steve.py
https://github.com/CallMeSteve/AppSec/tree/master/Assignment%201%20Sandbox
This sandbox restricts available resources and removes a number of builtin functions
- open is allowed, can write arbitrary files (beatSteve1.py)
- exec is allowed, can run code outside of sandbox (beatSteve2.py)

valcarcel.py
https://github.com/Justinvalcarcel/CS9163
This sandbox forces the code being run to import and wipe the sys.module dictionary. It also implements a blacklist and executes code
- can access arbitrary files by running the input function again (beatVal1.py)
- ads

ahhinav.py
https://github.com/abhinav1911/Assignment1
This sandbox creates an import and keyword whitelist then executes code.
- can make changes to sandbox before executing code

aot.py
https://github.com/aot221/SandboxEnvironment
This sandbox allows a user to log in and then execute code in a fresh namespace every time. Variables do not carry over between commands. Unfortunately this sandbox does not have support for file input of code.
- can access arbitrary files by using open(filename, 'rb').read() including the username and password files

pBjadhav.py
https://github.com/piyushbjadhav/Sandbox
This sandbox limits user resources, whitelists functions and then executes the code.

wLi.py
https://github.com/WilsonLiCode/SecureTuringCompleteSandbox
This sandbox uses a blacklist and a sort of white list. First it bans certain words then it clears the namespace except for a few important and relatively innocuous builtins.

a-sandbox.py
This sandbox bans all characters aside from numbers, important symbols (e.g. +-*/=) and the letter a.