Project title: Language Interpreters—case study in Python Virtual Machine [CPython]

Team members: Ryan Eggert

Brief description of project (1-3 paragraphs):

Python, unlike C, is typically interpreted instead of compiled. When one writes a program in C, he compiles it and produces a file of machine-specific instructions. These instructions are able to be run directly on the processor of the computer compiled on. Python, however, is usually turned into an interpreter-specific bytecode which is read by a 'virtual machine' and translated into machine-specific instructions at runtime.

I wish to examine the virtual machine used in language interpreters. What does it take as input, what resources does it use, what does it output? This seeks to answer or clarify, to the extent possible in two weeks, how high-level Python is turned into executable processor instructions.

2-3 references you plan to use

Python source code, especially `ceval.c`, "Python" directory

http://cython.org/

https://docs.python.org/3.3/library/dis.html

Deliverables:

The minimum deliverable would be a document.

Minimum:

- Description of virtual machine function in interpreted languages.
- Description of CPython Python interpretation/running process
- Description of Python Virtual Machine implemenation

Planned:

- Description of Python bytecode instructions
- Discussion of other (i.e., non-CPython) implementations' virtual machines/strategies

Stretch:

- Tool to view processor instructions for a given Python program.
  - Could be done by modifying python source code [C]?
- This may also require an overview of the instruction set used by Olin laptops.

Planned Deadlines

December 6: Have identified any non-CPython implementations worthy of mention or discussion.

December 9: Have researched VMs/related topics to depth sufficient to begin document writing.

December 13: Conclude Paper Writing. Begin assembling a summary presentation for the final period.

December 15: Present.