**Game Modifications DSL Design**

**Features**

* Pythonic/Lua-based syntax.
* Syntactically identical Lua tables.
* Local (module) scope, with a new global keyword if other mods/modules need to see that variable.
* while, do while and for loops (with Pythonic indentation but C-like for loops).
* if/else if/else statements (with Pythonic indentation/syntax).
* switch statements (with Pythonic indentation and Java-like syntax).
* try/catch statements (with Pythonic indentation and Java-like syntax). Figure out if the grammar needs to be changed to get this working with Lua tables/dispatching.
* and, or, is, not, in, break, continue, and return keywords.
* First-class functions.
* Libraries:
  + Standard library, which at least includes functional methods like map.
  + import statement. Possibly add in from \_ import \_?
* Numerical operators (+, -, \*, /, %, ^).
* Compound assignment operators (+=, -=, \*=, /=, %=, ^=).
* Comparison operators (>=, <=, >, <): Note, we will use is and not for equals and not equals, which will return true if the values are equal (and not references like in Python).
* Increment/decrement operators (i++, i--, ++i, --i). Probably not these.
* Negate operator (-n).
* Concatenate operator (++).
* Optional enforceable types:
  + boolean
  + int
  + float
  + null
  + string
  + function
  + var: Used for Lua tables.
  + *N.B. May also need an equivalent for the Lua types of CFunction and Userdata.*
* Events and event handlers (possibly defined in a library).
* Parentheses for specifying higher precedence.
* Parsing and type checking errors.
* Comments (// for single-line and /\* \*/ for multi-line).
* Extension features:
  + Timers
  + Macros
  + In editor API documentation reference/code completion.
  + Threads
  + IO support

**Implementation Details**

* PEG parser generator tool (initially trying TatSu).
* Type checker (most likely coded in Python or Lua).
* Still need to figure out how to translate to Lua code.
* Need to be careful to support anything present in the APIs.

**Grammar**