Progress Report A6

Vision: We are creating a python interpreter in OCaml. This involves a text user interface, a python parser, some kind of evaluation, assigning values to variables, return statements and for and while loops. We may include functions and objects if time permits in later sprints.

Summary of Progress: Thus far, we have implemented the parser for determining whether to assign a value to a variable or evaluate an expression and prints the resulting value. As of now, the parser is able to detect bracket for lists, parentheses, quotes enclosing strings, determining the type of an expression: whether it is unary, binary, a variable that was defined earlier, a list of expressions, or when it is just a simple value. The evaluator supports values of ints, strings, bools, floats, or lists. It also handles most of the built-in operators (both binary and unary), emulating the flexibility that Python has regarding mixing types when using binary operators (such as True + 1 evaluates to 2). We also have support for if/elif/else blocks and while loops.

Activity Breakdown:

Eric:

1. The refinement of the helper functions (including checking cross-types between arguments of a binary operator)
2. Exception raises for helper functions
3. State handling, which includes tracking up assigned variables in previous statements

Patrick:

1. Parsing of assignments and expressions
2. Errors and Error handling
3. Operators
4. Most of the syntax checking in parse
5. Lists

6. Cleaned up and refactored evalutation

William:

1. Helper functions of: addition, subtraction, multiplication, division, exponentiation, boolean operators, modulo, etc.
2. The evaluation function itself
3. State handling

Zaibo:

1. Parsing of if/elif/else blocks
2. Parsing of while loops
3. Parsing assignments and expressions.
4. Main interpreter

Productivity Analysis: We actually did pretty well and exceeded our own expectations. We initially aimed to support basic operations of just ints and strings. But we ended up getting operations of bools, floats, and lists done as well, although there are a few bugs in several corner cases that we have not taken care of. We also managed to start handling if statements.

Coding Standards Grades:

Scope Grade:

Excellent: Assignments, evaluating expressions, dynamic typing, if statements, conditional statements, exceptions

Satisfactory: Assignments, evaluating expressions, dynamic typing

Unsatisfactory: Assignments and evaluating expressions.

We grade ourselves Excellent because we implemented all the functionality in excellent. We even went past excellent by implementing while loops (although they do not work perfectly).

Goals for Next Sprint: We will add support for defining and calling functions. After we have function calls, we can implement many of the built in python functions. For example, list.append() is not currently supported. We will also add more functionalities for arrays such as indexing and slicing. Finally, we will write a wrapper for the interpreter so that we can input a python file.

Coding standards grades:

Comprehensibility: 1

Testing: -1

Documentation: -1

Format: 0