BUILDING AN INTERPRETER IN RPYTHON

Juozas Kaziukėnas


hello
my name is
@JUOKAZ

WHAT IS AN INTERPRETER

INTERPRETER IS

- Source code parser
- Bytecode interpretation loop
- Standard library

4 STEPS

- Lexing - turn a string into a list of tokens
- Parsing - turn a list of tokens into an Abstract Source Tree (AST)
- Generate bytecode
- Interpreting - run eval() in a loop

BYTECODE

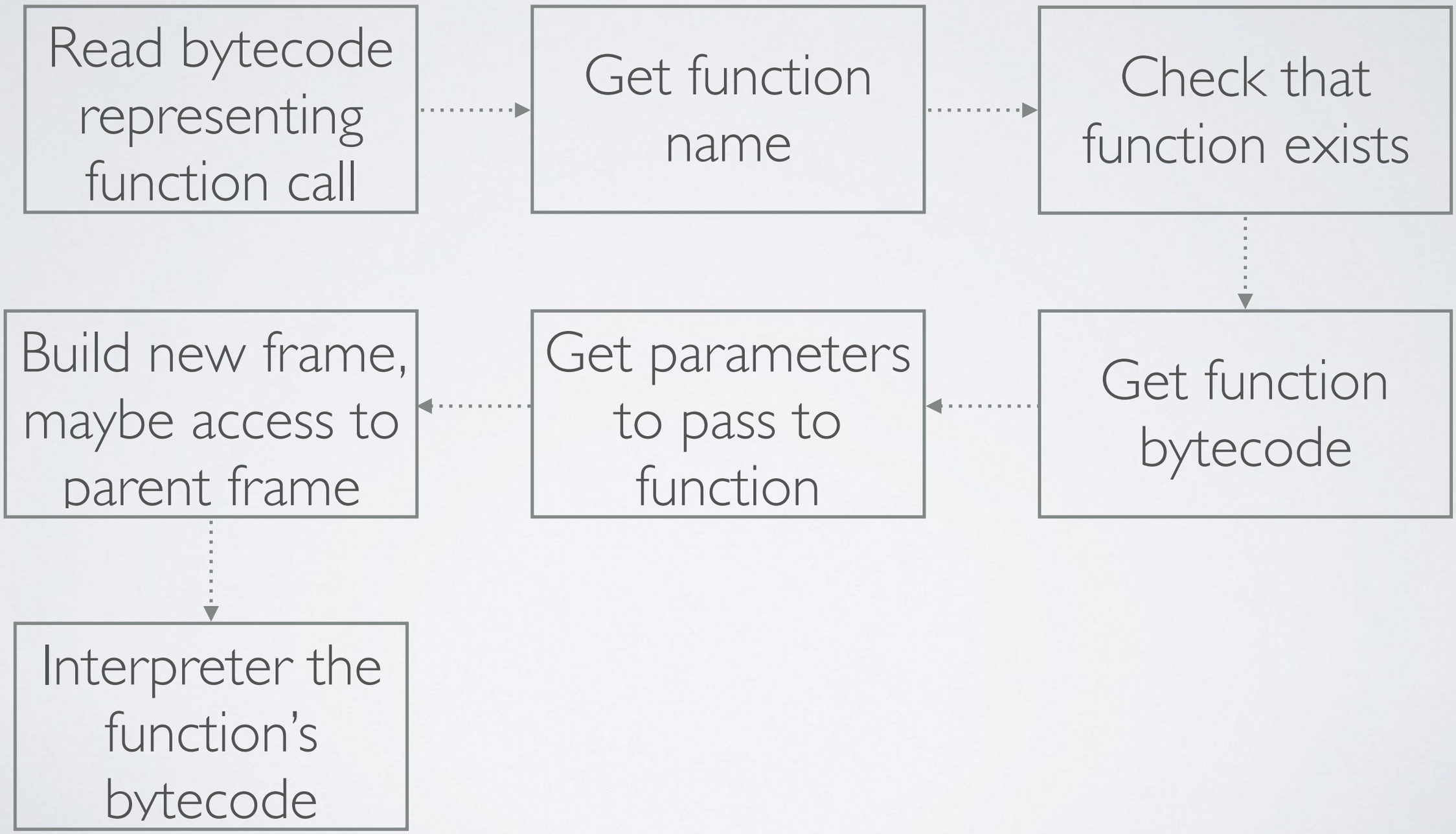
```
def foo():
        a = 2
        b = 3
        return a + b
```

Output bytecode

```
import dis
dis.dis(foo)
```

```
2           0 LOAD_CONST               1 (2)
            3 STORE_FAST               0 (a)
3           6 LOAD_CONST               2 (3)
            9 STORE_FAST               1 (b)
4          12 LOAD_FAST                0 (a)
           15 LOAD_FAST                1 (b)
           18 BINARY_ADD
           19 RETURN_VALUE
```

FUNCTION CALL



JIT

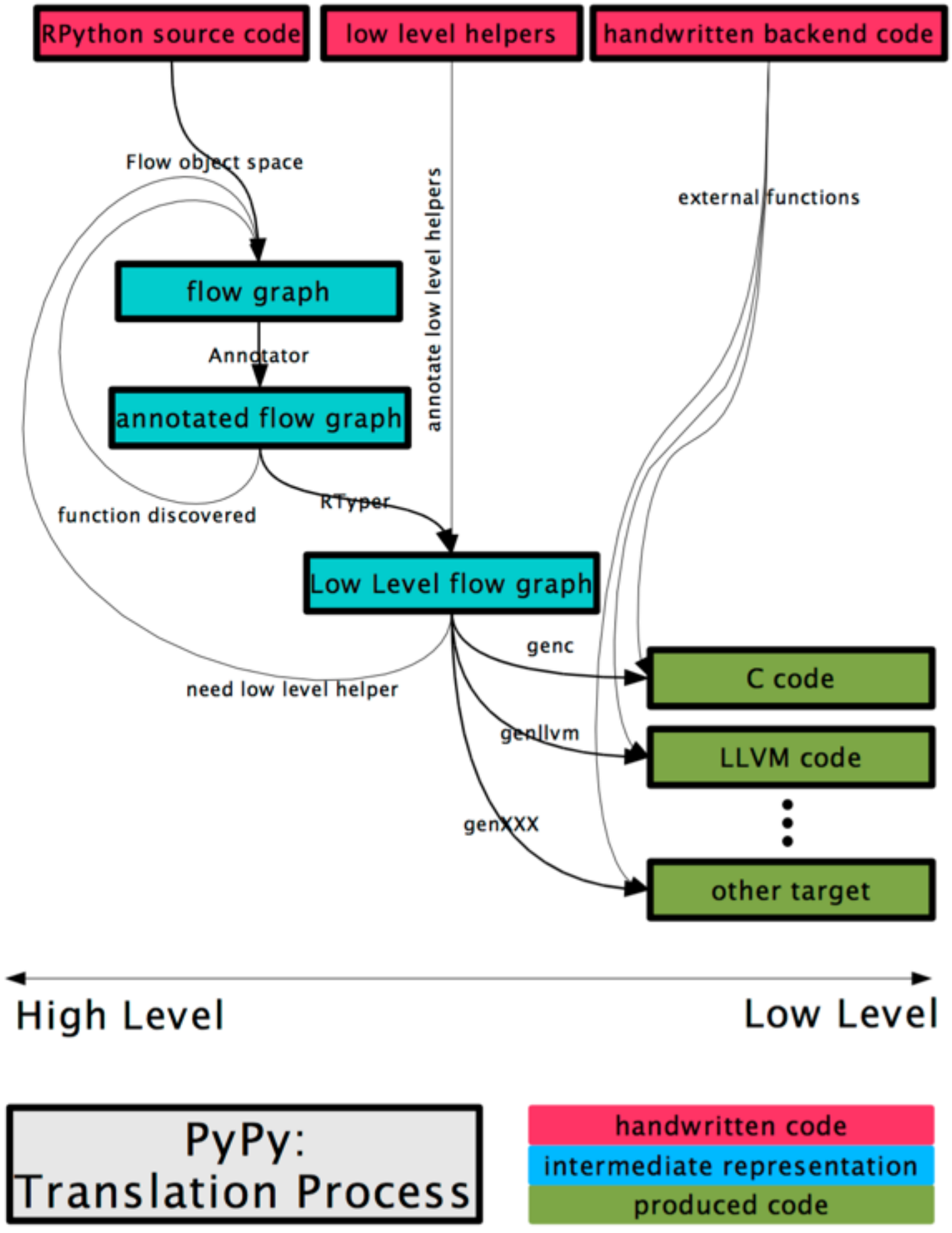
- Most modern interpreters have JIT
- Track runtime, look for optimizations
- On-demand machine code generation
- Most complicated part of the toolchain

WHAT IS RPYTHON

PYPY IS WRITTEN IN RPYTHON

RPYTHON

- Subset of Python
- `rlib` set of libraries
- Ideal for writing interpreters
- JIT & GC for free
- Gets translated to C and compiled



PyPy: Translation Process

handwritten code
intermediate representation
produced code

WHEN COMPILED, IT CAN BE AS FAST OR FASTER AS WRITING A PROGRAM IN C

CAN BE EXECUTED/TESTED LIKE ANY OTHER PYTHON PROGRAM

TYPE SYSTEM

```
def entry_point(argv):
    x = 123 # ok
    x = '456' # error!
```

TYPE SYSTEM

```
def entry_point(argv):
    if len(argv) == 1:
        x = None
    else:
        x = 0
    print x+1+2 # error!
    return 0
```

INHERITANCE

```
class Parent(object):
    pass

class ChildA(Parent):
    attr_only_on_this_child = 12

class ChildB(Parent):
    pass

def method(myinstance):
    ssert isinstance(myinstance, ChildA) # required
    print(child.attr_only_on_this_child)

method(ChildA())
```

JIT - IMMUTABLE FIELDS

```
class SomeClass(object):
    _immutable_fields_ = ['bytecode', 'args[*]']

    def __init__(self, bytecode, args):
        self.bytecode = bytecode
        self.args = args[:]
```

JIT - ELIDABLE

```
class Cell:
    def __init__(self, slot):
        self.slot = slot

@jit.elidable
def lookup(name):
    return namespace[name]

cell = lookup(name)
return cell.slot
```

WORKING WITH RPYTHON

1. Write valid Python

2. Modify it until it's valid RPython too

HOW I BUILT AN INTERPRETER

PyHP

https://github.com/juokaz/pyhp

PYHP

- Implements basic PHP functionality
- Includes debug tools and a basic HTTP server
- Suite of tests to check all functionality
- Thanks to JIT - very fast
- Built by modifying the sample interpreter for PHP

EBNF GRAMMAR

```
VARIABLENAME: “\$[a-zA-Z_][a-zA-Z0-9_]*";
variable: <VARIABLENAME>;

expression   : <variable>
             | <literal>
             ;

assignmentexpression : expression >assignmentoperator< assignmentexpression
                     | <expression>
                     ;

assignmentoperator  : "=" | "\*=" | "\/=" | "\%=" | "\+=" | "\-=" | "<<="
                    | ">>=" | ">>>=" | "&=" | "^=" | "\|=" | ".="
                    ;

ifstatement : ["if"] ["("] comparisonexpression [")"] statement  ["else"] statement
            | ["if"] ["("] comparisonexpression [")"] statement
            ;

statement : <block>
          | <assignmentexpression> [";"]
          | <ifstatement>
          | <returnstatement> [“;"]
          ;
```

DATATYPES

```
class W_Boolean(W_Root):
    _immutable_fields_ = ['boolval']

    def __init__(self, boolval):
        self.boolval = boolval

    def str(self):
        if self.boolval is True:
            return u"true"
        return u"false"

    def __deepcopy__(self):
        obj = instantiate(self.__class__)
        obj.boolval = self.boolval
        return obj

    def is_true(self):
        return self.boolval
```

TESTS

```
def test_running(self):
    out = self.run("""$x = 1;
    print $x;""")
    assert out == "1"

def test_if_and(self):
    out = self.run("""
    $x = 1;
    $y = 2;
    if ($x >= 1 && $y < 2) {
        print $x;
    } else {
        print $y;
    }""")
    assert out == "2"

def test_discards_assignment(self):
    """ if stack is not consumed
        this will overflow"""
    program = "$i = 1;"
    for i in range(1, 20):
        program += "$i = 2;"
    self.run(program)
```

```
def test_function_call_pass_by_value(self):
    out = self.run("""function test($a) {
        $a = 3;
    }

    $i = 5;
    test($i);

    print $i;
    """)
    assert out == "5"

def test_function_call_pass_by_reference(self):
    out = self.run("""function test(&$a) {
        $a = 3;
    }

    $i = 5;
    test($i);

    print $i;
    """)
    assert out == "3"
```

RUN PYHP

```
docker pull juokaz/pyhp   # container with RPython setup
make build                # build PyHP into an executable
make bench                # run the bench.php
```

PHP 7

```
simple              0.068
simplecall          0.010
simpleucall         0.042
simpleudcall        0.042
mandel              0.205
mandel2             0.189
ackermann(7)        0.056
ary(50000)          0.004
ary2(50000)         0.004
ary3(2000)          0.082
fibo(30)            0.155
hash1(50000)        0.018
hash2(500)          0.012
heapsort(20000)     0.041
matrix(20)          0.046
nestedloop(12)      0.110
sieve(30)           0.024
strcat(200000)      0.006
------------------------
Total               1.113
```

PyHP

```
simple              0.034
simplecall          0.002
simpleucall         0.002
simpleudcall        0.002
mandel              0.017
mandel2             0.019
ackermann(7)        0.045
ary(50000)          0.006
ary2(50000)         0.014
ary3(2000)          0.018
fibo(30)            0.054
hash1(50000)        0.026
hash2(500)          0.016
heapsort(20000)     0.030
matrix(20)          0.012
nestedloop(12)      0.016
sieve(30)           0.010
strcat(200000)      0.009
------------------------
Total               0.333
```

MAKING PHP UNICODE

Use unicode strings for all function names, variable names and string…

Browse files

… values.

master

juokaz committed on May 14, 2015 — 1 parent 46a037e — commit 2566f21c2237400ba9d8a7c6173ff6cc818031f5

Showing **13 changed files** with **286 additions** and **224 deletions**.

Unified | Split

BIGGEST ISSUES

- Lack of documentation
- Googling for errors yields no results
- Late-stage translation errors take a long time to debug

LESSONS LEARNED

- Re-implementing std library takes a lot of time
- Implementing language features requires knowing every edge case (PHP has a spec now though)
- Some PHP-specific features are a nightmare to figure out
- Function calls are expensive

WHERE TO START

BASICS

```
def entry_point(argv):
    # this is your program's main function
    return 0

def target(driver, args):
    # this is run at compile time
    return entry_point, None
```

ENTRY POINT

```
def entry_point(argv):
    filename = argv[0]
    try:
        source = read_file(filename)
    except OSError:
        print 'File not found %s' % filename
        return 1

    ast = source_to_ast(source)

    bc = compile_ast(ast, ast.scope, filename)

    intrepreter = Interpreter()
    intrepreter.run(bc)

    return 0
```

PARSER

```
from rpython.rlib.parsing.ebnfparse import parse_ebnf, make_parse_function

grammar_file = 'grammar.txt'
grammar = py.path.local(dir).join(grammar_file).read("rt")

regexs, rules, ToAST = parse_ebnf(grammar)

_parse = make_parse_function(regexs, rules, eof=True)

def parse(code):
    t = _parse(code)
    return ToAST().transform(t)
```

AST

```
class Transformer(RPythonVisitor):
    def visit_ifstatement(self, node):
        condition = self.dispatch(node.children[0])
        ifblock = self.dispatch(node.children[1])
        if len(node.children) > 2:
            elseblock = self.dispatch(node.children[2])
        else:
            elseblock = None
        return operations.If(condition, ifblock, elseblock)

def source_to_ast(source):
    ast = parse(source)
    transformer = Transformer()
    return transformer.dispatch(ast)
```

BYTECODE

```
class Print(Node):
    def __init__(self, expr):
        self.expr = expr

    def compile(self, ctx):
        self.expr.compile(ctx)
        ctx.emit('PRINT')

    def str(self):
        return u'Print (%s)' % self.expr.str()

def compile_ast(ast, scope, name):
    bc = ByteCode(name, scope.symbols)
    ast.compile(bc)
    return bc
```

INTERPRETER

```
class Interpreter(object):
    def run(self, bytecode):
        frame = Frame(self, bytecode)
        if bytecode._opcode_count() == 0:
            return None

        pc = 0
        while True:
            if pc >= bytecode._opcode_count():
                return None

            opcode = bytecode._get_opcode(pc)

            if isinstance(opcode, RETURN):
                return frame.pop()

            opcode.eval(self, frame)

            if isinstance(opcode, BaseJump):
                new_pc = opcode.do_jump(frame, pc)
                pc = new_pc
                continue
            else:
                pc += 1
```

ADDING JIT

```
driver = jit.JitDriver(reds=[‘frame’],
                       greens=['pc', 'bytecode'],
                       virtualizables=['frame'])

class Interpreter(object):
    def run(self, bytecode):
        frame = Frame()

        pc = 0
        while True:
            driver.jit_merge_point(pc=pc, bytecode=bytecode, frame=frame)

            opcode = bytecode._get_opcode(pc)

            opcode.eval(self, frame)

            if isinstance(opcode, BaseJump):
                new_pc = opcode.do_jump(frame, pc)
                if new_pc < pc:
                    driver.can_enter_jit(pc=new_pc, bytecode=bytecode, frame=frame)
                pc = new_pc
                continue
            else:
                pc += 1
```

COMPILE

Compile with no optimizations

```
python rpython/bin/rpython -O0 pyhp.py
```

Run the interpreter

```
./pyhp-c example.php
```

COMPILE WITH JIT

Compile with JIT support

```
python rpython/bin/rpython --opt=jit pyhp.py
```

DEBUG JIT

Generate debug trace file

```
PYPYLOG=jit-log-opt:jit.txt ./pyhp bench.php
```

Plot the trace as a graph

```
python rpython/tool/logparser.py \
draw-time jit.txt --mainwidth=8000 filename.png
```

RESOURCES

- PyPy blog http://morepypy.blogspot.com
- RPython docs http://rpython.readthedocs.io/en/latest/index.html
- Ruby interpreter https://github.com/topazproject/topaz
- PyHP interpreter http://github.com/juokaz/pyhp

One more thing...

PYHP.JS

(PYTHON + JAVASCRIPT) + PHP

https://github.com/juokaz/pyhp.js

QUESTIONS?

THANKS!

Juozas Kaziukėnas
@juokaz