Algorithms and Programming Languages Exam: Problem 5
Nick Creel - Due April 17, 2020

Question:

> Discuss the strengths and weaknesses of the Python, Javascript, and C programming languages as you see them. What sorts of problems or situations are good fits to these languages, and why? Which do you personally like, and why? Be specific, giving code examples to make your comparisons and conclusions concrete.

Answer:

Python is the language I've used most often, as it's the first language I ever learned, so I can speak mostly to it's advantages and disadvantages, and why I use it more than I use other languages. I think Python is a particularly useful general purpose language, with some exceptions. For example, because Python is a higher order programming language, it is theoretically slower than a a lower-order programming language like C. However, even with the runtime speed boost provided by C, I find developing code in Python to be a much faster, less confusing process. And in cases where the speed difference is monumental, using a tool like Cython allows a Python programmer to write in Python syntax and have the code translated and compiled into C code, rather than interpreted by the Python interpreter.

Python and Javascript are both languages that come with their own "garbage collector," which is an important feature as anyone writing code in either of these languages does not have to worry about memory management. In C, memory management is a necessity even for simple operations on arrays. To illustrate the difference, I'll demonstrate how arrays are created in Python, Javascript, and C.

Python:
```
     myArray = [] # this is technically called a list in python, but
                  # the python list is the equivalent structure to
                  # arrays in other languages.
```
Javascript:
```
     let myArray = []
```
C:
```
     ptr = (float*) malloc(100 * sizeof(float));
OR
     float myArray[10]
```

In both Python and Javascript, the user is allowed to declare an array without specifying the type of data stored in the array or the length of the array. However, due to C's strict rules on memory allocation and typing, a C programmer must declare both the type of data to be stored in the array and the length of the array. This way, the compiled program knows exactly how much memory to allocate. When declaring an array using the second method, this strict definition makes it difficult to increase the size of the array in case more space is needed. Dynamic memory management can be accomplished using malloc, calloc, free, and other functions, but successful dynamic memory management can be difficult to achieve for programmers who do not have a lot of experience with memory management; sometimes, the amount of memory needed can be underestimated, and memory can either clog up quickly or be freed too early. Furthermore, the first C example with malloc returns a pointer to a memory location, NOT an array, and working with pointers in C is another tricky skill to master, as they are more akin to addresses of objects than they are the objects themselves.

While C makes users deal with pointers more explicitly, that does not mean that pointers are not used in Python and C. Both Python and Javascript variables are actually pointers to memory that contain the value which is assigned to them. However, because both of these languages obscure the pointers for the sake of simplicity, not understanding how pointers work can cause a lot of trouble when debugging. For example, see this Python code:

```
>>> a = [1,2,3]
>>> b = a
>>> b.append(4)
>>> print(a)
[1, 2, 3, 4]
```

Even though I appended the number 4 to the array b, when I print a, the original list has also been changed. This is because there are not two lists, but only one with two different pointers to the same location. Because this is not immediately visible, debugging a problem that stems from this memory issue can be difficult without a step by step debugging tool. In practice, once I first ran into this issue, I did not face it again and learned how to avoid it, but even so, it’s relatively easy to make a mistake like this and not notice. The tradeoff with Python’s garbage collection is this slight ambiguity about memory access, but overall I think learning how Python manages memory *for you* is less intensive than learning how to manage memory yourself with C.

Of course, C is not a bad language for general purpose use, and is especially well suited for certain applications; C is particularly useful for systems programming. Even the Python interpreter is written in C, so C’s applicability to compiler design is also obvious in such a successful language. I think anyone interested in low-level programming, embedded systems, memory management, and operating systems is also right to use C for their work.

While Javascript is just as usable as Python, it is not a general purpose programming language by any means. Javascript is typically run in the browser in conjunction with HTML and CSS, and its fundamental functionality relies on events, such as when a user clicks “add to cart” or downloads a file from a server. In reality, each event function is like its own miniature Javascript program that is run when a certain signal is received, whether that signal is user input or messages across the web. Javascript is also particularly well suited to manipulate the DOM, otherwise known as the Document Object Model. This model for web design assumes that each webpage is made of nested objects, which can be manipulated either individually or by certain groupings. Changing the text displayed in a paragraph, for example, is as simple as this one line code, where the programmer searches for an element with a particular id and changes the inner HTML contents of that element:

```
document.getElementById(“about”).innerHTML = “new text”
```

I would generally advise against writing any kind of command line tool or interpreter with javascript due to it’s asynchronous nature. I worked for two weeks trying to figure out how to read user input from the command line and then print it out again when writing an interpreter, but to accomplish even the simplest task like this, I had to download an external tool through NPM built specifically for synchronous line reading. See: https://www.npmjs.com/package/readline-sync.

This issue not only reveals how Javascript is not suitable as a general purpose scripting language, but also shows how much Javascript relies on external dependencies in common use cases. While Javascript is certainly usable for most web purposes without these dependencies, many web developers use Javascript in addition to front-end design frameworks such as React, Angular, Vue, and many more. These libraries often implement their own special objects, often tooled to prevent code repetition and simplify page rendering, and most of them emphasize reactive, mobile first design. While this is not a bad thing by any means, these libraries bloat the file size of Javascript code considerably, and I would argue that learning to work with them and their means of manipulating the DOM (document object model) is more difficult than simply learning how to manipulate the DOM using vanilla Javascript. For example, to change the content of an HTML element using React, the process is much more complicated than the example I listed above. Suddenly the user is required to keep track of state using React's built in tools, and to change state via functions which are not part of vanilla Javascript. I ran into this problem when building my poetry website for Plan, as I tried to initially write the code using React, and each time I wanted to change the text of a paragraph or Component, I had to create another component with the text I wanted and render that component, rather than simply changing the inner HTML.

While I find this feature of Javascript particularly frustrating, Python also relies on dependencies for certain tasks, but I find that there are fewer packages looking to solve similar tasks and reinventing the wheel. For example, when I was working with MusicXML files over the course of my research fellowship, I was able to find one library that satisfied all of my needs and that had great documentation. Python is especially suited for scientific computing thanks to the Scipy, Numpy, and Matplotlib libraries, which are widely used and fairly well documented. Python would also be my language of choice for machine learning and data science, as a number of tools exist already for that purpose, such as PyTorch and TensorFlow. It seems that academia has greatly invested in the development of tools for Python, considering the fact that there are conferences where new packages for Scipy are introduced yearly. I believe this is less frustrating only because I find the built in functionality of Python to be largely sufficient for my purposes, but I feel like I must always search for a new library to learn for Javascript in order to accomplish any task other than web programming.

In short, I think if someone's looking for a good general purpose language with good documentation and out-of-the-box functionality, then Python is a good choice. While learning C as a first language is not impossible, it requires the programmer to learn more about memory management than they might understand at the introductory level, and Javascript is a bit niche to accomplish every task. C is still a good general purpose language for the intermediate or advanced programmer, and is especially useful when runtime is a consideration. Javascript is mostly useful for the web, but it is especially well suited for that purpose compared to the other two languages; Python is often used for server side programming and sometimes used for client side code, but I have not seen C used for web programming in any context.