Code From Nothing

Procedural Generation of Python Source Code

Kirill Borisov

EUROPYTHON
2021 Jul 26-Aug 1 Online

Greetings!

- I’m Kirill Borisov
- 15+ years of programming experience
- Creator of **pybetter** & **BlackConnect**
- In love with everything “*code*”

About this talk

- We will talk about how code is written
- Cover a little bit of parsing
- Introduce **Hypothesmith**
- Dive deeper into how it works

```
print(“Hello, world!”)
```

Code… What is it, really?

- Code is our bread & butter
- Code is usually written “by hand"
- But who checks it?
- Whole cottage industry of “linters"

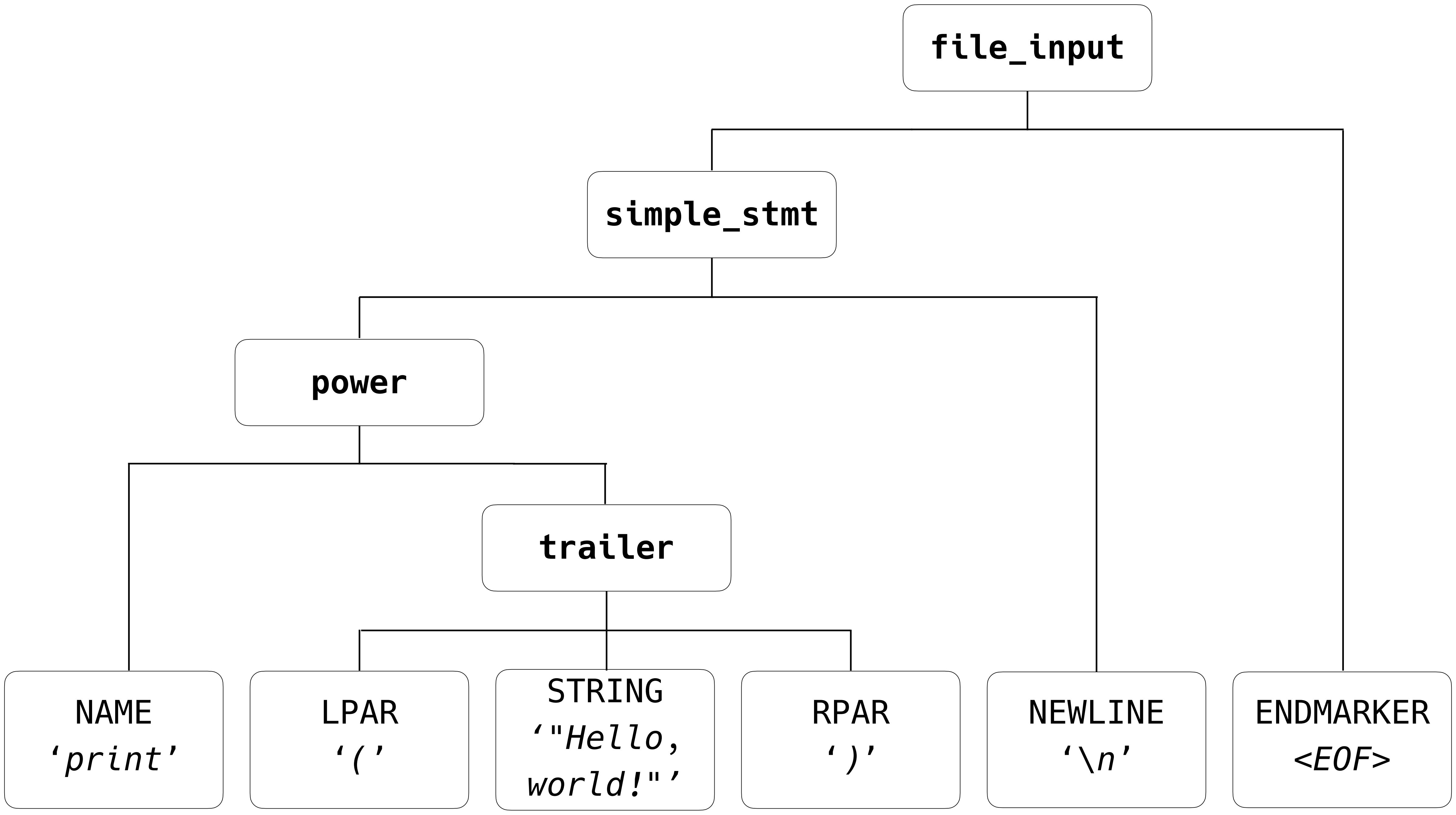
```
1: [    NAME     ] |   0 - 5    | print("Hello, world!");
1: [     OP      ] |   5 - 6    | print("Hello, world!");
1: [   STRING    ] |   6 - 21   | print("Hello, world!");
1: [     OP      ] |  21 - 22   | print("Hello, world!");
1: [     OP      ] |  22 - 23   | print("Hello, world!");
1: [   NEWLINE   ] |  23 - 24   |
2: [ ENDMARKER   ] |   0 - 0    | <EOF>
```

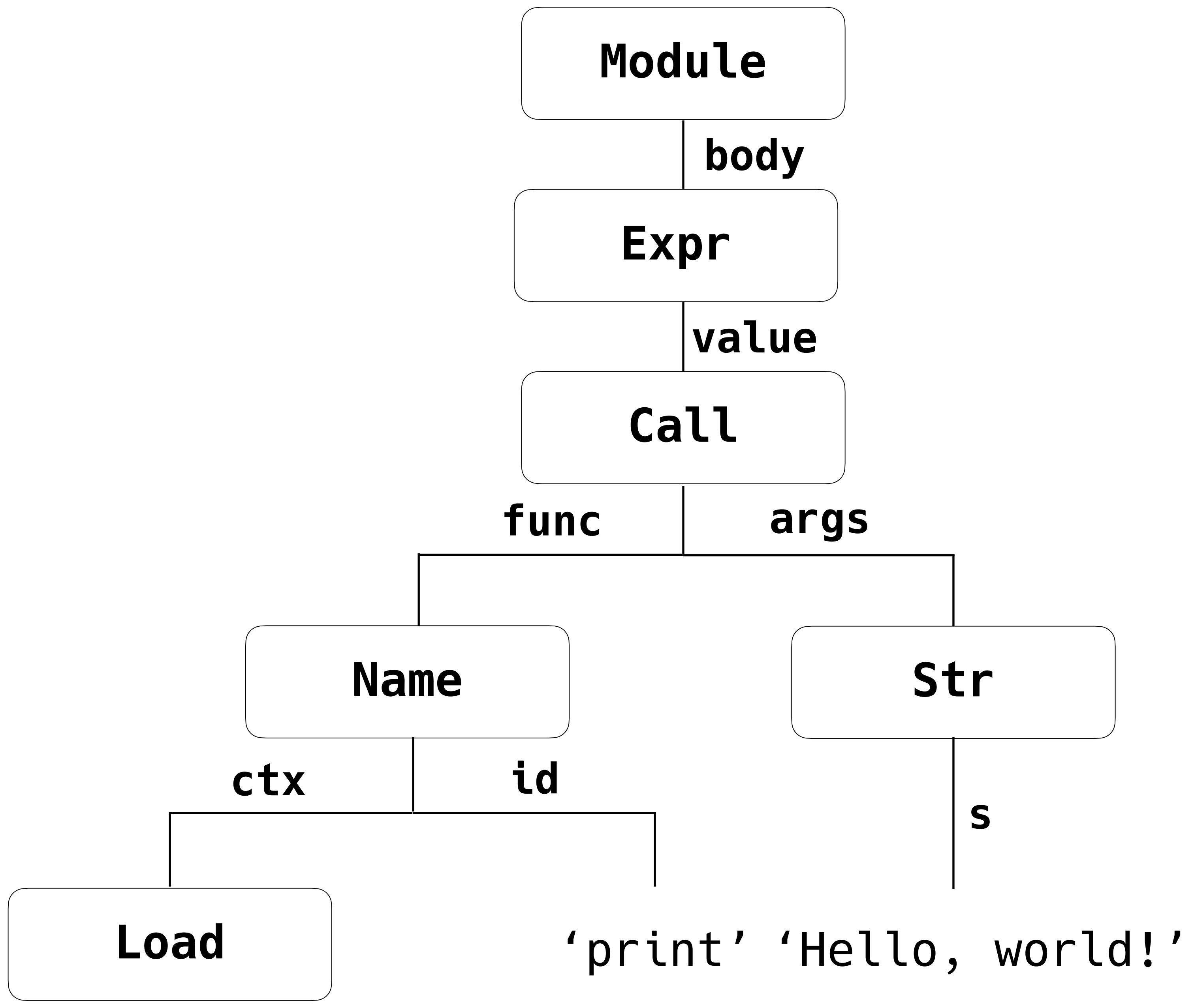
```
file_input:  (NEWLINE | stmt)* ENDMARKER

stmt:         simple_stmt | compound_stmt

simple_stmt: small_stmt (';' small_stmt)* [';']
NEWLINE

small_stmt:  (expr_stmt | print_stmt  | del_stmt |
              pass_stmt | flow_stmt |
              import_stmt | global_stmt |
              exec_stmt | assert_stmt)
```


file_input
simple_stmt
power
trailer
NAME 'print'
LPAR '('
STRING '"Hello, world!"'
RPAR ')'
NEWLINE '\n'
ENDMARKER <EOF>


Module
body
Expr
value
Call
func
args
Name
Str
ctx
id
s
Load
‘print’
‘Hello, world!’

Linters and autoformatters

- They "read" your code
- Code style, security checks, complexity…
- Some can also modify it!
- **pep8**, **pyflakes**, **black**...

Checking the Checkers

- Use hand-crafted examples?
- You need a lot of permutations
- Limited by your imagination
- Real world will surprise you!

Random acts of code

- Random set of characters as input
- Take one which compiles!
- "Infinite monkey theorem", anyone?
- Highly impractical in terms of time

…In computing, **procedural generation** is a method of creating data algorithmically as opposed to manually, typically through a combination of human-generated assets and algorithms coupled with computer-generated randomness and processing power….

(Source: Wikipedia)

https://en.wikipedia.org/wiki/Procedural_generation

Structure is the king

- *Rules* on how to arrange things
- *Patterns* for generatng things
- Need to cover whole of the language
- Sounds like a *grammar*, isn’t it?

Grammar as a template

- It can be represented as a tree
- Rules (*non-terminals*) as nodes
- Text (*terminals*) as leaves
- Just do random walk through the tree

Enter *Hypothesis*

https://hypothesis.works/

- *Property-based testing*
- Generates wide range of input data
- Based on *QuickCheck paper*
- Can do “hill-climbing search”

```python
def valid_branch_names():
    return st.text(
        alphabet=letters, min_size=1, max_size=112).map(lambda t: t.lower()) | st.just("master")
```

```python
from hypothesis import assume

@given(branch_name=valid_branch_names())
def test_checkout_new_branch(self, branch_name):
    assume(branch_name != "master")
    tmpdir = FilePath(self.mktemp())
    tmpdir.makedirs()
    repo = Repository.initialize(tmpdir.path)
    repo.checkout(branch_name, create=True)
    self.assertEqual(branch_name, repo.get_active_branch())
```

Source: https://hypothesis.works/articles/incremental-property-based-testing/

LarkStrategy

https://github.com/lark-parser/lark

- **Lark** is a parsing toolkit for Python
- Parses language grammar into a tree
- Select subsets of nodes on each step
- Generates *terminals* from regexes

Python is quirky

- *Indentaton* to mark blocks of code
- Identifiers must be *UTF-8 encodable*
- Lot of AST post-processing
- New PEG parser in Python 3.10

Enter *Hypothesmith*

https://github.com/Zac-HD/hypothesmith

- Inspired by **CSmith**
- Strategy for generating Python code
- Works around mentioned quirks
- Has support for per-node generation

```python
import hypothesmith
from hypothesis import given, settings, HealthCheck

settings.register_profile(
    "slow_example_generation",
    suppress_health_check=HealthCheck.all(),
    deadline=None,
)
settings.load_profile("slow_example_generation")


@given(generated_source=hypothesmith.from_grammar())
@settings(max_examples=1000)
def test_no_crashes_on_valid_input(generated_source):
    print(generated_source.encode("utf-8"))
    print("---------")
```

```
\n
\n\n
#0
\nA\n
A\n\n
A\n
ifA:A\n
withA:A\n\n
```

```
classA:A\n
\n\nA\n
\n\n\n
forAinA:A\n
ifA:A\n\nA\n
\r\n
\nA\n\n\n
#0A\n
```

```
from……import*;global\xc2\xba;nonlocal\xc5\x9df\xf0
\xaa\x9e\xb4\xc3\x91\xf0\xa8\xa4\x81Z;\r\n
```

```
import\xc5\x90\xc5\x8a\xf0\x97\x8c\x85.\xc3
\x8fBc\xc3\x83\xec\x99\xa7L4\xc5\x95\xc5\xb
3\xf0\xac\xb1\x80\xf0\xac\xbb\xa6\xf0\x90\x
a6\xb2\xc3\x90q\xf0\xa4\x9f\xa2\xc3\x86a\xf
0\xa4\xae\x9f\xc3\xba1as\xc3\x92\xc3\x81\xc
3\x9c\xc3\x95o\xc4\x9f\xc5\x8c\xc5\x87\xc4\
x87;nonlocal\xc5\x8b\xf0\xa2\x8d\xa6\xc4\xb
1\xc5\xba\xc4\x96\xf0\xa5\xba\x96\xc4\xa8\x
c3\xab\xe3\xa7\x90,\xc4\xa0\xc4\xb4\xc4\x8c
\xc5\x95,\xc5\xa8\xc5\x84\xc3\xa9\xc4\xba\x
f0\xa1\xb0\xa6\xc2\xb7\xc5\x9f\xc3\x9d\xe3\
xb5\xb9\xe5\xb7\xa6\xc4\xac\xc5\xa9\xc4\x84
,\xc5\x90\xc5\x8a\xf0\x97\x8c\x85,\xc5\xad\
xc5\xb1\xf0\xab\x9c\x8d\xc3\xad\xc5\xac,
\xe6\x95\x9a\n \t\t\t   \t
\t\r\n#\xc2\x93\xc2\x81\r\r\n\t#\xc2\xa9
```

Targeted search

- Use metrics to find better examples
- Targets:
  - Bytecode instructions
  - Total number of AST nodes
  - Number of *unique* AST node types
- Longer and more complicated code

Bugs found with Hypothesmith

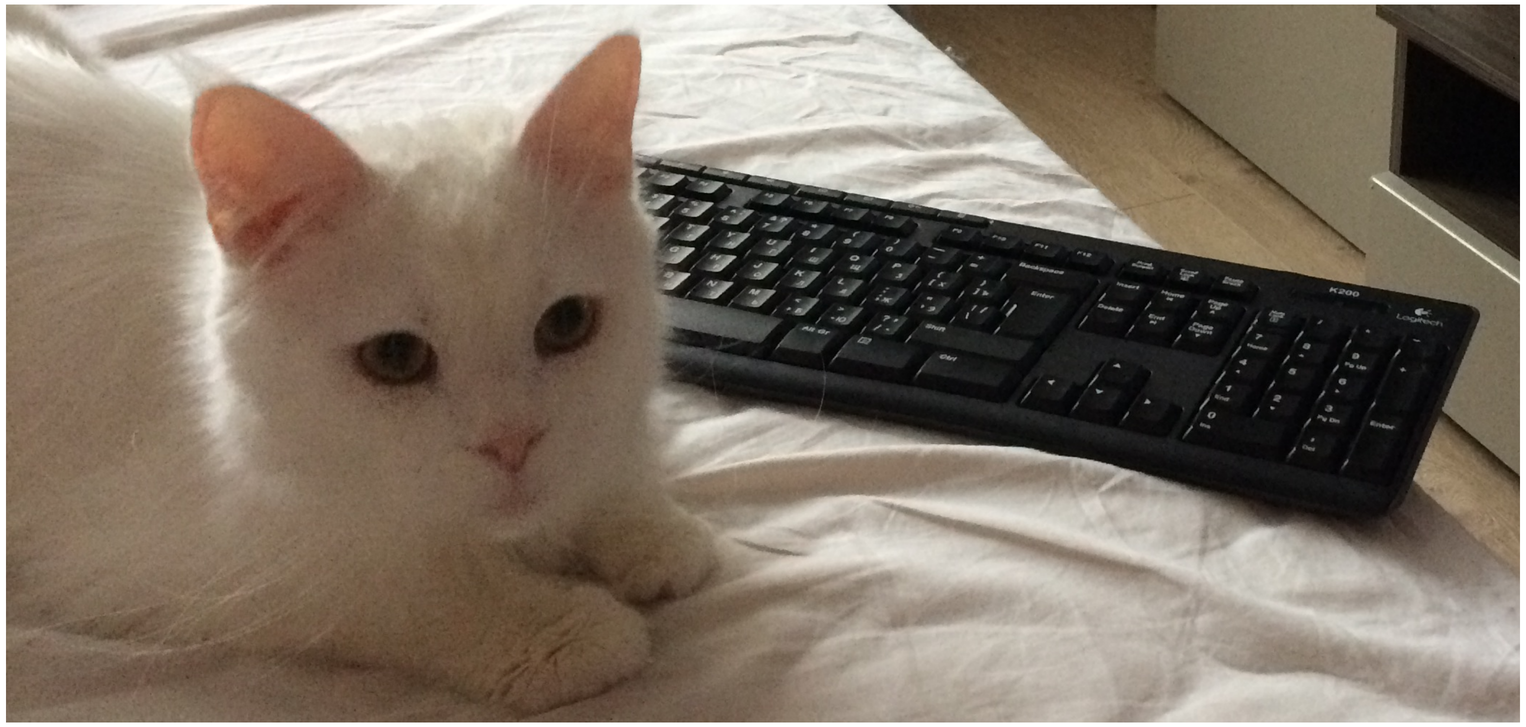
- BPO-40661 - Python parser segfault
- BPO-38953 - *tokenize* bug
- **lib2to3** errors on `\r` **in** comment
- ***black*** fails on files ending in `\`
- Round-trip bugs in **LibCST**

Caveats

- Most generated code is *gibberish*
- It can only serve as a *smoke test*
- No support for AST postprocessing
- Can be quite slow

Further reading

- How Hypothesis Works
- Finding and Understanding Bugs in C Compilers
- QuickCheck – A Lightweight Tool for Random Testing of Haskell Programs
- Compilers: Principles, Techniques and Tools



Questions?

- Thank you!

lensvol

lensvol@gmail.com