Dynamic Versus Static Typing

Rushi Shah

23 August 2015

Want to know why people my age hate Java so much? Its because Java yells at you. Whenever you screw something up, Java will be the first one to let you know. And they don't like that! There was even a time when I didn't like that. So I, like many others, made the switch to Python and applauded the dynamic typing. I could do whatever I wanted! I could say the variable `x` was a number, and then the very next line be like "Psych! `x` is actually a string!".

Python is doublethink. Python is like

> "Yeah, of course `x` is a string. Why wouldn't it be a string?"

Java will take none of that. Java will be like

> "You LITERALLY just said `x` was a number. Did you lie to me?"

The age old question of dynamically versus statically typed languages will probably never reach a conclusion because they both have their merits. Rather than one "winning out" over the other, they will continue to mature and diverge because they are both good for different things. Java's type system leaves something to be desired and Haskell swoops in and fills that desire. Python seems good for now, but I'm sure something will come along and one up it eventually like Python 3 was supposed to.

Now if you read this Stack Overflow answer on dynamic vs. static typing, you would be surprised to realize that both options sound pretty bad. Whenever you are given a choice in CS (and typically life in general), that means that all your options have merit for some reason or the other. Its up to you to decide what you value and make your decisions based on that priority. I personally value structure and order. I want the computer to work for me and tell me when I make a mistake so I can fix it. I think that since you should be documenting your code anyways for humans to read, the compiler 'ought to pick up on those annotations as well. Like I said, I wasn't always like this: I hated Java's type system just like everyone else. But when I discovered Haskell's type system I learned that static typing was for me.

1 Python (dynamic) vs. Haskell (static)

This is a bit of python code. Without having any information about the code that has been replaced by the ellipses, what can you tell me about what this function does? Not much, right?

```
def foobar(a, b):
    ...
    #This basically creates a function called foobar in Python
    #It takes two variables as arguments
```

Compare that to the following Haskell code:

```
foobar :: a -> a -> a
foobar = ...
--This defines a function foobar in Haskell
--It takes two arguments of any type (both are same type)
--It also returns something of the same type
```

Given this example of strongly typed code, you KNOW, without a doubt, that foobar is one of two functions (it either returns the first argument or returns the second argument. That's it.). That's kind of neat, right, because your type signature just described what the function is doing. For a more in depth explanation of this code, check out this explanation of parametric polymorphism. In the python code, you have no idea what the function is doing or what it is going to return so you have to actually look through all the (perhaps badly written) code.

This of course was slightly rectified with the creation of "docstrings" that document for the reader what the function is supposed to do. But that is just a workaround, and one that a majority of Python programmers don't even use. But hey, let's say that every Python programmer out there starts using docstrings for code readability. That still doesn't help the code at all! The code is not made stronger by this extra work that is put into documentation. With Haskell type signatures, though, its different. The compiler understands your documentation just as well (if not better) than any other reader of your program would. That is what I mean when I say that I want the computer to do the work for me and tell me if I'm actually doing what I want to be doing.

Is this a bit more up front effort? Yeah it is. You have to know what you want a function to do. But you should know what you want your function to do! If you don't, then maybe the problem isn't the type system...

2 See also

SO: Dynamic type languages versus static type languages
SO: What do people find so appealing about dynamic languages
CIS194: Explanation of parametric polymorphism and the Haskell type signature