NPRG045 Project Proposal: OPythn

Marcel Goh

February 26, 2019

1 Scope

The aim of this project is to create an interpreter for OPythn, a subset of the Python programming language. The process will be split into two main parts. First, the OPythn code will be parsed and compiled into intermediary bytecode. We will then implement a stack-based virtual machine to interpret this bytecode and produce the behaviour specified by the OPythn program.

2 Language Design

The OPythn compiler and interpreter will support a sizable, functional subset of Python. This includes primitive types and their operations, named functions, lists, dictionaries, and classes. Features that can be implemented in terms of this base language will be provided as libraries, written in OPythn. If the timeline permits, OPythn can be extended to support more advanced constructions such as anonymous functions, list comprehensions, generators, and coroutines.

3 Implementation

The front-end the program will be written in OCaml. This consists of getting the user input, displaying program output, as well as functionality for lexing, parsing, and compiling the OPythn code. The lexing and parsing will be accomplished using `ocamllex` and `ocamlyacc`, unless more flexibility is required than those libraries can provide, in which case hand-written rules will be applied.

The target bytecode will read by a virtual machine and for this step, we will consider writing certain functions in C and calling them into the OCaml code if necessary, via OCaml's interface with C. This may facilitate low-level operations on bitstrings, for instance. Designing the bytecode and virtual machine is a core part of the project. The virtual machine will be stack-based with typed data. The plan is to have one-byte instructions, which should allow a relatively wide instruction set. This means that different types of data can be handled by separate instructions. A significant advantage of writing the bytecode interpreter in OCaml (as opposed to switching to pure C for this step) will be that we can make use of OCaml's native garbage collector to manage OPythn's garbage collection.

4 Target Platforms

OPythn will be developed on macOS and aim to support both macOS and Linux operating systems. Users will be able to interact with OPythn either by calling the program with source code files as arguments or via a top-level REPL (read-eval-print loop).