```
# -*- coding: utf-8 -*-
from __future__ import division
import csv
from optparse import OptionParser
import numpy as np
import os
import copy
import pandas as pd

scores = {
'' : 0,
'Prefer not to answer': 0,
'Strongly Agree': 5,
'Agree': 4,
'Neither Agree/Disagree': 3,
'Disagree': 2,
'Strongly Disagree':1}

reverse_scores = {
'': 0,
'Prefer not to answer': 0,
'Strongly Agree': 1,
'Agree': 2,
'Neither Agree/Disagree': 3,
'Disagree': 4,
'Strongly Disagree':5}

overclaim_score = {
'':0,
'0 - Never heard of it': 0,
'1' : 1,
'2 -  Somewhat Familiar': 1,
'3': 1,
'4 - Very Familiar':1}

binary_score = {
'':0,
'True':1,
'False':0,
'TRUE': 1,
'FALSE': 0}

confidence_score = {
'':0,
'0%' : 0.0,
'10%' : 0.1,
'20%' : 0.2,
'30%' : 0.3,
'40%' : 0.4,
'50%' : 0.5,
```

```
'60%' : 0.6,
'70%' : 0.7,
'80%' : 0.8,
'90%' : 0.9,
'100%' : 1.0
}

estimation_scores = {
'' : 0,
'Extremely Easy': 1,
'Somewhat Easy': 2,
'Average': 3,
'Somewhat Difficult': 4,
'Extremely Difficult':5}

confidence_score_template = {
'0.0':0,'0.1':0,'0.2':0,'0.3':0,'0.4':0,'0.5':0,'0.6':0,'0.7':0,'0.8':
0,'0.9':0,'1.0':0,}

#print (type(scores))

def find_subject(subject_start_question, first_subject_headers,
second_subject_headers,
                 third_subject_headers, first_subject, second_subject,
third_subject):

    subject_data = []
    subject_headers = []

    if first_subject_headers[0] == subject_start_question:
        subject_data = first_subject
        subject_headers = first_subject_headers
    if second_subject_headers[0] == subject_start_question:
        subject_data = second_subject
        subject_headers = second_subject_headers
    if third_subject_headers[0] == subject_start_question:
        subject_data = third_subject
        subject_headers = third_subject_headers

    return subject_data, subject_headers

def write_data(data_file, data, final_answers, user_ids):
    num_records = 0
    for row in data:
        if row[0] in user_ids:
            print('Respondent ID', row[0] ,'already recorded')
        else:
            print('Processing Respondent ID', row[0])
            for column in row:
```

```
                column = column.replace('\"','')
                data_file.write('\"'+column+'\",')
            data_file.write(final_answers[num_records])
            num_records += 1
            data_file.write('\n')



def run(filepath):


    #filepath = '.csv'
    data = []
    user_ids = []
    with open(filepath) as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=',')
        line_count = 0
        headers = next(csv_reader)
        second_headers = next(csv_reader)
        #for header in headers:
            #print (header)
        for row in csv_reader:

            data.append(row)


    #print('!!!!!!!!!!!!!!!!!!!!',user_ids)
    num_columns = len(data[0])

    split_file = filepath.split('.')
    subjects = split_file[0].split('_')

    subject_start_questions = ['Out of twenty-nine Logic questions,
how many questions do you estimate you will answer correctly?',
                               'Out of twenty-five English grammar
questions, how many questions do you estimate you will answer
correctly?',
                               'Out of twenty-seven Nuclear Weapon
Science & Technology questions, how many questions do you estimate you
will answer correctly?']


    logic_index = headers.index(subject_start_questions[0])
    english_index = headers.index(subject_start_questions[1])
    nw_index = headers.index(subject_start_questions[2])

    subject_indeces = [nw_index,english_index,logic_index]
    sorted_subject_indeces = sorted(subject_indeces)

    data = np.array(data)
```

```
    preliminary_subjects = data[:,0:sorted_subject_indeces[0]]
    first_subject =
data[:,sorted_subject_indeces[0]:sorted_subject_indeces[1]]
    second_subject =
data[:,sorted_subject_indeces[1]:sorted_subject_indeces[2]]
    third_subject = data[:,sorted_subject_indeces[2]:num_columns-1]

    final_answers = data[:,-1]
    #print(final_answers)

    headers = np.array(headers)
    preliminary_headers = headers[0:sorted_subject_indeces[0]]
    preliminary_second_headers =
second_headers[0:sorted_subject_indeces[0]]
    first_subject_headers =
headers[sorted_subject_indeces[0]:sorted_subject_indeces[1]]
    second_subject_headers =
headers[sorted_subject_indeces[1]:sorted_subject_indeces[2]]
    third_subject_headers =
headers[sorted_subject_indeces[2]:num_columns-1]
    #print(headers.shape)
    #print(subject_indeces)



    logic_data, logic_headers =
find_subject(subject_start_questions[0], first_subject_headers,
second_subject_headers,
                third_subject_headers, first_subject, second_subject,
third_subject)
    english_data, english_headers =
find_subject(subject_start_questions[1], first_subject_headers,
second_subject_headers,
                third_subject_headers, first_subject, second_subject,
third_subject)
    nw_data, nw_headers = find_subject(subject_start_questions[2],
first_subject_headers, second_subject_headers,
                third_subject_headers, first_subject, second_subject,
third_subject)


    master_filename = 'data_master.csv'
    summary_filename = 'data_summary.csv'
    score_filename = 'data_scores_master.csv'
    confidence_filename = 'data_confidence.csv'

    final_header = '\"' + headers[num_columns-1] + '\"'

    if not os.path.isfile(confidence_filename):
```

```
    confidence_file = open(confidence_filename, "w")
    confidence_file.write('Respondent ID,')
    confidence_file.write('When Logic Score = 0,')
    for i in range(10):
        confidence_file.write(',')

    confidence_file.write('When Logic Score = 1,')
    for i in range(10):
        confidence_file.write(',')

    confidence_file.write('When Grammar Score = 0,')
    for i in range(10):
        confidence_file.write(',')

    confidence_file.write('When Grammar Score = 1,')
    for i in range(10):
        confidence_file.write(',')

    confidence_file.write('When NW Score = 0,')
    for i in range(10):
        confidence_file.write(',')

    confidence_file.write('When NW Score = 1,')
    for i in range(10):

        confidence_file.write(',')

    confidence_file.write('\n')

    confidence_file.write(',')
    for i in range(6):
        confidence_file.write('0.0,')
        confidence_file.write('0.1,')
        confidence_file.write('0.2,')
        confidence_file.write('0.3,')
        confidence_file.write('0.4,')
        confidence_file.write('0.5,')
        confidence_file.write('0.6,')
        confidence_file.write('0.7,')
        confidence_file.write('0.8,')
        confidence_file.write('0.9,')
        confidence_file.write('1.0,')

    confidence_file.write('\n')
else:
    confidence_file = open(confidence_filename, "r")
    lines=confidence_file.readlines()
    lines=lines[:-1]
    confidence_file.close()
```

```
    confidence_file = open(confidence_filename, "w")
    #cWriter = csv.writer(confidence_file, delimiter=',')
    for line in lines:
        confidence_file.write(line)
    confidence_file.close()
    confidence_file = open(confidence_filename, "a+")




if not os.path.isfile(master_filename):
    master_file = open(master_filename, "w")
    for header in preliminary_headers:
        header = header.replace('\"','')
        master_file.write('\"'+header+'\",')
    for header in logic_headers:
        header = header.replace('\"','')
        master_file.write('\"'+header+'\",')
    for header in english_headers:
        header = header.replace('\"','')
        master_file.write('\"'+header+'\",')
    for header in nw_headers:
        header = header.replace('\"','')
        master_file.write('\"'+header+'\",')
    master_file.write(final_header)

    master_file.write('\n')

    for header in preliminary_second_headers:
        master_file.write('\"'+header+'\",')
    master_file.write('\n')
else:
    master_file = open(master_filename, "a+")
    with open(master_filename) as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=',')
        headers = next(csv_reader)
        second_headers = next(csv_reader)
        #for header in headers:
            #print (header)
        for row in csv_reader:
            user_ids.append(row[0])


if not os.path.isfile(score_filename):
    score_file = open(score_filename, "w")
    for header in preliminary_headers:
        header = header.replace('\"','')
        score_file.write('\"'+header+'\",')
```

```
    for header in logic_headers:
        header = header.replace('\"','')
        score_file.write('\"'+header+'\",')
    for header in english_headers:
        header = header.replace('\"','')
        score_file.write('\"'+header+'\",')
    for header in nw_headers:
        header = header.replace('\"','')
        score_file.write('\"'+header+'\",')
    score_file.write(final_header)
    score_file.write('\n')
    for header in preliminary_second_headers:
        score_file.write('\"'+header+'\",')
    score_file.write('\n')
else:
    score_file = open(score_filename, "a+")

if not os.path.isfile(summary_filename):
    summary_file = open(summary_filename, "w")
    summary_file.write(',')
    summary_file.write(',')
    summary_file.write('Locus of Control,')
    summary_file.write(',')
    summary_file.write(',')
    summary_file.write('Big 5 Personality,')
    summary_file.write(',')
    summary_file.write(',')
    summary_file.write(',')
    summary_file.write(',')
    summary_file.write('Overclaiming,')
    summary_file.write(',')
    summary_file.write(',')
    summary_file.write(',')
    summary_file.write(',')
    summary_file.write(',')
    summary_file.write('Decision Style,')
    summary_file.write(',')
    summary_file.write(',')
    summary_file.write(',')
    summary_file.write(',')
    summary_file.write(',')
    summary_file.write(',')
    summary_file.write(',')
    summary_file.write(',')
    summary_file.write(',')
    summary_file.write('Social Desiribility,')
    summary_file.write('Metacognitive Awareness Inventory,')
    summary_file.write(',')
    summary_file.write(',')
    summary_file.write(',')
    summary_file.write(',')
```

```
summary_file.write(',')
summary_file.write(',')
summary_file.write(',')
summary_file.write('Logic Task,')
summary_file.write(',')
summary_file.write(',')
summary_file.write(',')
summary_file.write(',')
summary_file.write(',')
summary_file.write(',')
summary_file.write(',')
summary_file.write(',')
summary_file.write(',')
summary_file.write(',')
summary_file.write('Grammar Task,')
summary_file.write(',')
summary_file.write(',')
summary_file.write(',')
summary_file.write(',')
summary_file.write(',')
summary_file.write(',')
summary_file.write(',')
summary_file.write(',')
summary_file.write(',')
summary_file.write(',')
summary_file.write('NW Task,')
summary_file.write(',')
summary_file.write(',')
summary_file.write(',')
summary_file.write(',')
summary_file.write(',')
summary_file.write(',')
summary_file.write(',')
summary_file.write(',')
summary_file.write(',')
summary_file.write(',')
summary_file.write(',')
summary_file.write('Global Summary,')
summary_file.write('\n')


summary_file.write('Respondent ID,')
summary_file.write('Date,')
summary_file.write('Internal,')
summary_file.write('Chance,')
summary_file.write('Powerful Others,')
summary_file.write('Extraversion,')
summary_file.write('Agreeableness,')
```

```
        summary_file.write('Conscientiousness,')
        summary_file.write('Neuroticism,')
        summary_file.write('Openness,')
        summary_file.write('Accuracy,')
        summary_file.write('Bias,')
        summary_file.write('Accuracy Nuclear,')
        summary_file.write('Bias Nuclear,')
        summary_file.write('Accuracy with Nuclear,')
        summary_file.write('Bias with Nuclear,')
        summary_file.write('Avoidant,')
        summary_file.write('Dependent,')
        summary_file.write('Confident,')
        summary_file.write('Anxious,')
        summary_file.write('Vigilant,')
        summary_file.write('Spontaneous,')
        summary_file.write('Intuitive,')
        summary_file.write('Respected,')
        summary_file.write('Brooding,')
        summary_file.write('Social Desiribility,')
        summary_file.write('Declarative,')
        summary_file.write('Procedural,')
        summary_file.write('Conditional,')
        summary_file.write('Planning,')
        summary_file.write('Information Management,')
        summary_file.write('Comprehension Monitoring,')
        summary_file.write('Debugging Strategies,')
        summary_file.write('Evaluation,')
        summary_file.write('Pre-task Logic Correct Answers Estimate,')
        summary_file.write('Pre-task Logic Accuracy Esimate,')
        summary_file.write('Pre-task Logic Sandian Comparison
Estimate,')
        summary_file.write('Pre-task Logic Difficulty Estimate,')
        summary_file.write('Correct Logic Answers,')
        summary_file.write('Logic Accuracy,')
        summary_file.write('Logic Confidence Avg,')
        summary_file.write('Post-task Logic Correct Answers Esimate,')
        summary_file.write('Post-task Logic Accuracy Esimate,')
        summary_file.write('Post-task Logic Sandian Comparison
Estimate,')
        summary_file.write('Post-task Logic Difficulty Estimate,')
        summary_file.write('Post-task Logic Overestimation Score,')
        summary_file.write('Pre-task Correct Grammar Answers
Estimate,')
        summary_file.write('Pre-task Grammar Accuracy Esimate,')
        summary_file.write('Pre-task Grammar Sandian Comparison
Estimate,')
        summary_file.write('Pre-task Grammar Difficulty Estimate,')
        summary_file.write('Correct Grammar Answers,')
        summary_file.write('Grammar Accuracy,')
        summary_file.write('Grammar Confidence Avg,')
```

```
        summary_file.write('Post-task Correct Grammar Answers
Esimate,')
        summary_file.write('Post-task Grammar Accuracy Esimate,')
        summary_file.write('Post-task Grammar Sandian Comparison
Estimate,')
        summary_file.write('Post-task Grammar Difficulty Estimate,')
        summary_file.write('Post-task Grammar Overestimation Score,')
        summary_file.write('Pre-task NW Correct Answers Esimate,')
        summary_file.write('Pre-task NW Accuracy Esimate,')
        summary_file.write('Pre-task NW Sandian Comparison Estimate,')
        summary_file.write('Pre-task NW Difficulty Estimate,')
        summary_file.write('NW Correct Answers,')
        summary_file.write('NW Accuracy,')
        summary_file.write('NW Confidence Avg,')
        summary_file.write('Post-task NW Correct Answers Esimate,')
        summary_file.write('Post-task NW Accuracy Esimate,')
        summary_file.write('Post-task NW Sandian Comparison
Estimate,')
        summary_file.write('Post-task NW Difficulty Estimate,')
        summary_file.write('Post-task NW Overestimation Score,')

        summary_file.write('Estimated Best Subject,')
        summary_file.write('Actual Best Subject,')

        summary_file.write('Overall Correct Answers,')
        summary_file.write('Overall Correct Answers Estimate,')
        summary_file.write('Overall Overestimation Score,')
        summary_file.write('\n')
    #print('\"'+headers[num_columns-1]+'\"')
    else:
        summary_file = open(summary_filename, "a+")

    data =
np.hstack((preliminary_subjects,logic_data,english_data,nw_data))
    headers =
np.hstack((preliminary_headers,logic_headers,english_headers,nw_header
s))

    write_data(master_file,data,final_answers, user_ids)
    #print (data[0])

    num_records = 0
    for row in data:

        if row[0] in user_ids:
            pass
        else:
            #print (row)
            user_id = row[0]
            summary_file.write(user_id+',')
```

```
            confidence_file.write(user_id+',')
            #print(user_id)
            summary_file.write(row[2]+',')
            for i in range(38):
                column = row[i]
                score_file.write('\"'+column+'\",')

            #Locus of Control
            internal = scores[row[38]]+scores[row[39]]+scores[row[40]]
            chance = scores[row[41]]+scores[row[42]]+scores[row[43]]
            powerful_others = scores[row[44]]+scores[row[45]]
+scores[row[46]]

            summary_file.write(str(internal)+','+str(chance)
+','+str(powerful_others)+',')



            for i in range(38,47):
                score = scores[row[i]]
                score_file.write('\"'+str(score)+'\",')

    #************Big 5 personality******************#
            personality_question_1 = scores[row[47]]
            score_file.write('\"'+str(personality_question_1)+'\",')
            personality_question_2 = reverse_scores[row[48]]
            score_file.write('\"'+str(personality_question_2)+'\",')
            personality_question_3 = scores[row[49]]
            score_file.write('\"'+str(personality_question_3)+'\",')
            personality_question_4 = scores[row[50]]
            score_file.write('\"'+str(personality_question_4)+'\",')
            personality_question_5 = scores[row[51]]
            score_file.write('\"'+str(personality_question_5)+'\",')
            personality_question_6 = reverse_scores[row[52]]
            score_file.write('\"'+str(personality_question_6)+'\",')
            personality_question_7 = scores[row[53]]
            score_file.write('\"'+str(personality_question_7)+'\",')
            personality_question_8 = reverse_scores[row[54]]
            score_file.write('\"'+str(personality_question_8)+'\",')
            personality_question_9 = reverse_scores[row[55]]
            score_file.write('\"'+str(personality_question_9)+'\",')
            personality_question_10 = scores[row[56]]
            score_file.write('\"'+str(personality_question_10)+'\",')
            personality_question_11 = scores[row[57]]
            score_file.write('\"'+str(personality_question_11)+'\",')
            personality_question_12 = reverse_scores[row[58]]
            score_file.write('\"'+str(personality_question_12)+'\",')
            personality_question_13 = scores[row[59]]
            score_file.write('\"'+str(personality_question_13)+'\",')
            personality_question_14 = scores[row[60]]
```

```
score_file.write('\"'+str(personality_question_14)+'\",')
personality_question_15 = scores[row[61]]
score_file.write('\"'+str(personality_question_15)+'\",')
personality_question_16 = scores[row[62]]
score_file.write('\"'+str(personality_question_16)+'\",')
personality_question_17 = scores[row[63]]
score_file.write('\"'+str(personality_question_17)+'\",')
personality_question_18 = reverse_scores[row[64]]
score_file.write('\"'+str(personality_question_18)+'\",')
personality_question_19 = scores[row[65]]
score_file.write('\"'+str(personality_question_19)+'\",')
personality_question_20 = scores[row[66]]
score_file.write('\"'+str(personality_question_20)+'\",')
personality_question_21 = reverse_scores[row[67]]
score_file.write('\"'+str(personality_question_21)+'\",')
personality_question_22 = scores[row[68]]
score_file.write('\"'+str(personality_question_22)+'\",')
personality_question_23 = reverse_scores[row[69]]
score_file.write('\"'+str(personality_question_23)+'\",')
personality_question_24 = reverse_scores[row[70]]
score_file.write('\"'+str(personality_question_24)+'\",')
personality_question_25 = scores[row[71]]
score_file.write('\"'+str(personality_question_25)+'\",')
personality_question_26 = scores[row[72]]
score_file.write('\"'+str(personality_question_26)+'\",')
personality_question_27 = reverse_scores[row[73]]
score_file.write('\"'+str(personality_question_27)+'\",')
personality_question_28 = scores[row[74]]
score_file.write('\"'+str(personality_question_28)+'\",')
personality_question_29 = scores[row[75]]
score_file.write('\"'+str(personality_question_29)+'\",')
personality_question_30 = scores[row[76]]
score_file.write('\"'+str(personality_question_30)+'\",')
personality_question_31 = reverse_scores[row[77]]
score_file.write('\"'+str(personality_question_31)+'\",')
personality_question_32 = scores[row[78]]
score_file.write('\"'+str(personality_question_32)+'\",')
personality_question_33 = scores[row[79]]
score_file.write('\"'+str(personality_question_33)+'\",')
personality_question_34 = reverse_scores[row[80]]
score_file.write('\"'+str(personality_question_34)+'\",')
personality_question_35 = reverse_scores[row[81]]
score_file.write('\"'+str(personality_question_35)+'\",')
personality_question_36 = scores[row[82]]
score_file.write('\"'+str(personality_question_36)+'\",')
personality_question_37 = reverse_scores[row[83]]
score_file.write('\"'+str(personality_question_37)+'\",')
personality_question_38 = scores[row[84]]
score_file.write('\"'+str(personality_question_38)+'\",')
personality_question_39 = scores[row[85]]
```

```
score_file.write('\"'+str(personality_question_39)+'\",')
personality_question_40 = scores[row[86]]
score_file.write('\"'+str(personality_question_40)+'\",')
personality_question_41 = reverse_scores[row[87]]
score_file.write('\"'+str(personality_question_41)+'\",')
personality_question_42 = scores[row[88]]
score_file.write('\"'+str(personality_question_42)+'\",')
personality_question_43 = reverse_scores[row[89]]
score_file.write('\"'+str(personality_question_43)+'\",')
personality_question_44 = scores[row[90]]
score_file.write('\"'+str(personality_question_44)+'\",')

extraversion = (personality_question_1 +
                personality_question_6 +
                personality_question_11 +
                personality_question_16 +
                personality_question_21 +
                personality_question_26 +
                personality_question_31 +
                personality_question_36 )

agreeableness = (personality_question_2 +
                personality_question_7 +
                personality_question_12 +
                personality_question_17 +
                personality_question_22 +
                personality_question_27 +
                personality_question_32 +
                personality_question_37 +
                personality_question_42 )

conscientiousness = (personality_question_3 +
                    personality_question_8 +
                    personality_question_13 +
                    personality_question_18 +
                    personality_question_23 +
                    personality_question_28 +
                    personality_question_33 +
                    personality_question_38 +
                    personality_question_43 )

neuroticism = (personality_question_4 +
            personality_question_9 +
            personality_question_14 +
            personality_question_19 +
            personality_question_24 +
            personality_question_29 +
            personality_question_34 +
            personality_question_39 )
```

```
            openness = (personality_question_5 +
                        personality_question_10 +
                        personality_question_15 +
                        personality_question_20 +
                        personality_question_25 +
                        personality_question_30 +
                        personality_question_35 +
                        personality_question_40 +
                        personality_question_41 +
                        personality_question_44 )

            summary_file.write(str(extraversion)
+','+str(agreeableness)+','+
                                    str(conscientiousness)
+','+str(neuroticism)+','+str(openness)+',')
    #************End Big 5**********************#

    #***********Overclaiming******************#

            hits = 0
            false_alarms = 0
            nuclear_hits = 0
            nuclear_false_alarms = 0

            num_words = 0
            num_nuclear_words = 0
            fake_words = ['sentence stigma','pseudo-verb','shunt-
word', 'cholarine', 'ultra-lipid', 'plates of parallax']
            debatable_words = ['Gas Priming', 'Voltage inversion',
'Neutron focusing']

            for column in range(91,136):
                word = headers[column].strip()

                answer = overclaim_score[row[column]]
                score_file.write('\"'+str(answer)+'\",')
                if column < 121:
                    num_words += 1
                    if word in fake_words and answer == 1:
                        false_alarms += 1
                        #print(word)
                    if not word in fake_words and answer == 1:
                        hits += 1



                else:
                    num_nuclear_words += 1
                    if word in debatable_words and answer == 1:
                        nuclear_false_alarms += 1
```

```
                    if not word in debatable_words and answer == 1:
                        nuclear_hits += 1
                        #print(word)

            #print('hits',hits)
            #print('flase_alarms',false_alarms)
            num_fake_words = len(fake_words)
            num_real_words = num_words - num_fake_words
            #num_real_non_nuclear_words = num_real_words -
num_fake_words
            accuracy = ((hits)/(num_real_words)) - ((false_alarms)/
(num_fake_words))
            bias = (hits/num_real_words) + (false_alarms/
num_fake_words)

            summary_file.write(str(accuracy)+','+str(bias)+',')

            num_fake_nuclear_words = len(debatable_words)
            num_real_nuclear_words = num_nuclear_words -
num_fake_nuclear_words

            accuracy_nuclear = ((nuclear_hits)/
(num_real_nuclear_words)) - ((nuclear_false_alarms)/
(num_fake_nuclear_words))
            bias_nuclear = (nuclear_hits/num_real_nuclear_words) +
(nuclear_false_alarms/num_fake_nuclear_words)
            summary_file.write(str(accuracy_nuclear)
+','+str(bias_nuclear)+',')

            accuracy_with_nuclear = ((nuclear_hits+hits)/
(num_real_nuclear_words+num_real_words)) -
((nuclear_false_alarms+false_alarms)/
(num_fake_nuclear_words+num_fake_words))
            bias_with_nuclear = ((hits+nuclear_hits)/num_real_words +
num_nuclear_words) + ((false_alarms + nuclear_false_alarms)/
num_fake_nuclear_words+num_fake_words)

            summary_file.write(str(accuracy_with_nuclear)
+','+str(bias_with_nuclear)+',')
    #************ End Overclaiming ********************#
            score_file.write('\"'+str(row[136])+'\",')
    #********** Decision Style ***********************#



            avoidant = 0
            for column in range(137,142):
                #print(headers[column])
                avoidant += scores[row[column]]
```

```
                score_file.write('\"'+str(scores[row[column]])+'\",')

            dependent = 0
            for column in range(142,148):
                #print(headers[column])
                if column == 142:
                    dependent += reverse_scores[row[column]]

score_file.write('\"'+str(reverse_scores[row[column]])+'\",')
                else:
                    dependent += scores[row[column]]
                    score_file.write('\"'+str(scores[row[column]])
+'\",')

            confident = 0
            for column in range(148,153):
                #print(headers[column])
                if column == 149 or column == 151:
                    confident += reverse_scores[row[column]]

score_file.write('\"'+str(reverse_scores[row[column]])+'\",')
                else:
                    confident += scores[row[column]]
                    score_file.write('\"'+str(scores[row[column]])
+'\",')
            #print(confident)

            anxious = 0
            for column in range(153,158):
                #print(headers[column])
                anxious += scores[row[column]]
                score_file.write('\"'+str(scores[row[column]])+'\",')
            #print(anxious)

            vigilant = 0
            for column in range(158,164):
                #print(headers[column])
                vigilant += scores[row[column]]
                score_file.write('\"'+str(scores[row[column]])+'\",')
            #print(vigilant)

            spontaneous = 0
            for column in range(164,168):
                #print(headers[column])
                spontaneous += scores[row[column]]
                score_file.write('\"'+str(scores[row[column]])+'\",')
            #print(spontaneous)

            intuitive = 0
            for column in range(168,173):
```

```
            #print(headers[column])
            intuitive += scores[row[column]]
            score_file.write('\"'+str(scores[row[column]])+'\",')
        #print(intuitive)

        respected = 0
        for column in range(173,175):
            #print(headers[column])
            respected += scores[row[column]]
            score_file.write('\"'+str(scores[row[column]])+'\",')
        #print(respected)
        #
        brooding = 0
        for column in range(175,180):
            #print(headers[column])
            brooding += scores[row[column]]
            score_file.write('\"'+str(scores[row[column]])+'\",')
        #print(brooding)

        summary_file.write(str(avoidant)+',')
        summary_file.write(str(dependent)+',')
        summary_file.write(str(confident)+',')
        summary_file.write(str(anxious)+',')
        summary_file.write(str(vigilant)+',')
        summary_file.write(str(spontaneous)+',')
        summary_file.write(str(intuitive)+',')
        summary_file.write(str(respected)+',')
        summary_file.write(str(brooding)+',')


#********** End Decision Making *********#

#******* Social Desirability ************#
#
        social_desiribility_score = 0
        for column in range(180,192):
            #print(headers[column])
            social_desiribility_score += scores[row[column]]
            score_file.write('\"'+str(scores[row[column]])+'\",')
        #print(social_desiribility_score)
        summary_file.write(str(social_desiribility_score)+',')

##******** End Social Desirability ************#
#
##******** Metacognitive Awareness Inventory ************#
#
        declarative_knowledge = 0
        procedural_knowledge = 0
        conditional_knowledge = 0
        planning = 0
```

```
            information_management = 0
            comprehension_monitoring = 0
            debugging_strategies = 0
            evaluation = 0

            question_number = 1
            for column in range(192,244):

                answer = binary_score[row[column].strip()]
                score_file.write('\"'+str(answer)+'\",')

                if question_number in [5,10, 12, 16, 17, 20, 32, 46]:
                    #print(column, row[column])
                    declarative_knowledge +=
binary_score[row[column].strip()]

                if question_number in [3, 14, 27, 33]:
                    #print(headers[column])
                    #print(column, row[column])
                    procedural_knowledge +=
binary_score[row[column].strip()]

                if question_number in [15, 18, 26, 29, 35]:
                    #print(headers[column])
                    #print(column, row[column])
                    conditional_knowledge +=
binary_score[row[column].strip()]

                if question_number in [4, 6, 8, 22, 23, 42, 45]:
                    #print(headers[column])
                    #print(column, row[column])
                    planning += binary_score[row[column].strip()]

                if question_number in [9, 13, 30, 31, 37, 39, 41, 43,
47, 48]:
                    #print(headers[column])
                    #print(column, row[column])
                    information_management +=
binary_score[row[column].strip()]

                if question_number in [1, 2, 11, 21, 28, 34, 49]:
                    #print(headers[column])
                    #print(column, row[column])
                    comprehension_monitoring +=
binary_score[row[column].strip()]

                if question_number in [25, 40, 44, 51, 52]:
                    #print(headers[column])
                    #print(column, row[column])
                    debugging_strategies +=
```

```
binary_score[row[column].strip()]

                if question_number in [7, 19, 24, 36, 38, 50]:
                    #print(headers[column])
                    #print(column, row[column])
                    evaluation += binary_score[row[column].strip()]


                question_number += 1
                #print(headers[column])

            #print(evaluation)
            summary_file.write(str(declarative_knowledge)+',')
            summary_file.write(str(procedural_knowledge)+',')
            summary_file.write(str(conditional_knowledge)+',')
            summary_file.write(str(planning)+',')
            summary_file.write(str(information_management)+',')
            summary_file.write(str(comprehension_monitoring)+',')
            summary_file.write(str(debugging_strategies)+',')
            summary_file.write(str(evaluation)+',')

    ######## End Metacognitive Awareness Inventory #############
    #
    ######## Logic Task #######################################
    #
            for column in range(244, 246):
                score_file.write('\"'+str(row[column])+'\",')

            score_file.write(str(estimation_scores[row[246]])+',')
            #print(row[244])
            summary_file.write(row[244]+',')
            if row[244] in (None,''):
                summary_file.write(str(0)+',')
            else:
                summary_file.write(str(int(row[244])/(29))+',')
            summary_file.write((row[245])+',')
            summary_file.write(str(estimation_scores[row[246]])+',')

            is_confidence_question = False
            confidence_sum = 0

            logic_score = 0
            num_logic_questions = 0
            num_confidence_questions = 0
            confidence_scores_logic_correct =
copy.deepcopy(confidence_score_template)
            confidence_scores_logic_incorrect =
copy.deepcopy(confidence_score_template)

            #log_conf = 0
```

```
            for column in range(247,305):
                #print(headers[column])
                #print(is_confidence_question)
                #print(is_confidence_question)
                answer = row[column].strip()
                if is_confidence_question == True:
                    #print(user_id)
                    #log_conf+=1
                    #print(log_conf)
                    if not row[column].strip() in (None,''):
                        confidence =
confidence_score[row[column].strip()]

                        if score == 1:

confidence_scores_logic_correct[str(confidence)] =
confidence_scores_logic_correct[str(confidence)] +1
                        else:

confidence_scores_logic_incorrect[str(confidence)] =
confidence_scores_logic_incorrect[str(confidence)] +1

                score = 0
                if num_logic_questions == 0:
                    if answer == '27':
                        #print(answer)
                        logic_score += 1
                        score = 1

                if num_logic_questions == 1:
                    if answer == '$20':
                        #print(answer)
                        logic_score += 1
                        score = 1

                if num_logic_questions == 2:
                    if answer == '5 cents':
                        #print(answer)
                        logic_score += 1
                        score = 1

                if num_logic_questions == 3:
                    if answer == '4 days':
                        #print(answer)
                        logic_score += 1
                        score = 1

                if num_logic_questions == 4:
                    #print(headers[column])
                    if answer == '5 minutes':
```

```
            #print(headers[column])
            #print(answer)
            logic_score += 1
            score = 1

    if num_logic_questions == 5:
        if answer == '47 days':
            #print(answer)
            logic_score += 1
            score = 1

    if num_logic_questions == 6:
        if answer == 'has lost money':
            #print(answer)
            logic_score += 1
            score = 1

    if num_logic_questions == 7:
        if answer == 'Third':
            #print(answer)
            logic_score += 1
            score = 1

    if num_logic_questions == 8:
        if answer == '8':
            #print(answer)
            logic_score += 1
            score = 1

    if num_logic_questions == 9:
        if answer == 'Bella':
            #print(answer)
            logic_score += 1
            score = 1

    if num_logic_questions == 10:
        if answer == '0':
            #print(answer)
            logic_score += 1
            score = 1

    if num_logic_questions == 11:
        if answer == 'The Toyota':
            #print(answer)
            logic_score += 1
            score = 1

    if num_logic_questions == 12:
        if answer == 'The small hospital':
            #print(answer)
```

```
                        logic_score += 1
                        score = 1

                if num_logic_questions == 13:
                    if answer == 'Tyler':
                        #print(answer)
                        logic_score += 1
                        score = 1

                if num_logic_questions == 14:
                    if answer == '8 to 1':
                        #print(answer)
                        logic_score += 1
                        score = 1

                if num_logic_questions == 15:
                    if answer == '16 to 1':
                        #print(answer)
                        logic_score += 1
                        score = 1

                if num_logic_questions == 16:
                    if answer == '16 points':
                        #print(answer)
                        logic_score += 1
                        score = 1

                if num_logic_questions == 17:
                    #print(answer)
                    if answer == 'A player’s high average at the
beginning of the season may be just luck. The longer season provides a
more realistic test of a batter’s skill.':
                        #print('crroect')
                        logic_score += 1
                        score = 1

                if num_logic_questions == 18:
                    if answer == '1 out of 10':
                        #print(answer)
                        logic_score += 1
                        score = 1

                if num_logic_questions == 19:
                    if answer == 'c. Heads and tails are equally
probable on the sixth toss.':
                        #print(answer)
                        logic_score += 1
                        score = 1

                if num_logic_questions == 20:
```

```
                    if answer == 'People were cured by something
else':
                        #print(answer)
                        logic_score += 1
                        score = 1

                if num_logic_questions == 21:
                    if answer == 'A bank teller':
                        #print(answer)
                        logic_score += 1
                        score = 1

                if num_logic_questions == 22:
                    if answer == 'The crime rates of the two cities
closest to Middleton in location and size have decreased by 18% in the
same period.':
                        #print(answer)
                        logic_score += 1
                        score = 1

                if num_logic_questions == 23:
                    if answer == '1, 2, and 3':
                        #print(answer)
                        logic_score += 1
                        score = 1

                if num_logic_questions == 24:
                    if answer == '80% chance to win $60  and 20%
chance to win nothing':
                        #print(answer)
                        logic_score += 1
                        score = 1

                if num_logic_questions == 25:
                    if answer == 'The small container':
                        #print(answer)
                        logic_score += 1
                        score = 1

                if num_logic_questions == 26:
                    if answer == 'Strategy D':
                        #print(answer)
                        logic_score += 1
                        score = 1

                if num_logic_questions == 27:
                    if answer == 'I would turn off both the rented
movie and the free TV movie' or answer == 'I would continue to watch
both the rented movie and free TV movie':
                        #print(answer)
```

```
                        logic_score += 1
                        score = 1

                if num_logic_questions == 28:
                    if answer == 'The physician made the right
decision to operate for both Roger and Harold' or answer == 'The
physician should not have operated on either Harold or Roger':
                        #print(answer)
                        logic_score += 1
                        score = 1

                if is_confidence_question == True:
                    #print(row[column].strip())
                    num_confidence_questions += 1
                    confidence_sum +=
confidence_score[row[column].strip()]


                    if not row[column].strip() in (None,''):

score_file.write('\"'+str(confidence_score[row[column].strip()])
+'\",')
                    else:
                        score_file.write('\"'+'NA'+'\",')
                    is_confidence_question = False

                else:
                    num_logic_questions += 1
                    is_confidence_question = True

                    score_file.write('\"'+str(score)+'\",')


            #print('logic score',logic_score, 'out of',
num_logic_questions)
            #print('confidence avg',confidence_sum/
num_confidence_questions)
            summary_file.write(str(logic_score)+',')
            summary_file.write(str(logic_score/num_logic_questions)
+',')
            summary_file.write(str(confidence_sum/
num_confidence_questions)+',')
            if row[305] in (None,''):
                num_estimated_correct_logic = 0
            else:
                num_estimated_correct_logic = int(row[305])
            summary_file.write(row[305]+',')
            if num_estimated_correct_logic in (None,''):
                summary_file.write(str(0)+',')
            else:
```

```
                summary_file.write(str(num_estimated_correct_logic/
num_logic_questions)+',')
            summary_file.write((row[306])+',')
            summary_file.write(str(estimation_scores[row[307]])+',')
            if num_estimated_correct_logic in (None,''):
                summary_file.write(str(0)+',')
            else:
                summary_file.write(str(num_estimated_correct_logic-
logic_score) + ',')

            #print('incorrect',confidence_scores_logic_incorrect)
            #print('correct', confidence_scores_logic_correct)
            for key in confidence_scores_logic_incorrect.keys():

confidence_file.write(str(confidence_scores_logic_incorrect[key]) +
',')

            for key in confidence_scores_logic_correct.keys():

confidence_file.write(str(confidence_scores_logic_correct[key]) + ',')
            #confidence_file.write('\n')

    ##******** End Logic Task ***************************************#

    ##******** Grammar Task ****************************************#
            for column in range(305, 307):
                #print(row[column])
                score_file.write('\"'+str(row[column])+'\",')
            score_file.write(str(estimation_scores[row[307]])+',')

            for column in range(308, 310):
                #print(row[column])
                score_file.write('\"'+str(row[column])+'\",')

            score_file.write(str(estimation_scores[row[310]])+',')

            summary_file.write(row[308]+',')
            if row[308] in (None,''):
                summary_file.write(str(0)+',')
            else:
                summary_file.write(str(int(row[308])/(25))+',')
            summary_file.write((row[309])+',')
            summary_file.write(str(estimation_scores[row[310]])+',')

            is_confidence_question = False
            confidence_sum = 0
            confidence_scores_grammar_correct =
copy.deepcopy(confidence_score_template)
            confidence_scores_grammar_incorrect =
copy.deepcopy(confidence_score_template)
```

```
            grammar_score = 0
            num_grammar_questions = 0
            num_confidence_questions = 0
            #print('\n')
            #print('\n')
            #print('\n')
            #print('\n')

            for column in range(311,362):
                previous_grammer_score = grammar_score

                if is_confidence_question == True:

                    if not row[column].strip() in (None,''):
                        confidence =
confidence_score[row[column].strip()]
                        if score == 1:

confidence_scores_grammar_correct[str(confidence)] =
confidence_scores_grammar_correct[str(confidence)] +1
                        else:

confidence_scores_grammar_incorrect[str(confidence)] =
confidence_scores_grammar_incorrect[str(confidence)] +1

                score = 0
                if column == 341: #this is the shark question
                    #print(headers[column])
                    #print(is_confidence_question)
                    #print
                    score_file.write('\"'+str(row[column].strip())
+'\",')
                else:
                    #print(headers[column])
                    answer = row[column].strip()
                    #print(answer)

                    if num_grammar_questions == 0:
                        #print(num_grammar_questions,answer)
                        if answer == 'knew':
                            grammar_score += 1
                            score = 1

                    if num_grammar_questions == 1:
                        #print(num_grammar_questions,answer)
                        if answer == 'eighteenth and nineteenth':
                            grammar_score += 1
                            score = 1

                    if num_grammar_questions == 2:
```

```
                        #print(num_grammar_questions,answer)
                        if answer == 'an explanation of the impetus
for discussion of a potential bridge.':
                            grammar_score += 1
                            score = 1

                    if num_grammar_questions == 3:
                        #print(num_grammar_questions,answer)
                        if answer == 'accidents occur about three
times more often':
                            grammar_score += 1
                            score = 1

                    if num_grammar_questions == 4:

                        #print(num_grammar_questions,answer)
                        if answer == 'take into account':
                            grammar_score += 1
                            score = 1

                    if num_grammar_questions == 5:
                        #print(num_grammar_questions,answer)
                        if answer == 'In an effort to improve driver
and pedestrian safety, auto engineers often come up with ingenious
designs.':
                            grammar_score += 1
                            score = 1

                    if num_grammar_questions == 6:
                        #print(num_grammar_questions,answer)
                        if answer == 'switches for many years;':
                            grammar_score += 1
                            score = 1

                    if num_grammar_questions == 7:
                        #print(num_grammar_questions,answer)
                        if answer == 'road, particularly during':
                            grammar_score += 1
                            score = 1

                    if num_grammar_questions == 8:
                        #print(num_grammar_questions,answer)
                        if answer == 'NO CHANGE':
                            grammar_score += 1
                            score = 1

                    if num_grammar_questions == 9:
                        #print(num_grammar_questions,answer)
                        if answer == 'Two other systems are being
developed that will potentially make driving at night safer:':
```

```
                            grammar_score += 1
                            score = 1

                    if num_grammar_questions == 10:
                        #print(num_grammar_questions,answer)
                        if answer == 'their':
                            grammar_score += 1
                            score = 1

                    if num_grammar_questions == 11:
                        #print(num_grammar_questions,answer)
                        if answer == 'dog\'s':
                            grammar_score += 1
                            score = 1

                    if num_grammar_questions == 12:

                        #print(num_grammar_questions,answer)
                        if answer == 'Dogs are often fearful of
unusual or unfamiliar situations and people.':
                            grammar_score += 1
                            score = 1

                    if num_grammar_questions == 13:
                        #print(num_grammar_questions,answer)
                        if answer == 'zones, particularly':
                            grammar_score += 1
                            score = 1

                    if num_grammar_questions == 14:
                        #print(num_grammar_questions,answer)
                        if answer == 'a  condition   called
desynchronosis, commonly known as jet lag,':
                            grammar_score += 1
                            score = 1

                    if num_grammar_questions == 15:
                        #print(num_grammar_questions,answer)
                        if answer == 'without. He\'d sooner mail':
                            grammar_score += 1
                            score = 1

                    if num_grammar_questions == 16:
                        #print(num_grammar_questions,answer)
                        if answer == 'I assured her that the cheapest
possible option would more than suffice, and when she told me it was
only $15 per month, I arranged to have it installed.':
                            grammar_score += 1
                            score = 1
```

```
                    if num_grammar_questions == 17:
                        #print(num_grammar_questions,answer)
                        if answer == 'NO CHANGE':
                            grammar_score += 1
                            score = 1

                    if num_grammar_questions == 18:
                        #print(num_grammar_questions,answer)
                        if answer == 'began':
                            grammar_score += 1
                            score = 1

                    if num_grammar_questions == 19:
                        #print(num_grammar_questions,answer)
                        if answer == 'NO CHANGE':
                            grammar_score += 1
                            score = 1

                    if num_grammar_questions == 20:
                        #print(num_grammar_questions,answer)
                        if answer == 'that stood for':
                            grammar_score += 1
                            score = 1

                    if num_grammar_questions == 21:
                        #print(num_grammar_questions,answer)
                        if answer == 'NO CHANGE':
                            grammar_score += 1
                            score = 1

                    if num_grammar_questions == 22:
                        #print(num_grammar_questions,answer)
                        if answer == 'Deathstalker Scorpion which is
just about the only species':
                            grammar_score += 1
                            score = 1

                    if num_grammar_questions == 23:
                        #print(num_grammar_questions,answer)
                        if answer == 'NO CHANGE':
                            grammar_score += 1
                            score = 1

                    if num_grammar_questions == 24:
                        #print(num_grammar_questions,answer)
                        if answer == 'paints':
                            grammar_score += 1
                            score = 1

                    if is_confidence_question == True:
```

```
                        #print(row[column].strip())
                        num_confidence_questions += 1
                        confidence_sum +=
confidence_score[row[column].strip()]

                        if not row[column].strip() in (None,''):

score_file.write('\"'+str(confidence_score[row[column].strip()])
+'\",')
                        else:
                            score_file.write('\"'+'NA'+'\",')

                        is_confidence_question = False

                    else:
                        num_grammar_questions += 1
                        is_confidence_question = True
                        if grammar_score > previous_grammer_score:
                            score_file.write('\"'+str(1)+'\",')
                        else:
                            score_file.write('\"'+str(0)+'\",')




            #print('grammar score',grammar_score, 'out of',
num_grammar_questions)
            #print('garammar confidence avg',confidence_sum/
num_confidence_questions)
            summary_file.write(str(grammar_score)+',')
            summary_file.write(str(grammar_score/
num_grammar_questions)+',')
            summary_file.write(str(confidence_sum/
num_confidence_questions)+',')

            summary_file.write(row[362]+',')
            if row[362] in (None,''):
                summary_file.write(str(0)+',')
                num_estimated_correct_grammar = 0

            else:
                summary_file.write(str(int(row[362])/
num_grammar_questions)+',')
                num_estimated_correct_grammar = int(row[362])

            summary_file.write((row[363])+',')
            summary_file.write(str(estimation_scores[row[364]])+',')

            if row[362] in (None,''):
```

```
                summary_file.write(str(0)+',')
            else:
                summary_file.write(str(int(row[362])-grammar_score)
+',')

            for key in confidence_scores_grammar_incorrect.keys():

confidence_file.write(str(confidence_scores_grammar_incorrect[key]) +
',')

            for key in confidence_scores_grammar_correct.keys():

confidence_file.write(str(confidence_scores_grammar_correct[key]) +
',')
            #confidence_file.write('\n')

    #************ End Grammar Task ********************#

    #************ NW Task *****************************#
            for column in range(362, 364):
                score_file.write('\"'+str(row[column])+'\",')
            score_file.write(str(estimation_scores[row[364]])+',')

            for column in range(365, 367):
                score_file.write('\"'+str(row[column])+'\",')
            score_file.write(str(estimation_scores[row[367]])+',')

            summary_file.write(row[365]+',')
            if row[365] in (None,''):
                summary_file.write(str(0)+',')
            else:
                summary_file.write(str(int(row[365])/(27))+',')
            summary_file.write((row[366])+',')
            summary_file.write(str(estimation_scores[row[367]])+',')

            is_confidence_question = False
            confidence_sum = 0

            confidence_scores_nw_correct =
copy.deepcopy(confidence_score_template)
            confidence_scores_nw_incorrect =
copy.deepcopy(confidence_score_template)

            nw_score = 0
            num_nw_questions = 0
            num_confidence_questions = 0
            #print('\n')
            #print('\n')
            #print('\n')
            #print('\n')
```

```
            #nw_conf = 0
            for column in range(368,422):
                previous_nw_score = nw_score
                #print(headers[column])
                #print(is_confidence_question)
                #print
                answer = row[column].strip()

                if is_confidence_question == True:

                    if not row[column].strip() in (None,''):

                        confidence =
confidence_score[row[column].strip()]
                        #print(user_id)
                        #print(confidence)
                        #nw_conf += 1
                        #print(nw_conf)
                        if score == 1:


confidence_scores_nw_correct[str(confidence)] =
confidence_scores_nw_correct[str(confidence)] +1
                        else:

confidence_scores_nw_incorrect[str(confidence)] =
confidence_scores_nw_incorrect[str(confidence)] +1

                score = 0
                if num_nw_questions == 0:
                    if answer == 'Presidential authorization is
required to use a nuclear weapon':
                        nw_score += 1
                        score = 1

                if num_nw_questions == 1:
                    if answer == 'All of these are equally important':
                        nw_score += 1
                        score = 1

                if num_nw_questions == 2:
                    if answer == 'Leslie Groves':
                        nw_score += 1
                        score = 1

                if num_nw_questions == 3:
                    if answer == 'Used his prowess as arguably the
most recognized scientist in the US to plea for the government to
develop nuclear weapons':
                        nw_score += 1
```

```
                        score = 1

                if num_nw_questions == 4:
                    if answer == 'The Neutron':
                        nw_score += 1
                        score = 1

                if num_nw_questions == 5:
                    if answer == 'Not enough information to answer the
question':
                        nw_score += 1
                        score = 1

                if num_nw_questions == 6:
                    if answer == '233U':
                        nw_score += 1
                        score = 1

                if num_nw_questions == 7:
                    if answer == 'Nuclear weapon explosives are
safer':
                        nw_score += 1
                        score = 1

                if num_nw_questions == 8:
                    if answer == 'C,H,N,O':
                        nw_score += 1
                        score = 1

                if num_nw_questions == 9:
                    if answer == 'W76, W87, W88':
                        nw_score += 1
                        score = 1

                if num_nw_questions == 10:
                    if answer == 'Hanford, Oak Ridge, Los Alamos':
                        nw_score += 1
                        score = 1

                if num_nw_questions == 11:
                    if answer == 'Fission':
                        nw_score += 1
                        score = 1

                if num_nw_questions == 12:
                    if answer == 'US, UK, Russia, China, France':
                        nw_score += 1
                        score = 1

                if num_nw_questions == 13:
```

```
                    if answer == 'All of these are free-zones':
                        nw_score += 1
                        score = 1

                if num_nw_questions == 14:
                    if answer == 'Fast neutrons enable nuclear
weapons, Slow neutrons enable Reactors':
                        nw_score += 1
                        score = 1

                if num_nw_questions == 15:
                    #print(num_grammar_questions,answer)
                    if answer == 'Lise Meitner':
                        nw_score += 1
                        score = 1

                if num_nw_questions == 16:
                    #print(num_grammar_questions,answer)
                    if answer == '6.x':
                        nw_score += 1
                        score = 1

                if num_nw_questions == 17:
                    #print(num_grammar_questions,answer)
                    if answer == '42 U.S.C.':
                        nw_score += 1
                        score = 1

                if num_nw_questions == 18:
                    #print(num_grammar_questions,answer)
                    if answer == 'Radius and density':
                        nw_score += 1
                        score = 1

                if num_nw_questions == 19:
                    #print(num_grammar_questions,answer)
                    if answer == 'US Citizens':
                        nw_score += 1
                        score = 1

                if num_nw_questions == 20:
                    #print(num_grammar_questions,answer)
                    if answer == 'Safety':
                        nw_score += 1
                        score = 1

                if num_nw_questions == 21:
                    #print(num_grammar_questions,answer)
                    if answer == 'AQ Khan':
                        nw_score += 1
```

```
                        score = 1

                if num_nw_questions == 22:
                    #print(num_grammar_questions,answer)
                    if answer == 'All of these':
                        nw_score += 1
                        score = 1

                if num_nw_questions == 23:
                    #print(num_grammar_questions,answer)
                    if answer == 'Guns, Gates and Guards':
                        nw_score += 1
                        score = 1

                if num_nw_questions == 24:
                    #print(num_grammar_questions,answer)
                    if answer == 'The ability to hold a specific
target at risk':
                        nw_score += 1
                        score = 1

                if num_nw_questions == 25:
                    #print(num_grammar_questions,answer)
                    if answer == '1':
                        nw_score += 1
                        score = 1

                if num_nw_questions == 26:
                    #print(num_grammar_questions,answer)
                    if answer == 'Getting more energy out of a
reaction than is put into it':
                        nw_score += 1
                        score = 1




                if is_confidence_question == True:
                    #print(row[column].strip())
                    num_confidence_questions += 1
                    confidence_sum +=
confidence_score[row[column].strip()]

                    if row[column].strip() in (None, ''):
                        score_file.write('\"'+'NA'+'\",')
                    else:

score_file.write('\"'+str(confidence_score[row[column].strip()])
+'\",')
```

```
                    is_confidence_question = False

                else:
                    num_nw_questions += 1
                    is_confidence_question = True
                    if nw_score > previous_nw_score:
                        score_file.write('\"'+str(1)+'\",')
                    else:
                        score_file.write('\"'+str(0)+'\",')

            for column in range(422, 424):
                score_file.write('\"'+str(row[column])+'\",')
            score_file.write(str(estimation_scores[row[424]])+',')

            #print('nw score',nw_score, 'out of', num_nw_questions)
            #print('nw confidence avg',confidence_sum/
num_confidence_questions)
            summary_file.write(str(nw_score)+',')
            summary_file.write(str(nw_score/num_nw_questions)+',')
            summary_file.write(str(confidence_sum/
num_confidence_questions)+',')
            #print(nw_score)
            summary_file.write(row[422]+',')
            if row[422] in (None,''):
                summary_file.write(str(0)+',')
                num_estimated_correct_nw = 0
            else:
                num_estimated_correct_nw = int(row[422])
                summary_file.write(str(int(row[422])/(27))+',')

            summary_file.write((row[423])+',')
            summary_file.write(str(estimation_scores[row[424]])+',')

            if row[422] in (None,''):
                summary_file.write(str(0)+',')
            else:
                summary_file.write(str(int(row[422])-nw_score)+',')

            for key in confidence_scores_nw_incorrect.keys():

confidence_file.write(str(confidence_scores_nw_incorrect[key]) + ',')

            for key in confidence_scores_nw_correct.keys():

confidence_file.write(str(confidence_scores_nw_correct[key]) + ',')
            confidence_file.write('\n')



            summary_file.write(final_answers[num_records] + ',')
```

```
            max_score = max(logic_score/
num_logic_questions,grammar_score/num_grammar_questions,nw_score/
num_nw_questions)

            best_subject = ''
            if max_score == logic_score/num_logic_questions:
                best_subject = 'Logic'
            if max_score == grammar_score/num_grammar_questions:
                best_subject = 'English Grammar'
            if max_score == nw_score/num_nw_questions:
                best_subject = 'Nuclear weapon science and technology'

            summary_file.write(best_subject+',')

            overall_score = logic_score+grammar_score+nw_score

            summary_file.write( str(overall_score) + ',')

            overall_estimated_score =
num_estimated_correct_logic+num_estimated_correct_grammar+num_estimate
d_correct_nw

            summary_file.write(str(overall_estimated_score) + ',')

            summary_file.write(str(overall_estimated_score -
overall_score) + ',')

            score_file.write(final_answers[num_records])
            num_records += 1
            #master_file.write()

            score_file.write('\n')
            summary_file.write('\n')
    score_file.close()
    master_file.close()
    summary_file.close()
    confidence_file.close()

    confidence_file = open(confidence_filename, "a+")

    df = pd.read_csv(confidence_filename)

    columns = len(df.columns)
    rows = len(df.index)

    #
    confidence_file.write('Totals,')

    for i in range(1,columns-1):
```

```
        df1 = df.iloc[1:rows,i]
        #print(df1)
        #print(df1.values.sum())
        confidence_file.write(str(df1.values.sum())+',')
    #confidence_file.write('\n')



if __name__ == '__main__':
    parser = OptionParser()

    parser.add_option("-d", "--directory", metavar="STR",
            action="store", type=str, default=None,
            help="file to process")

    (options, args) = parser.parse_args()
    directory = options.directory

    for root, dirs, files in os.walk('./'+directory):
    #print(root)
        for name in files:
            if name.endswith(".csv"):
                print("./"+directory+'/'+name)
                run("./"+directory+'/'+name)
```