Dylan Johnson

CSC 330

Language Design and Implementation

Couldn’t find the course code?

Lesson 1:

The programming language I decided to look more into is Ruby. I have never written a program in Ruby. However, when I was an intern at Target, I heard all about it. I was supposed to have a chance to learn the language; unfortunately, my internship ended before we could make it happen. Considering my time at target was about 7 years ago now, anything I may have heard, I do not exactly remember. That said, looking into that language has been a treat for me. Even with other languages becoming more popular since then, I was excited to see how easy was to find resources, as well as how many resources are still out there. One of the downsides I came across was it seemed the language was not meant for performance intensive projects. When I hear that, however, it makes me think of using it in a Google/Social Media setting. Using it for something more personal or, call them, in-house projects would be a good fit. Something I noticed about the syntax was it seemed fairly similar to Python in the sense that it was very human readable and did not require the use of parentheses, rather indentation, for formatting. Ultimately, I think using the programming language would be a fun experience and I would really like to learn it. From the context of using it, and its platform, in theory it sounds like a good idea for projects that do not require scaling, but to say for certain I would like more experience with the language and what it can handle.

Lesson 2:

I chose to write the quicksort array using Ruby to see if I could figure it out and try out this new-to-me language. I’ve attached a screenshot as well as the text below. For the rest of this lesson, that being to talk about type system of the language, I would describe it as a strong, dynamically typed language. My reasoning for this assessment is that Ruby checks that variable types at runtime, instead of declaring them right in the code like Java does. In lesson one I called the syntax out for being similar to Python, which is another dynamically typed language, another similarity between the two is a variable can include multiple different data types within the single variable. As for it being strongly typed, this restricts the ability to execute the addition of a string and an integer. Basically, this means it will not automatically convert one data type to another, this must be done manually. Whereas in Python if you were to add a string variable to an integer variable it would convert the integer to a string and concatenate the two variables. As a result of this restriction, I believe it made the quicksort a bit easier. Say for example I had changed the build to allow a user to input a list to be sorted (I wanted to but was working with free online IDE’s for RUBY and it did not allow that), had the user added a non-integer character Ruby would have caught the issue and returned an error instead of converting the data types automatically (Note if part of the code manually converts the sort would work which comes to play with it being dynamically typed). On that same note of allowing user input, had the user entered all alphabetical characters, it would have been able to handle that sort as well.

A screenshot of a computer program

Description automatically generated

def quickSort(array)

# When/if array is of size 1, returns that

return array if array.length <= 1

# Temporary variable for random element in provided array

temp = array.delete\_at(rand(array.length))

# Arrays to store the remaining elements of the array

lessThan = []

greaterThan = []

# Loops through provided array

array.each do |element|

if element <= temp # if iteration value is less then the temp random number

lessThan << element # stores in lessThen

else

greaterThan << element # Else it stores in greaterThan

end

end

# Recursively returns the sorted arrays as a single array

## Meaning it continues sorting until the array is down to 1 in line 2

return \*quick\_sort(lessThan), temp, \*quick\_sort(greaterThan)

end

# Sample array

array = [1, 4, 7, 2, 5, 8, 3, 6, 9]

# Sorts and stores array in sortedArray

sortedArray = quickSort(array)

# Prints the Sorted array

puts sortedArray.join(', ')