Wilson Li Secure Turing Complete Sandbox Documentation

The Turing complete sandbox is implemented in Python 2.7 and is used to run other custom programs written in Python 2.7. The sandbox only supports .py files and running one program at a time.

Operations supported by Turing complete sandbox:
Custom programs written for the sandbox will have the same syntax as Python 2.7 but with a lot of the features being restricted.

All python reserved words are blacklisted except for the select few that are needed to implement any computable program.

Allowed reserved words:
[and, break, continue, elif, else, for, if, in, is, not, or, print, while]

Banned reserved words:
[as, assert, class, def, del, except, exec, finally, from, global, import, lambda, pass, raise, return, try, with, yield, ., _]

In addition, all built-in functions are removed, with the only built-ins white listed being:
[False, None, True]

With these restrictions in place, the sandbox prevents writing classes, functions, objects, importing modules, I/O, executing other code, and allocating memory on the heap. Since the sandbox does not allow return statements, any output for the program will need to be printed to the console.

Turing-Complete:
The sandbox is Turing complete because it can compute all computable problems. All the arithmetic operators, variable assignments, conditional statements, and loops are allowed to be used to implement a program to compute a problem. Since all computational operations are allowed, this means that a computation problem can be computed using a combination of these operations.

Example programs:

1. powersOfTwo.py

This program computes all the powers of 2 between 1 and 128 by using a while loop and multiplying the number by 2 at each iteration.

2. fibonacci.py

This program computes the first 10 Fibonacci numbers. It uses a while loop and at each iteration, it adds the 2 previous numbers together and then reassigns the variables accordingly.