```
import csv
import math
import sys
import re
def SigFigFormatter(filename):

    with open(filename,'r',newline='') as csvIn, open(filename[:-4]+"Sigfig.csv",'w',n
ewline='') as csvOut:
        reader = csv.DictReader(csvIn)
        writer_fieldnames = []
        sigfig_names = []
        nonfig_names = []
        for fname in reader.fieldnames:
            if len(fname) >= 4 and fname[-4:] == "_BST":
                sigfig_names.append(fname[0:-4])
                writer_fieldnames.append(fname[0:-4])
            elif len(fname) >= 4 and fname[-4:] == "_ERR":
                pass
            else:
                nonfig_names.append(fname)
                writer_fieldnames.append(fname)
        writer = csv.DictWriter(csvOut,fieldnames=writer_fieldnames)
        writer.writeheader()
        for row in reader:
            rowout = {}
            for col in nonfig_names:
                rowout[col] = row[col]
            for col in sigfig_names:
                rowout[col] = BstPlusMinusErr(row[col+"_BST"],row[col+"_ERR"])
            writer.writerow(rowout)

def BstPlusMinusErr(bst,err):
    #Work in Progress. Works for errors less 1, less so for others.
    bst = float(bst)
    err = float(err)
    if err == 0:
        return str(bst)
    else:
        errBit = math.floor(math.log10(err))
    errPl = errBit
    if err*10**(-1*errBit) < 2 :
        errOut = formatgButConfigurable(err,2)
    else :
        errOut = formatgButConfigurable(err,1)
    return truncateAtPlace(bst,errBit) + "$\pm$" + errOut

def formatgButConfigurable(num2fmt,sigfigs):
    if float(num2fmt) == 0 :
        return "0"
    sigfigs = int(sigfigs)
    fmt = "{:."+str(sigfigs)+"g}"
    gfmted = fmt.format(float(num2fmt))
    longstring = "{:.10f}".format(float(gfmted))
    pos = re.search('[123456789]',longstring).start()
    decpos = re.search('\.',longstring).start()
    zeros = decpos-pos-sigfigs
    if float(num2fmt) < 1 :
        zeros = 0
    return longstring[0:pos+sigfigs + ( -1*zeros if zeros < 0 else 0)] + '0'*(zeros if
 zeros > 0 else 0)
def truncateAtPlace(numIn,pl):
    numIn = round(numIn,-1*pl)
    if pl > 0 :
        numIn = int(numIn)
    elif pl < 0 :
```

```
        fmt = "{:." + str(int(-1*pl)) +"f}"
        numIn = fmt.format(float(numIn))
    return str(numIn)

if __name__ == "__main__":
    if len(sys.argv) > 1:
        SigFigFormatter(sys.argv[1])
    else :
        SigFigFormatter(input())
```