Q2:

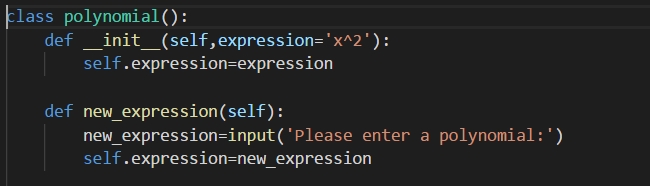
What can my program do?

1. deal with decimal representation
2. automatically deal with the calculation like x^2\*x^3
3. there’s no limit for the order of format. For example, 3\*x is the same as x\*3
4. regular and corresponding output format

What can’t my program do?

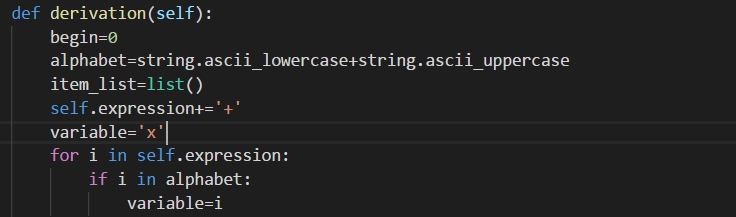
1. The number after the ‘^’ can’t be minus.
2. Can’t deal with the power calculation for numbers like ‘3^2’

Step1: build a class polynomial, set the default value of the instance ‘expression’. Ask the user to input an expression in proper format.

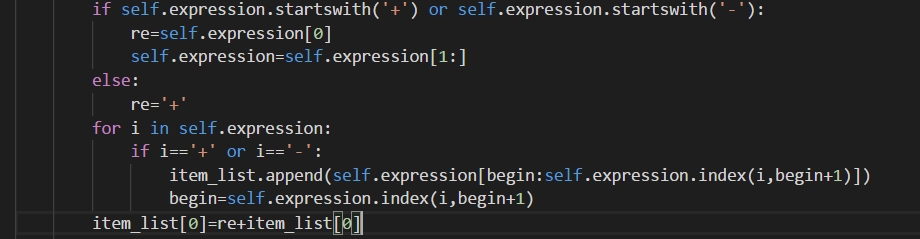


Step2: check what’s the variable of the expression. Firstly we build a list containing all the letters of both upper and lower. Then if there is a letter in the expression is also in the list, it is the variable. If not, we set a default variable x.





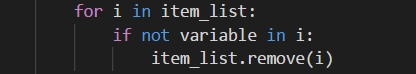
Step3: slicing the whole expression into several pieces by ‘+’ or ‘-‘. My method is special, because it can contain the sign together with the item. It’s easier for us to deal with the calculation later.



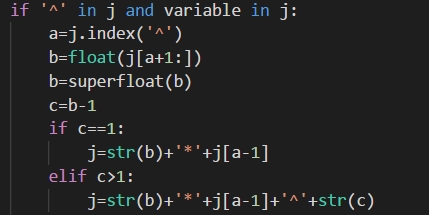
Step4: simplify each item in the item list basically. For example, if the item is +3\*x^2\*2\*x^3\*2, the program can simplify it into +3\*2\*2\*x^5:



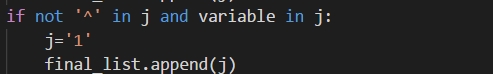
Step5: remove those items that don’t need to be considered:



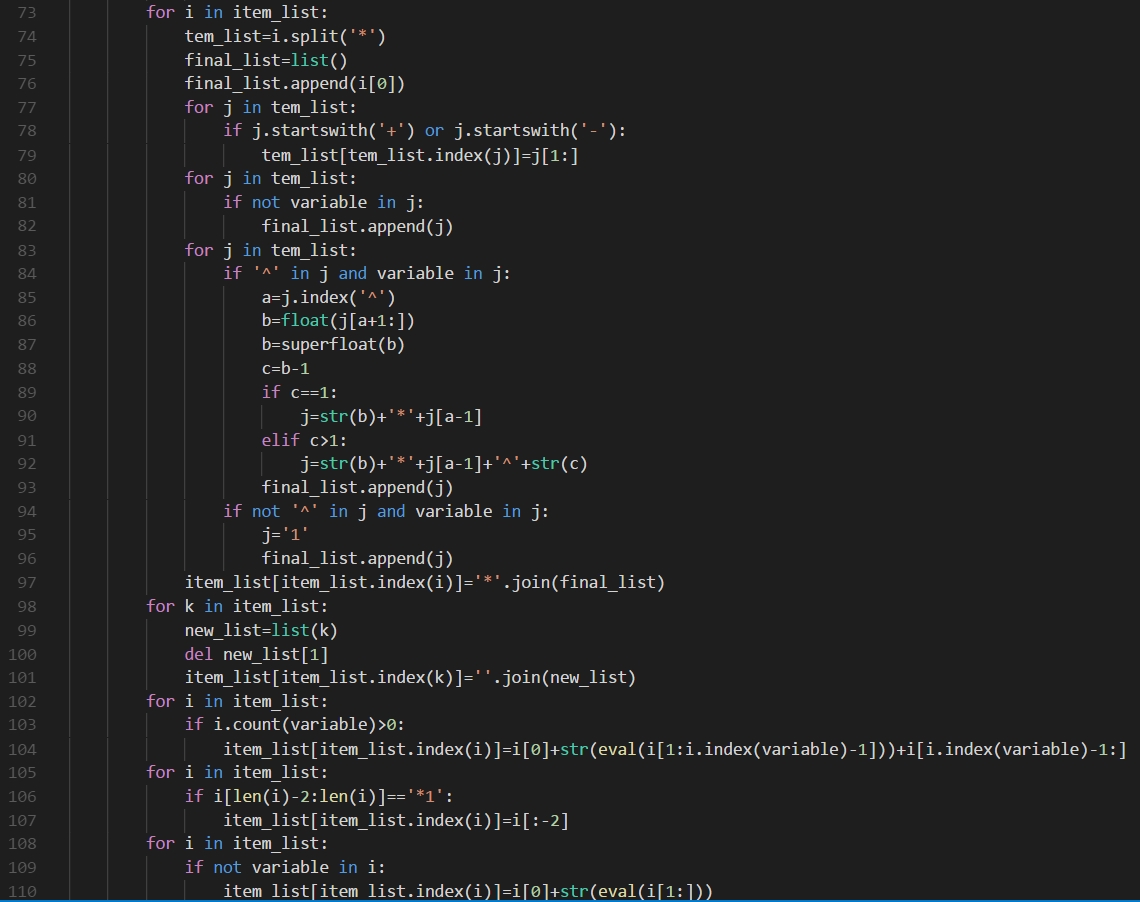
Step6: derivate for each item. The method in details: 1. split the small item into small pieces by ‘\*’. 2. Find the piece containing the variable. 3. Try to find ‘^’. 4. If the item containing both variable and ‘^’:



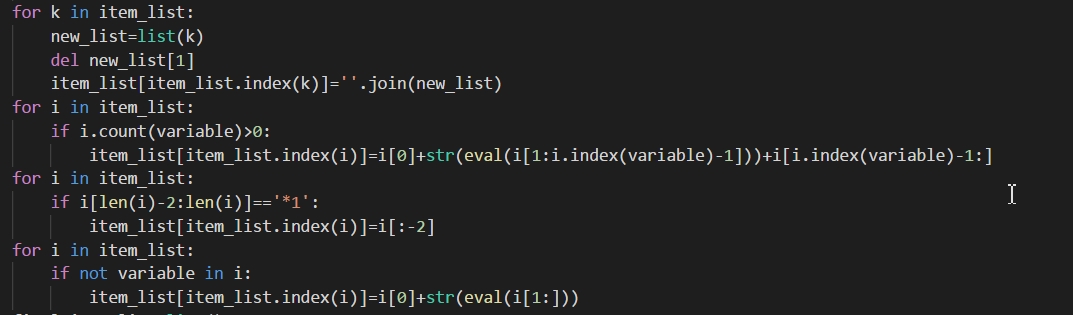
5.if the item only contains variable:



6. put the final list together, which is the new item of the item list (after derivation). The next picture is the whole process:



Step7: optimize the format of each item. For example, it will transform +3\*2\*4\*x^6 into +24\*x^6:



Step8: combine like terms. For example, it can combine x^2+3\*x^2-2\*x^2 into 2\*x^2.

final\_item\_list=list()

final\_item\_list.append(0)

for i in item\_list:

if variable in i:

if not i[i.index(variable):] in final\_item\_list:

final\_item\_list.append(i[i.index(variable):])

if not variable in i:

if i.startswith('-'):

final\_item\_list[0]-=superfloat(float(i[1:]))

else:

final\_item\_list[0]+=superfloat(float(i[1:]))

if final\_item\_list[0]==0:

del final\_item\_list[0]

else:

final\_item\_list[0]=str(final\_item\_list[0])

for i in final\_item\_list:

sum=0

if '^' in i and variable in i:

for j in item\_list:

if i in j:

if j[0]=='-':

if j[1]==variable:

sum=sum-1

else:

sum=sum-float(j[1:j.index('\*')])

if j[0]=='+':

if j[1]==variable:

sum=sum+1

else:

sum=sum+float(j[1:j.index('\*')])

if not '^' in i and variable in i:

for j in item\_list:

if not '^' in j and variable in j:

if j[0]=='-':

if j[1]==variable:

sum=sum-1

else:

sum=sum-float(j[1:j.index('\*')])

if j[0]=='+':

if j[1]==variable:

sum=sum+1

else:

sum=sum+float(j[1:j.index('\*')])

sum=str(superfloat(sum))

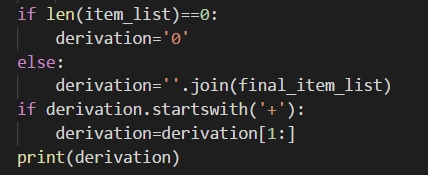
if not (sum.startswith('+') or sum.startswith('-')):

sum='+'+sum

if not (final\_item\_list.index(i)==0 and not variable in i):

final\_item\_list[final\_item\_list.index(i)]=sum+'\*'+final\_item\_list[final\_item\_list.index(i)]

Step9: output the result. Combine each item. If there’s no item in the item list, the derivation is 0.



Step10: def main() function

def main():

a=polynomial()

a.new\_expression()

a.derivation()

main()