from math import sin,cos,tan,radians

while 1:

try:

x = input()

x = x.split()

poped = []

if "ans" in x:

for i in range(len(x)):

if "ans"==x[i]:

x[i]=ans

if "sin" in x or "cos" in x or "tan" in x:

for i in range(len(x)):

if x[i]=="sin":

x[i]=sin(radians(float(x[i+1])))

poped.append(i+1)

elif x[i]=="cos":

x[i]=cos(radians(float(x[i+1])))

poped.append(i+1)

elif x[i]=="tan":

x[i]=tan(radians(float(x[i+1])))

poped.append(i+1)

for i in range(len(poped)-1,-1,-1):

x.pop(poped[i])

poped = []

if "\*\*" in x:

for i in range(len(x)):

if x[i]=="\*\*":

x[i]=float(x[i-1])\*\*float(x[i+1])

poped.append(i-1)

poped.append(i+1)

for i in range(len(poped)-1,-1,-1):

x.pop(poped[i])

poped = []

if "\*" in x or "/" in x:

for i in range(len(x)):

if x[i]=="\*":

x[i]=float(x[i-1])\*float(x[i+1])

poped.append(i-1)

poped.append(i+1)

if x[i]=="/":

x[i]=float(x[i-1])/float(x[i+1])

poped.append(i-1)

poped.append(i+1)

for i in range(len(poped)-1,-1,-1):

x.pop(poped[i])

poped = []

x = list(map(float,x))

print(sum(x))

except:

print(“Syntax Error”)

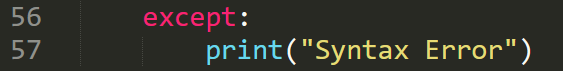
Illustration:

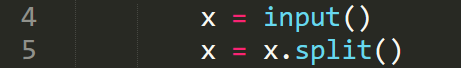
This idea behind that code is to create a calculator tree like, but as it would be unrewarding to replicate the python’s original mathematical calculator so I decided to build that tree with a linear code, that code spends for each existing operation O(n) n is the number of numbers plus the operations



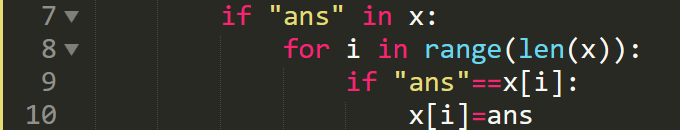
In this line of code I imported the main triangular functions and imported degree to radian convertor to use those functions

As the calculator should keep running, some measures must be taken like a loop and error detector

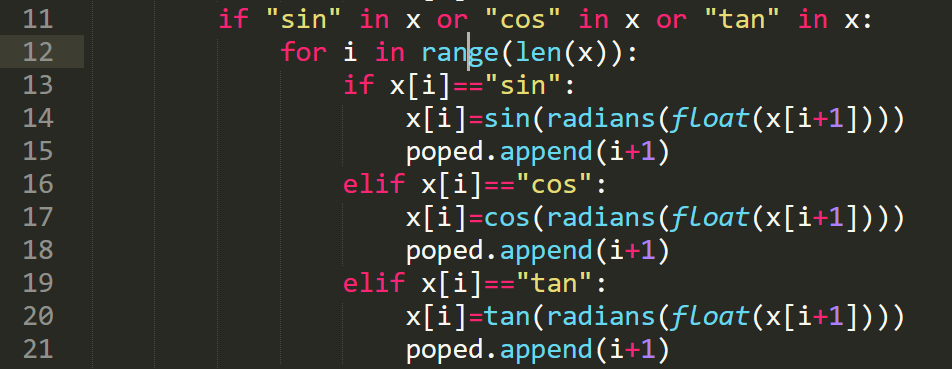
 

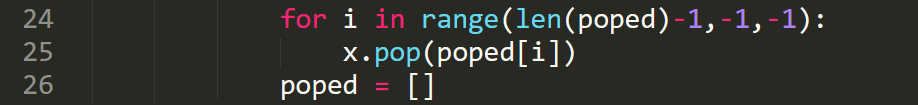


To import the mathematical operations and number



Those lines to make a use of the last calculated line and will be illustrated later

to reduce calculate the triangular functions and save them



Later, the numbers inserted in the operation should be deleted and only the outcome should be remembered, to solve list range problem, I used backword list to delete the no-use as a stack

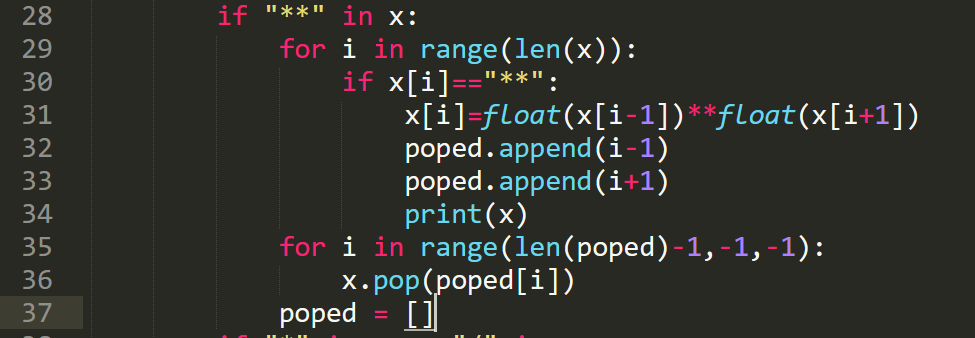
That is the main convention of my calculator, It does operations, save their numbers, then delete calculated number

The priority queue of the operations as follows:

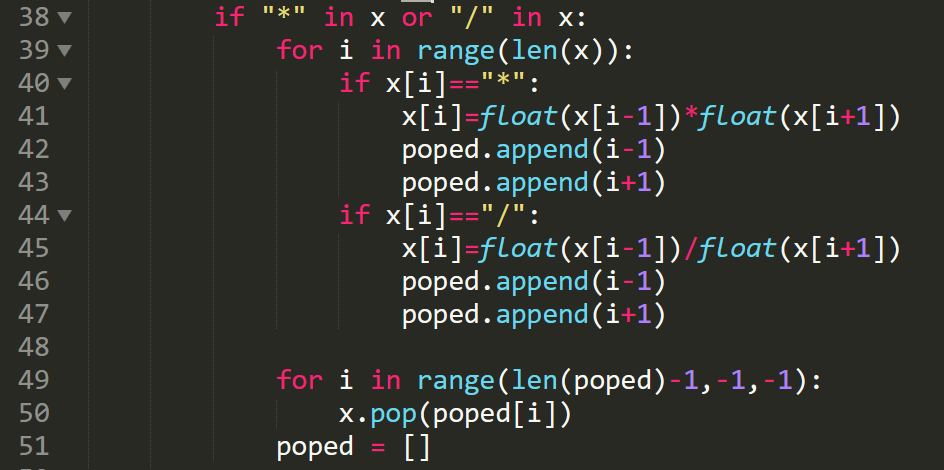
1. Using the last answer
2. Triangular functions
3. Power function
4. Multiplication and division
5. Summing and subtracting

Continueing my code illustration:

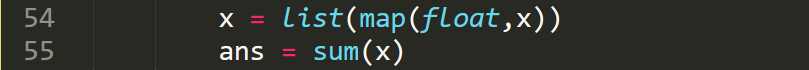
Power function:



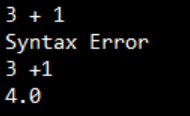
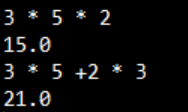
Multiplication and division:



And lastly summing elements and save the answer:

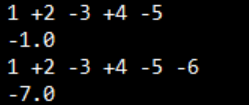


The simplicity of the code resulted in some deficiencies

1. The triangular functions and answer must be used with a co-efficient 
2. In any adding or subtracting operation: the sign must be close to the number 
3. No more than one multiplication or one division operation after another 
4. The operations must be seperated with space except the adding or subtracting

Results:

1. Adding and subtracting works efficient



1. Multiplication and division works efficient and compatible with adding and multiplication 
2. Power function works very efficient with adding and multiplication 
3. Triangular functions are efficient  
4. Adding the answer to a new operation works well

