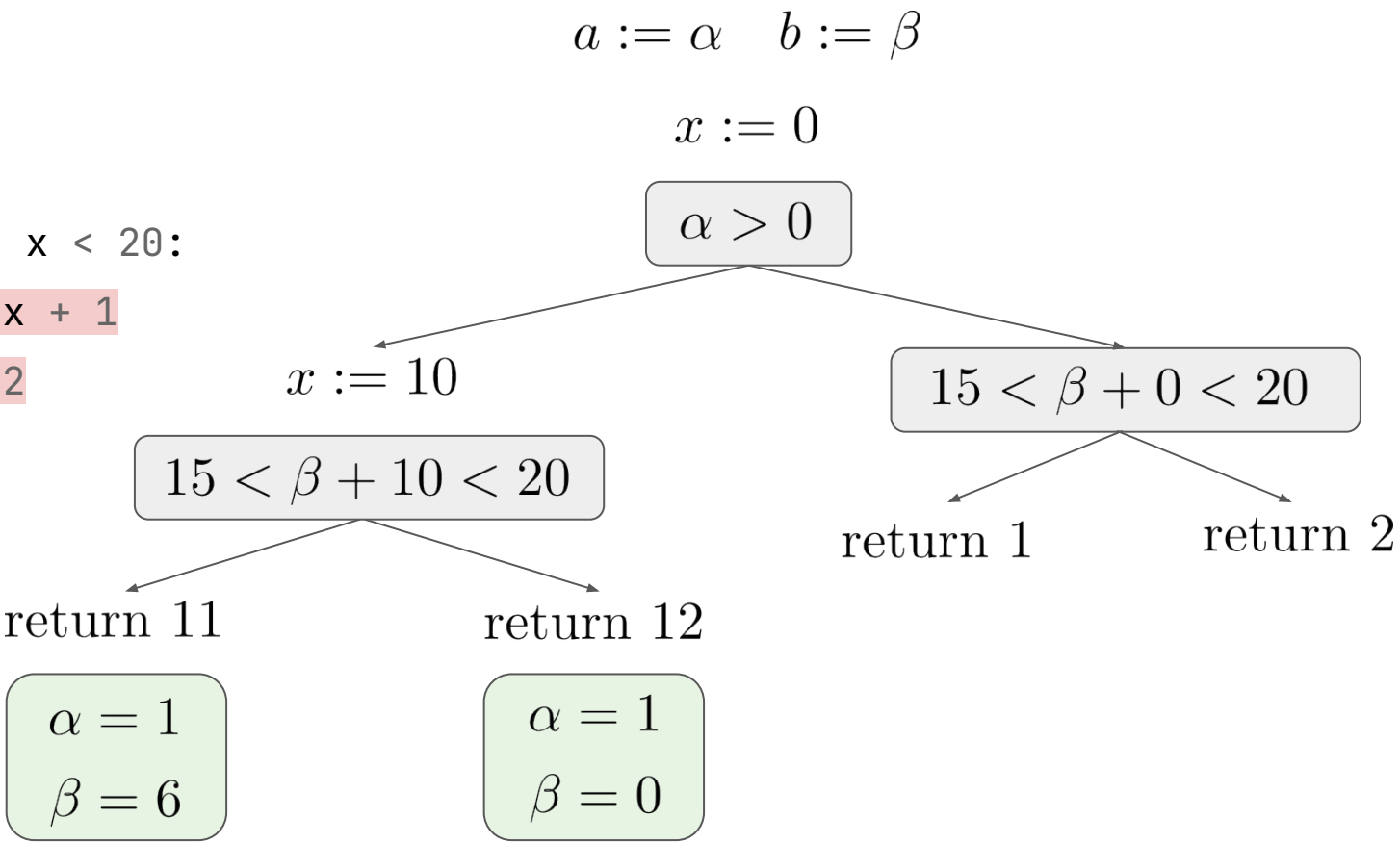
```
def function(a: int, b: int):
    x = 0
    if a > 0:
        x = 10
    if 15 < b + x < 20:
        return x + 1
    return x + 2
```



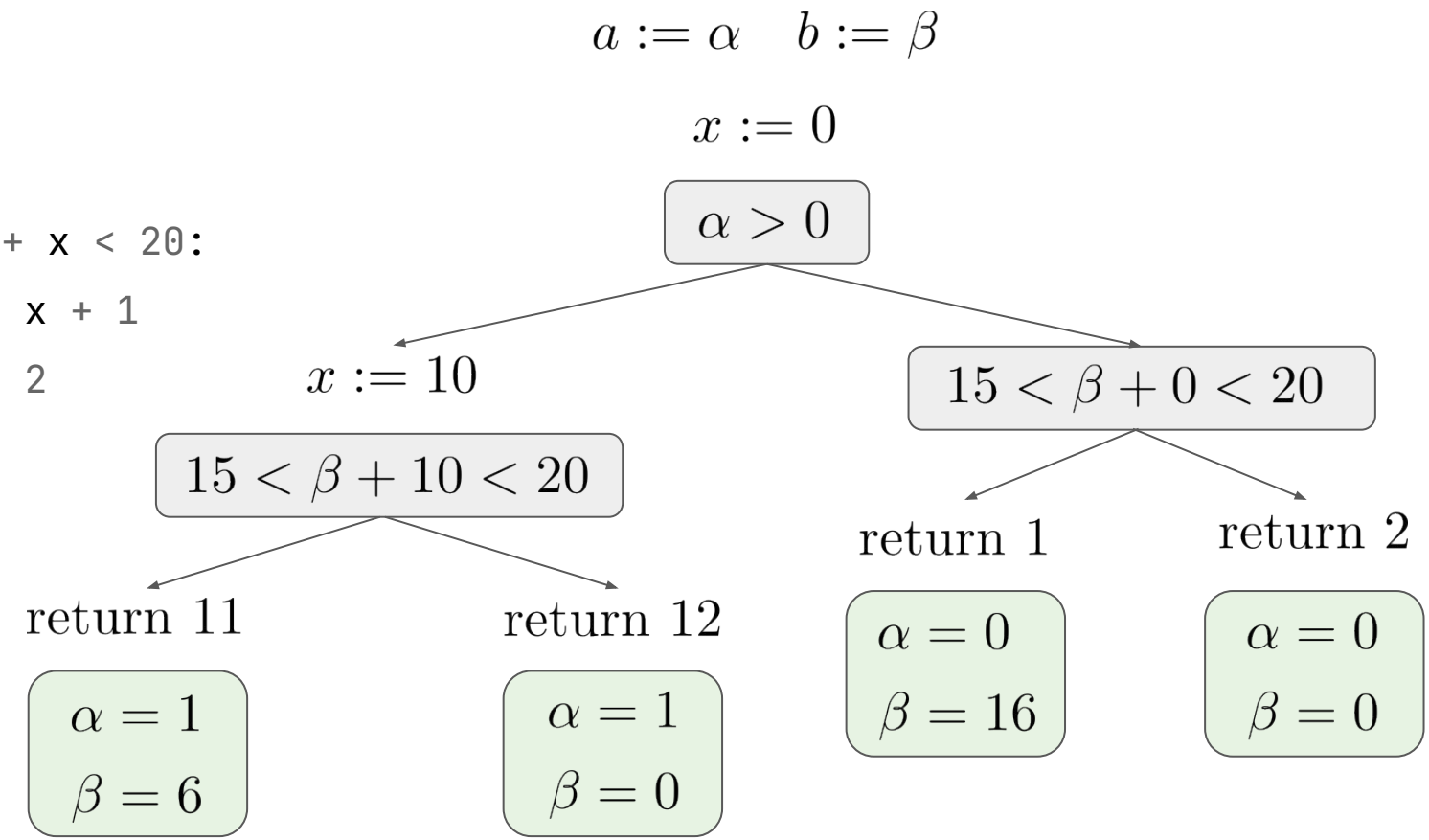
```
def function(a: int, b: int):
    x = 0
    if a > 0:
        x = 10
    if 15 < b + x < 20:
        return x + 1
    return x + 2
```



```python
def function(a: int, b: int):
    x = 0
    if a > 0:
        x = 10
    if 15 < b + x < 20:
        return x + 1
    return x + 2
```



```python
def function(a: int, b: int):
    x = 0
    if a > 0:
        x = 10
    if 15 < b + x < 20:
        return x + 1
    return x + 2
```



Существующие инструменты: символьное исполнение

https://github.com/ksluckow/awesome-symbolic-execution

Java

- Symbolic PathFinder (SPF) - Symbolic execution tool built on Java PathFinder. Supports multiple constraint solvers, lazy initialization, etc.
- JDart - Dynamic symbolic execution tool built on Java PathFinder. Supports multiple constraint solvers using JConstraints.
- CATG - Concolic execution tool that uses ASM for instrumentation. Uses CVC4.
- LimeTB - Concolic execution tool that uses Soot for instrumentation. Supports Yices and Boolector. Concolic execution can be distributed.
- Acteve - Concolic execution tool that uses Soot for instrumentation. Originally for Android analysis. Supports Z3.
- jCUTE - Concolic execution tool that uses Soot for instrumentation. Supports lp_solve.
- JFuzz - Concolic execution tool built on Java PathFinder.
- JBSE - Symbolic execution tool that uses a custom JVM. Supports CVC3, CVC4, Sicstus, and Z3.
- Key - Theorem Prover that uses specifications written in Java Modeling Language (JML).

Существующие инструменты: символьное исполнение

https://github.com/ksluckow/awesome-symbolic-execution

Python

- CrossHair - Symbolic execution tool for verifying properties of Python functions.
- PyExZ3 - Symbolic execution of Python functions. A rewrite of the NICE project's symbolic execution tool.

Java

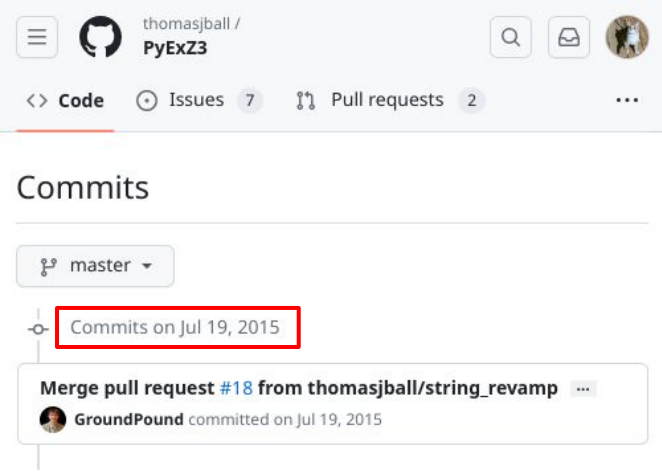
- Symbolic PathFinder (SPF) - Symbolic execution tool built on Java PathFinder. Supports multiple constraint solvers, lazy initialization, etc.
- JDart - Dynamic symbolic execution tool built on Java PathFinder. Supports multiple constraint solvers using JConstraints.
- CATG - Concolic execution tool that uses ASM for instrumentation. Uses CVC4.
- LimeTB - Concolic execution tool that uses Soot for instrumentation. Supports Yices and Boolector. Concolic execution can be distributed.
- Acteve - Concolic execution tool that uses Soot for instrumentation. Originally for Android analysis. Supports Z3.
- jCUTE - Concolic execution tool that uses Soot for instrumentation. Supports lp_solve.
- JFuzz - Concolic execution tool built on Java PathFinder.
- JBSE - Symbolic execution tool that uses a custom JVM. Supports CVC3, CVC4, Sicstus, and Z3.
- Key - Theorem Prover that uses specifications written in Java Modeling Language (JML).

Существующие инструменты: символьное исполнение

https://github.com/ksluckow/awesome-symbolic-execution

Python

- CrossHair - Symbolic execution tool for verifying properties of Python functions.
- PyExZ3 - Symbolic execution of Python functions. A rewrite of the NICE project's symbolic execution tool.



Java

- Symbolic PathFinder (SPF) - Symbolic execution tool built on Java PathFinder. Supports multiple constraint solvers, lazy initialization, etc.
- JDart - Dynamic symbolic execution tool built on Java PathFinder. Supports multiple constraint solvers using JConstraints.
- CATG - Concolic execution tool that uses ASM for instrumentation. Uses CVC4.
- LimeTB - Concolic execution tool that uses Soot for instrumentation. Supports Yices and Boolector. Concolic execution can be distributed.
- Acteve - Concolic execution tool that uses Soot for instrumentation. Originally for Android analysis. Supports Z3.
- jCUTE - Concolic execution tool that uses Soot for instrumentation. Supports lp_solve.
- JFuzz - Concolic execution tool built on Java PathFinder.
- JBSE - Symbolic execution tool that uses a custom JVM. Supports CVC3, CVC4, Sicstus, and Z3.
- Key - Theorem Prover that uses specifications written in Java Modeling Language (JML).

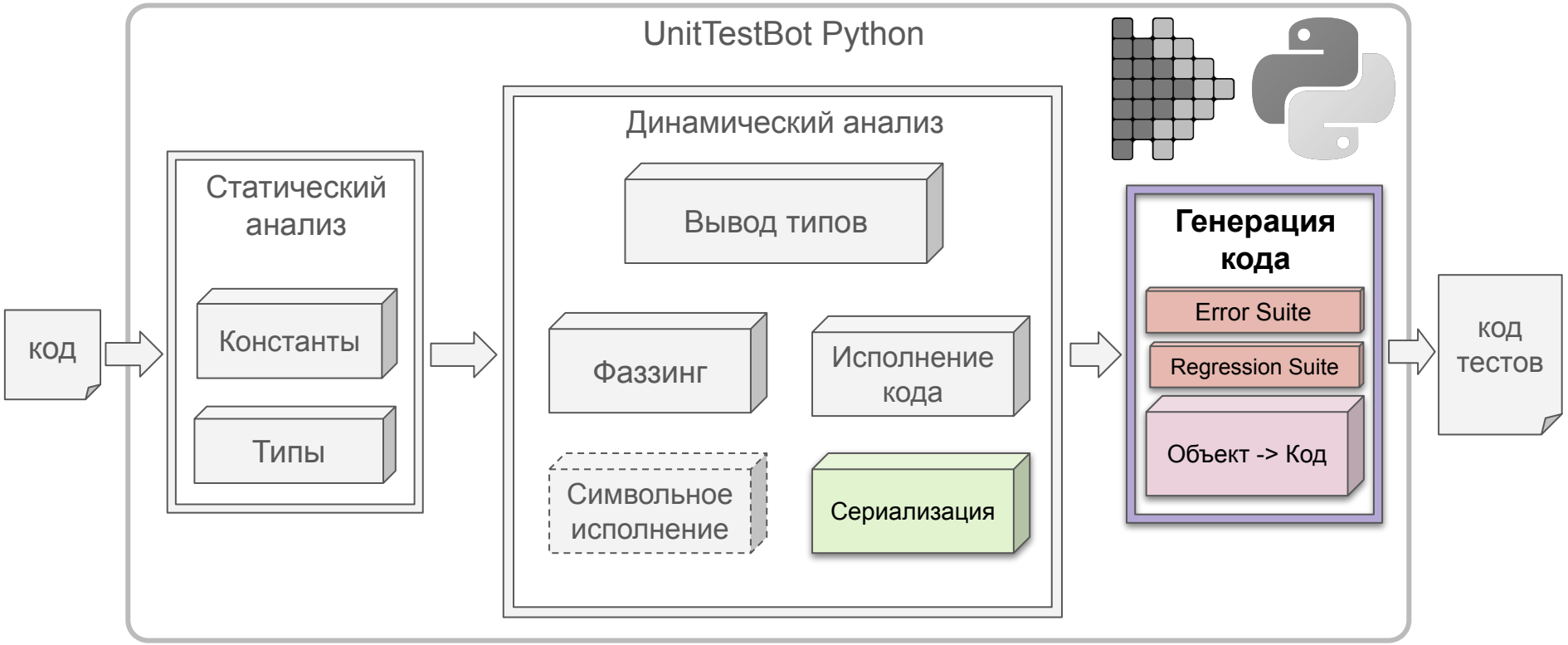
Сложности символьного исполнения

- Надо писать свой интерпретатор
- Экспоненциальный рост числа состояний
- Циклы, рекурсия
- …

Сложности символьного исполнения

- Надо писать свой интерпретатор
- Экспоненциальный рост числа состояний
- Циклы, рекурсия
- …
- **В Python: моделирование объектов произвольного типа**

Генерация текста программы



Arrange, Act, Assert

```
def test_merge_sort(self):
```

```
def merge_sort(pandas: list[Panda]):
    if len(pandas) == 1:
        return pandas
    mid = len(pandas) // 2
    sorted_left = merge_sort(pandas[:mid])
    sorted_right = merge_sort(pandas[mid:])
    return list(
        heapq.merge(
            sorted_left,
            sorted_right
        )
    )
```

Arrange, Act, Assert

```
def test_merge_sort(self):

    # Arrange
    pandas = [Panda(2), Panda(1)]
```

Arrange, **Act**, Assert

```
def test_merge_sort(self):

    # Arrange
    pandas = [Panda(2), Panda(1)]

    # Act
    actual = merge_sort(pandas)
```

Arrange, Act, Assert

```
def test_merge_sort(self):

    # Arrange
    pandas = [Panda(2), Panda(1)]

    # Act + Assert
    with self.assertRaises(RecursionError):
        merge_sort(pandas)
```

Ошибка!

Arrange, Act, **Assert**

```
def test_merge_sort(self):

    # Arrange
    pandas = [Panda(2), Panda(1)]

    # Act + Assert
    with self.assertRaises(RecursionError):
        merge_sort(pandas)
```

Ошибка!

```
def test_merge_sort(self):

    # Arrange
    pandas = [Panda(2), Panda(1)]

    # Act
    actual = merge_sort(pandas)

    # Asset
    self.assertEqual(actual, "???")
```

Регрессионный тест

```
self.assertEqual(actual, "???")
```

Что здесь должно быть?

Assert

```
actual = get_age("Bob")
self.assertEqual(actual, 42)
```

Assert

```
actual = get_age("Bob")
self.assertEqual(actual, 42)


actual = split_names("Alex Bob Cate")
self.assertEquals(
    actual,
    ["Alex", "Bob", "Cate"]
)
```

Assert

```
actual = get_age("Bob")
self.assertEqual(actual, 42)
```

```
actual = split_names("Alex Bob Cate")
self.assertEquals(
    actual,
    ["Alex", "Bob", "Cate"]
)
```

```
actual = generate_graph(5)

graph = Graph(5)
graph.add_edge(1, 2)
graph.add_edge(3, 4)
graph.add_edge(1, 5)
graph.add_edge(4, 5)

self.assertEqual(actual, graph)
```

Фаззинг

Фаззинг

Данные

Тестируемая функция

Данные

Модификации

Код теста

Как преобразовать объект в код?

repr

```
In [1]: repr(1)
Out[1]: '1'

In [2]: repr("123")
Out[2]: "'123'"

In [3]: repr([1, 2, "123"])
Out[3]: "[1, 2 '123']"
```

```
In [1]: eval('1')
Out[1]: 1

In [2]: eval("'123'")
Out[2]: '123'

In [3]: eval("[1, 2 '123']")
Out[3]: [1, 2, '123']
```

Как преобразовать объект в код?

repr

```
In [4]: class Panda:
            def __init__(self, weight):
                self.weight = weight
        repr(Panda(1))
Out[4]: '<__main__.Panda object at 0x7fe59dbd6ed0>'
```

Как преобразовать объект в код?

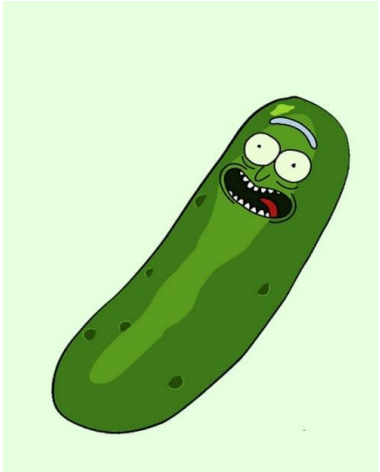
repr

```
In [4]: class Panda:
            def __init__(self, weight):
                self.weight = weight
        repr(Panda(1))
Out[4]: '<__main__.Panda object at 0x7fe59dbd6ed0>'


In [4]: eval('<__main__.Panda object at 0x7fe59dbd6ed0>')
Traceback (most recent call last):
  File <string>:1
    <__main__.Panda object at 0x7fe59dbd6ed0>
    ^
SyntaxError: invalid syntax
```

Как преобразовать объект в код?

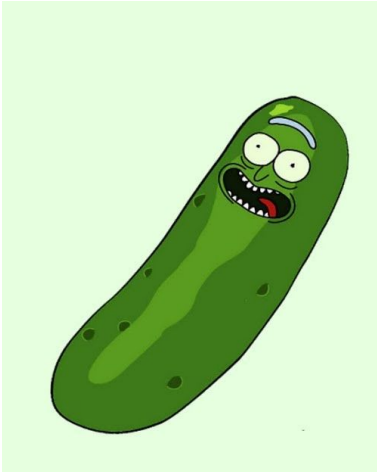
pickle

```
In [1]: pickle.dumps(1)
Out[1]: b'\x80\x04K\x01.'

In [2]: pickle.loads(b'\x80\x04K\x01.')
Out[2]: 1
```



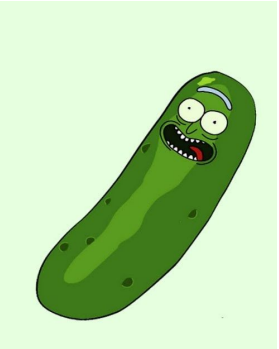
Как преобразовать объект в код?

pickle



```
In [1]: pickle.dumps(1)
Out[1]: b'\x80\x04K\x01.'

In [2]: pickle.loads(b'\x80\x04K\x01.')
Out[2]: 1

In [3]: pickle.dumps([1, 2, '123'])
Out[3]: b'\x80\x04\x95\x0f\x00\x00\x00\x00\x00\x00\x00]\x94(K\x01K\x02\x8c\x03123\x94e.'

In [4]: pickle.loads(
            b'\x80\x04\x95\x0f\x00\x00\x00\x00\x00\x00\x00]\x94(K\x01K\x02\x8c\x03123\x94e.'
        )
Out[4]: [1, 2, "123"]
```

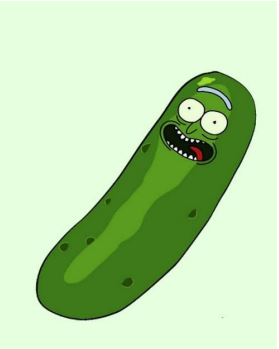
Как преобразовать объект в код?



pickle

```
In [5]: pickle.dumps(Panda(1))
Out[5]: b'\x80\x04\x95(\x00\x00\x00\x00\x00\x00\x00\x8c\x08__main__\
          \x94\x8c\x05Panda\x94\x93\x94)\x81\x94}\x94\x8c\x06weight\x94K\x01sb.'

In [6]: pickle.loads(
            b'\x80\x04\x95(\x00\x00\x00\x00\x00\x00\x00\x8c\x08__main__\
              \x94\x8c\x05Panda\x94\x93\x94)\x81\x94}\x94\x8c\x06weight\x94K\x01sb.'
        )
Out[6]: <__main__.Panda at 0x7f9222ac4a50>
```

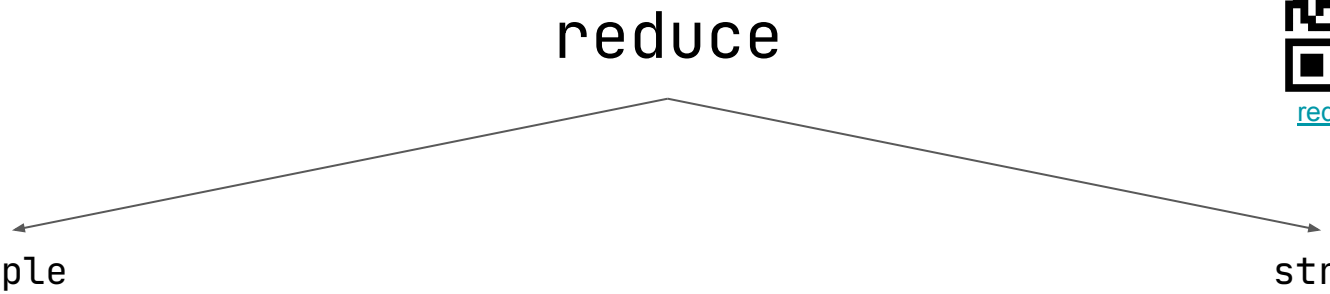
Как преобразовать объект в код?



pickle

```
b'\x80\x04K\x01.'

b'\x80\x04\x95\x0f\x00\x00\x00\x00\x00\x00\x00]\x94(K\x01K\x02\x8c\x03\
  123\x94e.'

b'\x80\x04\x95(\x00\x00\x00\x00\x00\x00\x00\x8c\x08__main__\
  \x94\x8c\x05Panda\x94\x93\x94)\x81\x94}\x94\x8c\x06weight\x94K\x01sb.'
```

Как преобразовать объект в код?



reduce документация

0. **«конструктор»**
1. **список аргументов для конструктора**
2. состояние (`__dict__` / `__setstate__`)
3. итератор элементов
4. итератор key-values
5. кастомный `__setstate__`

глобальная переменная

Как преобразовать объект в код?

0. **«конструктор»**
1. **список аргументов для конструктора**
2. состояние (`__dict__` / `__setstate__`)
3. итератор элементов
4. итератор key-values
5. кастомный `__setstate__`

reduce документация

```
In [1]: collections.Counter("12311242").__reduce__()
Out[1]:
(
    collections.Counter,
    ({'1': 3, '2': 3, '3': 1, '4': 1})
)
```

Как преобразовать объект в код?

0. **«конструктор»**
1. **список аргументов для конструктора**
2. состояние (`__dict__` / `__setstate__`)
3. итератор элементов
4. итератор key-values
5. кастомный `__setstate__`

reduce документация

```
In [1]: Panda(1).__reduce__()
Out[1]:
(
    <function copyreg._reconstructor(cls, base, state)>,
    (__main__.Panda, object, None),
    {'weight': 1}
)
```

Как преобразовать объект в код?

0. **«конструктор»**
1. **список аргументов для конструктора**
2. состояние (`__dict__` / `__setstate__`)
3. итератор элементов
4. итератор key-values
5. кастомный `__setstate__`

reduce документация

```
In [1]: Panda(1).__reduce__()
Out[1]:
(
    <function copyreg._reconstructor(cls, base, state)>,
    (__main__.Panda, object, None),
    {'weight': 1}
)
```

Как преобразовать объект в код?

0. **«конструктор»**
1. **список аргументов для конструктора**
2. состояние (`__dict__` / `__setstate__`)
3. итератор элементов
4. итератор key-values
5. кастомный `__setstate__`

reduce документация

```
In [1]: Panda(1).__reduce__()
Out[1]:
(
    <function copyreg._reconstructor(cls, base, state)>,
    (__main__.Panda, object, None),
    {'weight': 1}
)
```

```
(
    Panda.__new__,
    (__main__.Panda),
    {'weight': 1}
)
```

Как преобразовать объект в код?

0. **«конструктор»**
1. **список аргументов для конструктора**
2. состояние (`__dict__` / `__setstate__`)
3. итератор элементов
4. итератор key-values
5. кастомный `__setstate__`

reduce документация

```
In [1]: queue = collections.deque([1, 2, 3])
        queue.__reduce__()
Out[1]: (
    collections.deque,
    (),
    None,
    <_collections._deque_iterator at 0x7f92229ecef0>
 )

In [2]: list(deque.__reduce__()[3])
Out[2]: [1, 2, 3]
```

Как преобразовать объект в код?

0. **«конструктор»**
1. **список аргументов для конструктора**
2. состояние (`__dict__` / `__setstate__`)
3. итератор элементов
4. итератор key-values
5. кастомный `__setstate__`

```
In [1]: d = collections.defaultdict(int)
        d[1] += 2
        d.__reduce__()
Out[1]: (
    collections.defaultdict,
    (int,),
    None,
    None,
    <dict_itemiterator at 0x7f9222045490>
)
```

Как преобразовать объект в код?

0. **«конструктор»**
1. **список аргументов для конструктора**
2. состояние (`__dict__` / `__setstate__`)
3. итератор элементов
4. итератор key-values
5. кастомный `__setstate__`

```
In [1]: d = collections.defaultdict(int)
        d[1] += 2
        d.__reduce__()
Out[1]: (
    collections.defaultdict,
    (int,),
    None,
    None,
    <dict_itemiterator at 0x7f9222045490>
)

In [2]: list(d.__reduce__()[4])
Out[2]: [(1, 2)]
```

UnitTestBot

www.utbot.org/

github.com/UnitTestBot

gitlab.com/tamarinvs19/utfuzz

```python
def _to_utc_date_string(ts: date | datetime) -> str:
    """
    Coerce datetimes to UTC
    (assume localtime if nothing is given)
    """
    if isinstance(ts, datetime):
        try:
            ts = ts.astimezone(utc_tz)
        except:
            import tzlocal

            ts = ts.replace(tzinfo=tzlocal.get_localzone())
            ts = ts.astimezone(utc_tz)

    return ts.strftime("%Y%m%dT%H%M%SZ")
```

caldav репозиторий

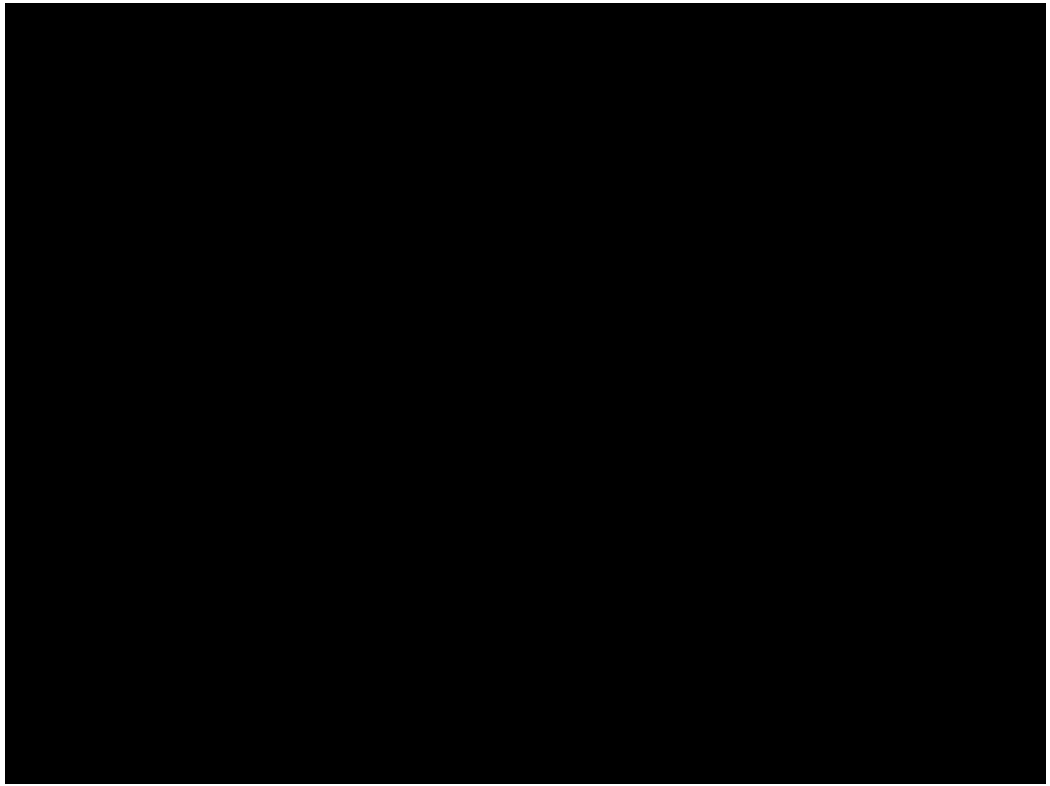
```python
def _to_utc_date_string(ts: date | datetime) -> str:
    """
    Coerce datetimes to UTC
    (assume localtime if nothing is given)
    """
    if isinstance(ts, datetime):
        try:
            ts = ts.astimezone(utc_tz)
        except:
            import tzlocal

            ts = ts.replace(tzinfo=tzlocal.get_localzone())
            ts = ts.astimezone(utc_tz)

    return ts.strftime("%Y%m%dT%H%M%SZ")
```

caldav репозиторий

```python
class TestTopLevelFunctions(unittest.TestCase):
    def test__to_utc_date_string(self):
        """
        ts = datetime.datetime(10, 10, 10, 10, 10, 10, 10)
        """
        actual = _to_utc_date_string(datetime.datetime(10, 10, 10, 10, 10, 10, 10))
        self.assertEqual('101010T073953Z', actual)

    def test__to_utc_date_string1(self):
        """
        ts = datetime.date(1, 1, 1)
        """
        actual = _to_utc_date_string(datetime.date(1, 1, 1))
        self.assertEqual('10101T000000Z', actual)

    def test__to_utc_date_string_with_exception(self):
        """
        ts = datetime.datetime(1, 1, 1, fold=1)
        """
        with self.assertRaises(OverflowError):
            _to_utc_date_string(datetime.datetime(1, 1, 1))
```

PR в caldav

Демо: PyCharm плагин



Вячеслав Тамарин
Екатерина Точилина

github.com/UnitTestBot