Journal Report 6

10/7/19-10/10/19
Jialin Tso
Computer Systems Research Lab
Period 2, White

Daily Log

Monday October 7

Finished writing expreg function so that curve is accurately graphed. Began writing numtoword function. As of this day, all numerical answers are converted.

Tuesday October 8

Numtoword function can convert "." to "point". Resolved problem of multiple digit numbers having to be separated by dashes (e.g. previously, 120 was written as "one-hundred-twenty"; now, the program recognizes "one hundred twenty") by writing a try-except segment previously attempted when the issue first arose.

Thursday October 10

Searched for answers that would require output other than numbers (e.g. indefinite integrals, matrices, lists of regression coefficients, one var stats) and listed all possible symbols. Enabled numtoword function to convert answers with left parentheses, right parentheses, left brackets, right brackets, x's, e's, pluses, minuses, multiplication signs, and division signs to their corresponding word forms.

Timeline

Date	Goal	Met
Sept 16	Solve integrals, derivatives, and permutations and combinations;	Indefinite and definite integrals and derivatives (not at specified value) solved. Permutations and combinations work for a certain format.
Sept 23	Solve matrices and find 1-var stats for user-inputted list.	Common matrix functions like addition, reduced row echelon, and determinant work. 1-var stats outputted.
Sept 30	Find regression equations from user-inputted lists	Linreg, Polyreg, and Expreg can generate equations and graphs.
Oct 7	Convert answers from expression to words and resolve problem of word numbers separated by dashes	No dashes needed between numbers. Many non-numerical answers can be displayed in words.
Oct 14	Convert all indefinite integrals and regression answers to words	

Reflection

While resolving the issue of having expreg output curves, I discovered that a certain number of point pairs (at least 4) must be inputted in expreg otherwise the curve generated would be a straight line. I also needed to remember that decimals like 1.57 are recognized by the program as ”one point five seven”, not ”one point fifty seven”.

When I was writing the numtoword function, I had a revelation: to solve the problem of needing dashes when typing multiple digit numbers, what if, instead of looping through each word, I loop through the indices corresponding to an array, where each element separated by a space has a unique index? Essentially, I changed ”for word in exp.split()” to ”for i in range(len(arr))”. This way, I can assure that the last typed word will be converted to a form that the expression evaluator can recognize and appended to the expression to be evaluated. Here is a snippet of code that I wrote this week:

```
 st = ""
 temp = ""
 realtemp = ""
 word = ""
 boo = True
 arr = []
 for w in exp.split():
    arr.append(w)
 for i in range(len(arr)):
   if arr[i] == "plus":
     word = "+"
   elif arr[i] == "times":
     word = "*"
#more elif statements
   temp += arr[i] + " "
   try:
      w2n.word_to_num(arr[i])
      realtemp = str(w2n.word_to_num(temp))
      boo = False
   except:
      boo = True
      temp = ""
      word = realtemp + word
      realtemp = ""
   if(i == len(arr) - 1) and boo == False:
      word = realtemp
      boo = True
   if boo == True:
      st += word
 print(st)
 a = eval(st)
 print(a)
 print(numtoword(a))
```

Because I was busy adding possibilities to the numtoword function, I didn't have time to correctly convert multiple digit numbers to their word forms after these additions. For next week, I will ensure that numbers like 66 are recognized as "sixty-six", not "six six" by writing a try except segment as I did for converting words to numbers.

Example of new input and output of current program: Input (before adding cases to numtoword function): 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: 24 twenty-four

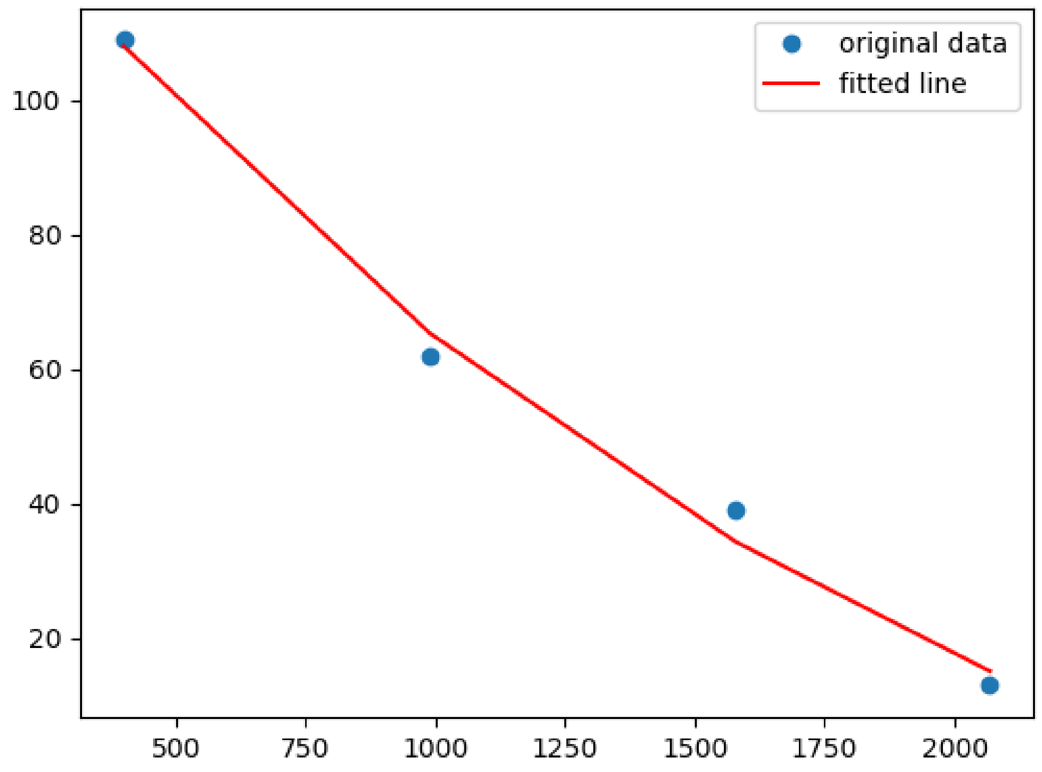
Input (implemented a while ago, but never demonstrated): combo left five comma two right (e.g. five choose two) Expression output: combo(5,2) Answer: 10 ten

Input: integrate left two comma x right Expression output: integral(2,x) Answer: 2*x, two times x

Input: exponential left left-bracket three hundred ninety nine point seven five comma nine hundred eighty nine point two five comma one thousand five hundred seventy eight point seven five comma two thousand sixty eight point two five right-bracket comma left-bracket one hundred nine comma six two comma thirty nine comma thirteen right-bracket right

Expression output: expreg([399.75,989.25,1578.75,2068.25],[109,62,39,13]) Answer (coefficients of curve equation): [1.94227476e+02 5.40780932e-04 -4.83497094e+01] (a*np.exp(b*x)+c format, different equation from calculator, but still pretty accurate) Expression output: left bracket one point nine four two two seven two four two e plus zero two minus five point four zero seven eight two five one one e minus zero four minus four point eight three four nine four one one six e plus zero one right bracket

Graph:


original data
fitted line