A Level Science Teaching Aid

By

Oscar Daniel

13MJO

Analysis

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.

Calculations follow:

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.

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:

[Warning: Image ignored]

So, the Variance is 21,704.

And the Standard Deviation is just the square root of Variance, so:

Standard Deviation: $\sigma = \sqrt{21{,}704} = 147.32... = 147$ (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 in 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 CO2: 2.34/44 = 0.0532

Then apply these to the equation.

(0.0532*8.314*273)/100,000 = 0.0012074921 mˆ3

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 litres = 0.0000562 mˆ3

100,000*0.0000562)/(8.314*273) = 0.00247607416 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 = qˆ2 = 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. 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 % of heterozygotes, can you figure out the % of homozygous dominant? Does the % of homozygous dominant, heterozygotes and homozygous recessive individuals add up to 100%? 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) = pˆ2 = 0.09 or 0.3ˆ2

An Example Question

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.

[Warning: Image ignored]

Feedback Reports

To produce the most effective package for my end user, I have asked several questions regarding their needs and expectations. From this I was able to select the most appropriate equations to address and how best to structure my program.

[Warning: Image ignored]

Proposed System

Pupils will need to be able to attempt 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. Should allow students and staff to log in. Will generate own questions. 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.

Objectives

Essential Objectives

1. (a) 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.
   (b) Background Explanation Of Problems- The program should be able to give a small amount of detail initially if the student should request it.
   (c) 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.
   (d) Unique User Accounts- Teachers and pupils should have different accounts with different privileges.
   (e) 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.
   (f) 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.
   (g) 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.

(h) 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. 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.

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

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

OOP 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.

Design

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

Etc.

ˆ Student Only
*Teacher Only

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 ASCHII upper case character.	Please enter a target grade.	A
Parent Email	Correct email format.	“String”.”string”@”string”.”string”	Please enter a valid email address.	q04ademeza@qegsss.org.uk

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.

1 Overall System Design

Input	Process	Storage	Output
Question Data	Store Question	Question Pickle	Question stored succes fully.
Question Data	Answer Question	Question Pickle	Question Answer.
Student Data	Add Student	Pupil Pickle	Pupil Data stored succes fully.

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 answer in/correctly. OR Answer Sent.
Student/Teacher Password	Log In	Retrieve from Pupil / Teacher pickle	Logged in as "person" OR Username password comb nation is incorrect.

2 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)

box = container("Cardboard box")
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.

Data Transformation

Hardy - Weinberg

To find the value of 2pq, the number of heterozygotes, we must find q and p. We are given a means of finding q^2 in the question:

recessiveInpop = ((totalpop * percentageReccessive) / 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 reccessive 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

Find k

To find the rate constant k, we take the values the student has been given, place them in the reaction class and send those variables to a function that solves the equation. Then format the answer correctly.

```
def rateEquation(myReaction):
intermediary = (pow(myReaction.conc_of_A, myReaction.order_of_A)*
(pow(myReaction.conc_of_B, myReaction.order_of_B)))
myReaction.rateOfReeac intermediary
    print((format_e(myReaction.k)))
    print("\n")
    return float(myReaction.noOfMoles)
```

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

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.

```
__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 is",
self.order_of_B, ".")
            print("This means our intermediary value is", self.intermediary, "."
                  "We then divide our rate of reaction by this number. The reate
of reaction is", self.rateOfReaction,".")
            print("This equals", self.k)
            print("Next we must find the units of k \n")
            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: "
                  "\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 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, "moldmˆ-3. The
conc of B is", self.conc_of_B, "moldmˆ-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. "
              "\We do this by adding the powers of the two reactants and
putting them into the formula: "
          "\nmolˆ-x dmˆ+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 molˆ-", x, "dmˆ+", 3 * x, "sˆ-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()
    [Warning: Image ignored] [Warning: Image ignored]
    [Warning: Image ignored]
```

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)
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 and 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.

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

This is the teacher main menu:

[Warning: Image ignored]

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.

[Warning: Image ignored]

[Warning: Image ignored]

This will use the validation from the table above.

As I noted that all of the fields needed to create a teacher are included in the create student form, I consolidated the two windows into one add new account form.

[Warning: Image ignored]

I then added validation to the create account form to check for uncompleted fields, if any of the fields do not comply with my validation and if a username is already taken by another user.

[Warning: Image ignored]

For my set questions window, I initially tried to create a tree view model to select pupils by form group and questions by type and question set. However by using python this did not seem possible so I decided to use combo boxes.

[Warning: Image ignored]

Class Descriptions
class teacher():
def __init__(self, username, account_type, password, forename, surname, middlename, pupils):

class Teacher_Main_Window(QtGui.QWidget):
These functions set up the GUI window and define what happens when the user interacts with the page:
def __init__(self):
QtGui.QWidget.__init__(self)
self.setupUi(self)
def setupUi(self, Form):
self.retranslateUi(Form)
QtCore.QMetaObject.connectSlotsByName(Form)
def retranslateUi(self, Form):
These functions open other pages:
def makeQSet(self):
This opens the Make Question Set form
def showSetQuestions(self):
This opens the set questions form
def showLoginPage(self):
This shows the login page
def showCreateNewAccountWindow(self):
This opens the Create new account form
def showNewq(self):
This opens the add question form

class createaccount(QtGui.QWidget):
def __init__(self):
def setupUi(self, Form):
self.retranslateUi(Form)
def retranslateUi(self, Form):
def checktype(self):
Validates inputs and dynamically updates form.
def usernameinuse(self):
Checks if username is already in use, opens pop up box.
def fieldsNotComplete(self):
Opens pop up window if all fields are not completed correctly
def create_new_Student(self):

Collates information from the form and creates a new Student object then pickles it.

def studentconfirmation(self):
Opens a pop up window when a new object is pickled. Closes create new account form.
def create_new_Teacher(self):
Collates information from the form and creates a new Teacher object then pickles it.
def teacherconfirmation(self):
Opens a pop up window when a new object is pickled. Closes create new account form.

```
class SetQuestions(QtGui.QDialog):
\ \ def __init__(self):
     \ \ def setupUi(self, Form):
   \ \ def populateforms(self):
        \ \ \ \ Dynamically populates the forms box from the \ \ \ \ \ \ students list.

\bigskip


\bigskip


\bigskip

\ \ def populatepupils(self, text):
\ \ \ \ Dynamically populates the pupils box by searching \ \ \ \ through the students list
    \ \ def populateQSets(self):
\ \ \ \ Dynamically populates the Qsets box by searching the \ \ \ \ Qsets list.

\bigskip

    \ \ def giveQuestions(self):
\ \ Adds the name of the selected question set to the \ \ ?questionstoanswer? of the selecte

\bigskip

class Student_Main_Window(QtGui.Qwidget):
def __init__(self):
def setupUi(self, Form):
def retranslateUi(self, Form):
def showLoginPage(self):
\ \ Opens the login page after the ?Log Out? button is \ \ pressed.
def launchPracticeQwindow(self):
```

```
\ \ Launches the Practice Questions Window.

\bigskip

class Practice_Questions_Window(QtGui.Qwidget):
def __init__(self):
def setupUi(self, Form):
def retranslateUi(self, Form):

\bigskip

def launch_hardy_weinberg(self):
\ \ Launches the hardy weinberg question form by \ \ creating an object.
def launchPracticeQWindow(self):
\ \ Opens the rate constant question window by creating \ \ an object.
class MakeQSet(QtGui.QWidget):
ListID = len(QSets.QList)+1
newQSet = self.newList
name = self.lineEdit.text()
owner = username
newQSet = Question_Set(ListID, name, newQSet, owner)
    def __init__(self, QSet):
    def setupUi(self, Form):
    def retranslateUi(self, Form):
    def removeQuestion(self):
\ \ Removes a question from the new QSet box.
    def makeNewSet(self):
Takes the questions in the new QSet box and pickles them as one QSet object
    def SetConfirmation(self):
Opens a dialogue box confirming the creation of a QSet and closes the form.
    def addQuestionToList(self):
\ \ Copies a question to the new QSet box.
    def populateQuestions(self, questions):
Dynamically adds questions to the questions box from the questions pickle.

\bigskip

class AddQuestion(QtGui.QWidget):
QID = len(questions)+1
values = [convertTofloat(self.firstValue.text()), convertTofloat(self.secondValue.text()),co
newQuestion = Question(QID, self.questiontype, values)
self.questiontype = text
text = self.QuestiontypeBox.activated[str]
def __init__(self):
def setupUi(self, Form):
def retranslateUi(self, Form):
```

```
def typechosen(self, text):
Detects the type of question chosen, then updates the form depending on the selected type.
def validate(self, text):
Checks if the values entered are correct.
def create(self):
Creates a new question object with the used values and then pickles it.
def questionconfirmation(self):
Opens a dialogue box confirming the creation of a new Question and that it has been pickled.
def fieldsIncorrect(self):
Opens a dialogue box telling the user that some entered fields are not correct.
def typeNotSelected(self):
Opens a dialogue box telling the user the question type is not selected.
class Question(object):
self.question_type = questiontype
     self.ID = ID
     self.values = values
     self.conc_of_A = self.values[0]
     self.conc_of_B = self.values[1]
     self.rateOfReaction = self.values[2]
     self.order_of_A= self.values[3]
     self.order_of_B= self.values[4]
\ \ def __init__(self, ID, questiontype, values):
Detects the type of question sent and sends it to the right function. Also initialises the c
def kQuestion(self):
If the question is a Rate constant question it sets the correct Values[] to be the correct v
def hardy_weinberg(self):
If the question is a hardy-weinberg question it sets the correct Values[x] to be the correct

\bigskip

class Question_Set(object):
def __init__(self, ID, name, newQList, owner):
        self.QID = ID
        self.name = name
        self.QList = newQList
        self.owner = owner
class Practice_Q_Window(QtGui.QWidget):
self.answer = ('%.3G' % self.thisQuestion.k)

\bigskip

\ \      def __init__(self):
    \ \ \ \ def setupUi(self, Form):
\ \ def retranslateUi(self, Form):
\ \ def checkAnswer(self):
Checks if the answer the user entered is the correct one.
```

```
    def wrongorright(self):
Opens a dialogue box telling the user if the answer they inputted is correct or not. If it i

\bigskip

class student():
isOntarget = boolean)
self.forename = str
self.username = str
self.account_type = str
self.password = str
self.surname = str
self.middlename = str
self.answered = questionsanswered
self.questionstoanswer = array of strings
self.answeredcorrectly = int
self.studentid = int
self.form = str
self.targetgrade = str

\bigskip

def __init__(self, studentid, username, account_type, password, forename, surname, middlenam
def percentage(self):
Finds the percentage of questions the student has answered correctly.
def findgrade(self):

\bigskip

class hardy_weinberg(QtGui.QWidget):
pops = array of int
recessiveInpop = float
randomnumber = int
totalpop = int
self.totalpop = int
     self.q_sqrdaspop = int
     self.qsqrd = int
     self.psqrd = int
     self.q = int
     self.p = p int
self.pq = int
\ \ def __init__(self):
\ \ def setupUi(self, hardy_weinberg):
def retranslateUi(self, hardy_weinberg):
def generate_H_W(self):
Generates all of the values for a hardy Weinberg question.
```

```
def checkAnswerpq(self):
Checks if the answer the user entered is the correct one.
class findKreaction():
self.conc_of_A = float
self.conc_of_B = float
self.order_of_A = float
self.order_of_B = float
self.rateNumber = float
self.rateExponent = float
self.rateOfReaction = r float
self.noOfMoles = float
self.intermediary = float
self.k = float
correct = boolean
def __init__(self, conc_of_A, conc_of_B, order_of_A, order_of_B, rateOfReaction):

\bigskip


\bigskip


\bigskip


\bigskip

User Manual
Contents:
This Product.....
Installation.....
First Time Setup?????????????????????????????????????????.00

\bigskip


\bigskip


\bigskip


\bigskip


\bigskip
```

This Product

\bigskip

\bigskip

This product has been designed to allow people of varying technical ability to use it. This

\bigskip

\bigskip

The primary objective of this software is to give teachers and students a platform to develo

\bigskip

\bigskip

\bigskip

\bigskip