SSSNAKELYZER

CS335: COMPILER DESIGN

PYTHON TO X86_64 COMPILER

Chitwan Goel (210295)

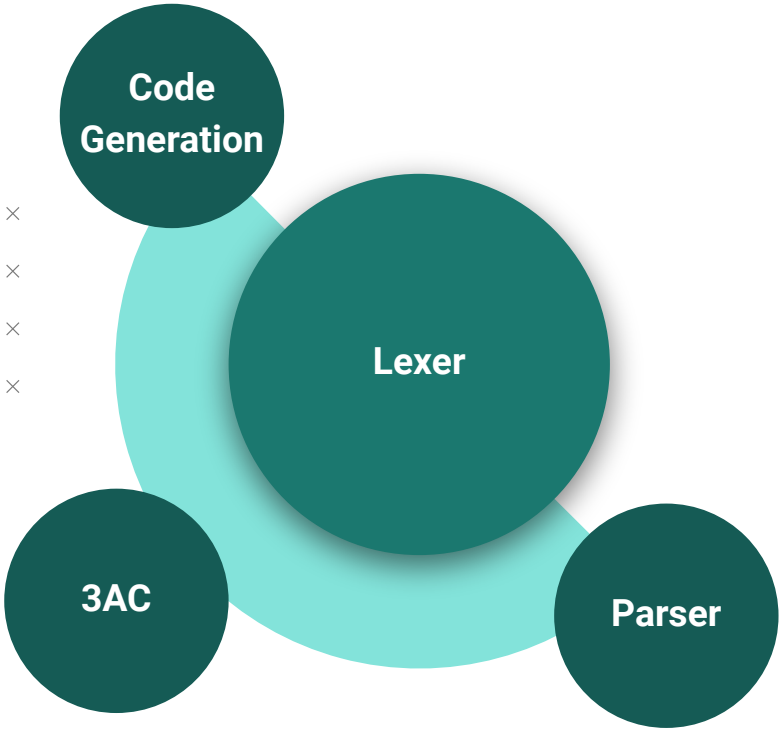
Shrey Bansal (210997)

Talin Gupta (211095)

ACKNOWLEDGEMENT

We would like to thank our professor, Mr. Swarnendu Biswas for teaching this course. This project seemed very difficult at first but with the constant guidance and clarifications from the instructor, this was made possible We would also like to thank our TA Sodanapalli Linga Reddy for constant evaluation and feedback. We're proud to be the last batch required to take this course as part of our department's mandatory curriculum.

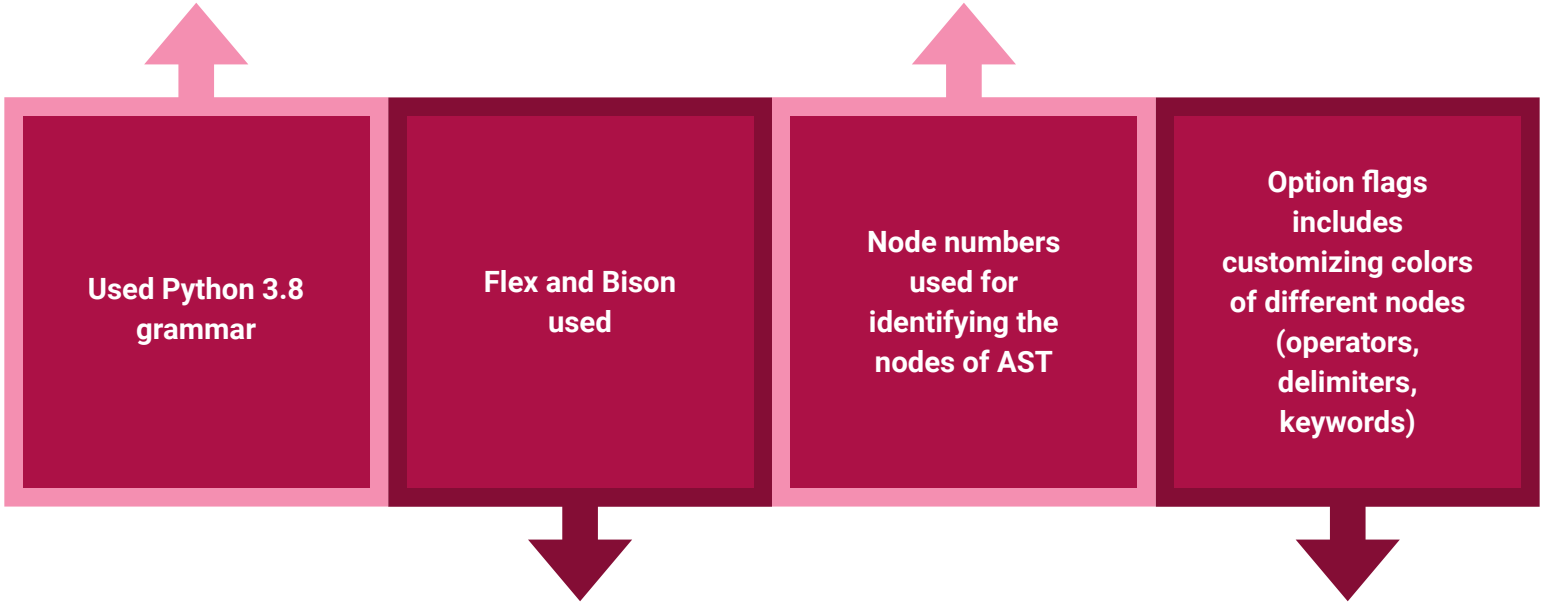
Introduction



Our compiler, sssnakelyzer can run a python code with all the necessary features supported, plus some additional ones.

It was a matter of great joy for us when we saw the python codes executing. Let's take you on a journey which spanned over a period of 2.5 months and saw our efforts bear fruit...

Milestone 1


Used Python 3.8 grammar
Flex and Bison used
Node numbers used for identifying the nodes of AST
Option flags includes customizing colors of different nodes (operators, delimiters, keywords)

Milestone 2

Type Checking

- Type errors thrown are similar to python
- Type compatibility checked based on implicit type casting
- Trailered expressions have recursive type-checking
- Four entries of print function for the 4 primitive types

3AC generated in the same pass

- There is a single pass which involves parsing, type checking and 3AC generation
- The 3AC code is stored in quadruples

3AC conventions

- Temporaries start with "`__`"
- Stack manipulation: `push_param`, `pop_param`, `+xxx`, `-xxx`
- Name mangling has been used with "`@`" as separator
- Special instructions for making strings, printing strings and booleans

Symbol Table

- Global symbol table - contains information about classes and global functions
- Class symbol table - contains entries for various attributes and member functions
- Function symbol table - contains the entries of formal parameters and local variables
- For inheritance, we perform lookup in class hierarchy

Milestone 3

Local Variables and Temporaries

- The local variables and temporaries are pushed on stack
- Their local offset relative to `rbp` is used for addressing them

Custom functions

- Custom functions for `strcmp`, `len`, `allocmem` print have been added
- `Print`, `strcmp1`, `allocmem` further call the glibc implementation of `printf`, `strcmp`, `malloc` respectively

Three AC to X86

- The code has been divided according to functions
- Code is generated function-wise
- We generate code through a template matching scheme

Calling conventions

- We pass parameters to functions through stack
- Return value is in register `rax`
- Popping of params is done in reverse order in which they appear in the body of function
- Register saving is done on calls

Optional Features Supported

- Implicit and explicit Line joining
- Multiple function calls in a line (nested function calls)
- Chained attributes (a.b.c.m(), a.b()[].m(), ...)
- Array index out of range check

CODE WALKTHROUGH

Talk is cheap. Show me the code

THANK YOU

- Team Pythoneers

Member	Contribution (%)
Chitwan Goel	33.33%
Shrey Bansal	33.33%
Talin Gupta	33.33%