Journal Report 2

9/9/19-9/12/19
Jialin Tso
Computer Systems Research Lab
Period 2, White

Daily Log

Monday September 9

Coded text to expression for factorial and logarithms. Looped the input prompt to allow user to type in as many expressions as desired.

Tuesday September 10

Implemented number to word conversion using Python's num2words library after numerical answer was correctly outputted. Spent most of class determining and programming the best way for calculator to know when an input number is complete (e.g. how will the calculator know the number is ″four thousand fifty″ instead of just ″four″?). Coded a try and except segment: if word-to-num was unable to convert the current word (e.g. ″minus″) appended to the saved string (e.g. so ″four thousand fifty minus″ would return an error) to a number, convert the saved string only. Otherwise, add the current word to the saved string and make sure not to add this new string to the expression to be solved.

Thursday September 12

Received help from Mr. White on setting up workstation to facilitate installation of libraries and code running without an online compiler. Installed num2words, word2number, and SpeechRecognition libraries. Once word2number was installed, realized that code segment from last class did not work: if the number was the last one in the expression, error would occur because code expected more words to come after the number (e.g. according to old code, ″six plus fifty″ would be converted to ″6+″, making eval() unable to perform computation). Began modifying code, but suddenly discovered that if dash was added between numbers, word2number would correctly work (e.g. ″one-hundred-fifty-eight″ converts to ″158″).

Timeline

Date	Goal	Met
Sept 3	Install Wabbitemu and convert text to expression for basic functions	Emulator works and program can calculate many functions (multiple digit number words separated by dashses)
Sept 9	Install SpeechRecognition and word2number; text to expression for factorial, logarithms, permutations and combinations, and other remaining useful functions	SpeechRecognition, word2number, and num2words installed. Factorial and logarithms work. Did not code permutations and combinations because correct word2number implementation was more essential.
Sept 16	Solve integrals, derivatives, and matrices; find 1-var stats for user-inputted list	
Sept 23	Begin experimenting with SpeechRecognition and implementing speech to text	
Sept 30	Find 2-var stats for user-inputted lists and find regression equations from lists	

Reflection

My main objective this week was to successfully convert number words to their actual symbols, so I spent much time thinking of how to handle words with multiple digits. My try and except segment would work only if it did not reach the last number. Before this realization, my thought process for why saving the number string and appending the next word to it would work was that even if the program did not reach the last word digit of the number, the current string would still produce a viable number. For instance, if I were to entirely convert ”one thousand fifty one,” word2num would be able to recognize ”one”, ”one thousand”, ”one thousand fifty”, and ”one thousand fifty one” as I slowly added each part of the number to the current string.

I found a temporary solution in place of the try and except segment: separating each component of the number with dashes. Although the program can now recognize the dashed numbers based on typed input, a challenge I'll be faced with is, ”How can SpeechRecognition convert the entire number to a dashed format?” I will continue searching for new ways to convert words to number without dashes, but for now, I'll keep this format so I can add more functions to the program.

Example of input and output of current program:

Input: factorial left parentheses one-thousand-nine-hundred-forty-six minus one-thousand-nine-hundred-forty-two right parentheses.

Expression output: math.factorial(1946-1942), Answer output: 24, Text output: Twenty four.

To make the factorial equation easier for the program to understand, I kept the typed format of factorial the same as eval()'s. I did the same for logarithms, where the typed format would be eval()'s ”log-argument-comma-base” format instead of say, ”log base two of eight.”