SICPy

Khooi Xin Zhe - A0228307H
Ang Jun Jie - A0228577N

Outline

- Overview
- Implementation & Design
  - High-level overview
  - Custom SICPy parser
    - Pits & falls of ANTLR4
  - Common Python backend
  - Execution flow
  - SICPy sublanguages
- Demo
- Contributions
- Future Enhancements
- Journey Recap
- Summary

Overview

What to expect?

Overview

- Objectives:
  - Series of sublanguages of Python – **SICPy**
  - Playground for SICPy – **SICPy Academy**
- Project Scope:
  - Adapt Source §1-4 to SICPy
  - Linting functionalities (e.g., autocompletion/ syntax highlighting)

Implementation & Design

What have we come up with?

High-level Overview

- Building on top of existing infrastructures:
  - x-slang
  - x-frontend
- SICPy specifications adapted from Source
- Custom SICPy parsers to restrict the use of specific syntaxes depending on the sublanguage variant
- Common Python backend for SICPy program execution

Custom SICPy Parser

- Three-stage parser
  - Syntax analysis (ANTLR4)*
  - AST generation (Python AST - ASDL)
  - Syntax restriction (custom “evaluator” – parser_python.ts)
- Design reasoning:
  - Syntactical errors flagged before execution
  - Fine-grained error messaging

** Currently disabled due to issues with the ANTLR4 Python3 grammar, see next slide.*

Pits & Falls of ANTLR4

- ANTLR4 grammar designed for interactive use
- If more than one “blocks” is passed to ANTLR4, only one is recognized…
  - Various attempts to fix the grammar has been carried out, but to no avail.
- Related open GitHub issues (7):
  - 1061, 1073, 1078, 1086, 1352, 1646, 1801

```
1+1
def f():
  pass

f()
```

```
(single_input \n)
```

Example 1

```
def x():
  if True:
    pass
  else:
    pass
  1+1
```

Example 2
(Pass???)

How it could have been…

SICPy Academy Playground

Run SICPy § 4 1000

```
1 def func():
2     1+1
3       2+2
4     |
```

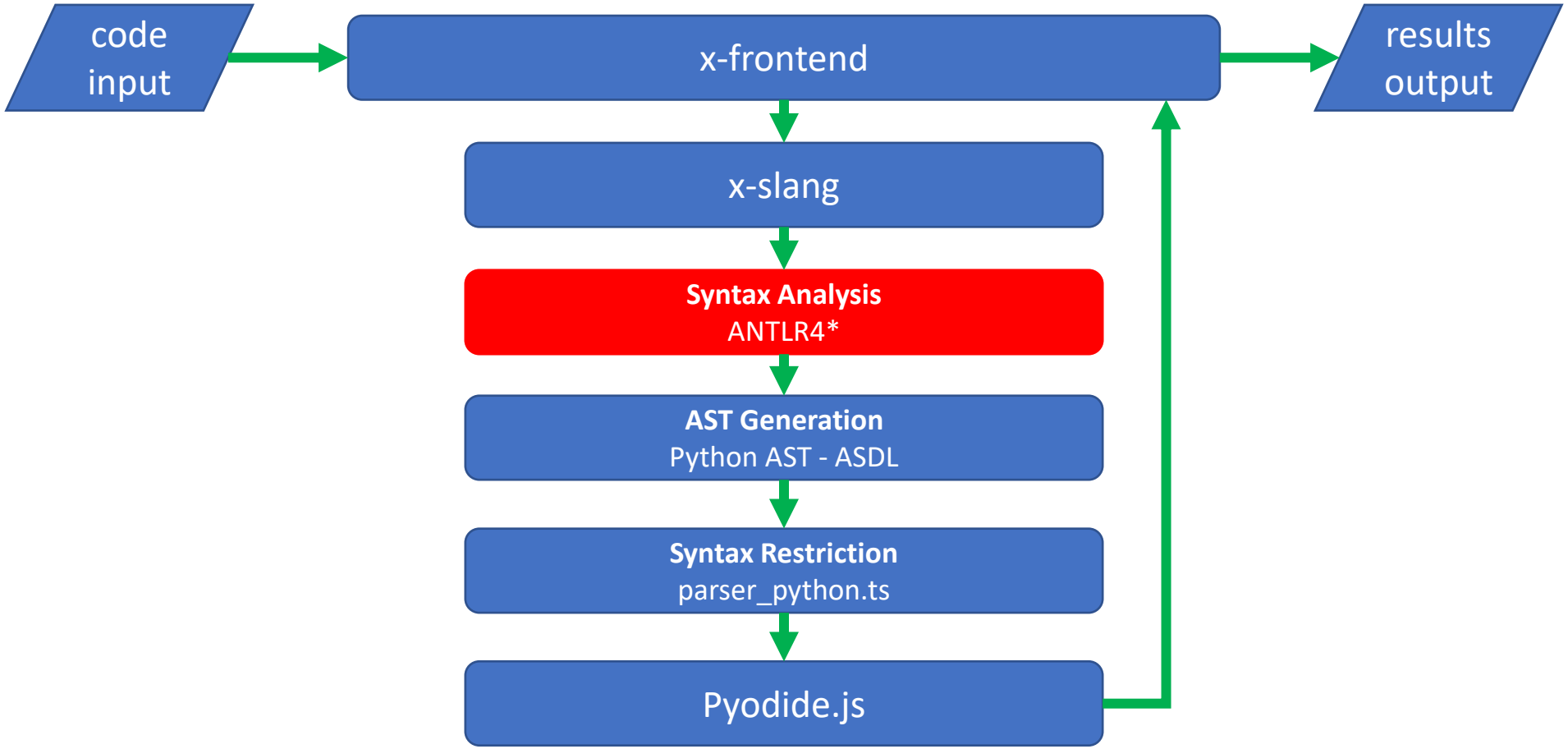
Expected:

```
line 3, col 6: extraneous input '      ' expecting {STRING, NUMBER, 'def', 'return',
'raise', 'from', 'import', 'global', 'nonlocal', 'assert', 'if', 'while', 'for', 'try',
'with', 'lambda', 'not', 'None', 'True', 'False', 'class', 'yield', 'del', 'pass',
'continue', 'break', 'async', 'await', NAME, '...', '*', '(', '[', '+', '-', '~', '{', '@',
DEDENT}
```

Common Python Backend

- Plethora of options available:
  - Skulpt.js
  - Pypy.js
  - Brython
  - **Pyodide.js**
- Selection criteria:
  - Fast (Web Assembly-based)
  - Well documented
  - Actively supported project (most recent commit – two days ago)
  - Recent Python version used (Python 3.8)
  - Extensibility (support for third party packages)

Execution Flow



** Currently disabled and bypassed.*

SICPy Sublanguages

- SICPy §0
  - arithmetic & boolean expressions
- SICPy §1
  - condition expressions
  - import statements
  - function declarations
  - function calls/ applications
  - declaration/ assignments
  - return statements
  - if-else statements
- SICPy §2
  - lists
- SICPy §3
  - loops (i.e., for, while)
  - continue, break, pass statements
  - list comprehension
- SICPy §4*
  - dictionaries
  - tuples
  - exception handling

** SICPy §4 contains additions on top of Source §4*

Demo

and ~~“evaluator” walkthrough~~ syntax analyzer walkthrough

Contributions

What did we deliver?

Contributions

- Recap on Deliverables:
  - ✓ Series of sublanguages of Python
    - Source §1 to SICPy (baseline goal)
    - Source §2 - 4 to SICPy (reach goal)
  - ✓ Playground for SICPy
    - Linting functionalities (reach goal)

- What we have delivered:
  - SICPy §0 – 4
  - SICPy Academy
    - w/ basic autocompletion, syntax highlighting
  - Updated and revised ANTLR4 Python3 grammar from 3.6 to 3.8*

*While we failed to patch the Python3 grammar for ANTLR4, we did revise the grammar to reflect the changes for Python 3.8 during the process.

Future Enhancements

What else can be done?

Future Enhancements

- Wrapping Pyodide.js in web workers
  - Frees up the main thread during program execution
  - Control program execution time
- Sublanguage-specific linting functionalities
  - Only highlight valid syntaxes given the sublanguage
  - More robust autocompletion supporting dynamic imports (e.g., import math)
- Improving error messaging to better fit with the context of the sublanguage
- Further investigate ANTLR4 grammar issues

Journey Recap

How the project has been?

Journey Recap

- First time working with web-related projects
  - Unfamiliar with various tools used in x-slang, x-frontend
- ANTLR4 – source of surprises
  - Code available on grammars-v4 may not work
  - Visitor vs Walkers?
- Applied ideas from the lab assignments to perform syntax restrictions while walking through the AST ~~like an "evaluator"~~ with a syntax analyzer.

Summary

Wrapping up!

Summary

- SICPy as an adaptation of Source
- SICPy Academy as a potential alternative for teaching Python

"For someone who is not yet a programmer, who wants to become a programmer, for those people Python is particularly easy to get."

– Guido van Rossum, creator of the Python programming language

Q&A

Found some bugs? Please let us know!
Mail to: khooixz@comp.nus.edu.sg. ☺