**RSE Assessment Task 1**

1. What is the total energy of the system?

-3.566690905823207

2. What is the total number of pair interactions?

28

3. How much would the total energy change in percentage if the energy contributions were rounded to the nearest integer number? I.e

Integer total energy: -12

Percentage error: 236.44631163322939

**Python Source Code:**

**import** **math**   
  
# constant  
k = **0.3062**  
  
particleData = [] # declares blank list for the processed data to go into, global because it will be used in several places  
  
def **readInput**():   
'''  
 Reads the user input line by line so it can read a table of any length in the provided format   
 the table can be copy and pasted in full or written line by line  
 Uses a while loop so it can be any length  
 returns the table as a list of each line as a string  
 '''  
 **print**("Enter the table of particles: **\n**")  
 dataTable = [] # the list of the input lines of the table  
 **while** True:  
 newLine = input() # reads each line   
 **if** newLine == "": # if the line is blank the table is done and returns it  
 **return** dataTable  
 **if** newLine[**0**].isdigit(): # checks the line starts with the index to ignore the colimn titles at the top  
 dataTable.append(newLine) # adds each line of the table to the list  
  
**def** **extractData**(dataTable):  
 '''  
 takes the list of input lines from readInput() and processes it into float values to be used in the calculations  
 adds the values read from the input into the particleData list  
 '''  
   
**for** index, line **in** enumerate(dataTable): # steps through each line of the input table, enumerates so the index can be used for the length of each line  
 startChar = -**1**  
 endChar=**0**  
 dataPoints = [**0.0**,**0.0**,**0.0**,**0.0**] # declares the list of float values as 4 0s because each line of the table has 4 values to be read  
 lineLength = len(dataTable[index]) # gets the length of the current line of the table  
 pointsExtracted = **0** # declares the amount of data points extracted from each line  
 **for** i **in** range(**0**,lineLength): # steps through each character of the current line  
 **if** pointsExtracted < **4**: # makes sure it only looks for data if its not all already taken  
 **if** pointsExtracted == **3**: # the charge doesnt have a " " after so it has a specific condition to read the charge (q)  
 **if** line[startChar+**1**:] == "1":  
 dataPoints[pointsExtracted] = **1.0**  
 pointsExtracted +=**1**  
 **elif** line[startChar+**1**:] == "-1":  
 dataPoints[pointsExtracted] = -**1.0**  
 pointsExtracted +=**1**  
 **elif** line[i] == " ": # uses the spaces in the input line to know where to look for the input values  
 startChar = i  
 **for** j **in** range(startChar + **1**, lineLength): # looks for the next " " after startChar in the line to get the data from inbetween  
 **if** dataPoints[pointsExtracted] == **0.0**: # checks this data point hasnt already been taken from the table  
 **if** line[j] == " ": # finds the next " " after the current data point  
 endChar = j  
 dataPoints[pointsExtracted] = (float(line[startChar:endChar])) # converts the data point from inbetween the spaces and converts it to float   
  
 pointsExtracted +=**1**  
 startChar = endChar # means that it will only extract after this data point on the next steps through the i loop  
 particleData.append({"x": dataPoints[**0**], "y": dataPoints[**1**], "z": dataPoints[**2**], "q": dataPoints[**3**]}) # adds all the data from this line to particle data list in a dictionary format  
**def** **calculateDistance**(particle1, particle2):  
 '''  
 Calculates the distance between any 2 particles passed through it  
 Using pythagoras in 3d space  
 '''  
 distance = math.sqrt((particle2["x"] - particle1["x"])\*\***2** + (particle2["y"] - particle1["y"])\*\***2** + (particle2["z"] - particle1["z"])\*\***2**)  
 **return** distance  
  
def **calculateInteractionEnergy**(charge1, charge2 , distance):  
 '''  
 Calculates the energy of one interaction using the constant k   
 and the charges and distance between 2 particles  
 '''  
 energy = k \* ((charge1 \* charge2) / distance)  
 **return** energy  
  
  
  
**def** **calculateTotalInteractions**(noOfParticles):  
 '''  
 Runs each interaction energy calculation and totals it and counts up the interactions  
 Uses a nested for loop to do a calculation with each other particle  
 Takes the number of particles and returns the total energy and the number of interactions  
 '''  
 totalEnergy = **0**  
 totalInteractions = **0**  
 **for** i **in** range(noOfParticles):  
 **for** j **in** range(**1**, noOfParticles):  
 **if** j > i: # makes sure the interaction hasnt already been done or isnt between a particle and itself  
 totalEnergy += calculateInteractionEnergy(particleData[i]["q"], particleData[j]["q"], calculateDistance(particleData[i], particleData[j])) # runs the energy calculation for each particle interaction  
 totalInteractions += **1**  
 **return** totalEnergy, totalInteractions  
  
**def** **calculateTotalIntegerInteractions**(noOfParticles):  
 '''  
 Does the same as calculateTotalInteractions but rounds the value of each interaction energy to and integer  
 '''  
 totalIntegerEnergy = **0**  
 **for** i **in** range(noOfParticles):  
 **for** j **in** range(**1**, noOfParticles):  
 **if** j > i:  
 totalIntegerEnergy += round(calculateInteractionEnergy(particleData[i]["q"], particleData[j]["q"], calculateDistance(particleData[i], particleData[j])))  
 **return** totalIntegerEnergy  
**def** **calculatePercentageError**(trueValue, approxValue):  
 '''  
 Calculates the percentage error using the true and approximate values  
 and returns it  
 '''  
 percentageError = ((trueValue - approxValue) / trueValue) \* **100**  
 **return** percentageError  
  
def **runCalculations**():  
 '''  
 Runs all the functions to get the data and process it  
 prints the energy and number of interactions and integer energy and percentage error  
 '''  
 extractData(readInput())  
 results = calculateTotalInteractions(len(particleData)) # results[0] is the total energy and [1] is the number of interactions  
 approxValue = calculateTotalIntegerInteractions(len(particleData))  
 percentageError = calculatePercentageError(results[**0**], approxValue)  
 **print**("Total energy of system: " + str(results[**0**]) + "**\n** Number of interactions: " + str(results[**1**]) + "**\n** Integer total energy: " + str(approxValue) + "**\n** Percentage error: " + str(percentageError))  
   
runCalculations()  
input()