

# Computing Practical Project

Oscar Daniel

February 2015

# Contents

<b>1</b>	<b>Analysis</b>	<b>2</b>
1.1	Background and Definition of Problem . . . . .	2
1.2	Users and the current system . . . . .	4
1.2.1	Data Flow Diagrams . . . . .	4
1.3	Feedback Reports . . . . .	6
1.4	Proposed System . . . . .	9
1.5	Objectives . . . . .	9
1.6	Data Information . . . . .	11
1.7	Object Orientated Programming Planning . . . . .	12
<b>2</b>	<b>Design</b>	<b>13</b>
2.1	Navigation . . . . .	13
2.1.1	GUI Navigation . . . . .	14
2.2	Validation . . . . .	14
2.3	Overall System Design . . . . .	16
2.4	Record Structure . . . . .	16
2.5	Data Transformation . . . . .	17
2.6	Security and Integrity . . . . .	21
2.7	Prototype . . . . .	22
2.7.1	Command Line Prototype Code . . . . .	22
2.7.2	GUI Prototype . . . . .	30

# Chapter 1

## Analysis

### 1.1 Background and Definition of Problem

At Queen Elizabeth Grammar School, pupils in year 13 are taught several calculations for their science subjects. They are tested on these every month or so, after the unit has been taught and students are assumed to understand the topics. Students are set questions during teaching time, however these results are not recorded and sometimes a struggling student may slip under a teacher's radar.

**Examples of A Level Calculations:**

#### **Rate of reaction equations**

A set of reaction equations used to deduce the rate of a reaction, the concentrations of reactants, their orders and the rate constant:  $k$ .

A reaction of reactants  $A + B \rightarrow D$  where A has the concentration  $0.0023 \text{ mol dm}^{-3}$  and the power 1. B has the concentration  $0.0154 \text{ mol dm}^{-3}$  with the power 2. Whilst the rate of reaction is  $0.000\,045 \text{ mol s}^{-1}$ .

*An example question.*

#### **Standard deviation**

A statistical value that is used ceaselessly in biology.

You and your friends have just measured the heights of your dogs (in millimetres): The heights (at the shoulders) are: 600mm, 470mm, 170mm, 430mm and 300mm. Find the standard deviation

Find out the Mean, the Variance, and the Standard Deviation.

Your first step is to find the Mean: The mean (average) height is 394 mm

Now we calculate each dog's difference from the Mean. To calculate the Variance, take each difference, square it, and then average the result:

So, the Variance is 21,704.

And the Standard Deviation is just the square root of Variance, so:  
Standard Deviation:  $\sqrt{21,704} = 147.32 = 147$  (to the nearest mm)

*An Example Question*

### **Ideal Gas Equation**

A calculation that is used in both chemistry and physics A level.

The ideal gas equation is  $PV = nRT$ . P is pressure in Pa, V is volume in cubic metres, n is number of moles and T is temperature in degrees Kelvin. R is the ideal gas constant which is measured as 8.314 Joules per Kelvin per Mole.

STP = 273K and 100kPa

Problem 1: Determine the volume of occupied by 2.34 grams of carbon dioxide gas at STP.

Our first job is to rearrange the equation to find V:  $V = nRT/P$

Then find moles of  $CO_2$ :  $2.34/44 = 0.0532$

Then apply these to the equation.

$$(0.0532 * 8.314 * 273)/100,000 = 0.0012074921m^3$$

Problem 2: A sample of argon gas at STP occupies 56.2 litres. Determine the number of moles of argon and the mass in the sample.

Rearrange to find n:  $n = PV/RT$

$$56.2 \text{ litres} = 0.0000562 m^3$$

$$(100,000 * 0.0000562)/(8.314 * 273) = 0.00247607416 \text{ moles. } Ar = 18g/Mole$$

$$18 * 0.00248 = 0.0446g$$

*An Example Question*

### **Hardy – Weinberg Equation**

A statistical test that is used to estimate the proportions of different genotypes in a population.

1. If 98 out of 200 individuals in a population express the recessive phenotype, what percent of the population would you predict would be heterozygotes? (a) I have given you information on the frequency of the homozygous recessive (or  $q^2$ ). So start by determining  $q^2$  and then solving for q.  $98/200 = q^2 = 0.49$   $\sqrt{0.49} = 0.7 = q$

(b) Now that you have q, you can solve for p. Remember there are only two alleles in the population, so if you add the frequency of the two alleles, you have accounted for all possibilities and it must equal 1. So

$$p + q = 1 \text{ Therefore: } p = 1 - q = 0.3$$

(c) Now what is the formula for heterozygotes? Think back to the Hardy-Weinberg equation – it is dealing with the genotypes of individuals in the population.  $2pq = 2 * 0.3 * 0.7 = 0.42$

(d) Now that you have figured out the percentage of heterozygotes, can you

figure out the percentage of homozygous dominant? Does the percentage of homozygous dominant, heterozygotes and homozygous recessive individuals add up to 100 percent? If not, you have made an error. Those are the only three genotypes possible with only two alleles and a simple dominant and recessive relationship.  $1 - (0.49 + 0.42) = p^2 = 0.09$  or  $0.3^2$

*An Example Question*

## 1.2 Users and the current system

Mr Fascione is the head of Science at QEGS. He focusses on student grades and student support alongside being a teacher.

Currently the only way for pupils' scores from homework to be tracked is by them being entered manually into a spreadsheet after being recorded on paper. There is also no way for pupils to have their problems solved, or explained step by step, other than referring to a text book, which is not interactive. Student feedback is also only monitored for tests, with this system every piece of homework could be used for student support.

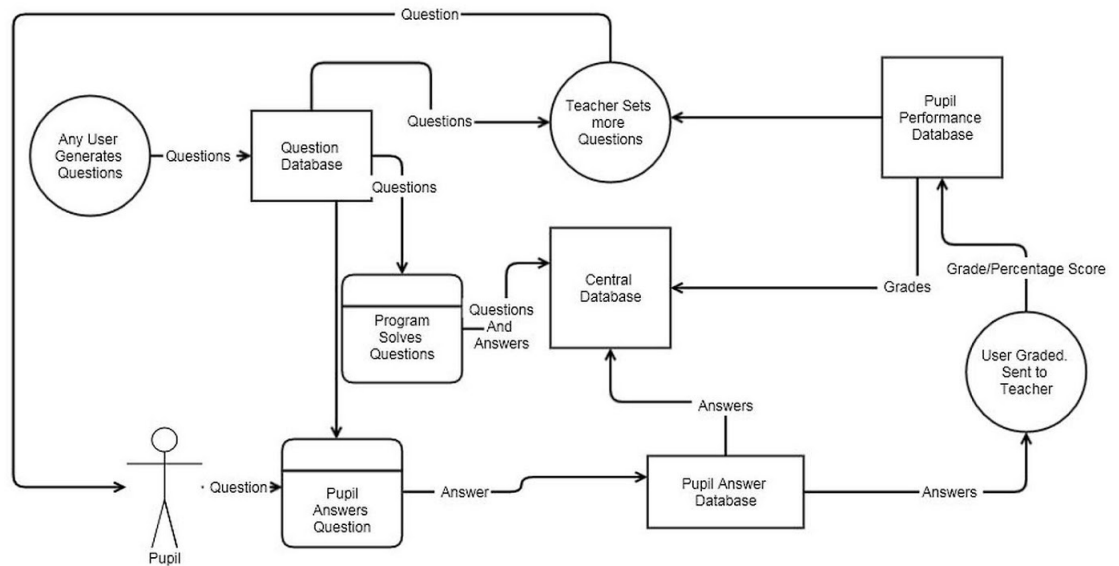
Pupils can only access a few problems that have been devised by teachers or examiners. These can also be hard for students to grasp, so a program which goes through these at the students' personal pace would be of great use. Teachers also have no digitally stored questions that can be easily sent to students, they must hand out photocopies and these are always being lost. A computer system would allow work to be set easily, without risk of students losing their things, as well as teachers not losing score sheets.

From my feedback and interviews from my end users I have been able to select the appropriate calculations for the computer package to use, and also have a framework to try and emulate, to best please my end user.

### 1.2.1 Data Flow Diagrams

To give a better understanding of how data flows in the current and proposed systems, I produced a data flow diagram for each. They highlight the areas of concern already mentioned and show how a computer package would eliminate them.





*The proposed system*

## 1.3 Feedback Reports

To get a better idea of exactly what type of system I should make, and what requirements that system would need to have, I asked my end users a few

Student Questionnaire

- 1) When studying calculations for you're a Level sciences, which calculations do you find most difficult?

Rate of Reaction, standard deviation, standard error, Hardy-Weinberg equation, Ideal Gas equation.

- 2) Would you benefit from being able to access sample questions?

Yeah.

- 3) What would help in your learning of calculations?

An interactive, step by step question answering program.

- 4) Do you think a computer aid would benefit your learning?

Yes

Alex Demeza - ☺

questions.



Teacher Questionnaire

1. Which calculations do you find students struggle with the most?

Calculations involving big and small numbers

2. Do you think an automatic grading system would improve student support?

Yes

3. Would a computer learning package aid in your teaching?

Aids learning, not teaching

4. What would you expect from a computer aid package?

My Maths type package

M. 1981

## 1.4 Proposed System

Pupils will need to be able to input their own problems and have them be solved in a step by step process where they can enter their answer for each step. The program will tell them if they were right and instructing them as to what they should be doing.

They will also be able to store their own problems and solutions in the central database.

Teachers will be allowed to set work to be done and see the scores.

Teachers and pupils will also have unique accounts with teachers having the ability to see pupil scores and set questions. If pupils are having difficulties then they can send a message to the teacher and vice versa. Teachers should be able to make new pupil and teacher accounts and delete old ones.

Allows students and staff to log in.

The program will generate its own questions but may not display questions in the correct format.

Only 100 questions will be stored at once as the memory taken up would be too high.

Pupil accounts would have to be entered in manually as school will not give me access to the student detail database.

## 1.5 Objectives

### Essential Objectives

1. Example Based Demonstration – The program should be able to provide a clear demonstration of how any of the equations work, by executing them for the user. It should be able to carry out this demonstration either on data entered by the user, or on data stored within the program/database.
2. Background Explanation Of Problems- The program should be able to give a small amount of detail initially if the student should request it
3. Interactive System allowing Users to Solve Equations – The user should be able to choose to attempt to solve the equations themselves – if this is the case the program should point out (or correct) any mistakes they may have made.

4. Unique User Accounts- Teachers and pupils should have different accounts with different privileges.
5. A System To Allow Users to Input and Store Questions – The users should be able to input their own questions, that the program can solve or walk them through, and have the option to store them.
6. A Database Holding Question Data- the program should run off of a back end database holding around 100 questions in total, which can randomly select a question, or allow a teacher to set specific questions for the pupil to answer.
7. A System For Teachers to set questions and Students Send Feedback – The program should allow teachers to select questions from the database to give to students. Students should be able to send feedback to their teachers.
8. A Database Holding Pupil Performance- Teachers should have access to the questions that pupils answered correctly and incorrectly so that they can monitor their performance and therefore provide support.

### **General Objectives**

1. Teachers should be able to create or delete user accounts.
2. A central database should collate all information into a practical form.
3. The program should be self-explanatory and not overly complex for the user.
4. Should use less than 100MB of storage.
5. Grade scores. Highlights students who are below their target grades
6. Allows for students to make comments on problems so the teacher can help them.
7. A log in system with passwords and usernames

### **Acceptable Limitations**

1. To allow the program to generate questions properly with the correct units would require the use of libraries and a lot more time than I have available.

2. As my school will not allow me access to the student details database, students must be entered in manually.
3. Only 100 questions stored at one time to control database size. As there may be hundreds of students inputting questions, the database may have thousands of questions at one time, which would make it cumbersome to use.
4. As the school already has a functioning email system, it would be unnecessary to recreate this system in my project.
5. As having a function to solve equations could lead to cheating, I will leave this out.

## 1.6 Data Information

Instead of using an SQL database to store information, as I am using an object oriented system I will use the Pickle function in Python to store my objects and classes as .pickle binary files. This allows for compact storage and easy data storage manipulation within python.

Data	Source	Destination	Type
Student Name	Input manually	Student File PICKLE	String
Student ID	Generated by Database	Student File PICKLE	Integer
Student Password	Input manually	Student File PICKLE	String
Student Test Scores	Generated by program	Student File PICKLE	Integer and String
Student Target Grade	Input manually	Student File PICKLE	String
Teacher Name	Input manually	Teacher File PICKLE	String
Teacher Password	Input manually	Teacher File PICKLE	String

Questions	Input manually	Question File	PICKLE	Arrays of Integers and Strings
-----------	----------------	---------------	--------	--------------------------------

## 1.7 Object Orientated Programming Planning

Access Type	Field Name	Field Type	Initial Value	Description
Private	Pupil Data	Strings and Integers	-	The data associated with a student. Contains information about them, i.e. form group.
Private	Question Data	Strings and Integers	-	Data contained in the question object. Gives format for insertion into program.

# Chapter 2

## Design

### 2.1 Navigation

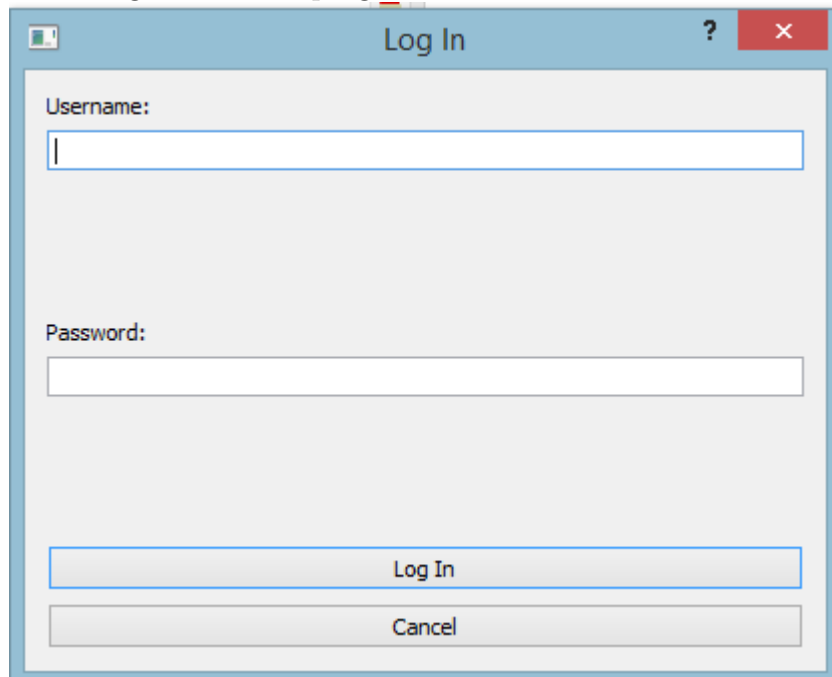
- Log in menu
  - Main Menu
    - \* Set Questions \*
    - \* Select Student/s\*
    - \* Select Question/s\*
    - \* See Set Work ^
    - \* Attempt Set Questions^
    - \* Practice
    - \* Finding K Questions
    - \* Select question from list / Randomly
    - \* Finding Rate Of Reaction Questions
    - \* Select question from list / Randomly
    - \* Messages
      - Open Messages
      - Create Message
      - Recipient Content

\* = *Teacher account only*

^ = *Student account only*

### 2.1.1 GUI Navigation

To make my program more user friendly, and using user feedback, I decided that a graphical user interface would be more appropriate. These forms show general navigation of the program.



The image shows a standard Windows-style dialog box titled "Log In". The title bar is light blue and contains a small icon on the left, the text "Log In" in the center, a question mark icon on the right, and a red button with a white "X" for closing. The main area of the dialog is light gray. It contains two text input fields. The first is labeled "Username:" and the second is labeled "Password:". Below these fields are two buttons: "Log In" and "Cancel".

*Log In Page*

## 2.2 Validation

Field Name	Validation Checks	Description	Error Message	Data Input
Gender	Listed Data ("M", "F")	Only allow M or F	Please select legal gender.	Radio Button
Name	Presence, Data Type.	Only allow names with letter characters, apostrophes or hyphens, make sure a name is present.	Please enter a name. / Please remove invalid characters from field.	F0rd Pr3f3ct
Form Group	Presence, Listed Data.	Allow only listed form groups.	Please select a form group from the list.	Drop Down
Password	Presence, correct password format.	Any string of ascii characters, including numbers and special characters.	Please enter a valid password.	Slartibartfast
Target Grade	A, B, C, D, E, F or U	A single ASCII upper case character.	Please enter a target grade.	A
Parent Email	Correct email format.	String.string'@'string.'string	Please enter a valid email address.	myEmail@email.net

The student class which I have designed incorporates all of the information above, apart from email and gender as it is going to be used at an all-boys school which stores parental contact information securely. Therefore those two data fields would not be appropriate to store in my program. The pupil class accepts input in this format: ( studentid, password, forename, surname, middlename, age, form, targetgrade, questionsanswered, questionstoanswer, answeredcorrectly)

The questionsanswered, questionstoanswer and answeredcorrectly values are worked out by other functions in the program and are set to 0 when a new student is created.



To track a students' performance, when they answer questions the `percentage()` `findgrade()` functions take their answered questions and find out if they are keeping to their current target grade.

## 2.3 Overall System Design

Input	Process	Storage	Output
Question Data	Store Question	Question Pickle	Question stored successfully.
Question Data	Answer Question	Question Pickle	Question Answer.
Student Data	Add Student	Pupil Pickle	Pupil Data stored successfully.
Question ID, Student ID	Set Question	Pupil Pickle	Questions sent to Pupil
Student Answer	Answer Question	Retrieve from answer box in window, check answer. Record if wrong or right. Send to pupil pickle.	Question answered in/correctly. OR Answer Sent.
Student/Teacher Password	Log In	Retrieve from Pupil / Teacher pickle	Logged in as "person" OR Username password combination is incorrect.

## 2.4 Record Structure

I plan to use pickle tool to store my objects as binary files.

The pickle module implements binary protocols for serializing and de-serializing a Python object structure. "Pickling" is the process whereby a Python object hierarchy is converted into a byte stream, and "unpickling" is the inverse operation, whereby a byte stream (from a binary file or bytes-like object) is converted back into an object hierarchy.

Pickle accepts inputs in this format: `pickle.dump(obj, file, protocol,)`

As an example:

```
class container():
    def __init__(self, name):
        self.objects = []
        self.name = name

    def sayName(self):
        return self.name

    def listObjects(self):
        for thing in self.objects:
            print (thing.sayName())

    def addObject(self,item):
        self.objects.append(item)

with open('data.pickle', 'wb') as f:
    pickle.dump(box, f, pickle.HIGHEST_PROTOCOL)
```

^ Stores class and instance as a binary file (.PICKLE)

```
with open('data.pickle', 'rb') as f:
    The protocol version used is detected automatically, so we do not have
to specify it.
    box = pickle.load(f)
```

^ Reads in the information from the file and decodes it.

## 2.5 Data Transformation

### Hardy - Weinberg

Uses the Hardy-Weinberg equation to find 2pq. Two variations of the class are used. One makes random numbers and one uses values from a question class.

### Pseudocode 1

```

FUNCTION calculateAnswer(self):
    totalpop <- totalpop
    recessiveInpop <- recessiveInpop
    q_sqrdaspop <- Round_To_n(recessiveInpop, 1) # now rounded
    qsqrd <- Round_To_n(q_sqrdaspop / totalpop, 2)
    # Finds q squared as a decimal.
    OUTPUT "recessives", q_sqrdaspop
    OUTPUT "qsqrd", qsqrd
    q <- Round_To_n(math.sqrt(qsqrd), 2)
    # finds the number of recessive alleles
    p <- Round_To_n(1 - q, 3) #finds the number of dominant alleles
    psqrd <- Round_To_n(p * p, 3) #finds the number of homozygous dominant
    OUTPUT "psqrd", psqrd
    pq <- Round_To_n(p * q, 3) # finds heterozygotes
ENDFUNCTION

```

## Pseudocode 2

```

FUNCTION generate_H_W(self):
    totalpop <- random.randint(10,1000)
    IF totalpop % 2 = 0:
        percentageRecessive <- random.randrange(0, 99, 2)
    ELSE:
        percentageRecessive <- random.randrange(1, 99, 2)
    ENDIF
    calculateAnswerfromRand()
    updateForm(hardy_weinberg)
ENDFUNCTION

FUNCTION calculateAnswerfromRand(self):
    percentageRecessive <- percentageRecessive
    totalpop <- totalpop
    recessiveInpop <- ((totalpop * percentageRecessive) / 100)
    #unrounded.Makes a percentage of the population have recessive alleles.
    q_sqrdaspop <- Round_To_n(recessiveInpop, 1) # now rounded
    qsqrd <- Round_To_n(q_sqrdaspop / totalpop, 3)
    # Finds q squared as a decimal.
    OUTPUT "recessives", q_sqrdaspop
    OUTPUT "qsqrd", qsqrd
    q <- Round_To_n(math.sqrt(qsqrd), 3)
    #finds the number of recessive alleles
    p <- Round_To_n(1 - q, 3)

```

```

#finds the number of dominant alleles
psqrd <- Round_To_n(p * p, 3)
#finds the number of homozygous dominant
OUTPUT "psqrd", psqrd
pq <- Round_To_n(p * q, 3) # finds heterozygotes
ENDFUNCTION

```

### Python Code 1

```

def calculateanswer(self)
    recessiveInpop = ((totalpop * percentageRecessive) / 100)
    # unrounded.Makes a percentage of the population have recessive alleles.

    q_sqrdaspop = Round_To_n(recessiveInpop, 1) # now rounded

    qsqrd = Round_To_n(q_sqrdaspop / totalpop, 3) # Finds q squared as a decimal.

    q = Round_To_n(math.sqrt(qsqrd), 3)
    # finds the number of recessive alleles and rounds

    p = Round_To_n(1 - q, 3)#finds the number of dominant alleles and rounds

    psqrd = Round_To_n(p * p, 3)#finds the number of homozygous dominant

    pq = Round_To_n(p * q, 3) # finds heterozygotes

```

### Python Code 2

```

def generate_H_W(self):

    self.totalpop = random.randint(10,1000)
    if self.totalpop % 2 == 0:
        self.percentageRecessive = random.randrange(0, 99, 2)
    else:
        self.percentageRecessive = random.randrange(1, 99, 2)
    self.calculateAnswerfromRand()
    self.updateForm(hardy_weinberg)

def calculateAnswerfromRand(self):
    percentageRecessive = self.percentageRecessive
    totalpop = self.totalpop

```

```

recessiveInpop = ((totalpop *
percentageRecessive) / 100) # unrounded.
Makes a percentage of the population have
recessive alleles.
q_sqrda = Round_To_n(recessiveInpop, 1)
#now rounded
qsqrda = Round_To_n(q_sqrda / totalpop, 3)
#Finds q squared as a decimal.
print("recessives", q_sqrda)
print("qsqrda", qsqrda)
q = Round_To_n(math.sqrt(qsqrda), 3)
#finds the number of recessive alleles
p = Round_To_n(1 - q, 3) #finds the number of dominant alleles
psqrda = Round_To_n(p * p, 3) #finds the number of homozygous dominant
print("psqrda", psqrda)
pq = Round_To_n(p * q, 3) # finds heterozygotes
print("p", p, "q", q)

print("2pq", pq * 2)

```

### Find k

Takes values sent by random generation or from a question and finds the answer. It then will update the question form.

### Pseudocode

```

FUNCTION findK(self, window):
    QtGui.QApplication.processEvents()
    n <- 0
    intermediary <- pow( conc_of_A,
order_of_A) * (pow( conc_of_B,
order_of_B))
    k <- rateOfReaction / intermediary
    correct <- False
    window.label.setText(_translate("Form", "The conc of reactant A is"
+ str("%.2E" % conc_of_A) +
" moldm-3. The conc of B is " + str("%.2E" %
conc_of_B) + " moldm-3\n"
"The order of A is " + str(
order_of_A) + " AND the order of B is "
+ str( order_of_B) + ". The rate of

```

```

        reaction is " + str("%.2E" % rateOfReaction), None))
ENDFUNCTION

```

## Python Code

```

def findK(self, window):

    QtGui.QApplication.processEvents()
    n = 0
    self.intermediary = pow(self.conc_of_A,
self.order_of_A) * (pow(self.conc_of_B,
self.order_of_B))
    self.k = self.rateOfReaction/self.intermediary
    correct = False
    window.label.setText(_translate
("Form", "The conc of reactant A is " + str("%.2E" %
self.conc_of_A) +
" moldm-3. The conc of B is "
+ str("%.2E" % self.conc_of_B) + "moldm-3\n"
"The order of A is " + str(
self.order_of_A) + " and the order of B is "
+ str(self.order_of_B) + ". The rate of reaction is " + str("%.2E" %
self.rateOfReaction) + " Find k of this
reaction.", None))

```

## 2.6 Security and Integrity

As Python is an interpreted language, students could in theory open it as a text file. An easy way around this would to use the current controls to not allow students access to the program files. However the program requires read write access to files in order to work. However data can be backed up to a secure location once a week. As the files are very small, each time the data is backed up it can be assigned a date

## 2.7 Prototype

I showed my end user an initial skeleton program that I had created. I responded to their requests and added a function to give the user an option to immediately have another question also, when their answer is incorrect then they are told it was incorrect and are asked to try again. With further testing I thought the user should be able to have the program to take them through the problem, as it is only a practice.

### 2.7.1 Command Line Prototype Code

```
author = 'Oscar'
import random
import decimal

def getChoice():
    choice = (input("Please choose an option. "))

    if choice == "1":
        print(" ")
        loadQuestion()
    if choice == "2":
        pass
    if choice == "4":
        pass
    return (choice)

def menu():
    choice = 0

    while choice != "q":
        print("1.Do a practice K question.")
        print("2.See a K example. ")
        print("3. Find units of K. ")
        print("4. Calculate the initial rate of reaction.")
        print("Press q to quit. ")
        choice = getChoice()
```

```

def format_e(n):
    a = '%E' % n
    return a

class findKreaction():
    def __init__(self, conc_of_A, conc_of_B,
                  order_of_A, order_of_B, rateOfReaction):
        self.conc_of_A = conc_of_A
        self.conc_of_B = conc_of_B
        self.order_of_A = order_of_A
        self.order_of_B = order_of_B
        # self.rateNumber = rateNumber
        # self.rateExponent = rateExponent
        self.rateOfReaction = rateOfReaction
        self.noOfMoles = (order_of_A + order_of_B)

    def walk_throughK(self):
        print("To calculate the rate constant, k, we must first
multiply the concentrations of our reactants together."
              "However the concentrations must be raised to the power
of their orders."
              "The concentrations of A and B are", self.conc_of_A, "and"
self.conc_of_B, "respectively."
              "The order of A is", self.order_of_A, "and the order of B
self.order_of_B, ".")
        print("This means our intermediary value is", self.intermediary,
              "We then divide our rate of reaction by this number. The r
of reaction is", self.rateOfReaction, ".")
        print("This equals", self.k)
        print("Next we must find the units of
k")
        print("To find the units of k we must cancel the number of moles

per dm cubed on both sides of the equation. "
              "We do this by adding the powers of the two reactants
and putting them into the formula: mol-x dm+x*3.
              Where x is the powers added together minus 1."

```



```

        "Unless the number of moles is
        one, then there are no units.")
if self.noOfMoles > 1:
    x = int((self.noOfMoles) - 1)

    print("The units are mol^-", x, "dm^+", 3 * x, "s^-1")
else:
    print("No units")

def findK(self):
    n = 0
    self.intermediary = pow(self.conc_of_A, self.order_of_A) *
    (pow(self.conc_of_B, self.order_of_B))
    self.k = self.rateOfReaction / self.intermediary
    correct = False
    print("The conc of reactant A is ", self.conc_of_A, "moldm^-3.
    The conc of B is", self.conc_of_B, "moldm^-3")
    print("The order of A is ", self.order_of_A, "and the order of B
    is", self.order_of_B)
    print("The rate of reaction is", self.rateOfReaction)
    while correct == False:
        k = self.k
        k = ("%3G" % (k))

        studentAnswer = input("Enter in your value of k
        to 3 sig fig: ")
        if studentAnswer == k:
            correct = True
            print("Well done, that was correct!")
            another = input("Would you like to try again? y/n")
            if another == "y":
                print(" ")
                loadQuestion()
            if another == "n":
                print("Goodbye!")
        if studentAnswer != k:
            n += 1
            print("Sorry that's not right, try again.")

```

```

        if n > 2:
            walkthrough = input("Would you like a walk-through?
            y/n")
            if walkthrough == "y":
                print(" ")
                findKreaction.walk_throughK(self)

def findunitsofK(self):
    print("To find the units of k we must cancel the number of moles
    per dm cubed on both sides of the equation. "
        "\nWe do this by adding the
        powers of the two reactants and
        putting them into the formula: "
        "\nmol-x dm+x*3. Where x is
        the powers added together minus 1."
        "\nUnless the number of moles is one, then
        there are no units.\n ")
    if self.noOfMoles > 1:
        x = int((self.noOfMoles) - 1)

        print("The units are mol-", x, "dm+", 3 * x, "s-1")
    else:
        print("No units")

    # def getRateOfReaction(self):
    # rateOfReaction = self.k*(pow(10,self.rateExponent))
    # return rateOfReaction

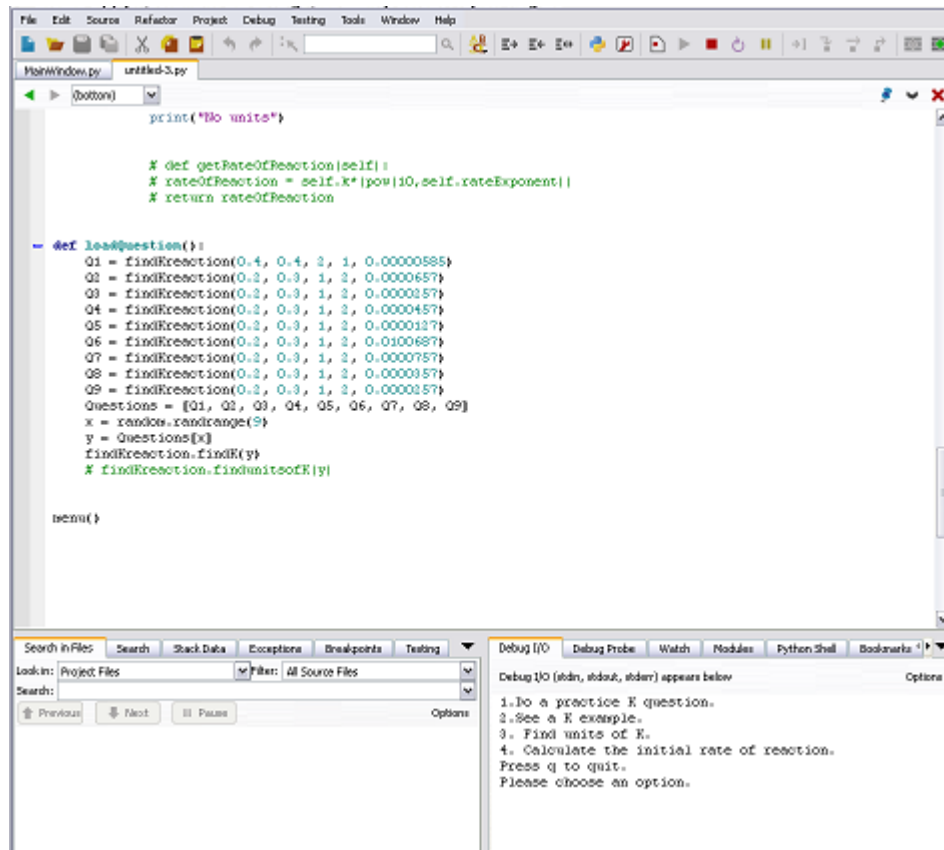
def loadQuestion():
    Q1 = findKreaction(0.4, 0.4, 2, 1, 0.00000585)
    Q2 = findKreaction(0.2, 0.3, 1, 2, 0.0000657)
    Q3 = findKreaction(0.2, 0.3, 1, 2, 0.0000257)
    Q4 = findKreaction(0.2, 0.3, 1, 2, 0.0000457)
    Q5 = findKreaction(0.2, 0.3, 1, 2, 0.0000127)
    Q6 = findKreaction(0.2, 0.3, 1, 2, 0.0100687)
    Q7 = findKreaction(0.2, 0.3, 1, 2, 0.0000757)

```

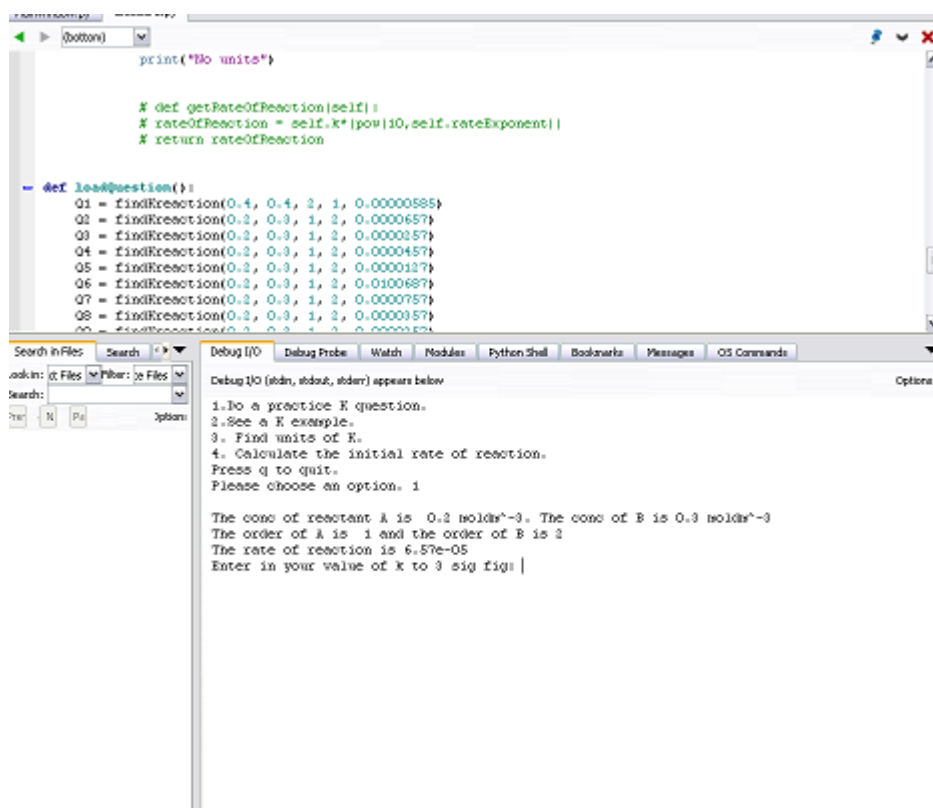
```
Q8 = findKreaction(0.2, 0.3, 1, 2, 0.0000357)
Q9 = findKreaction(0.2, 0.3, 1, 2, 0.0000257)
Questions = [Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q9]
x = random.randrange(9)
y = Questions[x]
findKreaction.findK(y)
# findKreaction.findunitsofK(y)
```

```
menu()
```

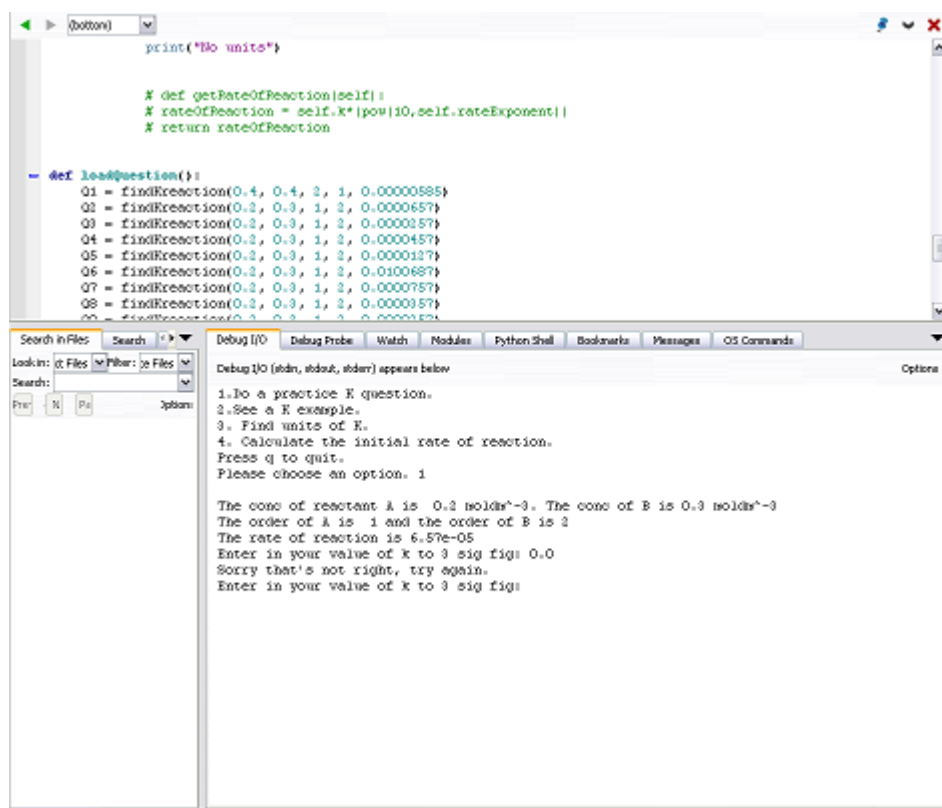
## Screenshots of CLI Prototype:



CLI Main Menu



*Attempting a question in CLI*



*Entering an incorrect answer*

After this I decided that if the student was only practising questions, they should not use up questions that teachers are marking them on. Therefore I created a function that generates random numbers, in a specific range, and rounds them to suit the question. This way, a student will never have encountered the questions they would be assessed on.

```

def generateK():
    sig = 3

    concofa = random.uniform(0.001, 0.9)
    concofa = round(concofa, sig-int(floor(log10(concofa)))-1)
    concofb = random.uniform(0.001, 0.9)
    concofb = round(concofb, sig-int(floor(log10(concofb)))-1)
    orderofA = random.randrange(1,3)
    orderofB = random.randrange(1,3)

```

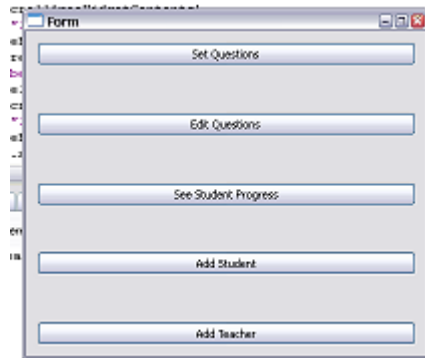


Figure 2.1: Early teacher main menu

```
rate = random.uniform(0.00000001, 0.0009)
rateofreaction = round(rate, sig-int(floor(log10(rate))))-1)
y = findKreaction (concofA, concofB, orderofA, orderofB, rateofreaction)
return y
```

This generates the numbers for the equation, creates a class using them, then returns it to the main program.

After consulting my end user again, and by using their request of something “MyMaths”-esque, I have implemented a GUI into my design using PyQt, which is event driven. This improves the ease of use of the program and also makes it more user friendly for people who do not use command line regularly.

## 2.7.2 GUI Prototype

To prototype the GUI I hand drew some of my windows and annotated functions:

I have also made a form window for the input of new users to the system. This window sends its’ variables to a function which adds the new student and repickles the list. This function is only available to teacher accounts. They can also create new teacher accounts in a similar manner.

The image shows a graphical user interface window titled "Form". Inside the window, the title "Create Student" is centered at the top. Below the title, there are seven input fields, each preceded by a label: "First Name:", "Middle Name(if applicable):", "Surname:", "Password:", "Age:", "Form:", and "Target Grade:". Each label is followed by a white rectangular text input box. At the bottom of the form area, there is a blue button with the text "Submit". The window has a standard operating system title bar with minimize, maximize, and close buttons on the right side.

Figure 2.2: The initial create student window



The image shows a graphical user interface for creating a student account. The main window is titled "Form" and contains a section titled "Create Student". The form has several input fields: "First Name:" with the value "Alex", "Middle Name(if applicable):" with the value "David", "Surname:" with the value "Demeza", "Password:" with the value "wololo", "Age:" with the value "17", "Form:" with the value "13JMG", and "Target Grade:" with the value "A". At the bottom of the form is a "Submit" button. A modal dialog box titled "python" is overlaid on the form, displaying the message "New Student Created." and an "OK" button.

Field	Value
First Name:	Alex
Middle Name(if applicable):	David
Surname:	Demeza
Password:	wololo
Age:	17
Form:	13JMG
Target Grade:	A

Figure 2.3: An account being created