How we got the numbers in Figure 3.

Python:

I used Python 2.5.2's standard library - everything in Lib/ that wasn't in Lib/site-packages. Additionally I used the Django codebase, version 1.2.3.

All instances were found by grepping for certain patterns and verifying that they were type tests by looking at the context they were found in (the previous and following few lines of code).

undefined:

Searched for "is None" and "is not None". The files in the ./test/ directory had many of these checks, but did not use them to refine types, so we did not count those instances. nces. same reasoning as for `is None`.

The count is likely an under-estimate. In many cases, instead of having a conditional "if x is not None:", the shorter form "if x:" is used. Some of the latter are just checking to see that "x" is not 0, or an empty string/list/tuple/dictionary, but others are indeed type tests. Counting these accurately would require more sophisticated code analysis than grep, however.

field-presence:

Searched for "\bhasattr(". The most common pattern was:

```
if hasattr(obj, <stringliteral>):
    obj.stringliteral(...)
```

typeof:

Searched for "\btype(". The word boundary is important, as apparently functions ending in "type(" are prevalent in the standard library. Again only counted instances that weren't in the ./test/ directory. There were a few more instances where it was used just for testing, such as "`failUnless(type(x) ==type(''))`," and where nothing more was done with the variable afterwards - I didn't count these. The rest were used for type refinement, for example in locale.py:

```
def setlocale(category, locale=None):
    if locale and type(locale) is not type(""):
        # convert to string
        locale = normalize(_build_localename(locale))
```

```
        return _setlocale(category, locale)
```

instanceof:

Searched for “\bisinstance(“. Removed lines containing “fail”, “unless”, or “self.assert_(“. The rest were of a form like:

```
    if isinstance(val, tuple):
        new_val = list(val)
```

An interesting note is that there were far more uses of “isinstance(“ than “type(“ in the Django code base. In many situations, either type test can be used (for example, “isinstance(x, str)” is mostly equivalent to “type(x) == str”), so it is mostly a matter of style.

JavaScript:

The code base was all the google gadgets as of April 22, 2010. Many of the gadgets had identical files, likely from using the same libraries, so out of the initial 3500 or so .js files, only 2757 unique files remained.

undefined:

Searched for one of ==, ===, !=, and !== followed by null or undefined, as both are used as sentinel values to indicate lack of a value. Among this codebase, explicit tests for null far predominated tests for undefined. Like in the Python codebase, JavaScript code often uses the “if (x)” or “if (!x)” idiom to test for undefined, and these were not counted.

field-presence:

There is no field-presence operator in JS. Instead, checks are done either like “if (obj.attr) { ... }” or “if (obj.attr == undefined) { ... }”. The latter were counted under the “undefined” tests. The former were not counted for the same reason “if (x)” was not counted in the undefined tests.

typeof:

Searched for “typeof”. Most of the results were from various implementations of XML or JSON parsers/emitters.

instanceof:

Searched for “instanceof”. There were very few hits, indicating this to not

be a popular pattern among google gadget developers.

Ruby:

For Ruby, I used the Ruby 1.8 standard library, as well as Ruby on Rails version 2.2.2. Although I was unfamiliar with Ruby, it was easy to find equivalent constructs to the Python and JavaScript ones. This indicates that scripting languages share the need for such tests, and fill the need one way or another.

undefined:

In Ruby, “nil” is used as the undefined-sentinel. nil is an object, just like in Python. All objects in Ruby have a nil? method, returning True if the object is the nil object. Grepping for “\.nil?” indicated that .nil? is used much like “is None” in Python:

```
./cgi/session.rb:         f.close unless f.nil?
--
./cgi.rb:           if c.nil? || c.empty?
./cgi.rb-             raise EOFError, "bad content body"
./cgi.rb-           end
./cgi.rb-           buf.concat(c)
./cgi.rb-           content_length -= c.size
```

I also grepped for “== nil” and “!= nil” patterns. These occurred less frequently, indicating that “.nil?” is more idiomatic, but served the same purpose.

field-presence:

All Ruby objects have a respond_to? method, which takes a symbol and returns True if the object contains a method with that name. There is a way to test for the presence of instance variables, instance_variable_present?, but the code I examined rarely used this method. Like in Python, much of the uses were to check for the presence of a method and immediately call it if present.

typeof:

I could not discover an equivalent of “type(“ in Python or “typeof” in JavaScript before the paper was submitted. I later discovered “obj.class” serves the same purpose as “type(obj)” in Python, so perhaps the pattern “obj.class == foo” is used in Ruby code.

instanceof:

There are three object methods that seemed to perform this function: kind_of?, instance_of?, and is_a?. Apparently kind_of? is exactly equivalent to is_a?, and the choice of which one to use is left to the programmer as whatever reads the best (quoth a Ruby user from StackOverflow: “Think `@honda.kind_of? Car` and `@person.is_a? Administrator`”). They both check the inheritance hierarchy, so “subclass.kind_of? superclass” yields True. instance_of?, on the other hand, does not check inheritance. instance_of? was used rarely - only 28 times in the standard library - but was used in a similar pattern as with kind_of? and is_a?.

```
./multi-tk.rb:    if args[-1].kind_of?(Hash)
./multi-tk.rb-      keys = _symbolkey2str(args.pop)
./multi-tk.rb-    else
./multi-tk.rb-      keys = []
./multi-tk.rb-    end

./multi-tk.rb:    if slave.kind_of?(MultiTkIp)
./multi-tk.rb-      slave.path
./multi-tk.rb:    elsif slave.kind_of?(String)
./multi-tk.rb-      slave
./multi-tk.rb-    else
./multi-tk.rb-      slave.to_s
```