Programming Project #5
CpSc 8270: Language Translation
Computer Science Division, Clemson University
Interpreting Spaceless Python: Phase I
Brian Malloy, PhD
October 27, 2014

## Due Date:

In order to receive 90% of the credit for this assignment, your submission must be submitted, using the `web handin` command, by 8 AM, Wednesday, November $12^{th}$ of 2014. If you are unable to complete the project by the due date, you may submit by 8 AM, Monday, November $19^{th}$ of 2014, to receive 80% of the grade.

## Project Specification:

Your project is to build *Phase I* of an interpreter for spaceless Python using the grammar, or a slightly modified version of the grammar, that you used in the previous project. To interpret a *Phase I* spaceless Python program, you must build an AST that includes structures to handle integer and floating point variables and constants, and all valid spaceless Python arithmetic expressions and output statements. However, for *Part I*, you do not have to interpret function calls, parameter lists, or decision structures.

We will discuss both the syntax and the semantics of spaceless Python, but when in doubt, we will use the syntax and semantics of Python, including *duck typing*, however the basic requirements only entail your incorporation of two types: integers and floating point variables and constants.

Nevertheless, when designing your solution to Phase I, you should be mindful of the issues that you will confront in Phase II. In particular, spaceless Python does not include explicit repetition structures, but we will simulate loops using recursive function calls.

In addition, you must alter the spaceless Python parser so that it performs error recovery. Your error recovery should specify the line number of the erroneous statement, and continue parsing. For an example of error recovery, see the example in the repo at:

`8270assets-2014/examples/bison/errorloc`

Finally, you must write at least five (5) programs that exercise the structures for Phase I, and your spaceless Python interpreter should be able to interpret these programs.

These are the base requirements for Phase I. However, you can receive more credit by adding additional functionality, which may entail augmenting the grammar. For example, you can augment the grammar to add global variables using the keyword global (see Python global statement). Another example might entail you extending the grammar to include statements outside of the scope of a function, similar to the statements that Python permits. There are many examples of extensions that you might use; for example, you might want to incorporate additional types, such as *strings*, or additional data structures such as lists. You can also add functionality of your choosing and these do not necessarily entail modifying the grammar.

To summarize the Phase I requirements:

1. Interpret arithmetic expressions
2. Interpret integers and floating point
3. Interpret straight-line code in functions
4. Interpret each function in a spaceless Python program
5. Augment the spaceless Python grammar to recover from syntax errors.

An example test program might be:

```
def f():
  x = 2
  y = 3
  print -y + x**3 -1
end

def main():
  sum = 2.5
  product = 2*2.5
  print product
end
```