Python Type Inference: If It Quacks Like a Duck...

Jay R Bolton

May 24, 2012

Introduction

Python is a wildly dynamic language whose program behavior depends almost entirely on the execution path. Despite being very multi-paradigm, it is also not a very complicated language, with an abstract grammar of only around 100 lines. The enforcement of type correctness in python is performed at run-time using Duck Typing, where we are only concerned with the comparative structure of two objects. We'll explore this further in sections below.

All languages can be statically typed, no matter how dynamic their current implementation may be. Our goal here is to perform static type checking (albeit in a limited form) to a very dynamic language with very minimal type enforcement. Our type checking system will be optional and will not affect your ability to run the program. The advantages of having such a system are:

- Retain 'agile development' without limiting possible correct programs.
- Still be able to analyze the type correctness of your programs, catching errors that you may not find in tests.
- Provide documentation and meaningful type annotations that can be helpful in understanding your program in a number of different ways, discussed in the last section.

The Type System

Here will be an overview of how python's type system works, with some formalized notation for it.

Type Inference Using Only Attributes

First formally define the type system...

$$\begin{aligned} type &= Obj\ \{attr : type,\ attr : type,\ \cdots\} \\ &\mid Builtin \\ attr &= name \\ Builtin &= Tuple(type,\ type,\ \cdots) \\ &\mid Int \mid Str \mid \cdots \end{aligned}$$

Then define all the inference rules...

Lookup

$$\frac{M \vdash e : \tau}{M \vdash e \ : \ \tau}$$

Assignment

$$\frac{M \vdash e : \tau}{M \vdash x = e,\ x \ : \ \tau}$$

Function Definition

$$\frac{M, [\text{params}] : \tau \vdash \ [body] : \{\text{`} * r' : \sigma\}}{M \vdash \ \text{def f}([\text{params}])\ [\text{body}] \ : \ \{\text{`} * r' : \sigma,\ \text{`} * p' : \tau\}}$$

Examples and Cases

Present an example program and walk through all the type inference rules.

Then discuss the edge cases and limitations of inferencing python, such as dynamic redefinition of attributes.

Improvement and Expansion

Talk about how the system could be expanded and all its potential uses.