Zurich Python User Group

funcparserlib

or how to write your own programming language...

Aaron Richiger
a.richi@bluewin.ch

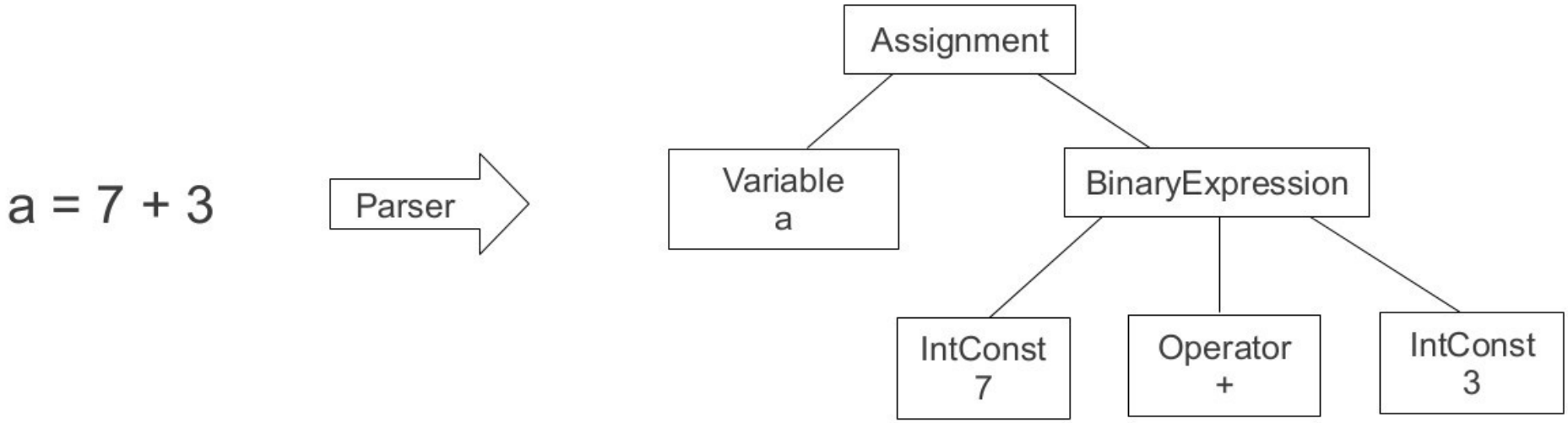
2. Mai 2013

Goals

- What is a parser?
- How to use funcparserlib?
- What's the visitor pattern useful for?
- What is an interpreter?
- How to write a parser and an interpreter for your own programming language?

What does a parser do?

- Input: Text
- Output: AST (Abstract Syntax Tree)



Parsing with python

- pyparsing
- ANTLR
- Many other tools exist

  Good overview: http://nedbatchelder.com/text/python-parsers.html

- funcparserlib

funcparserlib

- Pure python library, 500 LOC
- Author: Andrey Vlasovskikh
- Inspired by Haskell
- LL(*) parser
- Still under development

Performance

File Size	cjson	simplejson	funcparserlib	json-ply	pyparsing
6 KB	0 ms	45 ms	228 ms	n/a	802 ms
11 KB	0 ms	80 ms	395 ms	367 ms	1355 ms
100 KB	4 ms	148 ms	855 ms	1071 ms	2611 ms
134 KB	11 ms	957 ms	4775 ms	n/a	16534 ms
1009 KB	87 ms	6904 ms	36826 ms	n/a	116510 ms
User Code	0.9 KLOC	0.8 KLOC	0.1 KLOC	0.5 KLOC	0.1 KLOC
Library Code	0 KLOC	0 KLOC	0.5 KLOC	5.3 KLOC	3.7 KLOC

Fixing funcparserlib

Execute fix.py as admin

What it does

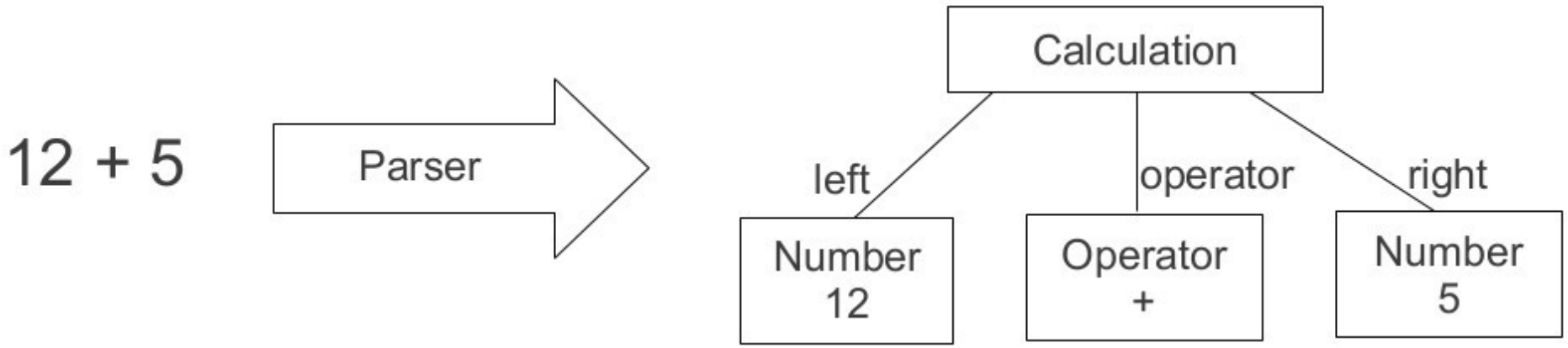
Replaces line 273 of parser.py:

Old: return (res, e.state)

New: return res, State(s.pos, e.state.max)

Example 1: Parsing Calculations

- Input: Simple additions
- Output: AST



Example 1: Parsing Calculations

- Parsers you will need:

+	Concatenates two parsers	calc = nr + operator + nr
|	One of both has to match	digit = one | two | ...
>>	Send match to a function	operator = op('+') >> Operator

- Files:
  - Parser: calc_parser.py — to modify
  - AST: calc_ast.py — read only
  - Helpers: funcparserlib_helpers.py — read only
  - Tests: calc_test.py — read only

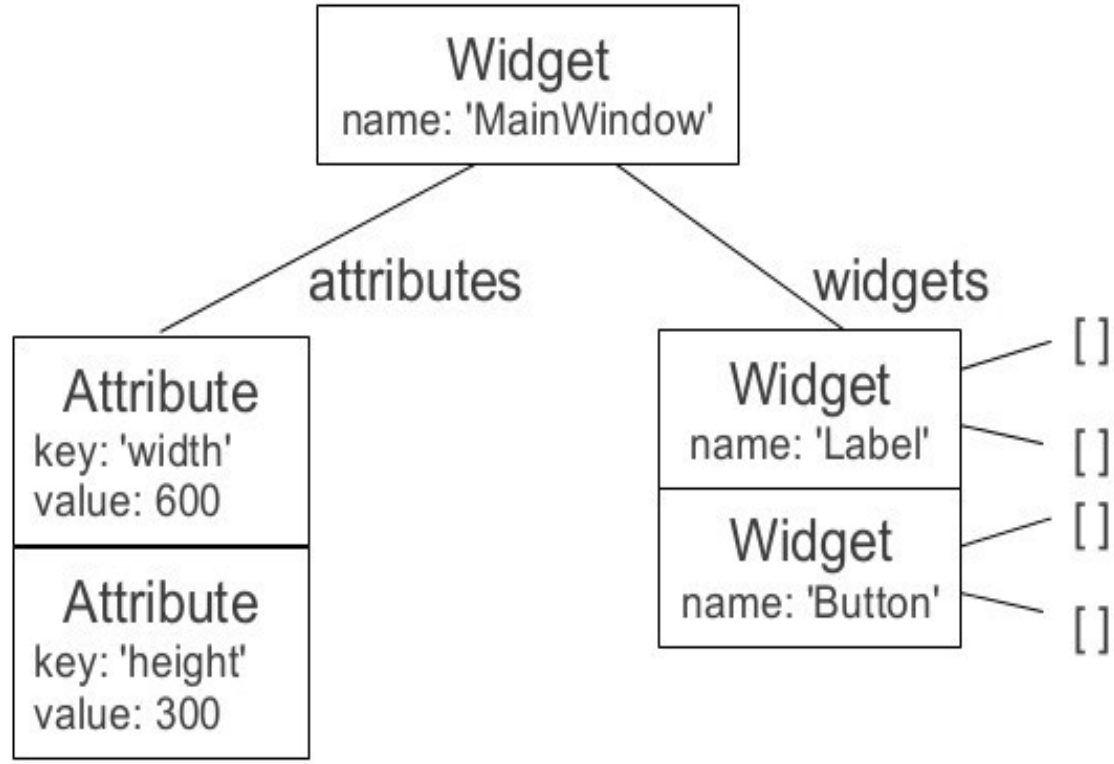
- Your task: Add support for subtractions: "12 – 5"

Example 2: pyml Parser

- Input: UI modeling language: pyml
- Output: AST

```
MainWindow {
    width = 600;
    height = 300;

    Label {}
    Button {}
}
```

Parser



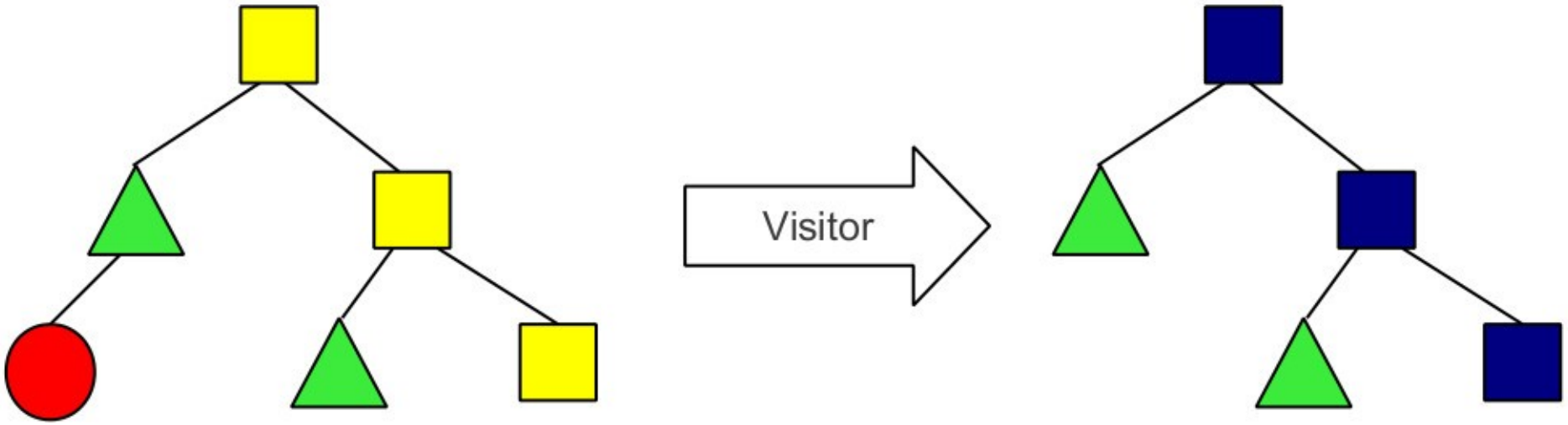
Example 2: pyml Parser

- Parsers you will need:
  - many() Match zero or more times nr = many(digit)
  - forward_decl() For recursive parsers
- Files:
  - Parser: pyml_parser.py to modify
  - AST: pyml_ast.py to modify
  - Input input.pyml read only
  - Helpers: funcparserlib_helpers.py read only
  - Tests: pyml_test.py read only
- Your task:
  - Add support for calculation and list of strings attribute values:

```
height = 1 + 2;
colors = ["red", "green"];
```

Visitor Pattern

Transformation of trees



Visitor Pattern: Python Implementation

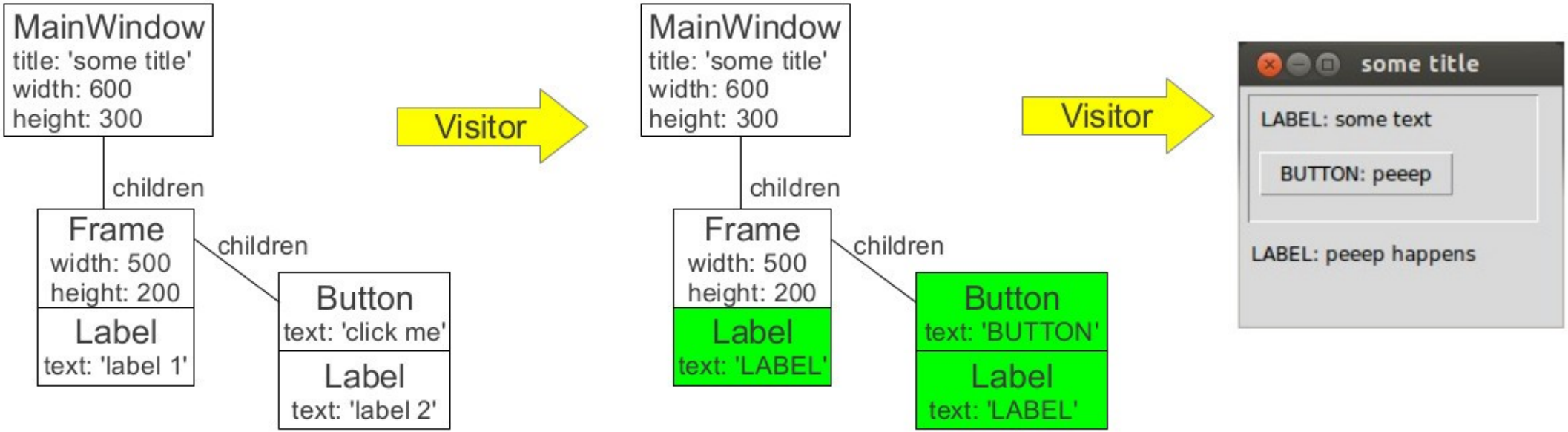
```
class BaseVisitor(object):
    def visit(self, tree):
        if isinstance(tree, Calculation):
            self.visit_calculation(tree)
        elif isinstance(tree, Number):
            self.visit_number(tree)

    def visit_calculation(self, calculation):
        self.visit(calculation.left)
        self.visit(calculation.right)

    def visit_number(self, number):
        pass
```

```
class IncrementVisitor(BaseVisitor):
    def visit_number(self, number):
        number.value += 1
```

```
ast = parser.parse("1 + 2")
incrementor = IncrementVisitor()
incrementor.visit(ast)
```

Example 3: pyml Visitor

- Input: pyml AST
- Output: Modified ASTs



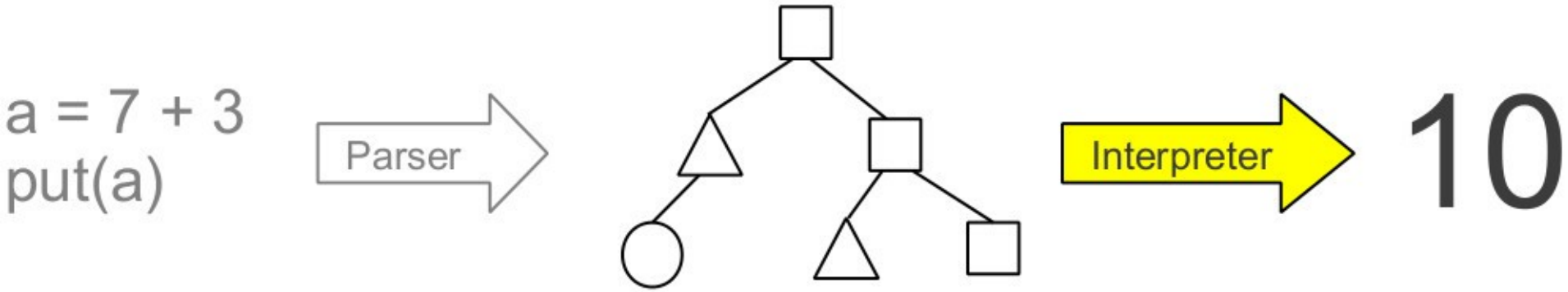
Example 3: pyml Visitor

- Files:
  - AST: pyml_ast.py — read only
  - Default visitor: base_visitor.py — read only
  - Visitor 1: text_change_visitor.py — read only
  - Visitor 2: politeness_visitor.py — to modify
  - Visitor 3: enlarge_visitor.py — to modify
  - Visitor 4: gui_visitor.py — read only
  - Coordinator: coordinator.py — read only
  - Tests: visitor_test.py — read only

- Your task:
  - Remove impolite text
  - Enlarge the UI

Interpreter

- Input: AST of a program
- Output: Program "execution"



Python – Interpreter (parser and interpreter)

Example 4: javali Interpreter

- Input: javali code
- Output: Output of the javali program

```
a = 10

while (a < 13) {
    a = a + 1
    put(a)
    b = 0

    while (b < 2) {
        b = b + 1
        put(b)
    }
}

put(3 + 5)
```


```
11
1
2
12
1
2
13
1
2
8
```

Example 4: javali Interpreter

- Files:
  - Parser: parser.py — to modify
  - AST: javali_ast.py — to modify
  - Default visitor: base_visitor.py — read only
  - Evaluator: evaluator_visitor.py — to modify
  - Interpreter: interpreter.py — read only
  - Tests: interpreter_test.py — read only

- Your task:
  - Comparison operators: >, <=, >=, ==, !=
  - Negative integers: a = -7
  - if-else statement: if (1 <2) {...} else {...}

Conclusion

- What is a parser?
- How to use funcparserlib?
- What's the visitor pattern useful for?
- What is an interpreter?
- How to write a parser and an interpreter for your own programming language?