AQA qualification training

A-level Computer Science

Focus on Non Exam Assessment

BOOKLET 4

Published date: Summer 2016 version 1.0

Connect Four – Maths Revision Tool

Contents

Analysis

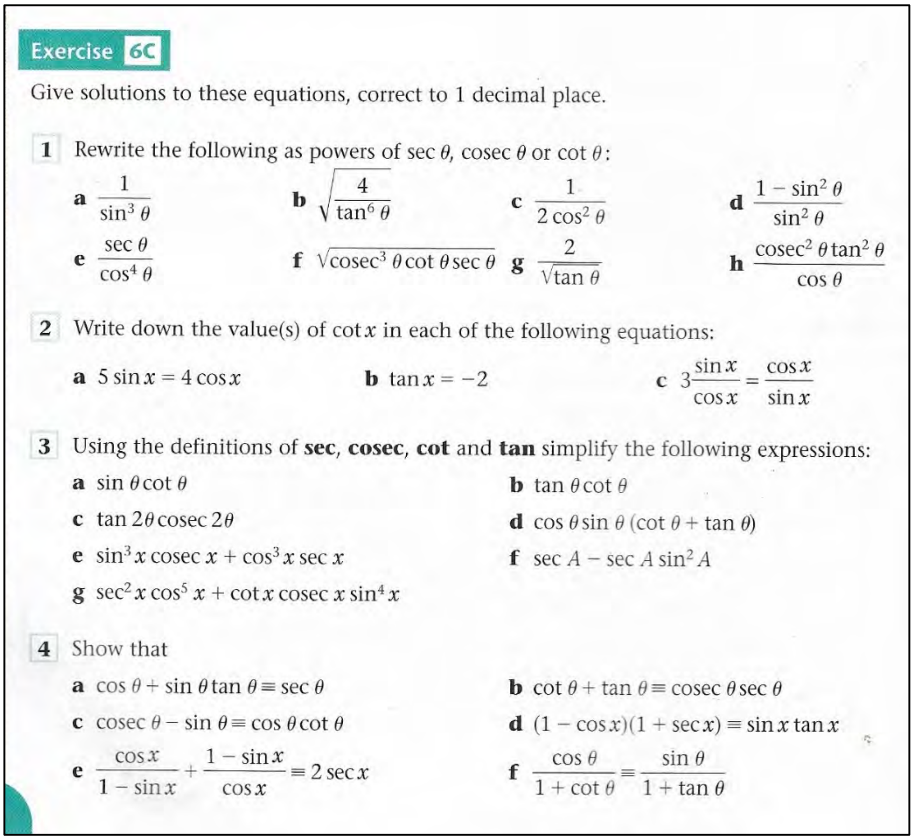
Problem background

At A High School, A level maths revision mainly consists of completing past paper questions and questions from a text book. This can make revision become tedious for the student, as there are a limited number of past paper questions for each topic, due to the number of past paper created. Other revision papers usually consist of repented questions from other papers, which can make some students simply remember how to answer a certain question and not understand how to solve it.

Current system

The current system involves the student finding a past paper, or text book question and then proceeding to answer question. For example, if a student wanted to revise differentiation (a topic which is frequent through A Level maths,) they would most likely find a past paper by either searching online ("C4 past paper" – would show results for past papers from the fourth core unit of A level maths,) or searching through first class, a service provided by the school for the use of email, as well as sharing of resources. Once the student has found a past paper, they would either work through in chronological order, or look for specific topics.

This is an example of a question from a textbook:



Exercise 6C

Give solutions to these equations, correct to 1 decimal place.

1 Rewrite the following as powers of sec θ, cosec θ or cot θ:

a $\frac{1}{\sin^3\theta}$ **b** $\sqrt{\frac{4}{\tan^6\theta}}$ **c** $\frac{1}{2\cos^2\theta}$ **d** $\frac{1-\sin^2\theta}{\sin^2\theta}$

e $\frac{\sec\theta}{\cos^4\theta}$ **f** $\sqrt{\operatorname{cosec}^3\theta\cot\theta\sec\theta}$ **g** $\frac{2}{\sqrt{\tan\theta}}$ **h** $\frac{\operatorname{cosec}^2\theta\tan^2\theta}{\cos\theta}$

2 Write down the value(s) of $\cot x$ in each of the following equations:

a $5\sin x = 4\cos x$ **b** $\tan x = -2$ **c** $3\frac{\sin x}{\cos x} = \frac{\cos x}{\sin x}$

3 Using the definitions of **sec**, **cosec**, **cot** and **tan** simplify the following expressions:

a $\sin\theta\cot\theta$ **b** $\tan\theta\cot\theta$

c $\tan 2\theta\operatorname{cosec}2\theta$ **d** $\cos\theta\sin\theta(\cot\theta+\tan\theta)$

e $\sin^3 x\operatorname{cosec}x+\cos^3 x\sec x$ **f** $\sec A-\sec A\sin^2 A$

g $\sec^2 x\cos^5 x+\cot x\operatorname{cosec}x\sin^4 x$

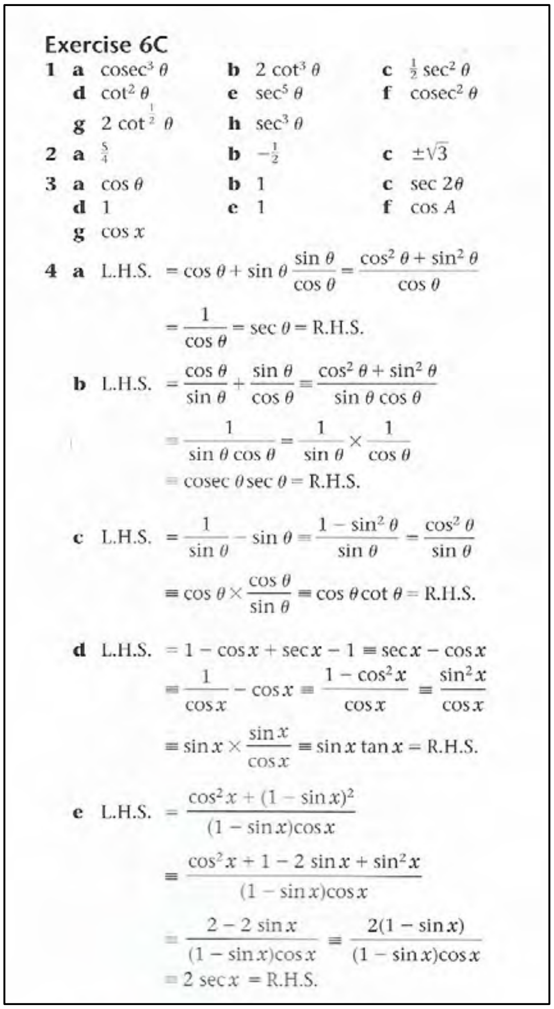
4 Show that

a $\cos\theta+\sin\theta\tan\theta\equiv\sec\theta$ **b** $\cot\theta+\tan\theta\equiv\operatorname{cosec}\theta\sec\theta$

c $\operatorname{cosec}\theta-\sin\theta\equiv\cos\theta\cot\theta$ **d** $(1-\cos x)(1+\sec x)\equiv\sin x\tan x$

e $\frac{\cos x}{1-\sin x}+\frac{1-\sin x}{\cos x}\equiv 2\sec x$ **f** $\frac{\cos\theta}{1+\cot\theta}\equiv\frac{\sin\theta}{1+\tan\theta}$

After the student has solved the question, they would look at the mark scheme for the question, to see whether they were correct. To do this for a past paper question the student would have to search for the year the paper was released followed by which unit the paper was in, for example: “C4 June 2012- mark scheme”. Alternatively, the student could go to first class, where the mark schemes and papers are available for the students at A High School. For this example, the answer would be found at the back of the text book, as shown:



Exercise 6C

1 **a** $\operatorname{cosec}^3 \theta$ **b** $2 \cot^3 \theta$ **c** $\frac{1}{2} \sec^2 \theta$
d $\cot^2 \theta$ **e** $\sec^5 \theta$ **f** $\operatorname{cosec}^2 \theta$
g $2 \cot^{\frac{1}{2}} \theta$ **h** $\sec^3 \theta$

2 **a** $\frac{5}{4}$ **b** $-\frac{1}{2}$ **c** $\pm\sqrt{3}$

3 **a** $\cos \theta$ **b** 1 **c** $\sec 2\theta$
d 1 **e** 1 **f** $\cos A$
g $\cos x$

4 **a** L.H.S. $= \cos \theta + \sin \theta \dfrac{\sin \theta}{\cos \theta} = \dfrac{\cos^2 \theta + \sin^2 \theta}{\cos \theta}$

$= \dfrac{1}{\cos \theta} = \sec \theta =$ R.H.S.

b L.H.S. $= \dfrac{\cos \theta}{\sin \theta} + \dfrac{\sin \theta}{\cos \theta} \equiv \dfrac{\cos^2 \theta + \sin^2 \theta}{\sin \theta \cos \theta}$

$\equiv \dfrac{1}{\sin \theta \cos \theta} = \dfrac{1}{\sin \theta} \times \dfrac{1}{\cos \theta}$

$\equiv \operatorname{cosec} \theta \sec \theta =$ R.H.S.

c L.H.S. $= \dfrac{1}{\sin \theta} - \sin \theta \equiv \dfrac{1 - \sin^2 \theta}{\sin \theta} = \dfrac{\cos^2 \theta}{\sin \theta}$

$\equiv \cos \theta \times \dfrac{\cos \theta}{\sin \theta} \equiv \cos \theta \cot \theta =$ R.H.S.

d L.H.S. $= 1 - \cos x + \sec x - 1 \equiv \sec x - \cos x$

$\equiv \dfrac{1}{\cos x} - \cos x \equiv \dfrac{1 - \cos^2 x}{\cos x} \equiv \dfrac{\sin^2 x}{\cos x}$

$\equiv \sin x \times \dfrac{\sin x}{\cos x} \equiv \sin x \tan x =$ R.H.S.

e L.H.S. $= \dfrac{\cos^2 x + (1 - \sin x)^2}{(1 - \sin x)\cos x}$

$\equiv \dfrac{\cos^2 x + 1 - 2 \sin x + \sin^2 x}{(1 - \sin x)\cos x}$

$\equiv \dfrac{2 - 2 \sin x}{(1 - \sin x)\cos x} \equiv \dfrac{2(1 - \sin x)}{(1 - \sin x)\cos x}$

$\equiv 2 \sec x =$ R.H.S.

If the user has answered the question incorrectly, they would most likely either refer to the example questions which have a step by step guide, or ask their teacher for assistance.

Algorithms used in the current system

Differentiation question

The student currently will apply the following method to solve a differentiation question:

Identify the power of the main bracket (as shown highlighted):

$$y = (Ax^{\mu} + Bx^{\beta})^{\alpha}$$

Find the differential of the main bracket (as shown highlighted):

$$y = \left(Ax^{\mu} + Bx^{\beta}\right)^{\alpha}$$

Giving:

$$u = Ax\mu + Bx\beta$$

$$\frac{du}{dx} = \mu Ax^{\mu\text{-}1} + \beta Bx^{\beta\text{-}1}$$

Use this information to find $\frac{dy}{dx}$ using the following formula:

$$\frac{dy}{dx} = \alpha\,(\frac{du}{dx})(\,Ax^{\mu} + Bx^{\beta}\,)^{\alpha\text{-}1}$$

$$\frac{dy}{dx} = \alpha\,(\,\mu Ax^{\mu\text{-}1} + \beta Bx^{\beta\text{-}1})(\,Ax^{\mu} + Bx^{\beta}\,)^{\alpha\text{-}1}$$

Substitute the given value for **x** to find the gradient at the point.

Example

(Scan example – in notepad)

Exponentials question

The student currently will apply the following method to solve an exponentials question:

$$\boldsymbol{A + Be^{(\frac{t}{x})} = C}$$

Minus the value of “**A**” from both sides.

$$\boldsymbol{Be^{(\frac{t}{x})} = C - A}$$

Divide both sides by the value “**B**”.

$$\boldsymbol{e^{(\frac{t}{x})} = \frac{C - A}{B}}$$

Take the natural log of both sides.

$$\boldsymbol{ln(e^{\left(\frac{t}{x}\right)}) = ln(\frac{C - A}{B})}$$
$$\boldsymbol{(\frac{t}{x})(ln(e) = ln(\frac{C - A}{B})}$$
$$\boldsymbol{\left(\frac{t}{x}\right) = ln(\frac{C - A}{B})}$$

Multiply both sides by the value of $\boldsymbol{x}$**.**

$$\boldsymbol{t = xln(\frac{C - A}{B})}$$

Example

(Scan example – in notepad)

Trigonometric equations question

The student currently will apply the following method to solve a trigonometric equation question:

$\boldsymbol{function = sin, cos\ or\ tan.}$ The function needed will change between questions. All values of $\boldsymbol{x}$ will be given between the range $\boldsymbol{0 \leq x \leq 360}$.

Identify the values $\boldsymbol{A, b\ and\ C.}$

$$\boldsymbol{a\ function^2(x) + b\ function(x) + c = 0}$$

Apply the quadratic equation with the values of $\boldsymbol{A, B\ and\ C}$, to get an answer in terms of the function. Two solutions will be generated because of the $\pm$ used in the quadratic formula.

$$\boldsymbol{\frac{-b\pm\sqrt{b^2-4ac}}{2a} = N, M}$$

This will result in the equation:

$$\boldsymbol{function(x) = N, M}$$

When $\boldsymbol{function = Cos}$:

$$\boldsymbol{x = Cos^{-1}(N, M)}$$

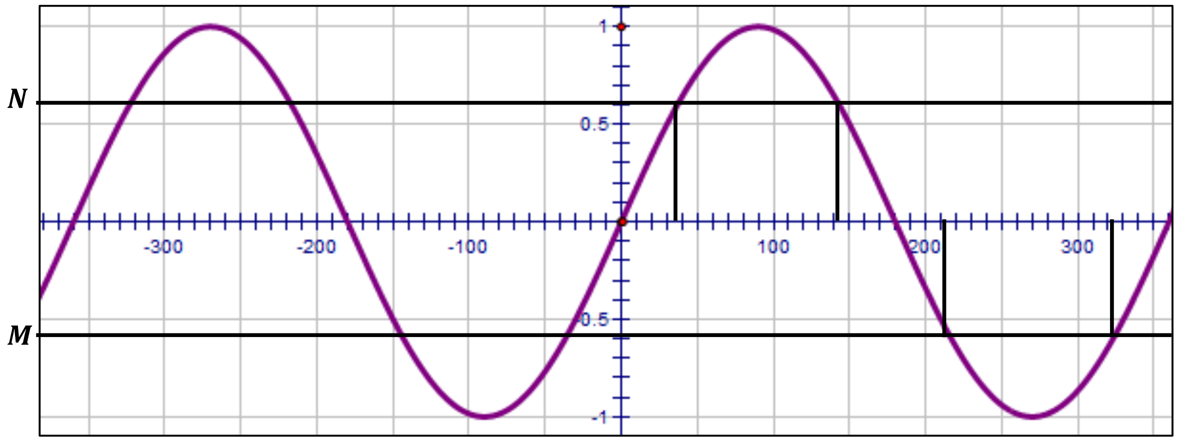
When $\boldsymbol{function = Sin}$:

$$\boldsymbol{x = Sin^{-1}(N, M)}$$

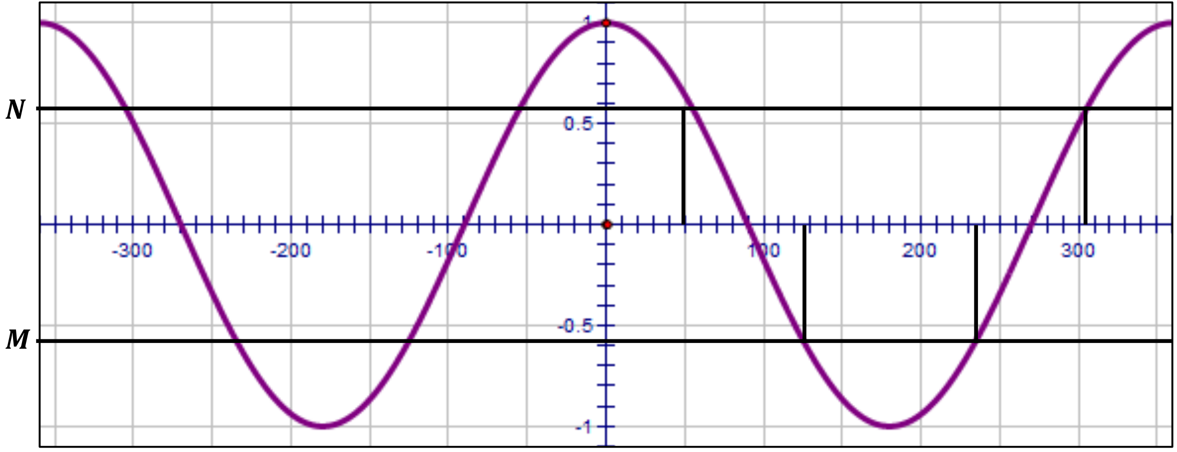
When $\boldsymbol{function = Tan}$:

$$\boldsymbol{x = Tan^{-1}(N, M)}$$

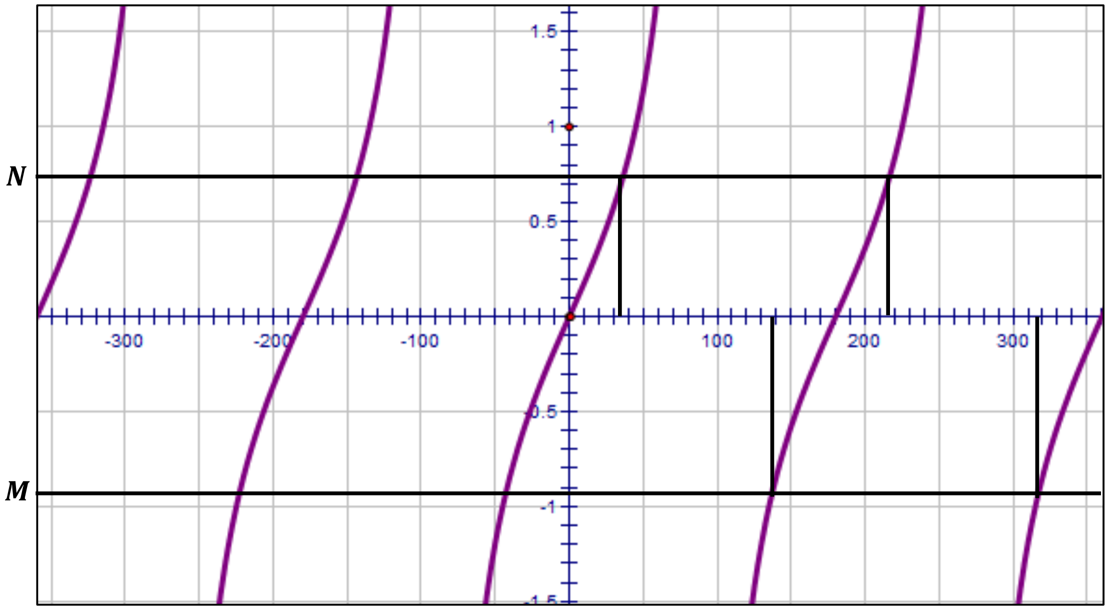
The values given by the calculator may not be in the range required. To get the values in this range, the student would have to sketch the graph of the function, where they can then see where the other values will lie, as shown in the graphs below. Any values of $\boldsymbol{x}$ not between 0 and 360 are not required. There may be multiple solutions.



Cosine Graph



Sine Graph



Tangent Graph

Example

(Scan example – in notepad)

Objectives

General objectives

1. The system must improve the logical and quick thinking skills of students.

2. To offer students a way to improve their maths skills as well as playing a problem solving game.

3. The target user of the system will be someone studying A2 maths.

4. The system should be easily understood by the target audience. The terminology used must be understood by the target audience. The language and grammar must be of a high standard.

5. The system should be user friendly as otherwise the terminology used could make the questions difficult to understand. User friendly systems are also more likely to not frustrate the user. For the system to be user-friendly it must have a general theme:

    - The layout must be consistent but must also be efficient to use.

    - All font sizes must be consistent and easy to read. The font used must also be consistent throughout.

    - The system must have a consistent colour scheme.

    - The themes should enable the user to confidently use the system and experience it to its full potential.

6. The system must be able to run quickly and efficiently.

7. The navigation of the system must be clear. The user must be able to navigate around the system with no problems.

Specific objectives

1. The system will be able to run in school, or on the student's personal computer.

2. The student will be able to complete a different set of questions each time, to stop them from being able to remember answers, and instead improve their maths skills.

3. The system will have at least three topics from the A2 maths syllabus.

4. The system should have different options to change the size of the game.

5. The student will be able to play the game against the computer, so it is not required to have two players.

6. The computers first move will be different each time to stop it from generating the same group of moves each game.

7. The user will only get a turn on the game if they get a question correct.

8. The type of question that is asked to the user is different each time.

9. The number of moves in which the student wins the game will be shown after completion of the game.

10. The number of questions that the user answers correctly in a row will be shown after completion of the game.

11. The game instructions will be shown as the user is playing the game, therefore will not require a separate instructions page.

12. The user will be rewarded in game when they get a set number of questions correct, in a row.

13. The user will have the ability to change the difficulty of the computer (how many moves are calculated in advanced).

Prospective Users and Acceptable Limitations

The prospective users of the system will be students studying A level maths at A High School. The students studying A level maths will have the needed computing knowledge to be able to use the application, and use it to improve their maths skills. As maths is a practice subject, the application should generate a new question before each of the users turns.

System limitations

1. Programming knowledge – Using the python library 'pygame,' meant I had to spend some time learning the functions which are available, and how they can be applied to my program.

- Version 1 –

  Working connect four game, no connection to questions.

- Version 2 –

  Working connect four game and question one. Question two partially working.

- Version 3 –

  Working connect four game and all three questions working. Game and questions currently not connected.

- Version 4 –

  Questions and game connected. Questions display in python shell rather than in pygame window, causing pygame to not respond until the loop has been exited.

- Version 5 –

  Questions and game connected with questions displaying in pygame window. Answer does not update on the screen when the user types, and instead has to be typed in the python shell window.

- Version 6 –

  Questions and game connected with questions and the users answer in the pygame window.

Data Sources and Destination

Data source for existing applications

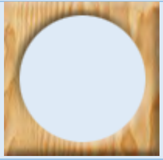
Data in the existing system			
Data	Source	Description	Destination
Sample questions	Edexcel C3 and C4 text books	Practice questions which the student can use to revise from. Also includes mark schemes where the student can check their work.	Student
Past exam questions	Edexcel website	Questions from the examinations that previous year have taken.	Student
Past paper mark schemes	Edexcel website	Answers to the examinations which previous years have taken.	Student

Data source for proposed application

Data in the proposed system			
Data	**Source**	**Description**	**Destination**
For all questions			
Generated question	Question subroutine	Before the user has a turn on the game, each time they will have to answer a randomly generated question. Printed onto the game window for the user to see.	question (variable)
Attempts	Question subroutine	The user will have five attempts at each question.	attempts (variable)
Question 1			
	Question1 subroutine	The first answer solved by the program. To be compared with the users solutions.	x1 (variable)

Solutions	Question1 subroutine	The third solution solved by the program, using x1. To be compared with the users solutions.	x1_2 (variable)
	Question1 subroutine	The second answer solved by the program. To be compared with the users solutions.	x2 (variable)
	Question1 subroutine	The forth solution solved by the program, using x2. To be compared with the users solutions.	x2_2 (variable)
Number of solutions	Question1 subroutine	Questions will have between one and four solutions,	usersolutionsno (variable)
Question 2 and Question 3			
Solution	Question2/Question3 subroutine	The solution to the question generated.	x1 (variable)

Data Dictionary

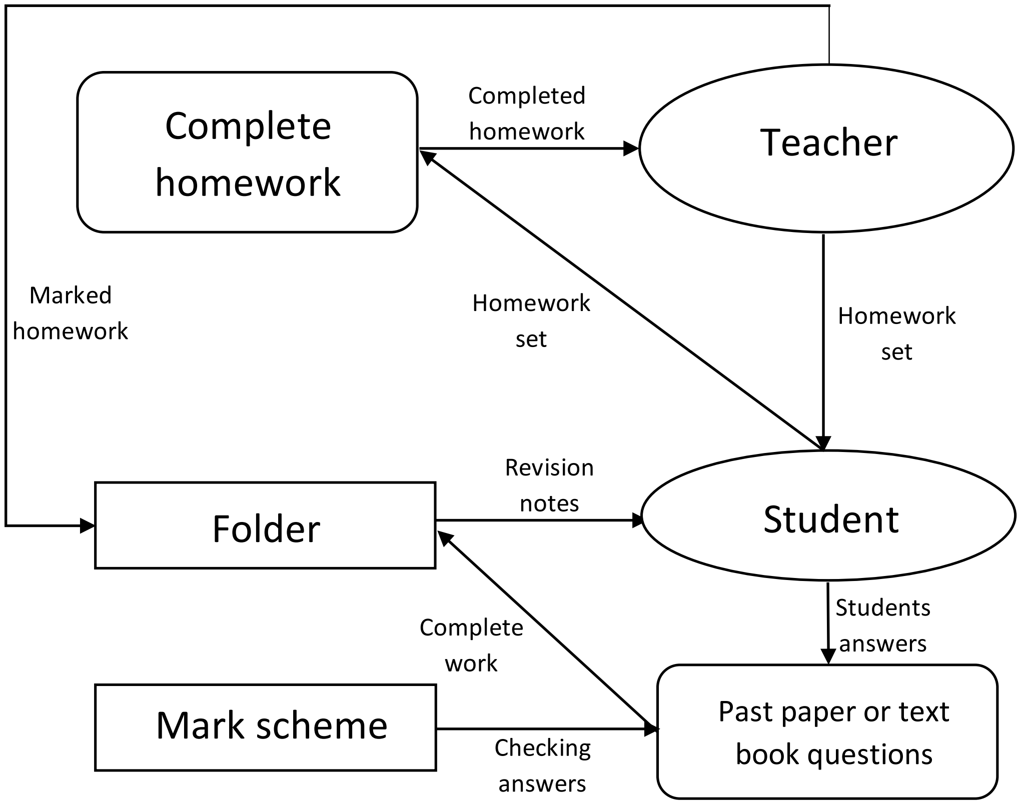
Field name	Data type	Example	Description
difficulty	Integer	1	How many moves the computer calculates in advance. Options being either "1" or "2."
Empty space	None	None	Used to find the lowest free space on a column.
Player	String	"player"	Used in subroutines to find which players turn, player or computer.
Computer	String	"Computer"	Used in subroutines to find which players turn, player or computer.
Red token	Image		The image used for the red counter.
Red token	Image		The image used for the black counter.
Board	Image		The image used to create the game board.
User Answer	String	"Incorrect"	Used to store if the user got the question correct.
Potential Moves	String	(3,4)	Finds the moves available for the computer.
Discriminant	Integer	$\sqrt{b^2 - 4ac}$	Used to stop the program from attempting to square root a negative number.
Quadratic	String	$y = Sin^2x + Sinx + 4$	Used to create a quadratic equation which can be printed onto the screen.
Question answer	Float	72.32	The answer to the question.
Solutions	Integer	4	The number of solutions to the question.
attempts	Integer	5	The number of attempts the user has at a question.

Data Flow Diagrams

Process = ▢ External entity = ⬭ Data Store = ▭ Data flow = →

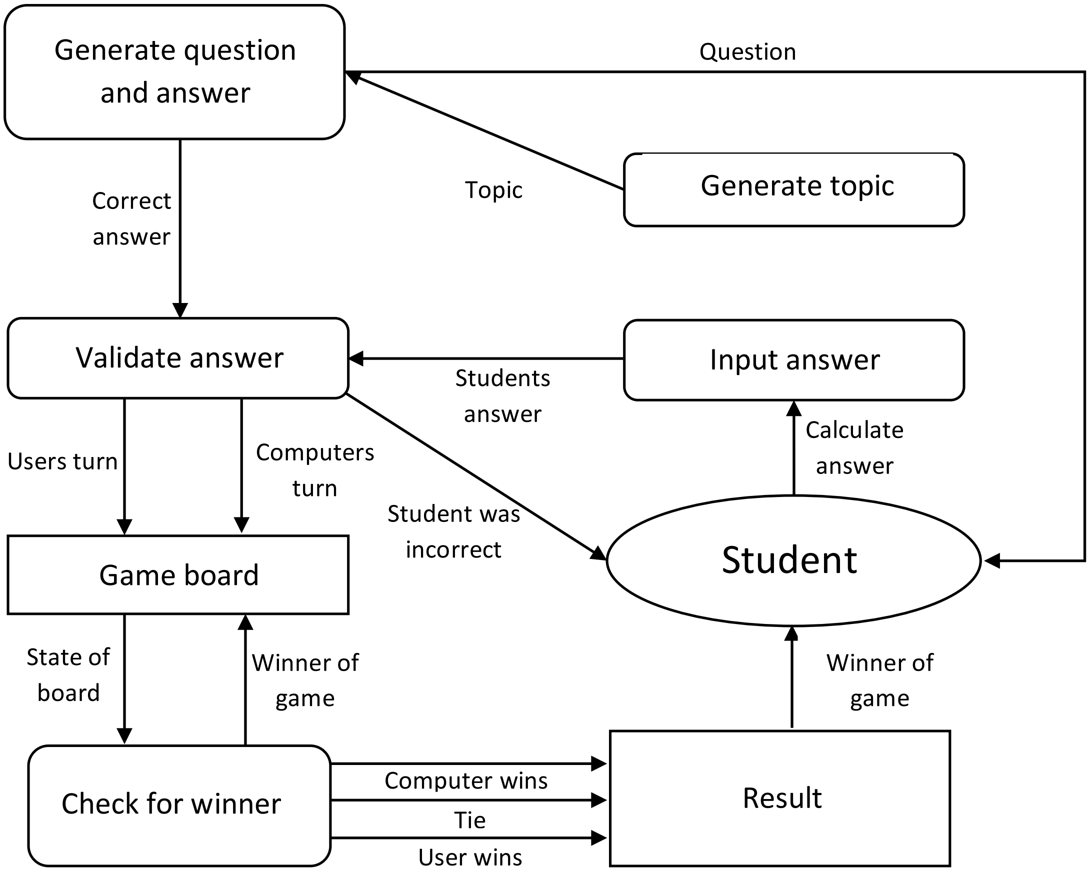
Existing

The data flow diagram shows how a student currently revises for an A level maths exam.



Proposed

The data flow diagram shows how a student will revise for an A level maths exam with the new system.



Evidence of Use of Appropriate Analysis Techniques

When writing the analysis, I kept in mind what the end user would want the program to achieve. The analysis of the current and proposed system was written up based on feedback from students studying maths at A High School. The user wanted the program to be a revision tool which provided more of an incentive to independently revise maths. The new system would provide this, as the incentive to win the game is in place.

It was also clear that the system would be able to run on many systems, which is why python was chosen. As it is completely free, and easy for the user to install.

The user also wanted there to be a range of topics, so the program did not become repetitive. Based on this I chose the three most common topics which have appeared in maths exam papers.

Design

Overall System Design

Inputs	Processes	Outputs
Question answer	Computers move	Computers move
Users turn on the game	Possible moves	Users move
Press escape to quit	Calculate question answers (Trigonometry)	Question was incorrect
	Calculate question answers (Exponentials)	Question was correct
	Calculate question answers (Differentials)	
	Winner of the game	

Description of Modular Structure of System

Menu

The menu is the first page which the user will see when the program ran. On this page, there will be two buttons, one to start the game, and one which will quit the game. Above this, there will be the title of the game, “Connect 4”. There will be counters which are animated to fall down the screen while the user hasn’t clicked one of the buttons.

Game screen

This screen will show the state of the game board. The user will see this screen at the very start of the game, and every time a question is answered. When it is the computers turn on the game, the black counter will be animated to move up and over the top of the board, and then will be dropped down the column which is calculated to be the best move. When it is the users turn, they will have to drag and drop a red counter over the top of the board, which it will then be animated to fall down the selected column.

Question screen

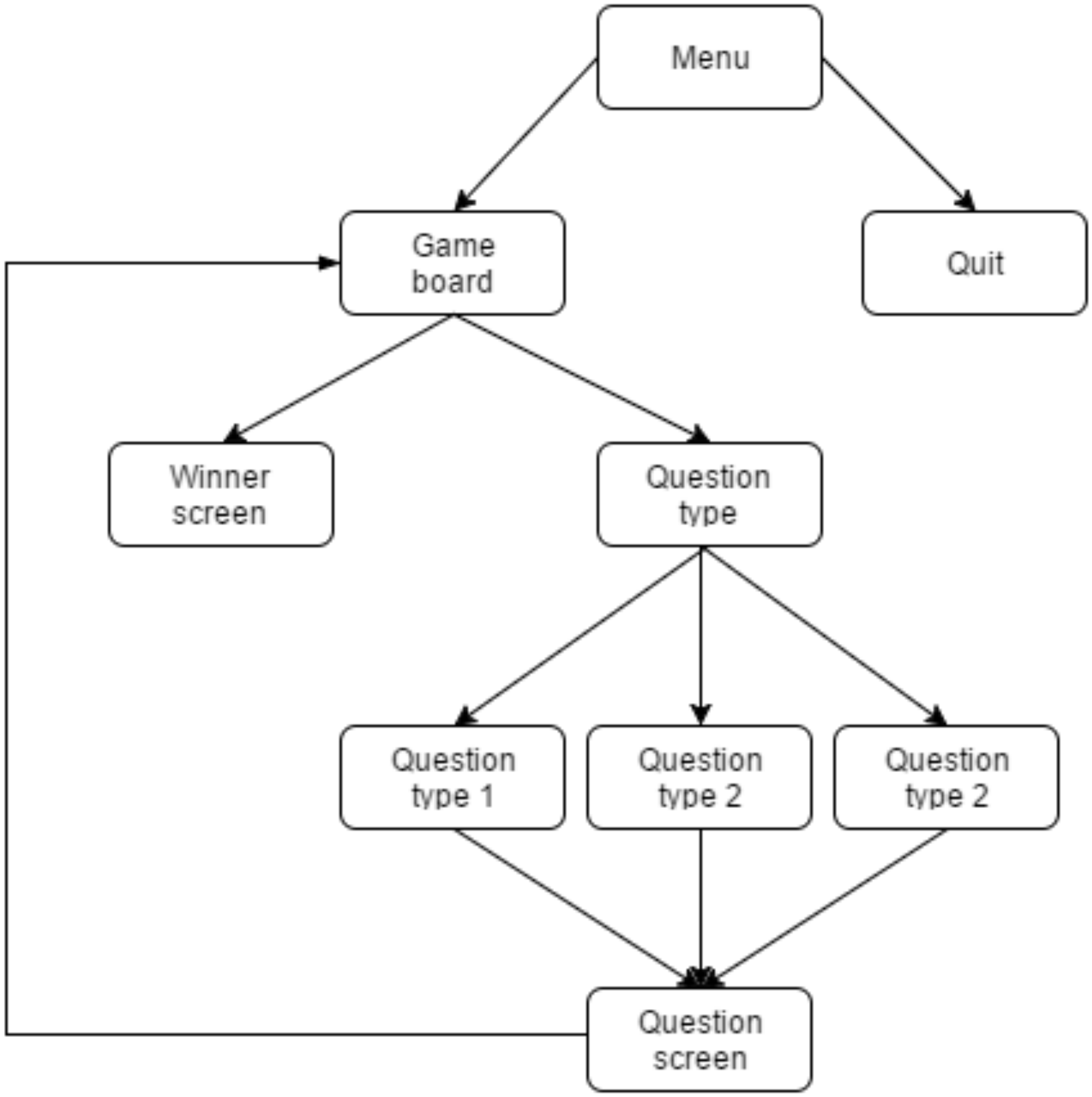
This is the screen which the user will be taken to after the computer has had a turn. The screen will look the same regardless of the type of question, with the only difference being the text. The user will calculate their answer and they type it in on this screen. They will have five attempts to correctly answer this question.

To get the type of question which will be displayed on the screen, a random selection out of the three will add the equation to the question.

End game screen

This screen will be displayed when either the computer or the user has won the game. An image will be displayed showing who has won, or if the game was a tie.

This diagram shows how the pages are linked together:



Definition of data requirements (Design Data Dictionary)

Refer to Data dictionary in Analysis section.

Validation Required

Validation will ensure that data is input accurately. The data will need to be processed to ensure that it is within an acceptable range. This is needed especially for the users input to answer a question. The answer which they type will be converted to a string to avoid errors when converting to a float. For example, if the user would to try and type in a string for an answer, and then the program proceeded try and convert it to a float, this could potentially crash the program depending on what characters they typed. To keep the data which is inputted from the user consistent, the answers will be taken to two decimal places.

Field Name	Validation Checks	Description	Error message	Data	Caught
Correct number of solutions	if solutions == 1 and usersolutionsno == str(solutions):	Validates that the user has typed in the correct number of solutions	Incorrect number of solutions	3 (When solutions = 2)	Yes
User solution (question 1)	usersolution1 != str(x1) and attempts > 1	Validates that the user still has attempts left to answer the question when they correctly answered it.	Incorrect solution, 4 attempts remaining.	Wrong answer to the question	Yes
User solution (question 2)	usersolution != str(x1) and tries > 1:	Validates that the user still has attempts left to answer the question when they correctly answered it.	Incorrect solution, 4 attempts remaining.	Wrong answer to the question	Yes

User solution (question 3)	usersolution != str(sol1) and attempts > 1	Validates that the user still has attempts left to answer the question when they correctly answered it.	Incorrect solution, 4 attempts remaining.	Wrong answer to the question	Yes
User dragging token over full column	if column < 0 or column >= (boardWidth) or board[column][0] != empty:	Stops the user from placing a token on a full column.	No message displayed. Token will not be placed in column. User will re-do their turn.	Dragging token over full column.	Yes
Answer to question calculated by program	format(float(x1),'.2f')	Formats the answer calculated to two decimal places. The question will state that the users answer should be given to two decimal places.	No message.	1.34	Yes

Algorithm Design

Start game

Start game
This pseudo code shows how the game will initiate when it is run.
Algorithm

```
IF firstGame = True:
    Turn = computer
    ShowHelp = True
ELSE:
    IF randomInteger[0,1] => 0:
        Turn = computer
    ELSE:
        Turn = player
    END IF
    ShowHelp = False
END IF
mainBoard = GetNewBoard
WHILE True:
    IF turn => player:
        GetPlayerMove
        IF showHelp:
            showHelp = False
        END IF
        IF IsWinner:
            winnerImage = PlayerWinnerImage
            BREAK
        END IF
        Turn = computer
    ELSE:
        Column = GetComputerMove
        AnimateComputerMoving
        MakeMove
        IF IsWinner:
            WinnerImage = TieWinnerImage
            BREAK
        END IF
        Turn = Player
    END IF
    IF Board = Full
        WinnerImage = TieWinnerImage
    END IF
END WHILE
WHILE True:
    DrawBoard
    DisplayWinnerImage
    FOR Event:
        IF Event => QUIT:
            QUIT
        END IF
    END FOR
END WHILE
```

Draw board

Draw board

This pseudo code shows how the game board will be displayed on the screen.

Algorithm

```
FillBackground(BackgroundColour)
SpaceSize = 50
SpaceRectangle = Rectangle(0,0,SpaceSize,SpaceSize)
FOR x in RANGE[BoardWidth]:
    FOR y in RANGE[BoardHeight]:
        SpaceRectangle[TopLeft] = Top left of board
        IF Board[x][y] => red:
            DisplayImage(RedTokenImage)
        ELSEIF Board[x][y] => black:
            DisplayImage(BlackTokenImage)
        END IF
    END FOR
END FOR
IF ExtraToken <> None:
    IF ExtraToken[Colour] = red:
        DisplayImage(RedTokenimage,Extra space)
    END IF
    IF ExtraToken[Colour] = black:
        DisplayImage(BlackTokenimage, (ExtraToken[x], ExtraToken[y], Extra space))
    END IF
END IF
FOR x in RANGE[BoardWidth]:
    FOR y in RANGE[BoardHeight]:
        SpaceRectangle[TopLeft] = Top left of board
        DisplayImage(BoardImage, SpaceRectangle)
    END FOR
END FOR
DisplayImage(RedTokenImage, RedPileRectangle)
DisplayImage(BlackTokenImage, BlackPileRectangle)
```

Player Move

Player Move

This pseudo code shows how the players move is retrieved.

Algorithm

```
GetQuestionType
DraggingToken = False
TokenX, TokenY = None, None
IF UserAnswer = Correct:
    WHILE True:
        FOR Event:
            IF Event => QUIT:
                QUIT
            ELSEIF Event => MouseDown AND NOT DraggingToken:
                DraggingToken = True
                TokenX, TokenY = Event.Position
            ELSEIF Event => MouseMotion AND DraggingToken:
                TokenX, TokenY = Event.Position
            ELSEIF Event =>MouseUp AND DraggingToken:
                IF TokenY <YMargin AND TokenX > XMargin AND tokenX < WindowWidth – Xmargin:
                    Column = (tokenX - xMargin) / spaceSize
                    IF IsValidMove:
                        AnimateDroppingToken
                        Board[column][GetLowestSpace] = red
                        DrawBoard
                    END IF
                END IF
                TokenX, TokenY = None, None
                DraggingToken = False
            END IF
        END FOR
        IF TokenX <> None and TokenY <> None:
            DrawBoard(board, Users position)
        ELSE:
            DrawBoard(Board)
        END IF
        IF IsFirstMove:
            DisplayImage(ArrowImage)
        END IF
    END WHILE
IF UserAnswer => Incorrect:
    DrawBoard(Board)
END IF
```

Animate dropping token

Animate dropping token

This pseudo code shows how the token is made to look like it is falling down a column on the board.

Algorithm

```
X = XMargin + Column * SpaceSize
Y = YMargin - SpaceSize
DropSpeed = 1.0

WHILE True:
    Y = Y + DropSpeed
    IF (Y – YMargin)/SpaceSize >=lowestEmptySpace:
        RETURN
    END IF
    DrawBoard (Board, New Move)
END WHILE
```

Getting computers move

Getting computers move

This pseudo code shows how the computers move is retrieved.

Algorithm

```
PotentialMoves = GetPotentialMoves(Board, Black, Difficulty)
BestMoveFitness = -1
FOR a IN PotentialMoves:
    IF PotentialMoves[a] > BestMoveFitness and IsValidMove(Board, a):
        BestMoveFitness = PotentialMoves[a]
    END IF
END FOR
BestMoves = []
FOR a IN PotentialMoves:
    IF PotentialMoves[a] > BestMoveFitness and IsValidMove(Board, a):
        BestMoves.APPEND[a]
    END IF
END FOR
RETURN RandomChoice(BestMoves)
```

Getting potential moves

Getting potential moves

This pseudo code shows how the potential moves the computer can take are found.

Algorithm

```
IF lookAhead = 0 OR isBoardFull(board):
  RETURN [0] * boardWidth
ENDIF
IF tile = red:
  enemyTile <- black
ELSE:
  enemyTile <- red
ENDIF
potentialMoves <- [0] * boardWidth
FOR firstMove in range(boardWidth):
  dupeBoard <- copy.deepcopy(board)
  IF not isValidMove(dupeBoard, firstMove):
    continue
  ENDIF
  makeMove(dupeBoard, tile, firstMove)
  IF isWinner(dupeBoard, tile):
    potentialMoves[firstMove] <- 1
    break
  ELSE:
    IF isBoardFull(dupeBoard):
      potentialMoves[firstMove] <- 0
      ELSE:
         for counterMove in range(boardWidth):
            dupeBoard2 <- copy.deepcopy(dupeBoard)
            IF not isValidMove(dupeBoard2, counterMove):
               continue
            ENDIF
            makeMove(dupeBoard2, enemyTile, counterMove)
            IF isWinner(dupeBoard2, enemyTile):
               potentialMoves[firstMove] <- -1
               break
            ELSE:
               results <- getPotentialMoves(dupeBoard2, tile, lookAhead - 1)
               potentialMoves[firstMove] += (sum(results) / boardWidth) / boardWidth
           ENDIF
      ENDIF
   ENDIF
ENDFOR
RETURN potentialMoves
```

Differentiation question

Getting computers move

This pseudo code shows how the differentiation question is generated.

Algorithm

```
p=True
while p = True:
   A <- random.randint(-10,10)
   while A = 0:
      A <- random.randint(-10,10)
   ENDWHILE
   B <- random.randint(1,10)
   while B = 0:
      B <- random.randint(1,10)
   ENDWHILE
   C <- random.randint(-10,10)
   while C = 0:
      C <- random.randint(-10,10)
   ENDWHILE
   D <- random.randint(1,10)
   while D = 0:
      D <- random.randint(-10,10)
   ENDWHILE
   E <- random.randint(1,10)
   while E = 0 OR E = 1:
      E <- random.randint(-10,10)
   ENDWHILE
   X <- random.randint(-20,20)
   while X = 0 OR X = -1 OR X = 1:
      X <- random.randint(-15,15)
```

```
    ENDWHILE
    while (D = 1 AND B = 1) OR D = B:
        D <- random.randint(-10,10)
        B <- random.randint(-10,10)
    ENDWHILE
    sol1 <- 0.00001
    F <- str((X)**(B-1))
    IF F = "0":
        F <- 0
    ELSE:
        F <- F[-1]
    ENDIF
    G <- str((C*D)*(X)**(D-1))
    IF G = "0":
        G = 0
    ELSE:
        G <- G[-1]
    ENDIF
    H <- str(A*(X)**(B))
    IF H = "0":
        H <- 0
    ELSE:
        H <- H[-1]
    ENDIF
    I <- str(C*(X)**(D))
    IF I = "0":
        I = 0
    ELSE:
        I <- I[-1]
    ENDIF
    WHILE A = 0 OR B = 0 OR C = 0 OR D = 0 OR E = 0 OR E = 1 OR X = 0 OR X = 1 OR X = -1 OR (B = 1 AND D == 1) OR B == D or (A*X**(B)+C*X**(D)) == 0 or E*((A*B)*(X)**(B-1)+(C*D)*(X)**(D-1)) == 0 or F[-1] == "0" or G[-1] == "0" or H[-1] == "0" or I[-1] == "0":
        A <- random.randint(-10,10)
        WHILE A = 0:
            A <- random.randint(-10,10)
        ENDWHILE
        B <- random.randint(1,10)
        WHILE B = 0:
            B <- random.randint(1,10)
        ENDWHILE
        C <- random.randint(-10,10)
        WHILE C = 0:
            C <- random.randint(-10,10)
        ENDWHILE
        D <- random.randint(1,10)
        WHILE D = 0:
            D <- random.randint(-10,10)
        ENDWHILE
        E <- random.randint(1,10)
        WHILE E = 0 OR E = 1:
            E <- random.randint(-10,10)
        ENDWHILE
        X <- random.randint(-15,15)
        WHILE X = 0 OR X = 1 OR X = -1:
            X <- random.randint(-15,15)
        ENDWHILE
        WHILE (B = 1 AND D = 1) OR B = D:
            B <- random.randint(-10,10)
            D <- random.randint(-10,10)
        ENDWHILE
        F <- str((X)**(B-1))
        IF F = "0":
            F <- 0
        ELSE:
            F <- F[-1]
        ENDIF
        G <- str((C*D)*(X)**(D-1))
        IF G = "0":
            G = 0
        ELSE:
            G <- G[-1]
        ENDIF
        H <- str(A*(X)**(B))
        IF H = "0":
            H <- 0
        ELSE:
            H <- H[-1]
        ENDIF
        I <- str(C*(X)**(D))
        IF I = "0":
            I = 0
        ELSE:
            I <- I[-1]
        ENDIF
    ENDWHILE
    sol1 <- E*((A*B)*(X)**(B-1)+(C*D)*(X)**(D-1))*((A*(X)**(B))+(C*(X)**(D)))**(E-1)
    IF sol1 < -500 OR sol1 > 500 OR type(sol1) != int:
        p <- True
    ELSE:
        equation <- "y="+"("+str(A)+"x^("+str(B)+")+"+str(C)+"x^("+str(D)+"))"+"^"+str(E)
        IF C < 0:
            equation <- "y <- ("+str(A)+"x^("+str(B)+")"+str(C)+"x^("+str(D)+"))"+"^"+str(E)
        ENDIF
        IF C > 0:
            equation <- "y <- ("+str(A)+"x^("+str(B)+")+"+str(C)+"x^("+str(D)+"))"+"^"+str(E)
        ENDIF
        IF C = 1:
            equation <- "y <- ("+str(A)+"x^("+str(B)+")+"+"x^("+str(D)+"))"+"^"+str(E)
        ENDIF
        IF C = -1:
            equation <- "y <- ("+str(A)+"x^("+str(B)+")-"+"x^("+str(D)+"))"+"^"+str(E)
        ENDIF
        IF D = 1:
            equation <- "y <-("+str(A)+"x^("+str(B)+")+("+str(C)+"x)"+"^"+str(E)
        ENDIF
        IF A = 1 AND C < 0:
            equation <- "y <- ("+"x^("+str(B)+")"+str(C)+"x^("+str(D)+"))"+"^"+str(E)
        ENDIF
        IF A = 1 AND C > 0:
            equation <- "y <- ("+"x^("+str(B)+")+"+str(C)+"x^("+str(D)+"))"+"^"+str(E)
        ENDIF
        IF A = 1 AND C = 1:
            equation <- "y <- ("+"x^("+str(B)+")+"+"x^("+str(D)+"))"+"^"+str(E)
        ENDIF
        IF A = 1 AND C = -1:
            equation <- "y <- "+"("+"x^("+str(B)+")-"+"x^("+str(D)+"))"+"^"+str(E)
        ENDIF
        IF A = -1 AND C < 0:
            equation <- "y <- (-x^("+str(B)+")"+str(C)+"x^("+str(D)+"))"+"^"+str(E)
        ENDIF
        IF A = -1 AND C > 0:
            equation <- "y <- (-x^("+str(B)+")+"+str(C)+"x^("+str(D)+"))"+"^"+str(E)
        ENDIF
        IF A = -1 AND C = 1:
            equation <- "y <- (-x^("+str(B)+")+"+"x^("+str(D)+"))"+"^"+str(E)
        ENDIF
        IF A = -1 AND C = -1:
            equation <- "y <- (-x^("+str(B)+")-"+"x^("+str(D)+"))"+"^"+str(E)
        ENDIF
        IF B = 1 AND C > 0:
            equation <- "y <- ("+str(A)+"x+"+str(C)+"x^("+str(D)+"))"+"^"+str(E)
        ENDIF
        IF B = 1 AND C < 0:
            equation <- "y <- ("+str(A)+"x"+str(C)+"x^("+str(D)+"))"+"^"+str(E)
        ENDIF
        IF B = 1 AND C = 1:
            equation <- "y <- ("+str(A)+"x+"+"x^("+str(D)+"))"+"^"+str(E)
        ENDIF
        IF B = 1 AND C = -1:
            equation <- "y <- ("+str(A)+"x-"+"x^("+str(D)+"))"+"^"+str(E)
        ENDIF
        IF D = 1 AND C > 0:
            equation <- "y <- ("+str(A)+"x^("+str(B)+")+"+str(C)+"x)"+"^"+str(E)
        ENDIF
        IF D = 1 AND C < 0:
            equation <- "y <- ("+str(A)+"x^("+str(B)+")"+str(C)+"x)"+"^"+str(E)
        ENDIF
        IF D = 1 AND C = 1:
            equation <- "y <- ("+str(A)+"x^("+str(B)+")+"+"x)"+"^"+str(E)
        ENDIF
        IF D = 1 AND C = -1:
            equation <- "y <- ("+str(A)+"x^("+str(B)+")-"+"x)"+"^"+str(E)
        ENDIF
        IF D = 1 AND A = 1:
            equation <- "y <- ("+"x^("+str(B)+")+("+str(C)+"x)"+"^"+str(E)
        ENDIF
        IF D = 1 AND A = 1 AND C > 0:
            equation <- "y <- ("+"x^("+str(B)+")+"+str(C)+"x)"+"^"+str(E)
        ENDIF
        IF D = 1 AND A = 1 AND C < 0:
            equation <- "y <- ("+"x^("+str(B)+")"+str(C)+"x)"+"^"+str(E)
        ENDIF
        IF D = 1 AND A = 1 AND C = -1:
            equation <- "y <- ("+"x^("+str(B)+")-x)"+"^"+str(E)
        ENDIF
        IF D = 1 AND A = -1 AND C > 0:
            equation <- "y <- (-x^("+str(B)+")+"+str(C)+"x)"+"^"+str(E)
        ENDIF
        IF D = 1 AND A = -1 AND C < 0:
            equation <- "y <- (-x^("+str(B)+")"+str(C)+"x)"+"^"+str(E)
        ENDIF
        IF D = 1 AND A = -1 AND C = -1:
            equation <- "y <- (-x^("+str(B)+")-x)"+"^"+str(E)
        ENDIF
        IF D = 1 AND A = 1 AND C = 1:
            equation <- "y <- (x^("+str(B)+")"+"+x)"+"^"+str(E)
        ENDIF
        IF D = 1 AND C = 1 AND A = -1:
            equation <- "y <- (-x^("+str(B)+")+x)"+"^"+str(E)
        ENDIF
        IF B = 1 AND C > 0 AND A = -1:
            equation <- "y <- (-x+"+str(C)+"x^("+str(D)+"))"+"^"+str(E)
        ENDIF
        IF B = 1 AND C < 0 AND A = -1:
            equation <- "y <- (-x"+str(C)+"x^("+str(D)+"))"+"^"+str(E)
        ENDIF
        IF B = 1 AND C = 1 AND A = -1:
            equation <- "y <- "+"(-x+x^("+str(D)+"))"+"^"+str(E)
        ENDIF
        IF B = 1 AND C = -1 AND A = -1:
            equation <- "y <- (-x-x^("+str(D)+"))"+"^"+str(E)
        ENDIF
        IF A = 1 AND B = 1 AND C > 0:
            equation <- "y <- (x+"+str(C)+"x^("+str(D)+"))"+"^"+str(E)
        ENDIF
        question <- str("Find the gradient on the curve: "+equation+", when x <- "+str(X))
        DisplayText(Question)
        p <- False
    ENDIF
ENDWHILE
OUTPUT (sol1)
attempts <- 5
usersolution <- ""
DisplayText("Solution")
while 1:
    IF KEYDOWN:
        IF KEY = ENTER: BREAK
        ENDIF
        usersolution += Key
        IF Key = BACKSPACE:
            usersolution <- usersolution[0:-2]
        ENDIF
        DisplayText(usersolution)
    ENDIF
ENDWHILE
while usersolution != str(sol1) AND attempts > 1:
    attempts <- attempts - 1
    usersolution <- ""
       while 1:
    IF KEYDOWN:
        IF KEY = ENTER: BREAK
        ENDIF
        usersolution += Key
        IF Key = BACKSPACE:
            usersolution <- usersolution[0:-2]
        ENDIF
        DisplayText(usersolution)
    ENDIF
ENDWHILE
IF usersolution = str(sol1):
    UserAnswer <- "Correct"
ELSE:
    UserAnswer <- "Incorrect"
    Wait(2 Seconds)
ENDIF
OUTPUT (UserAnswer)
RETURN UserAnswer
```

Class definitions

All of the classes used are inherited from the Pygame library. Pygame allows for object orientated programming through per defined functions.

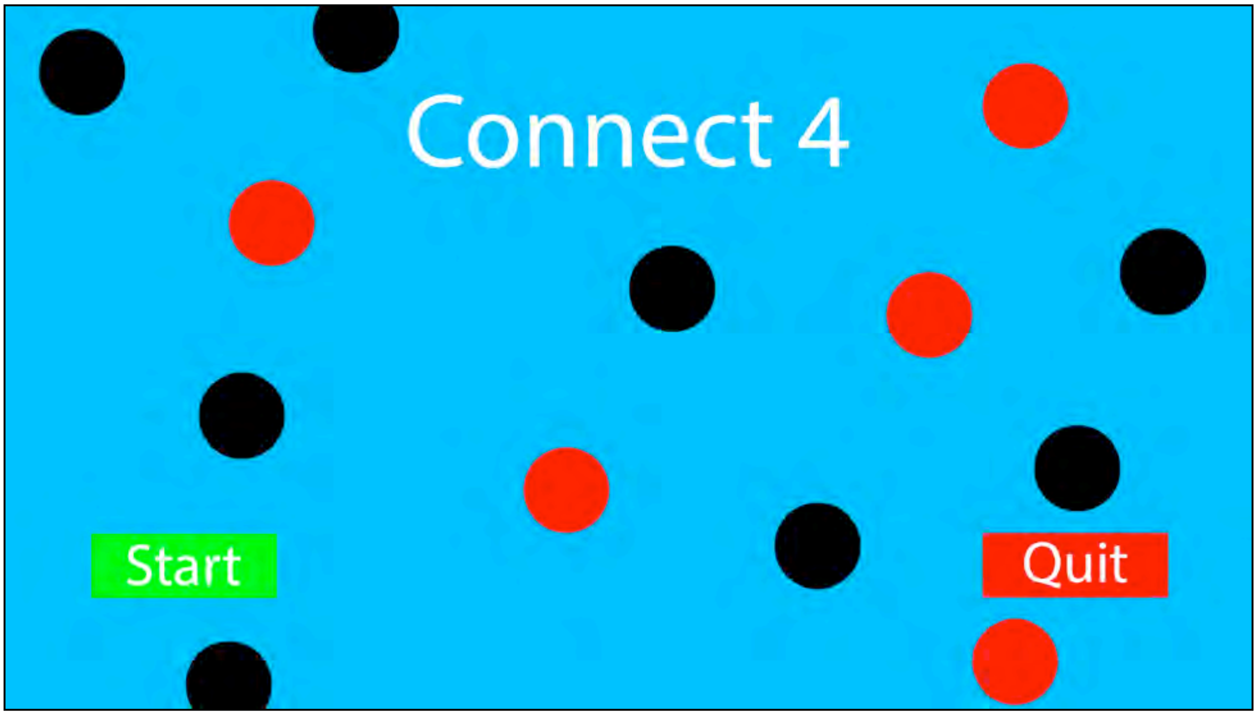
Some of the classes used are buttons, however, this is not a direct feature provided by pygame. To create buttons I used rectangles and overlaid text. To give the button more of an interactive appearance, the colour of the button will become brighter when the user hovers their mouse over it.

Other use of classes involves animating images, and placing images on the screen in general. This is called the "blit" function with in pygame. The mouse position and key presses are also used to make the game interactive for the user.

User interface

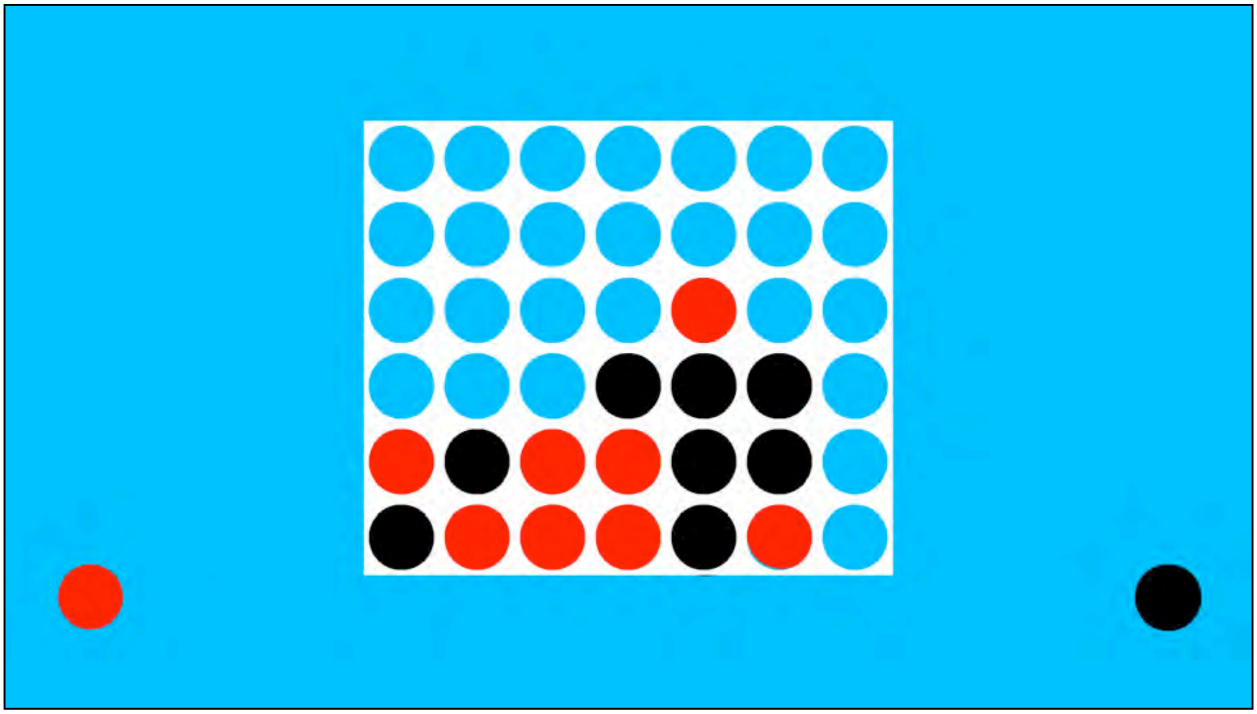
All screenshots used for the user interface section are prototypes and are subject to change.

Menu screen



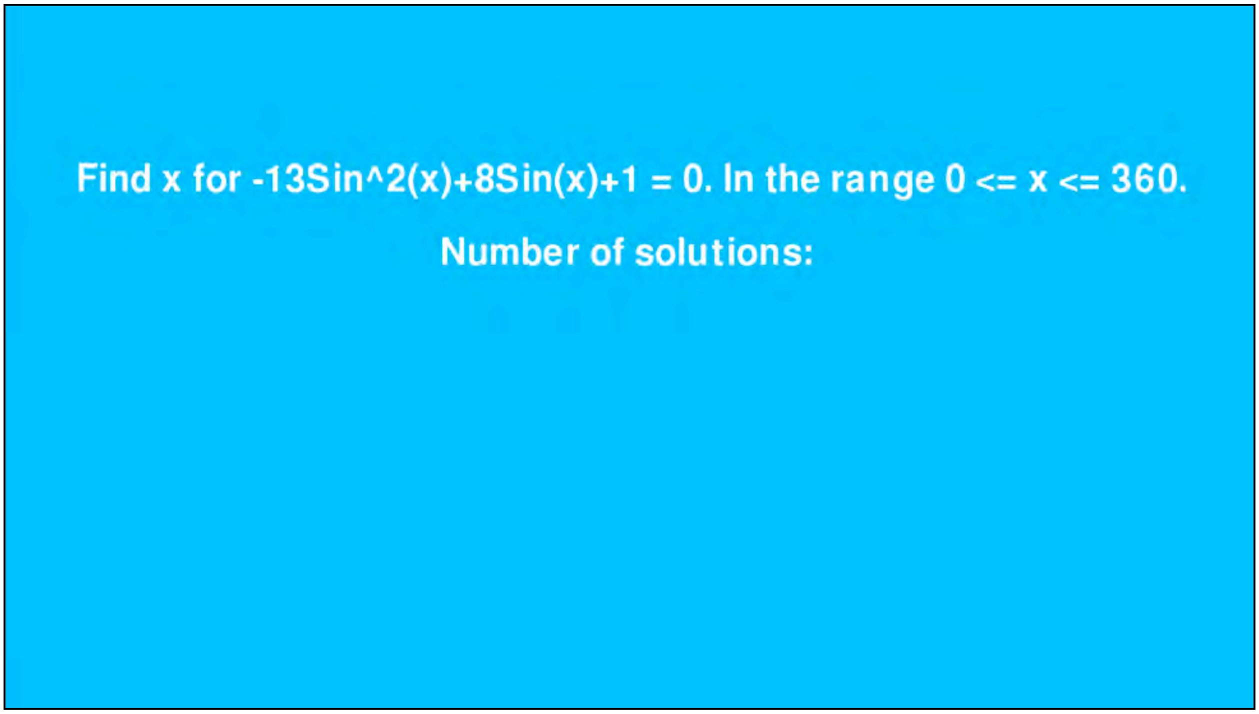
I chose to have a menu screen to clearly show what the game is. There will be two buttons, to start the game and to quit the game. The counters will be animated to fall and rise on the screen to give the system more of an interactive feel.

Game screen



The game screen is kept simplistic, as there will be a separate screen for the question. This is to stop the screen from looking overcrowded, and keep the program user friendly. The default game board will be a standard, 7x6 grid, however there will be functionality to change this. As the size of the board increases, the size of the tokens and individual board positions will be scaled down. This is to ensure that all of the board will fit onto the game window. The maximum size of the board will be 9x9, to stop any scaling problems, with images becoming unclear, or click boxes becoming too small.

Question one screen

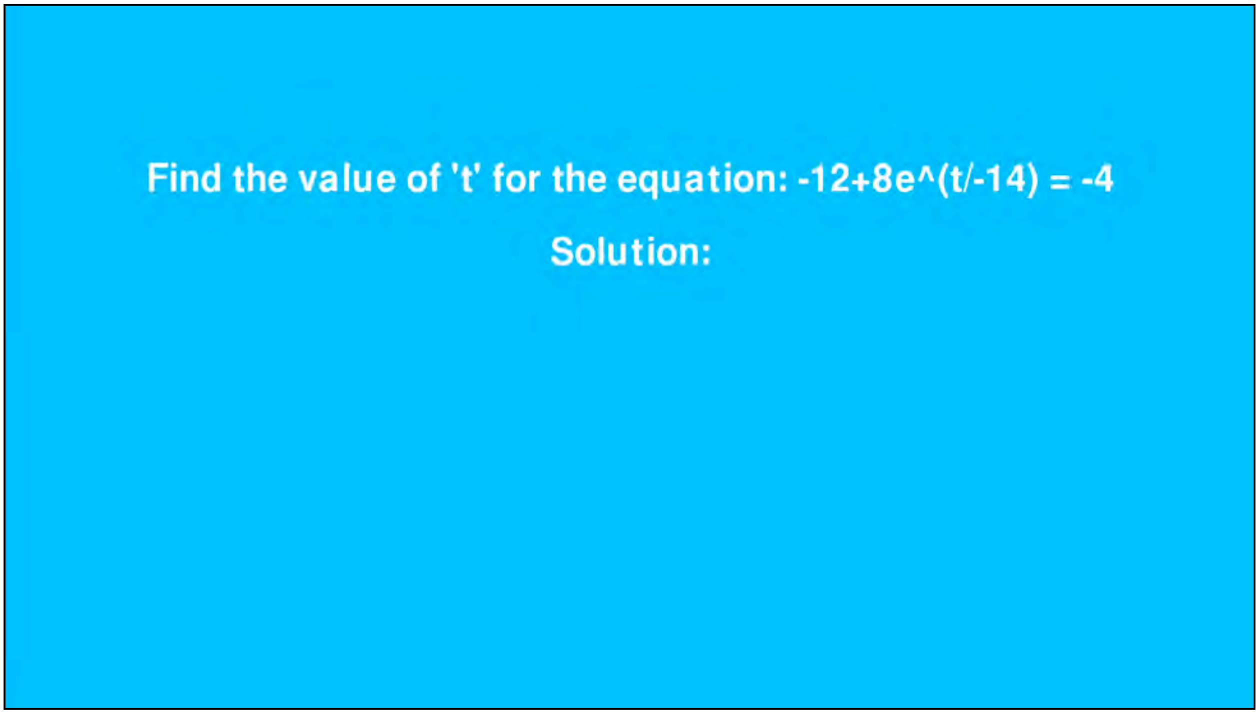


Find x for -13Sin^2(x)+8Sin(x)+1 = 0. In the range 0 <= x <= 360.

Number of solutions:

This screen will appear immediately after the computer has had a turn at the game. The equation shown on the screen will be replaced each time, as it is randomly generated. For the user to input their answer, they simply begin typing, and what they type will appear on screen. The colour choices allow for the text to be easily read, increasing the user friendliness of the program. The font used will be Arial, as this font comes pre-installed with windows, and will not cause problems when launching the program with fonts that are not installed. The text "Number of solutions", will change to "solution one", "Solution two", "solution three" and "solution four", depending on how many solution there are to the question.

Question two screen

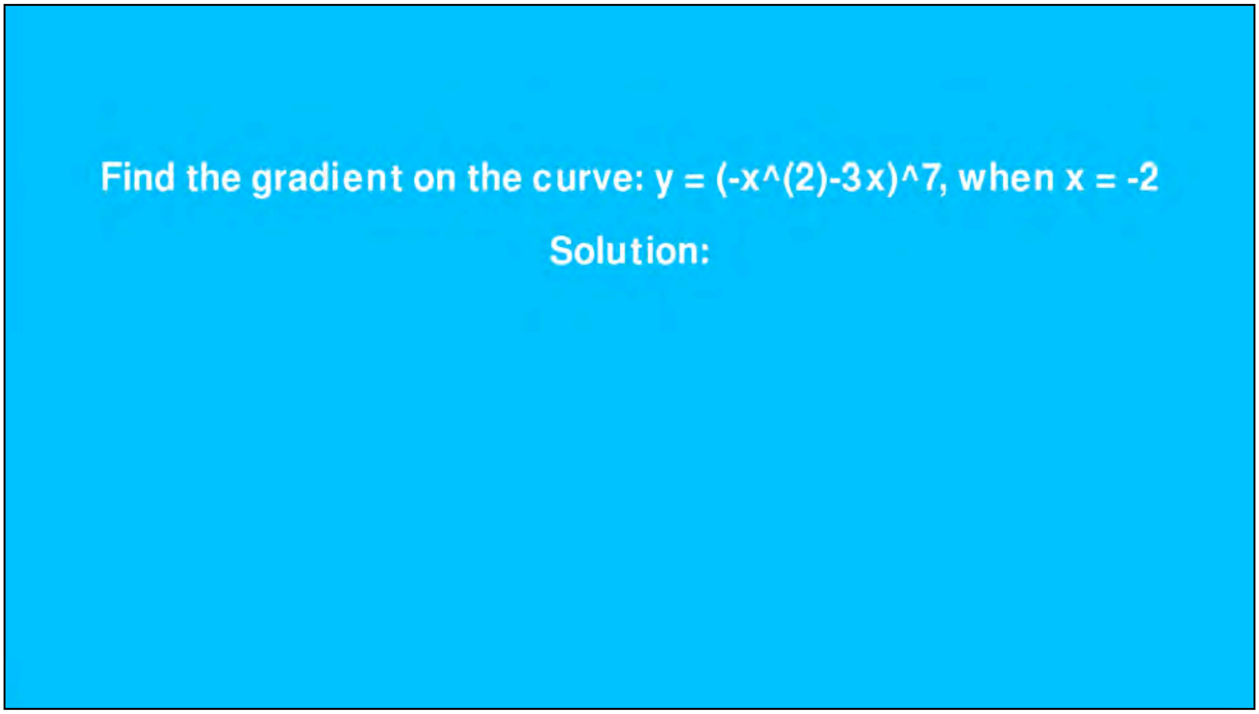


Find the value of 't' for the equation: -12+8e^(t/-14) = -4

Solution:

This screen is very similar looking to the question one screen. The only difference being the question. Once again, the question will be randomly generated each time, and will be displayed to the user before their turn. To input their answer, the user can simply begin typing, and what they have typed will be displayed on the screen.

Question three screen

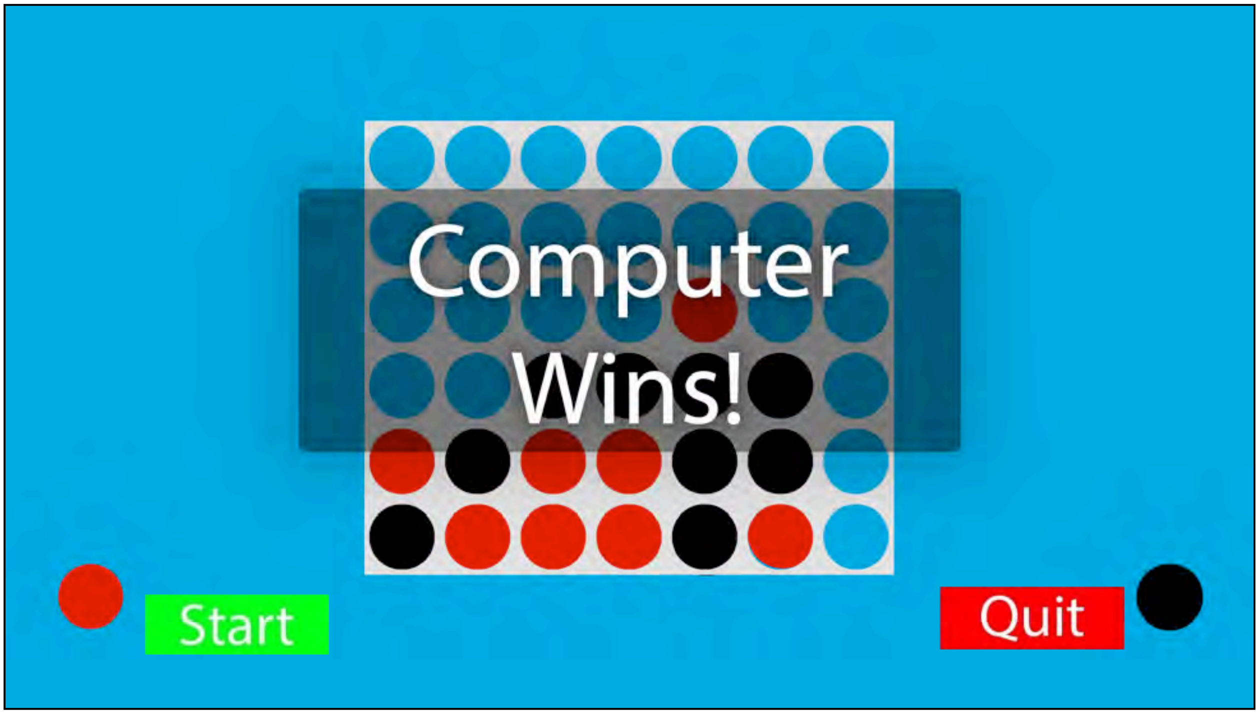


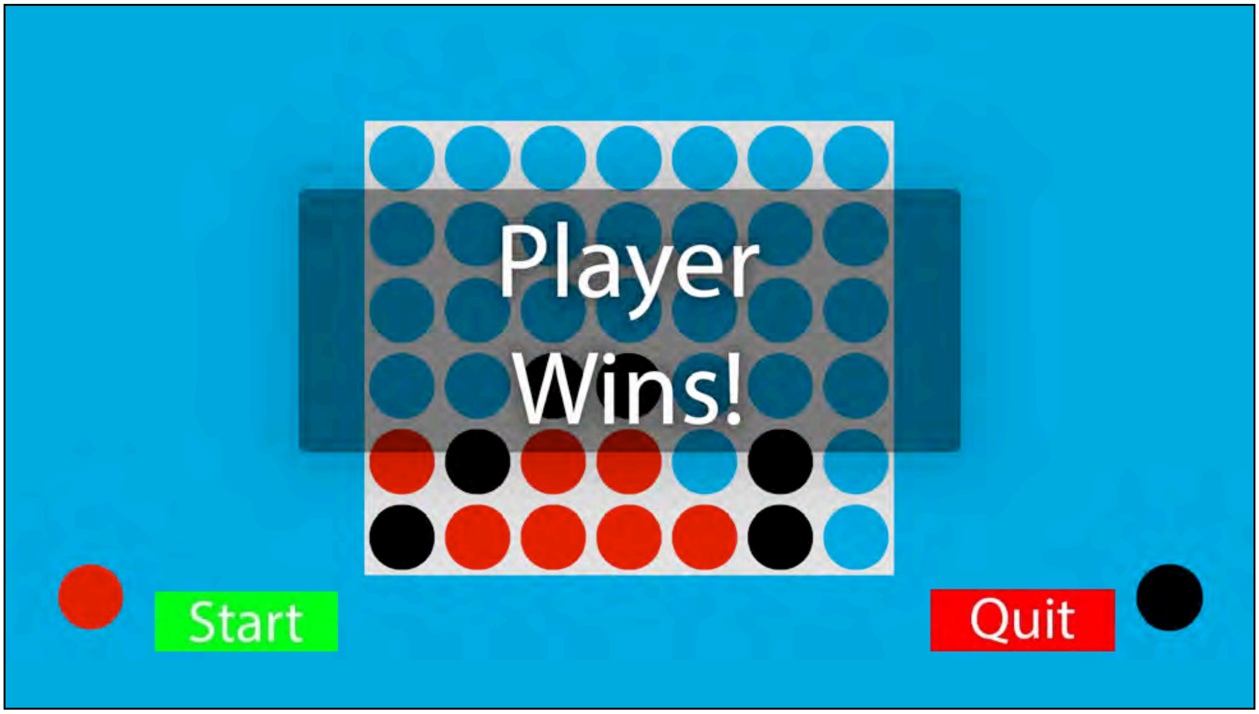
Find the gradient on the curve: y = (-x^(2)-3x)^7, when x = -2

Solution:

This screen is very similar looking to the question one and two screen. The only difference being the question. Once again, the question will be randomly generated each time, and will be displayed to the user before their turn. To input their answer, the user can simply begin typing, and what they have typed will be displayed on the screen.

Winner Screen





These two images show the screen which will appear when the game has been won. The buttons from the menu screen will appear again, to give the user the option to quit or play the game again. Using white text with a darker background allows for a greater contrast, and for the winner to be clearly displayed on screen.

Testing

Testing plan

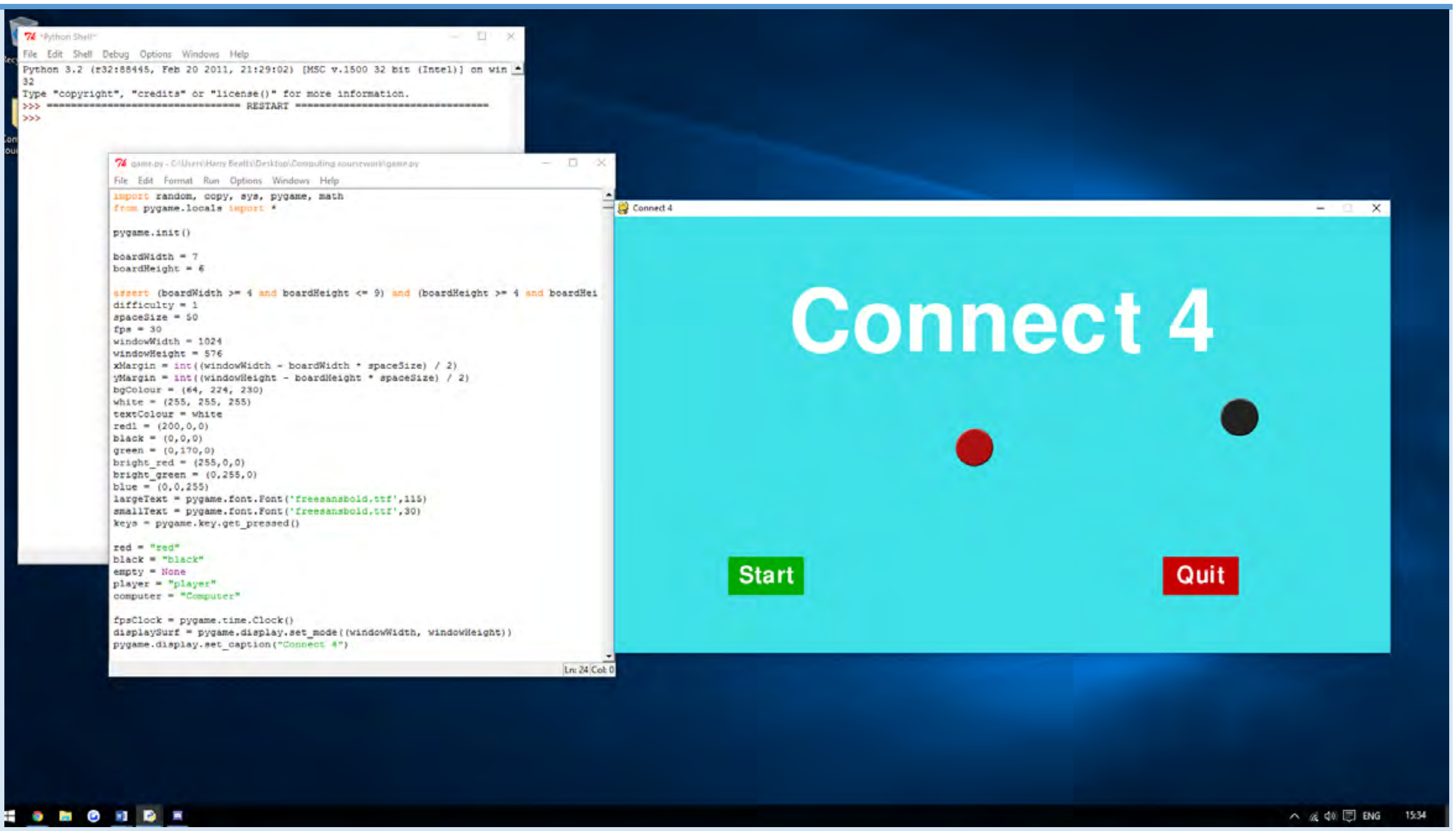
The tests of the program include all of the core functions which need to work to ensure that the program is not going to crash. My testing will involve uses of typical, erroneous and extreme data types. Using a range of data types will test whether the program is able withstand the inputs from the user, and the data which it is generating its self. Each test will have a screenshot, to show the proof that the test was either successful or unsuccessful. Each screenshot will also have a short description explaining what is happening.

Test ID	Description	Data type	Expected result	Pass/Fail
1.	Start button on menu page should not activate when clicking very close to it.	Extreme	No change.	Pass
2.	Start button should activate when clicked on.	Typical	Game should begin.	Pass
3.	Start button on the menu page should turn a brighter green when mouse if hovered over it.	Typical	Brighter green.	Pass
4.	Quit button on menu page should not activate when clicking very close to it.	Extreme	No change.	Pass
5.	Quit button should activate when clicked on.	Typical	Game should close.	Pass
6.	Quit button on the menu page should turn a brighter red when mouse if hovered over it.	Typical	Brighter red.	Pass
7.	The user should not be able to have the first turn when starting the game for the first time.	Typical	Computer should always have the first turn, when the game is run for the first time.	Pass
8.	The user should not be able to take a turn on the game with the wrong colour counter.	Erroneous	No change when the user drags wrong counter.	Pass
9.	The question one answers should only be accepted to two decimal places.	Extreme	“Incorrect” message.	Pass

10.	Question one should only accept float data types as acceptable inputs.	Erroneous	“Incorrect” message.	Pass
11.	The user should only have five total attempts at any question.	Typical	“Incorrect solution, X attempts remaining. Solution:”, message.	Pass
12.	The user should not have a turn when a question is answered incorrectly.	Typical	“Incorrect” message.	Pass
13.	When the user types to answer a question it should display on screen as they type.	Typical	Text typed should appear on screen.	Pass
14.	When the user has a turn for the first time, a help message should be displayed.	Typical	Help message displayed	Pass
15.	The game should be won if there are four, or more, counters in a row of either colour.	Typical, when four in a row. Extreme when five or more.	Game won image displayed.	Pass
16.	The question two answers should only be accepted to two decimal places.	Extreme	“Incorrect” message.	Pass
17.	Question two should only accept float data types as acceptable inputs.	Erroneous	“Incorrect” message.	Pass
18.	The question three answers should only be accepted as integers.	Extreme	“Incorrect” message.	Pass
19.	The discriminant generated in question one should always be a positive value.	Extreme	Positive value.	Pass
20.	The randomly generated integer values should be within the given parameters.	Typical	Values in the range given.	Pass
21.	Values returned for answers should be within 0<=x<=360, in question one.	Typical	Values in the range given.	Pass
22.	Menu tokens should offset over time.	Typical	Tokens offset	Pass
23.	Menu tokens should stay looping around the game window, when the user has not selected an option.	Typical	Tokens on loop.	Pass

24.	The computer should not try to place a token in a full column.	Erroneous	This result for a potential move should not be returned.	Pass
25.	The user should not be allowed to place a token in a full column.	Erroneous	Turn should not be allowed.	Pass
26.	Start button on game won screen page should not activate when clicking very close to it.	Extreme	No change.	Pass
27.	Start button on the game won screen should activate when clicked on.	Typical	Game should begin.	Pass
28.	Start button on the game won screen should turn a brighter green when mouse if hovered over it.	Typical	Brighter green.	Pass
29.	Quit button on menu page should not activate when clicking very close to it.	Extreme	No change.	Pass
30.	Quit button should activate when clicked on.	Typical	Game should close.	Pass
31.	Start button on the game won page should turn a brighter red when mouse if hovered over it.	Typical	Brighter red.	Pass

Screenshots

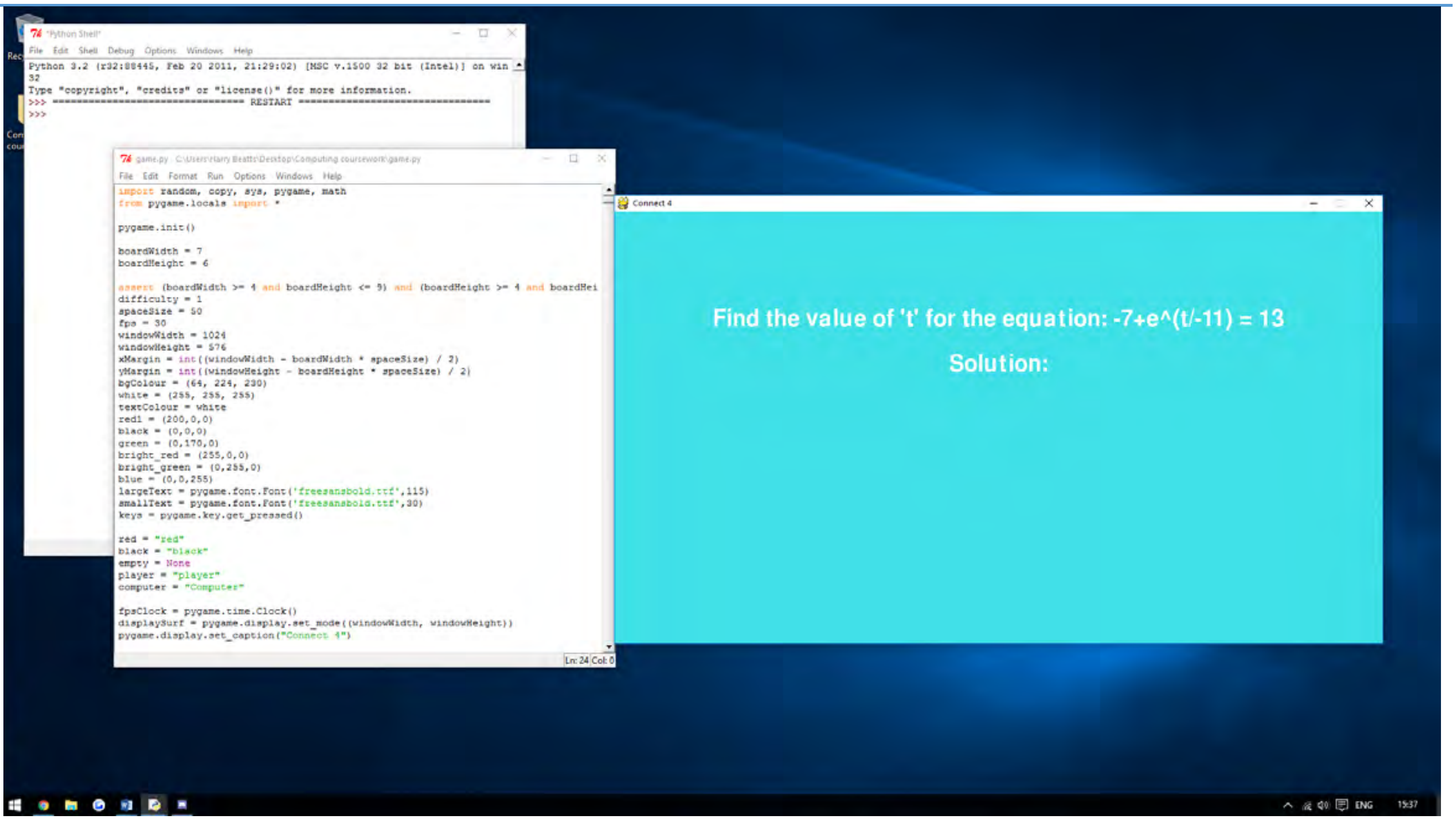
Test ID	Screenshot
1.	 Screenshot shows that there is no effect when the user clocks very close to the button.

Python Shell
File Edit Shell Debug Options Windows Help
game.py - C:\Users\Harry Bealts\Desktop\Computing coursework\game.py
File Edit Format Run Options Windows Help
Ln: 24 Col: 0
Connect 4
Connect 4
Start
Quit
ENG
15:34

```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
```

```
import random, copy, sys, pygame, math
from pygame.locals import *

pygame.init()

boardWidth = 7
boardHeight = 6

assert (boardWidth >= 4 and boardHeight <= 9) and (boardHeight >= 4 and boardHei
difficulty = 1
spaceSize = 50
fps = 30
windowWidth = 1024
windowHeight = 576
xMargin = int((windowWidth - boardWidth * spaceSize) / 2)
yMargin = int((windowHeight - boardHeight * spaceSize) / 2)
bgColour = (64, 224, 230)
white = (255, 255, 255)
textColour = white
red1 = (200,0,0)
black = (0,0,0)
green = (0,170,0)
bright_red = (255,0,0)
bright_green = (0,255,0)
blue = (0,0,255)
largeText = pygame.font.Font('freesansbold.ttf',115)
smallText = pygame.font.Font('freesansbold.ttf',30)
keys = pygame.key.get_pressed()

red = "red"
black = "black"
empty = None
player = "player"
computer = "Computer"

fpsClock = pygame.time.Clock()
displaySurf = pygame.display.set_mode((windowWidth, windowHeight))
pygame.display.set_caption("Connect 4")
```

2.


Python Shell
File Edit Shell Debug Options Windows Help
game.py - C:\Users\Harry Beatts\Desktop\Computing coursework\game.py
File Edit Format Run Options Windows Help
Connect 4
Ln: 24 Col: 0
ENG
15:37

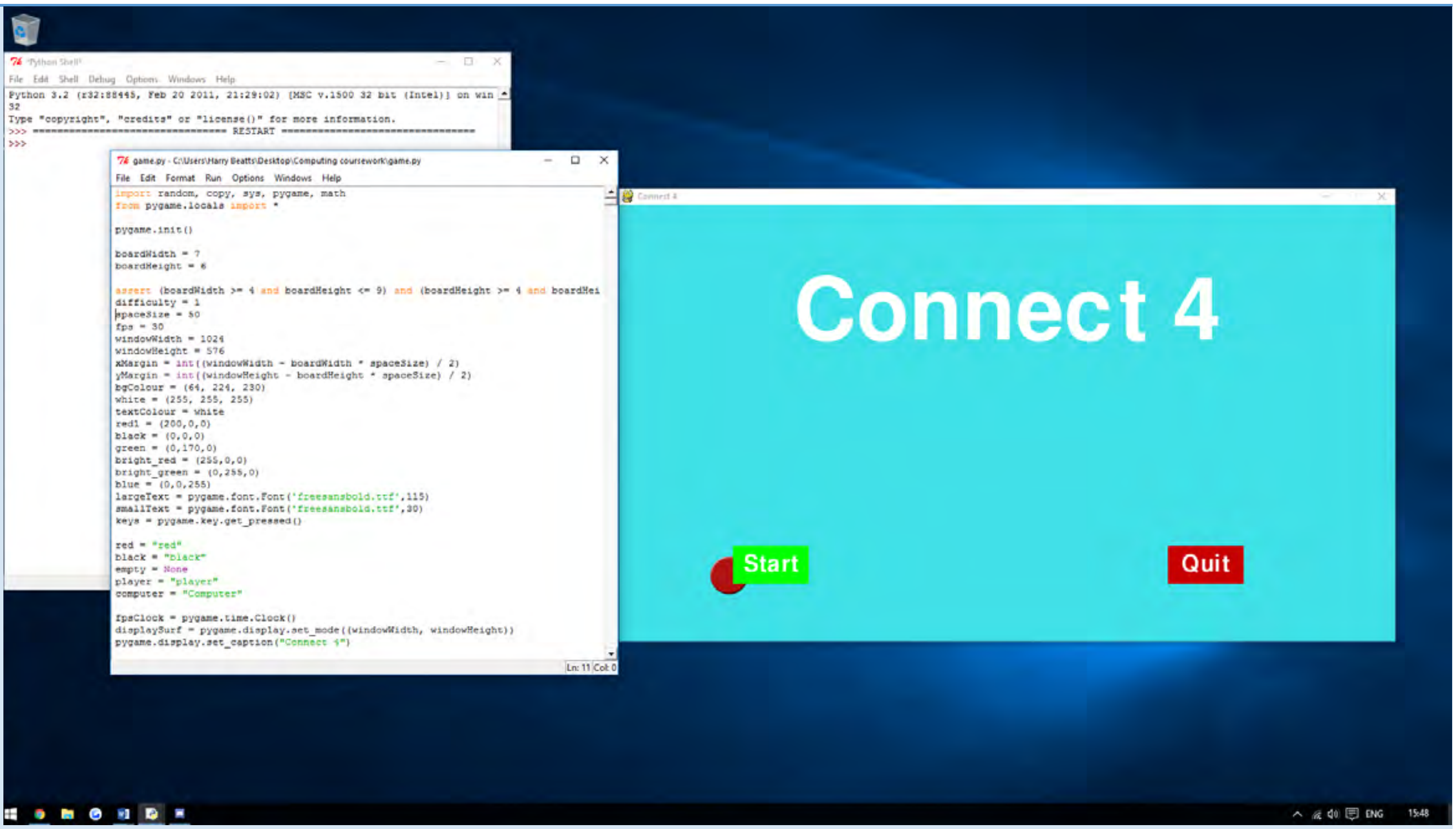
```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
```

```python
import random, copy, sys, pygame, math
from pygame.locals import *

pygame.init()

boardWidth = 7
boardHeight = 6

assert (boardWidth >= 4 and boardHeight <= 9) and (boardHeight >= 4 and boardHei
difficulty = 1
spaceSize = 50
fps = 30
windowWidth = 1024
windowHeight = 576
xMargin = int((windowWidth - boardWidth * spaceSize) / 2)
yMargin = int((windowHeight - boardHeight * spaceSize) / 2)
bgColour = (64, 224, 230)
white = (255, 255, 255)
textColour = white
red1 = (200,0,0)
black = (0,0,0)
green = (0,170,0)
bright_red = (255,0,0)
bright_green = (0,255,0)
blue = (0,0,255)
largeText = pygame.font.Font('freesansbold.ttf',115)
smallText = pygame.font.Font('freesansbold.ttf',30)
keys = pygame.key.get_pressed()

red = "red"
black = "black"
empty = None
player = "player"
computer = "Computer"

fpsClock = pygame.time.Clock()
displaySurf = pygame.display.set_mode((windowWidth, windowHeight))
pygame.display.set_caption("Connect 4")
```

```
Find the value of 't' for the equation: -7+e^(t/-11) = 13

Solution:
```

The game begins when the start button is clicked on.

3.



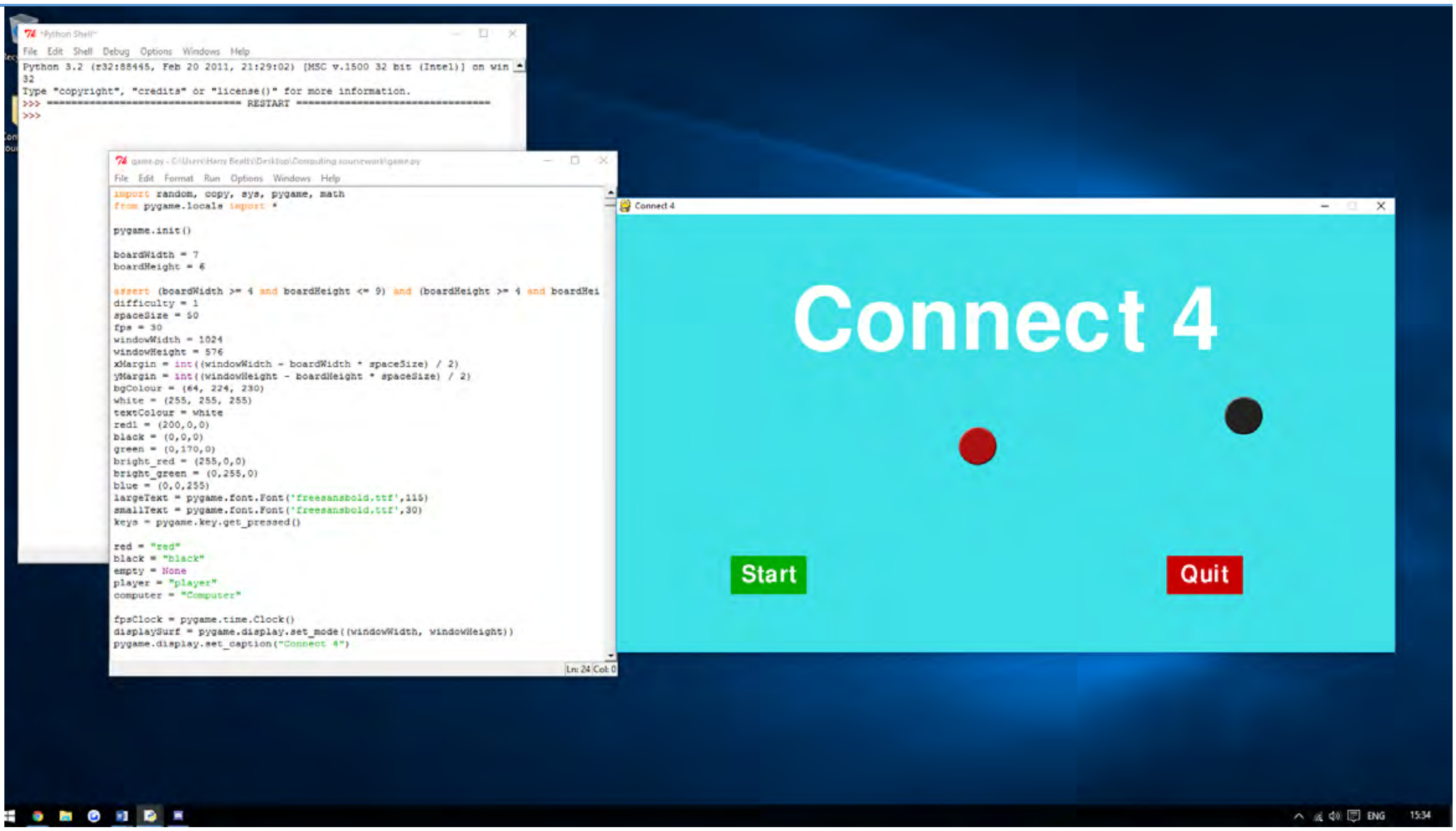
Python Shell
File Edit Shell Debug Options Windows Help
game.py - C:\Users\Harry Beatts\Desktop\Computing coursework\game.py
File Edit Format Run Options Windows Help
Ln: 11 Col: 0
Connect 4
Connect 4
Start
Quit
ENG
15:48

```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
```

```
import random, copy, sys, pygame, math
from pygame.locals import *

pygame.init()

boardWidth = 7
boardHeight = 6

assert (boardWidth >= 4 and boardHeight <= 9) and (boardHeight >= 4 and boardHei
difficulty = 1
spaceSize = 50
fps = 30
windowWidth = 1024
windowHeight = 576
xMargin = int((windowWidth - boardWidth * spaceSize) / 2)
yMargin = int((windowHeight - boardHeight * spaceSize) / 2)
bgColour = (64, 224, 230)
white = (255, 255, 255)
textColour = white
red1 = (200,0,0)
black = (0,0,0)
green = (0,170,0)
bright_red = (255,0,0)
bright_green = (0,255,0)
blue = (0,0,255)
largeText = pygame.font.Font('freesansbold.ttf',115)
smallText = pygame.font.Font('freesansbold.ttf',30)
keys = pygame.key.get_pressed()

red = "red"
black = "black"
empty = None
player = "player"
computer = "Computer"

fpsClock = pygame.time.Clock()
displaySurf = pygame.display.set_mode((windowWidth, windowHeight))
pygame.display.set_caption("Connect 4")
```

The start button changes to a brighter green when the mouse is hovered over it.

4.



Python Shell
File Edit Shell Debug Options Windows Help

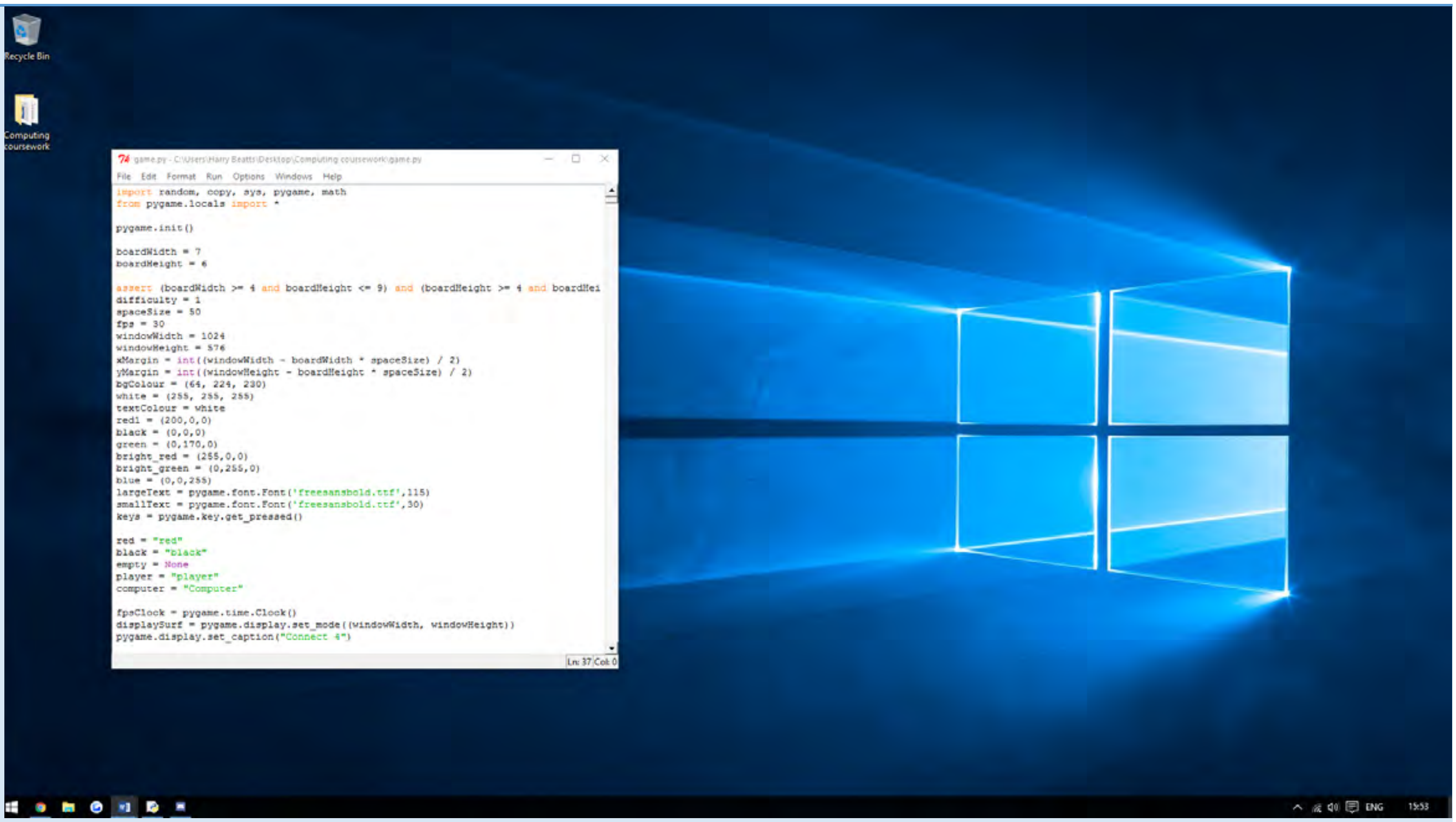
```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
```

game.py - C:\Users\Harry Bealts\Desktop\Computing coursework\game.py
File Edit Format Run Options Windows Help

```
import random, copy, sys, pygame, math
from pygame.locals import *

pygame.init()

boardWidth = 7
boardHeight = 6

assert (boardWidth >= 4 and boardHeight <= 9) and (boardHeight >= 4 and boardHei
difficulty = 1
spaceSize = 50
fps = 30
windowWidth = 1024
windowHeight = 576
xMargin = int((windowWidth - boardWidth * spaceSize) / 2)
yMargin = int((windowHeight - boardHeight * spaceSize) / 2)
bgColour = (64, 224, 230)
white = (255, 255, 255)
textColour = white
red1 = (200,0,0)
black = (0,0,0)
green = (0,170,0)
bright_red = (255,0,0)
bright_green = (0,255,0)
blue = (0,0,255)
largeText = pygame.font.Font('freesansbold.ttf',115)
smallText = pygame.font.Font('freesansbold.ttf',30)
keys = pygame.key.get_pressed()

red = "red"
black = "black"
empty = None
player = "player"
computer = "Computer"

fpsClock = pygame.time.Clock()
displaySurf = pygame.display.set_mode((windowWidth, windowHeight))
pygame.display.set_caption("Connect 4")
```

Screenshot shows that there is no effect when the user clicks very close to the button.

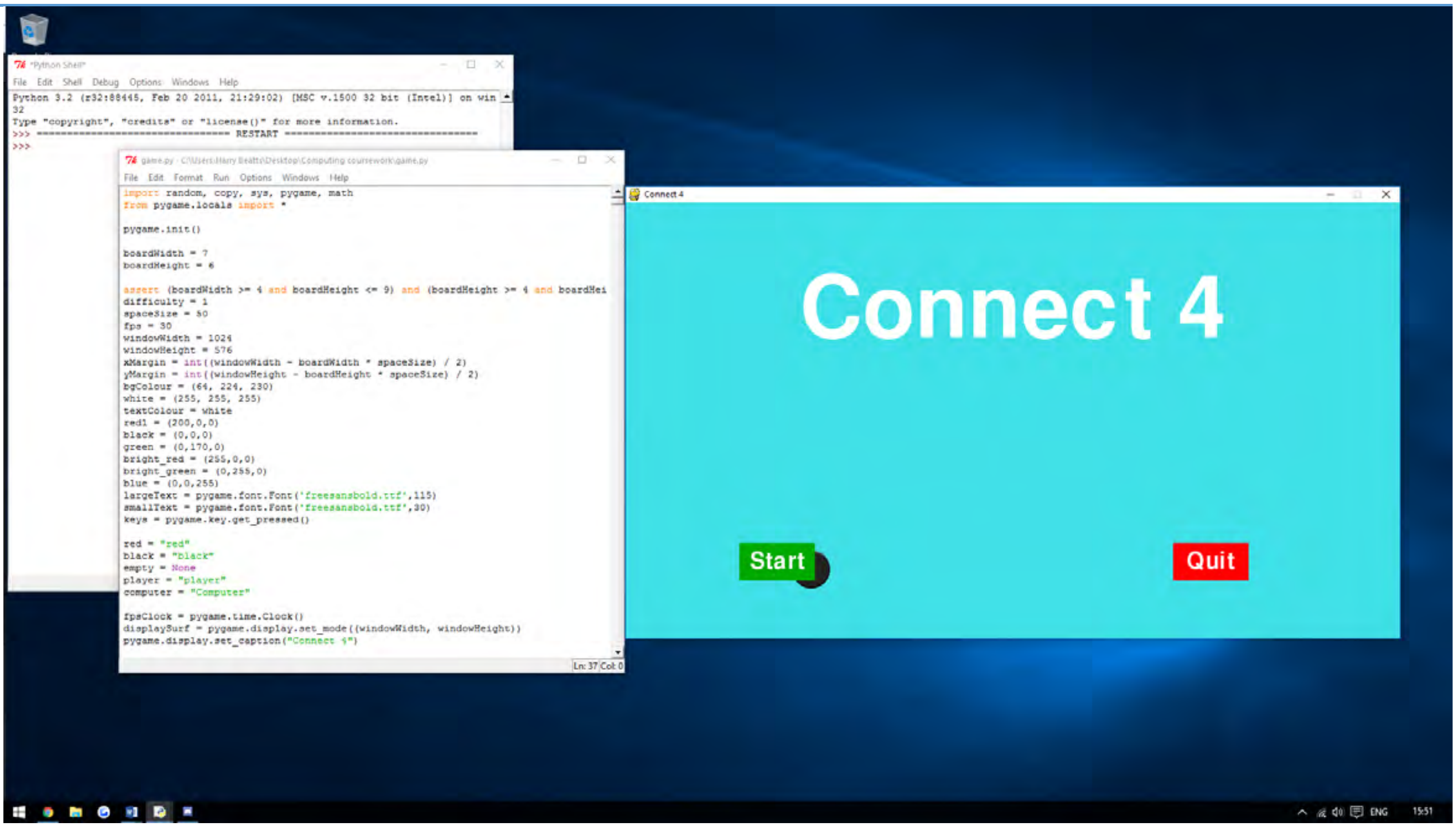
5,


Recycle Bin
Computing coursework
game.py - C:\Users\Harry Beatts\Desktop\Computing coursework\game.py
File Edit Format Run Options Windows Help
Ln: 37 Col: 0
ENG
15:53

```
import random, copy, sys, pygame, math
from pygame.locals import *

pygame.init()

boardWidth = 7
boardHeight = 6

assert (boardWidth >= 4 and boardHeight <= 9) and (boardHeight >= 4 and boardHei
difficulty = 1
spaceSize = 50
fps = 30
windowWidth = 1024
windowHeight = 576
xMargin = int((windowWidth - boardWidth * spaceSize) / 2)
yMargin = int((windowHeight - boardHeight * spaceSize) / 2)
bgColour = (64, 224, 230)
white = (255, 255, 255)
textColour = white
red1 = (200,0,0)
black = (0,0,0)
green = (0,170,0)
bright_red = (255,0,0)
bright_green = (0,255,0)
blue = (0,0,255)
largeText = pygame.font.Font('freesansbold.ttf',115)
smallText = pygame.font.Font('freesansbold.ttf',30)
keys = pygame.key.get_pressed()

red = "red"
black = "black"
empty = None
player = "player"
computer = "Computer"

fpsClock = pygame.time.Clock()
displaySurf = pygame.display.set_mode((windowWidth, windowHeight))
pygame.display.set_caption("Connect 4")
```

The game closes when the quit button was clicked on.

6.



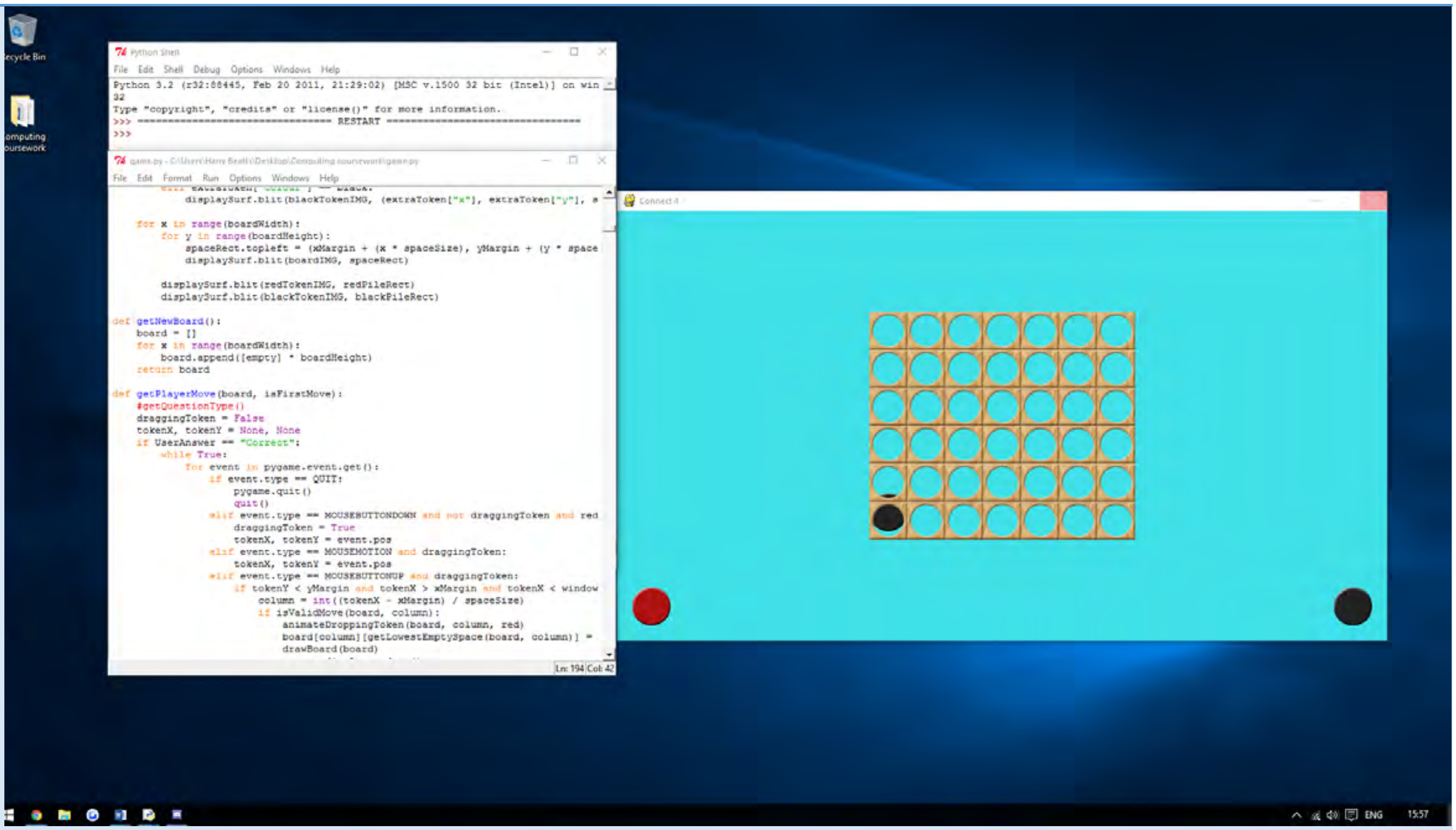
```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
```

```
import random, copy, sys, pygame, math
from pygame.locals import *

pygame.init()

boardWidth = 7
boardHeight = 6

assert (boardWidth >= 4 and boardHeight <= 9) and (boardHeight >= 4 and boardHei
difficulty = 1
spaceSize = 50
fps = 30
windowWidth = 1024
windowHeight = 576
xMargin = int((windowWidth - boardWidth * spaceSize) / 2)
yMargin = int((windowHeight - boardHeight * spaceSize) / 2)
bgColour = (64, 224, 230)
white = (255, 255, 255)
textColour = white
red1 = (200,0,0)
black = (0,0,0)
green = (0,170,0)
bright_red = (255,0,0)
bright_green = (0,255,0)
blue = (0,0,255)
largeText = pygame.font.Font('freesansbold.ttf',115)
smallText = pygame.font.Font('freesansbold.ttf',30)
keys = pygame.key.get_pressed()

red = "red"
black = "black"
empty = None
player = "player"
computer = "Computer"

fpsClock = pygame.time.Clock()
displaySurf = pygame.display.set_mode((windowWidth, windowHeight))
pygame.display.set_caption("Connect 4")
```

Python Shell
File Edit Shell Debug Options Windows Help
game.py - C:\Users\Harry Beatts\Desktop\Computing coursework\game.py
File Edit Format Run Options Windows Help
Ln: 37 Col: 0
Connect 4
Connect 4
Start
Quit
ENG
15:51

The quit button changes to a brighter red when the mouse is hovered over it.

7.

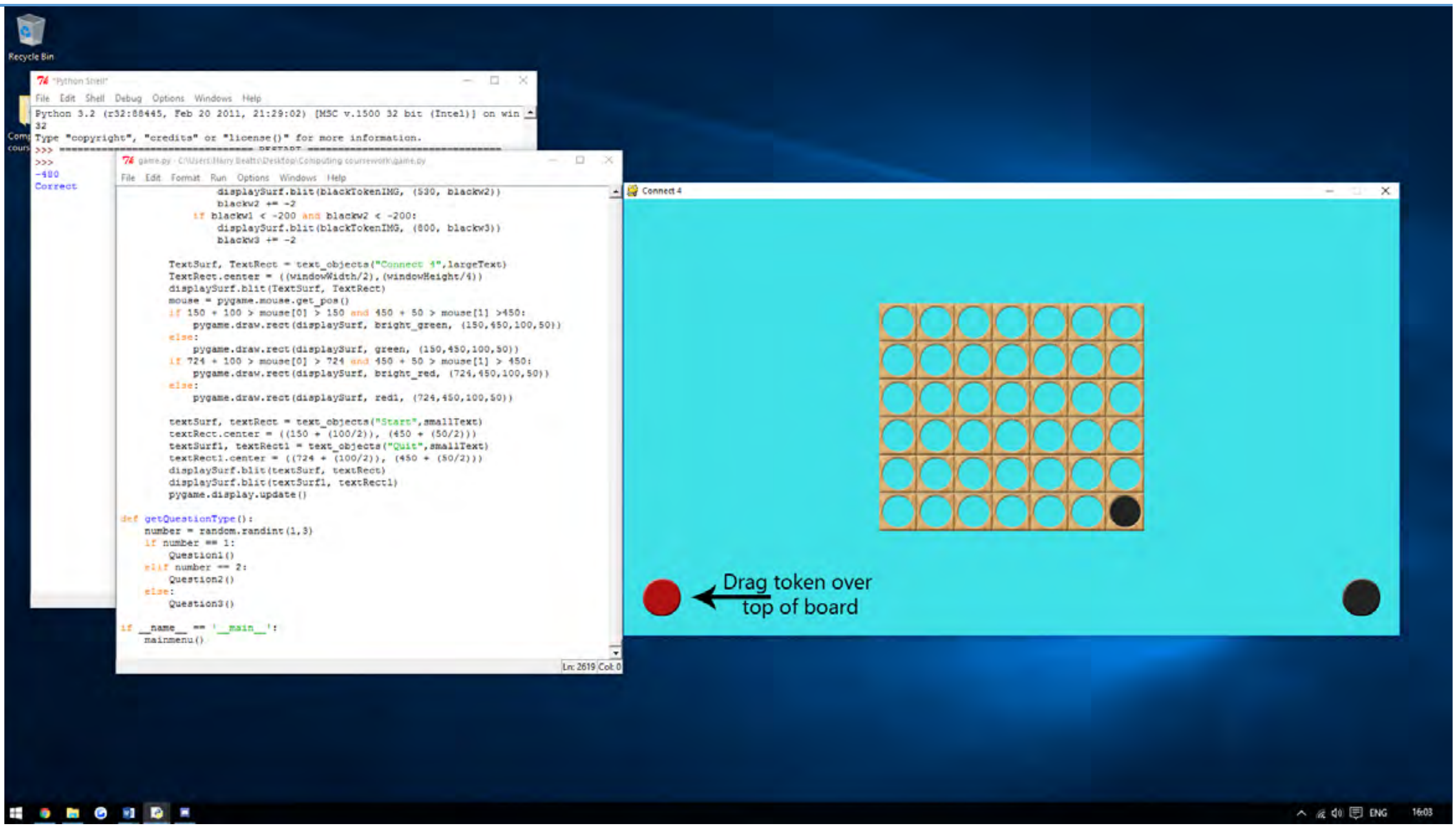

Recycle Bin
Computing coursework
Python Shell
File Edit Shell Debug Options Windows Help
game.py - C:\Users\Harry Beall\Desktop\Computing coursework\game.py
File Edit Format Run Options Windows Help
Ln: 194 Col: 42
Connect 4
ENG
15:57

```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
```

```
            displaySurf.blit(blackTokenIMG, (extraToken["x"], extraToken["y"], s

    for x in range(boardWidth):
        for y in range(boardHeight):
            spaceRect.topleft = (xMargin + (x * spaceSize), yMargin + (y * space
            displaySurf.blit(boardIMG, spaceRect)

        displaySurf.blit(redTokenIMG, redPileRect)
        displaySurf.blit(blackTokenIMG, blackPileRect)

def getNewBoard():
    board = []
    for x in range(boardWidth):
        board.append([empty] * boardHeight)
    return board

def getPlayerMove(board, isFirstMove):
    #getQuestionType()
    draggingToken = False
    tokenX, tokenY = None, None
    if UserAnswer == "Correct":
        while True:
            for event in pygame.event.get():
                if event.type == QUIT:
                    pygame.quit()
                    quit()
                elif event.type == MOUSEBUTTONDOWN and not draggingToken and red
                    draggingToken = True
                    tokenX, tokenY = event.pos
                elif event.type == MOUSEMOTION and draggingToken:
                    tokenX, tokenY = event.pos
                elif event.type == MOUSEBUTTONUP and draggingToken:
                    if tokenY < yMargin and tokenX > xMargin and tokenX < window
                        column = int((tokenX - xMargin) / spaceSize)
                        if isValidMove(board, column):
                            animateDroppingToken(board, column, red)
                            board[column][getLowestEmptySpace(board, column)] =
                            drawBoard(board)
```

The computer will always have the first turn when the game starts.

8.



Recycle Bin
Python Shell
File Edit Shell Debug Options Windows Help

```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
-480
Correct
```

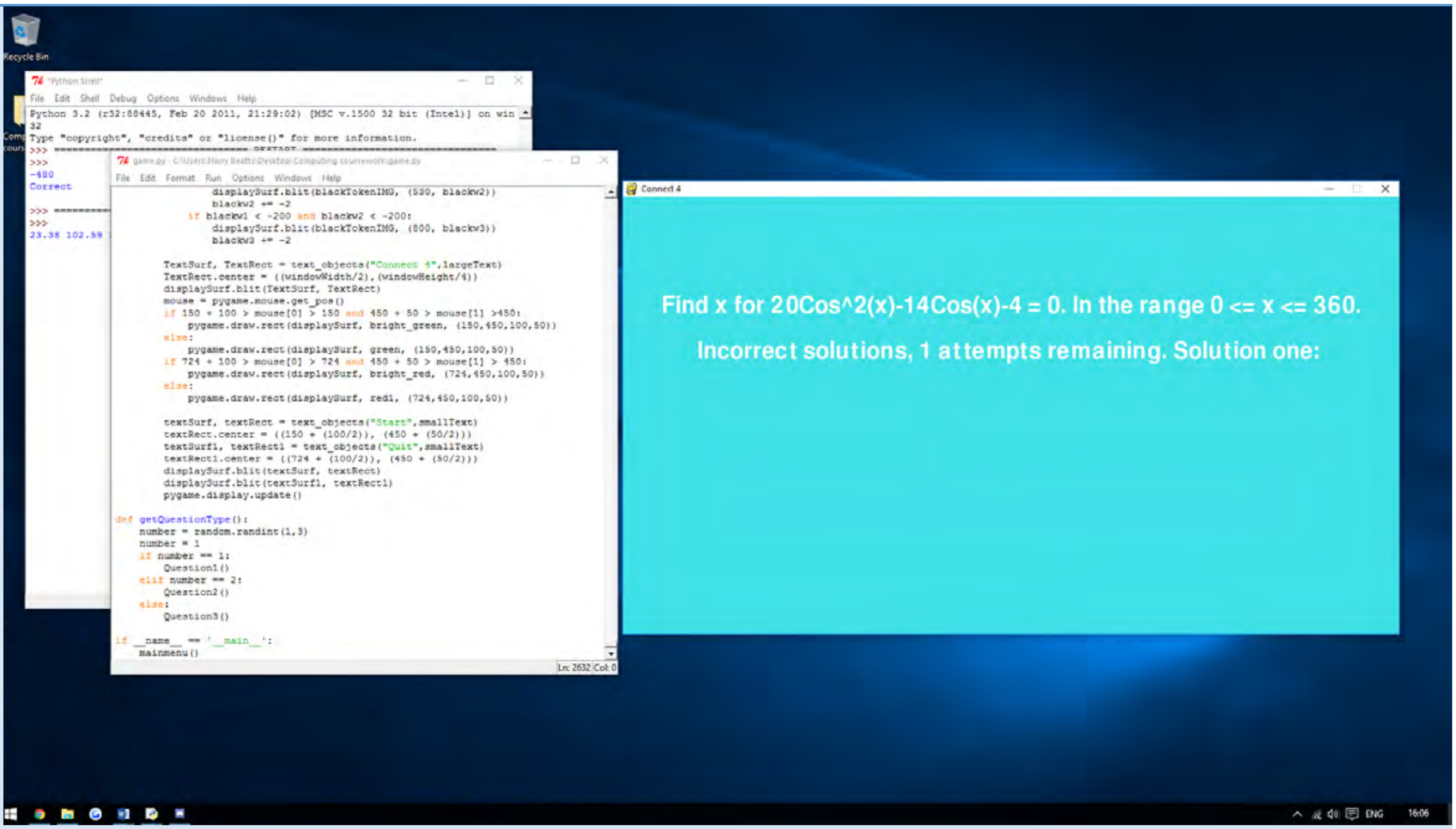
game.py - C:\Users\Harry Beatt\Desktop\Computing coursework\game.py
File Edit Format Run Options Windows Help

```
                displaySurf.blit(blackTokenIMG, (530, blackw2))
                blackw2 += -2
            if blackw1 < -200 and blackw2 < -200:
                displaySurf.blit(blackTokenIMG, (800, blackw3))
                blackw3 += -2

        TextSurf, TextRect = text_objects("Connect 4",largeText)
        TextRect.center = ((windowWidth/2),(windowHeight/4))
        displaySurf.blit(TextSurf, TextRect)
        mouse = pygame.mouse.get_pos()
        if 150 + 100 > mouse[0] > 150 and 450 + 50 > mouse[1] >450:
            pygame.draw.rect(displaySurf, bright_green, (150,450,100,50))
        else:
            pygame.draw.rect(displaySurf, green, (150,450,100,50))
        if 724 + 100 > mouse[0] > 724 and 450 + 50 > mouse[1] > 450:
            pygame.draw.rect(displaySurf, bright_red, (724,450,100,50))
        else:
            pygame.draw.rect(displaySurf, red1, (724,450,100,50))

        textSurf, textRect = text_objects("Start",smallText)
        textRect.center = ((150 + (100/2)), (450 + (50/2)))
        textSurf1, textRect1 = text_objects("Quit",smallText)
        textRect1.center = ((724 + (100/2)), (450 + (50/2)))
        displaySurf.blit(textSurf, textRect)
        displaySurf.blit(textSurf1, textRect1)
        pygame.display.update()

def getQuestionType():
    number = random.randint(1,3)
    if number == 1:
        Question1()
    elif number == 2:
        Question2()
    else:
        Question3()

if __name__ == '__main__':
    mainmenu()
```

Drag token over top of board

No effect when the user tries to drag the wrong colour token.

9.


Recycle Bin
Python Shell
File Edit Shell Debug Options Windows Help
game.py - C:\Users\Harry Beatt\Desktop\Computing coursework\game.py
File Edit Format Run Options Windows Help
Ln: 2632 Col: 0
Connect 4
ENG
16:06

```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
-480
Correct

>>> ==========
>>>
23.38 102.59
```

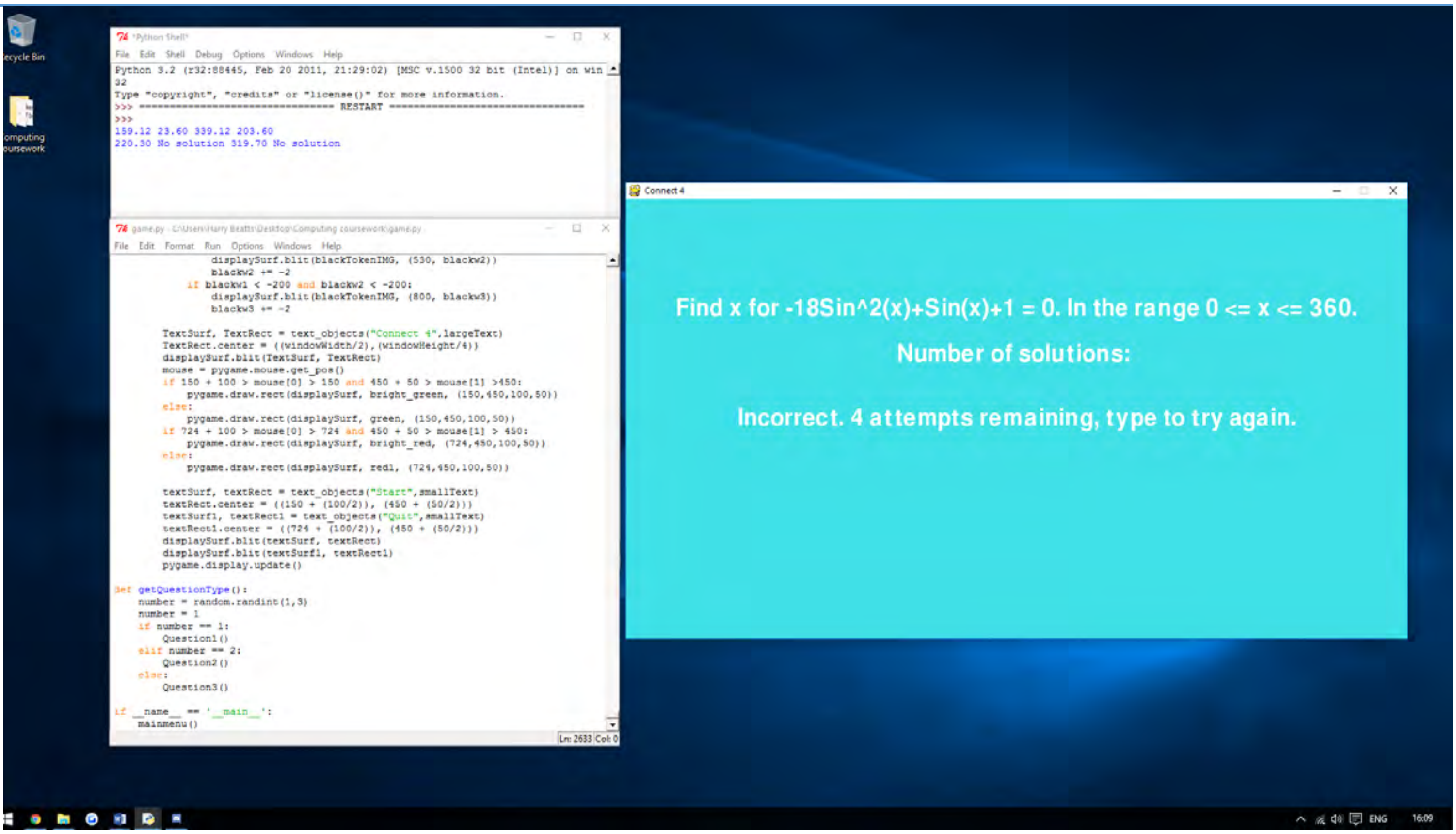
```
                displaySurf.blit(blackTokenIMG, (530, blackw2))
                blackw2 += -2
            if blackw1 < -200 and blackw2 < -200:
                displaySurf.blit(blackTokenIMG, (800, blackw3))
                blackw3 += -2

        TextSurf, TextRect = text_objects("Connect 4",largeText)
        TextRect.center = ((windowWidth/2),(windowHeight/4))
        displaySurf.blit(TextSurf, TextRect)
        mouse = pygame.mouse.get_pos()
        if 150 + 100 > mouse[0] > 150 and 450 + 50 > mouse[1] >450:
            pygame.draw.rect(displaySurf, bright_green, (150,450,100,50))
        else:
            pygame.draw.rect(displaySurf, green, (150,450,100,50))
        if 724 + 100 > mouse[0] > 724 and 450 + 50 > mouse[1] > 450:
            pygame.draw.rect(displaySurf, bright_red, (724,450,100,50))
        else:
            pygame.draw.rect(displaySurf, red1, (724,450,100,50))

        textSurf, textRect = text_objects("Start",smallText)
        textRect.center = ((150 + (100/2)), (450 + (50/2)))
        textSurf1, textRect1 = text_objects("Quit",smallText)
        textRect1.center = ((724 + (100/2)), (450 + (50/2)))
        displaySurf.blit(textSurf, textRect)
        displaySurf.blit(textSurf1, textRect1)
        pygame.display.update()

def getQuestionType():
    number = random.randint(1,3)
    number = 1
    if number == 1:
        Question1()
    elif number == 2:
        Question2()
    else:
        Question3()

if __name__ == '__main__':
    mainmenu()
```

Find x for 20Cos^2(x)-14Cos(x)-4 = 0. In the range 0 <= x <= 360.

Incorrect solutions, 1 attempts remaining. Solution one:

Incorrect message displaying when the user’s answer is not given to two decimal places.

10.



Recycle Bin
Computing coursework
Python Shell
File Edit Shell Debug Options Windows Help
game.py - C:\Users\Harry Beatts\Desktop\Computing coursework\game.py
File Edit Format Run Options Windows Help
Ln: 2633 Col: 0
Connect 4
ENG
16:09

```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
159.12 23.60 339.12 203.60
220.30 No solution 319.70 No solution
```

```
                displaySurf.blit(blackTokenIMG, (530, blackw2))
                blackw2 += -2
            if blackw1 < -200 and blackw2 < -200:
                displaySurf.blit(blackTokenIMG, (800, blackw3))
                blackw3 += -2

        TextSurf, TextRect = text_objects("Connect 4",largeText)
        TextRect.center = ((windowWidth/2),(windowHeight/4))
        displaySurf.blit(TextSurf, TextRect)
        mouse = pygame.mouse.get_pos()
        if 150 + 100 > mouse[0] > 150 and 450 + 50 > mouse[1] >450:
            pygame.draw.rect(displaySurf, bright_green, (150,450,100,50))
        else:
            pygame.draw.rect(displaySurf, green, (150,450,100,50))
        if 724 + 100 > mouse[0] > 724 and 450 + 50 > mouse[1] > 450:
            pygame.draw.rect(displaySurf, bright_red, (724,450,100,50))
        else:
            pygame.draw.rect(displaySurf, red1, (724,450,100,50))

        textSurf, textRect = text_objects("Start",smallText)
        textRect.center = ((150 + (100/2)), (450 + (50/2)))
        textSurf1, textRect1 = text_objects("Quit",smallText)
        textRect1.center = ((724 + (100/2)), (450 + (50/2)))
        displaySurf.blit(textSurf, textRect)
        displaySurf.blit(textSurf1, textRect1)
        pygame.display.update()

def getQuestionType():
    number = random.randint(1,3)
    number = 1
    if number == 1:
        Question1()
    elif number == 2:
        Question2()
    else:
        Question3()

if __name__ == '__main__':
    mainmenu()
```

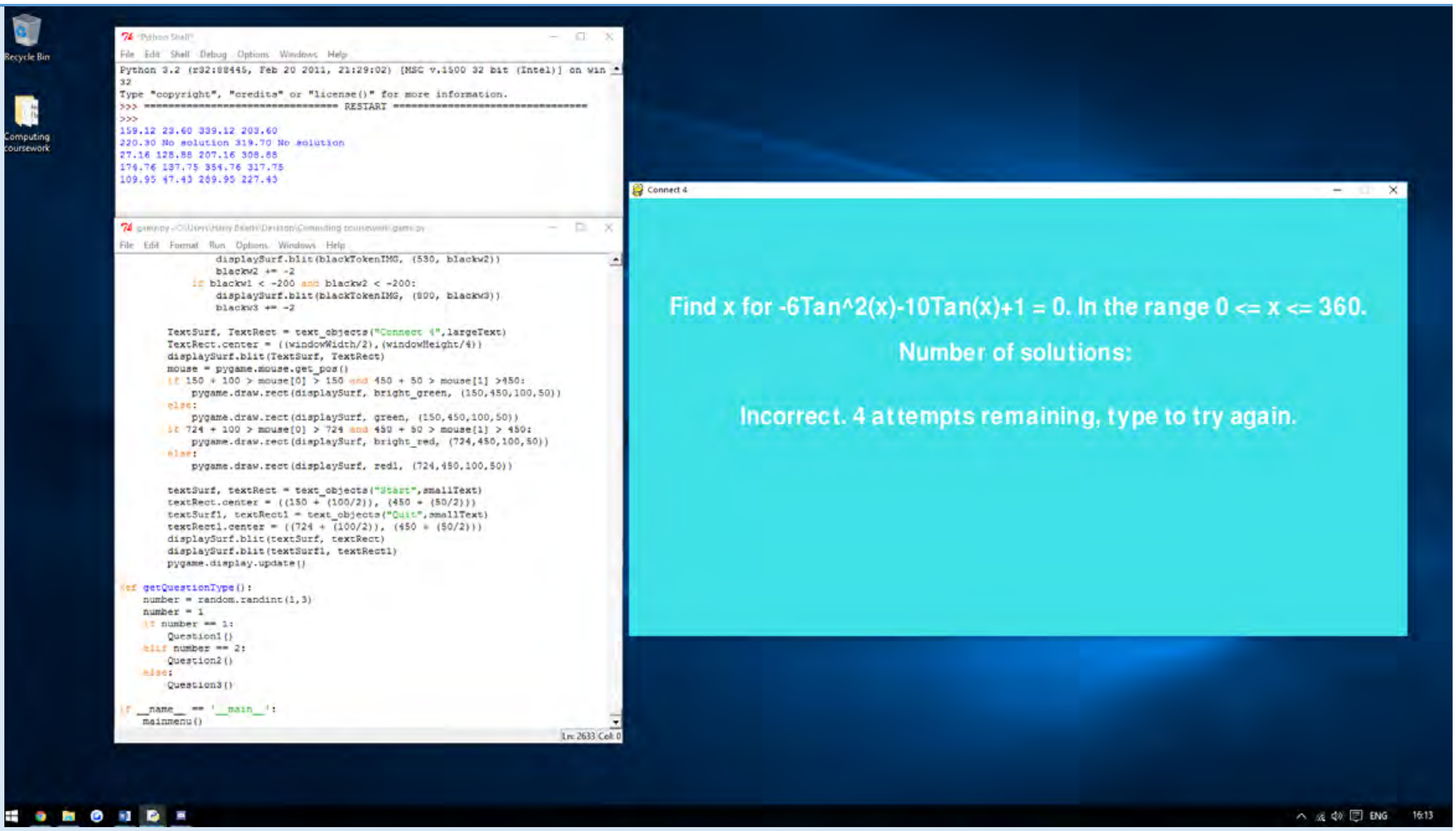
Find x for -18Sin^2(x)+Sin(x)+1 = 0. In the range 0 <= x <= 360.

Number of solutions:

Incorrect. 4 attempts remaining, type to try again.

Non float datatypes not accepted. Error message is displayed, showing the answer is incorrect.

11.



```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
159.12 23.60 339.12 203.60
220.30 No solution 319.70 No solution
27.16 128.88 207.16 308.88
174.76 137.75 354.76 317.75
109.95 47.43 289.95 227.43
```

```
                displaySurf.blit(blackTokenIMG, (530, blackw2))
                blackw2 += -2
            if blackw1 < -200 and blackw2 < -200:
                displaySurf.blit(blackTokenIMG, (800, blackw3))
                blackw3 += -2

        TextSurf, TextRect = text_objects("Connect 4",largeText)
        TextRect.center = ((windowWidth/2),(windowHeight/4))
        displaySurf.blit(TextSurf, TextRect)
        mouse = pygame.mouse.get_pos()
        if 150 + 100 > mouse[0] > 150 and 450 + 50 > mouse[1] >450:
            pygame.draw.rect(displaySurf, bright_green, (150,450,100,50))
        else:
            pygame.draw.rect(displaySurf, green, (150,450,100,50))
        if 724 + 100 > mouse[0] > 724 and 450 + 50 > mouse[1] > 450:
            pygame.draw.rect(displaySurf, bright_red, (724,450,100,50))
        else:
            pygame.draw.rect(displaySurf, red1, (724,450,100,50))

        textSurf, textRect = text_objects("Start",smallText)
        textRect.center = ((150 + (100/2)), (450 + (50/2)))
        textSurf1, textRect1 = text_objects("Quit",smallText)
        textRect1.center = ((724 + (100/2)), (450 + (50/2)))
        displaySurf.blit(textSurf, textRect)
        displaySurf.blit(textSurf1, textRect1)
        pygame.display.update()

def getQuestionType():
    number = random.randint(1,3)
    number = 1
    if number == 1:
        Question1()
    elif number == 2:
        Question2()
    else:
        Question3()

if __name__ == '__main__':
    mainmenu()
```

Find x for -6Tan^2(x)-10Tan(x)+1 = 0. In the range 0 <= x <= 360.

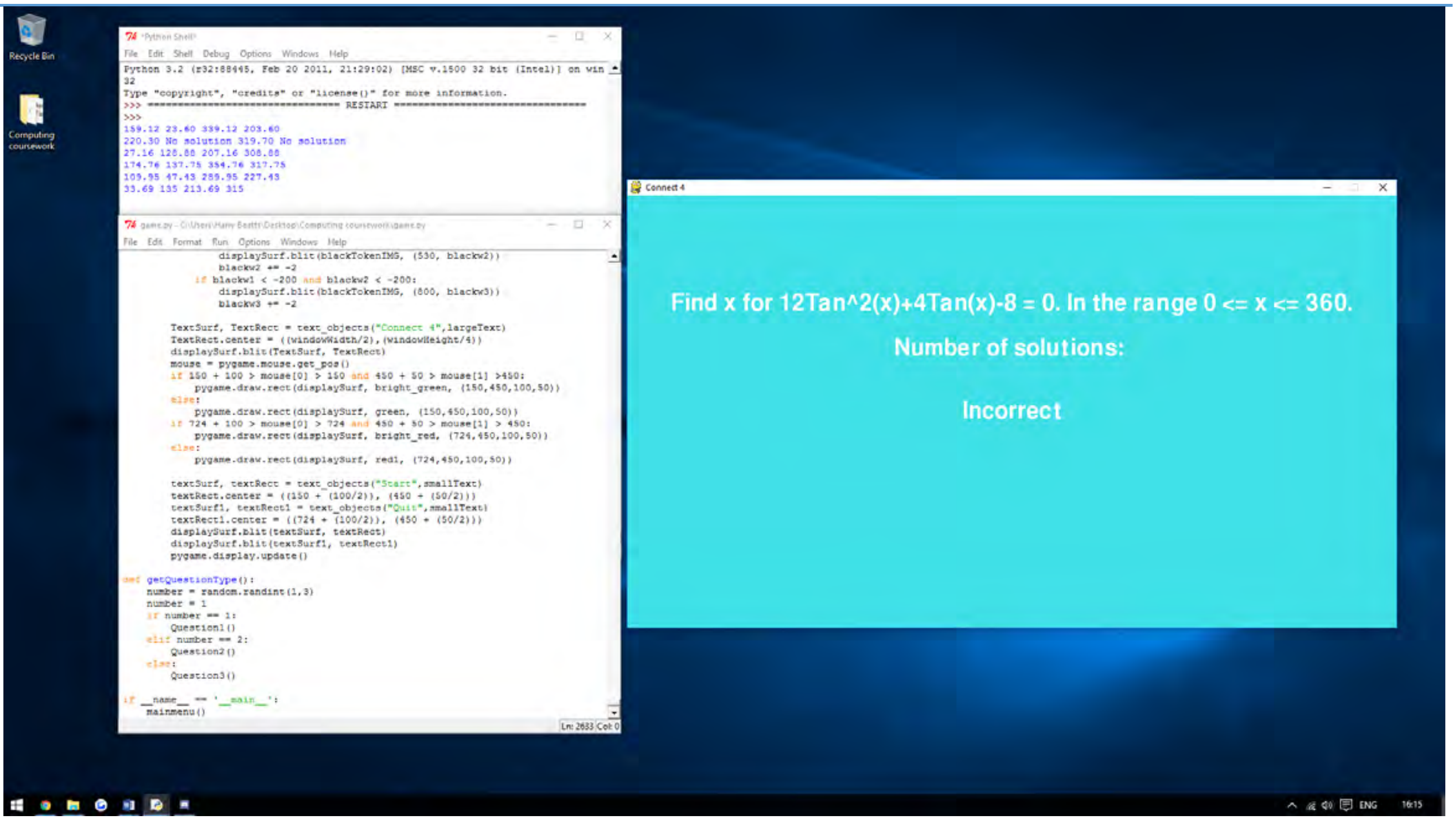
Number of solutions:

Incorrect. 4 attempts remaining, type to try again.

Python Shell
File Edit Shell Debug Options Windows Help
game.py - C:\Users\Harry Beath\Desktop\Computing coursework\game.py
File Edit Format Run Options Windows Help
Ln: 2633 Col: 0
Connect 4
Recycle Bin
Computing coursework
ENG
16:13

Four attempts remaining after the first appempt is incorrect. Shows that there is five total attempts at a question.

12.



Recycle Bin
Computing coursework
Python Shell
File Edit Shell Debug Options Windows Help

```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
159.12 23.60 339.12 203.60
220.30 No solution 319.70 No solution
27.16 128.88 207.16 308.88
174.76 137.75 354.76 317.75
109.95 47.43 289.95 227.43
33.69 135 213.69 315
```

game.py - C:\Users\Harry Beatt\Desktop\Computing coursework\game.py
File Edit Format Run Options Windows Help

```
                displaySurf.blit(blackTokenIMG, (530, blackw2))
                blackw2 += -2
            if blackw1 < -200 and blackw2 < -200:
                displaySurf.blit(blackTokenIMG, (800, blackw3))
                blackw3 += -2

        TextSurf, TextRect = text_objects("Connect 4",largeText)
        TextRect.center = ((windowWidth/2),(windowHeight/4))
        displaySurf.blit(TextSurf, TextRect)
        mouse = pygame.mouse.get_pos()
        if 150 + 100 > mouse[0] > 150 and 450 + 50 > mouse[1] >450:
            pygame.draw.rect(displaySurf, bright_green, (150,450,100,50))
        else:
            pygame.draw.rect(displaySurf, green, (150,450,100,50))
        if 724 + 100 > mouse[0] > 724 and 450 + 50 > mouse[1] > 450:
            pygame.draw.rect(displaySurf, bright_red, (724,450,100,50))
        else:
            pygame.draw.rect(displaySurf, red1, (724,450,100,50))

        textSurf, textRect = text_objects("Start",smallText)
        textRect.center = ((150 + (100/2)), (450 + (50/2)))
        textSurf1, textRect1 = text_objects("Quit",smallText)
        textRect1.center = ((724 + (100/2)), (450 + (50/2)))
        displaySurf.blit(textSurf, textRect)
        displaySurf.blit(textSurf1, textRect1)
        pygame.display.update()

def getQuestionType():
    number = random.randint(1,3)
    number = 1
    if number == 1:
        Question1()
    elif number == 2:
        Question2()
    else:
        Question3()

if __name__ == '__main__':
    mainmenu()
```

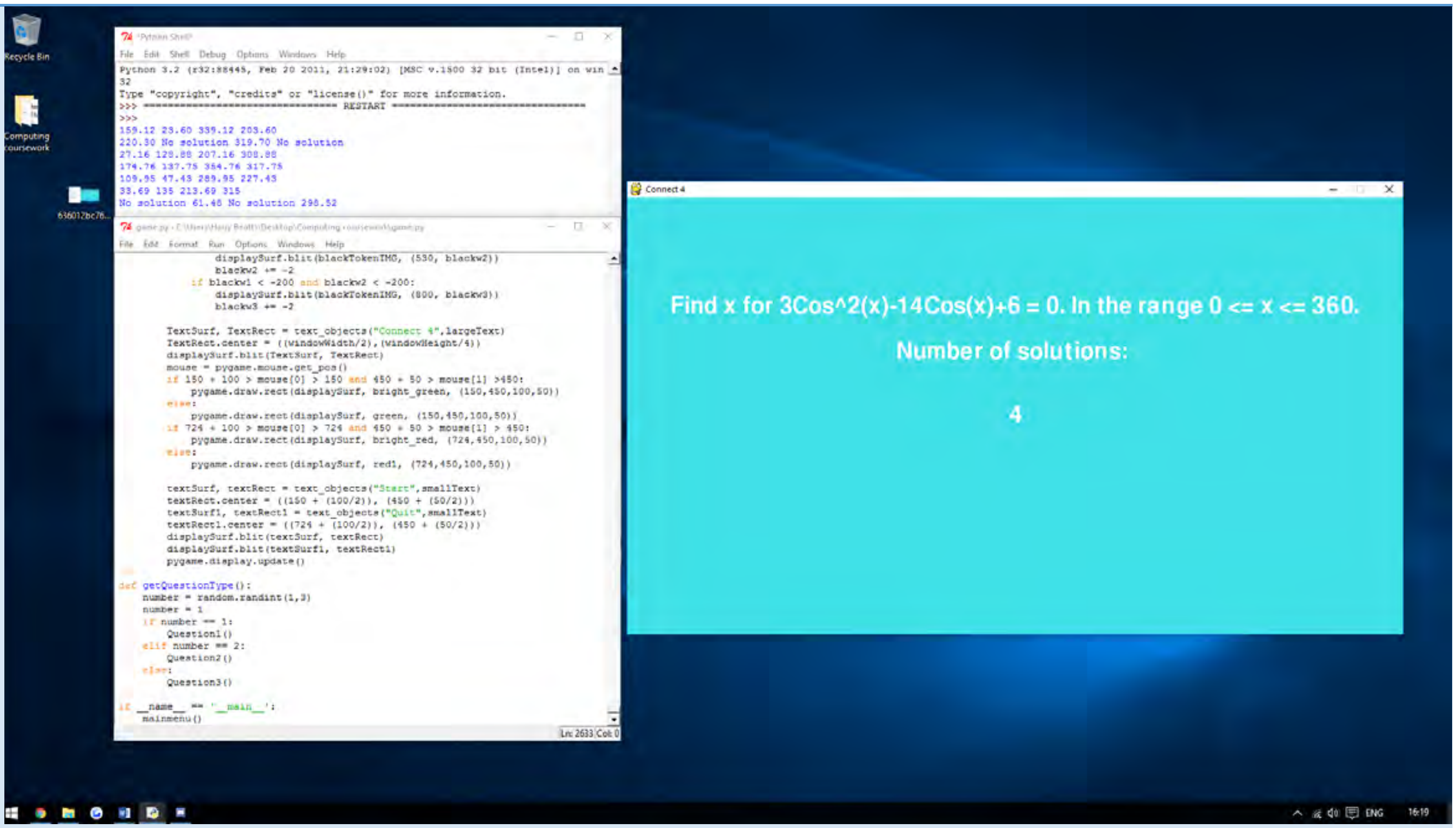
Find x for 12Tan^2(x)+4Tan(x)-8 = 0. In the range 0 <= x <= 360.

Number of solutions:

Incorrect

The incorrect message is displayed when the user is out of turns.

13.


Recycle Bin
Computing coursework
636012bc76...
Python Shell
File Edit Shell Debug Options Windows Help
game.py - C:\Users\Harry Beatt\Desktop\Computing coursework\game.py
File Edit Format Run Options Windows Help
Ln: 2633 Col: 0
Connect 4
ENG
16:19

```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
159.12 23.60 339.12 203.60
220.30 No solution 319.70 No solution
27.16 128.88 207.16 308.88
174.76 137.75 354.76 317.75
109.95 47.43 289.95 227.43
33.69 135 213.69 315
No solution 61.48 No solution 298.52
```

```
                displaySurf.blit(blackTokenIMG, (530, blackw2))
                blackw2 += -2
            if blackw1 < -200 and blackw2 < -200:
                displaySurf.blit(blackTokenIMG, (800, blackw3))
                blackw3 += -2

        TextSurf, TextRect = text_objects("Connect 4",largeText)
        TextRect.center = ((windowWidth/2),(windowHeight/4))
        displaySurf.blit(TextSurf, TextRect)
        mouse = pygame.mouse.get_pos()
        if 150 + 100 > mouse[0] > 150 and 450 + 50 > mouse[1] >450:
            pygame.draw.rect(displaySurf, bright_green, (150,450,100,50))
        else:
            pygame.draw.rect(displaySurf, green, (150,450,100,50))
        if 724 + 100 > mouse[0] > 724 and 450 + 50 > mouse[1] > 450:
            pygame.draw.rect(displaySurf, bright_red, (724,450,100,50))
        else:
            pygame.draw.rect(displaySurf, red1, (724,450,100,50))

        textSurf, textRect = text_objects("Start",smallText)
        textRect.center = ((150 + (100/2)), (450 + (50/2)))
        textSurf1, textRect1 = text_objects("Quit",smallText)
        textRect1.center = ((724 + (100/2)), (450 + (50/2)))
        displaySurf.blit(textSurf, textRect)
        displaySurf.blit(textSurf1, textRect1)
        pygame.display.update()

def getQuestionType():
    number = random.randint(1,3)
    number = 1
    if number == 1:
        Question1()
    elif number == 2:
        Question2()
    else:
        Question3()

if __name__ == '__main__':
    mainmenu()
```

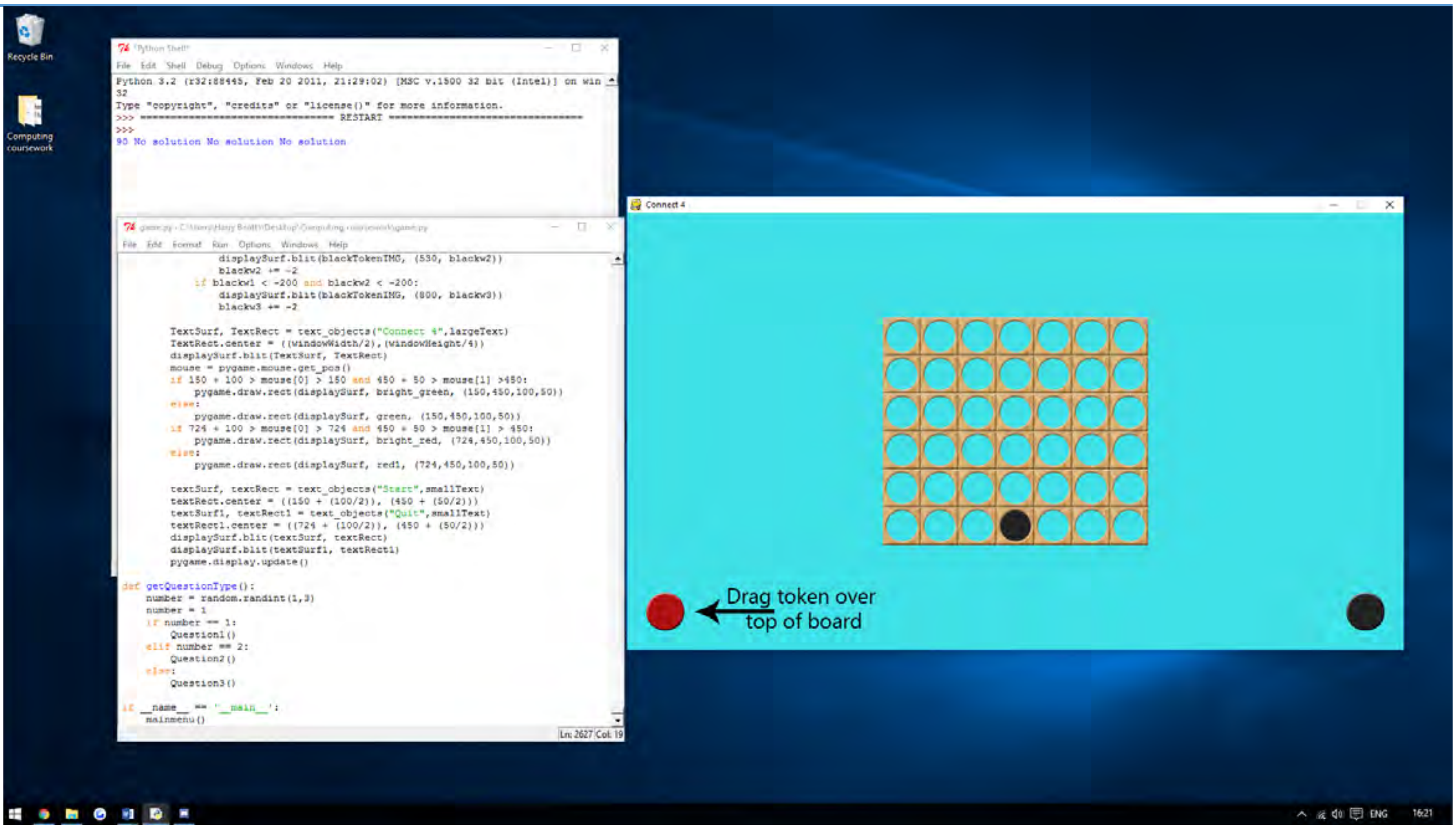
Find x for 3Cos^2(x)-14Cos(x)+6 = 0. In the range 0 <= x <= 360.

Number of solutions:

4

What is being types is being displayed on the game window.

14.



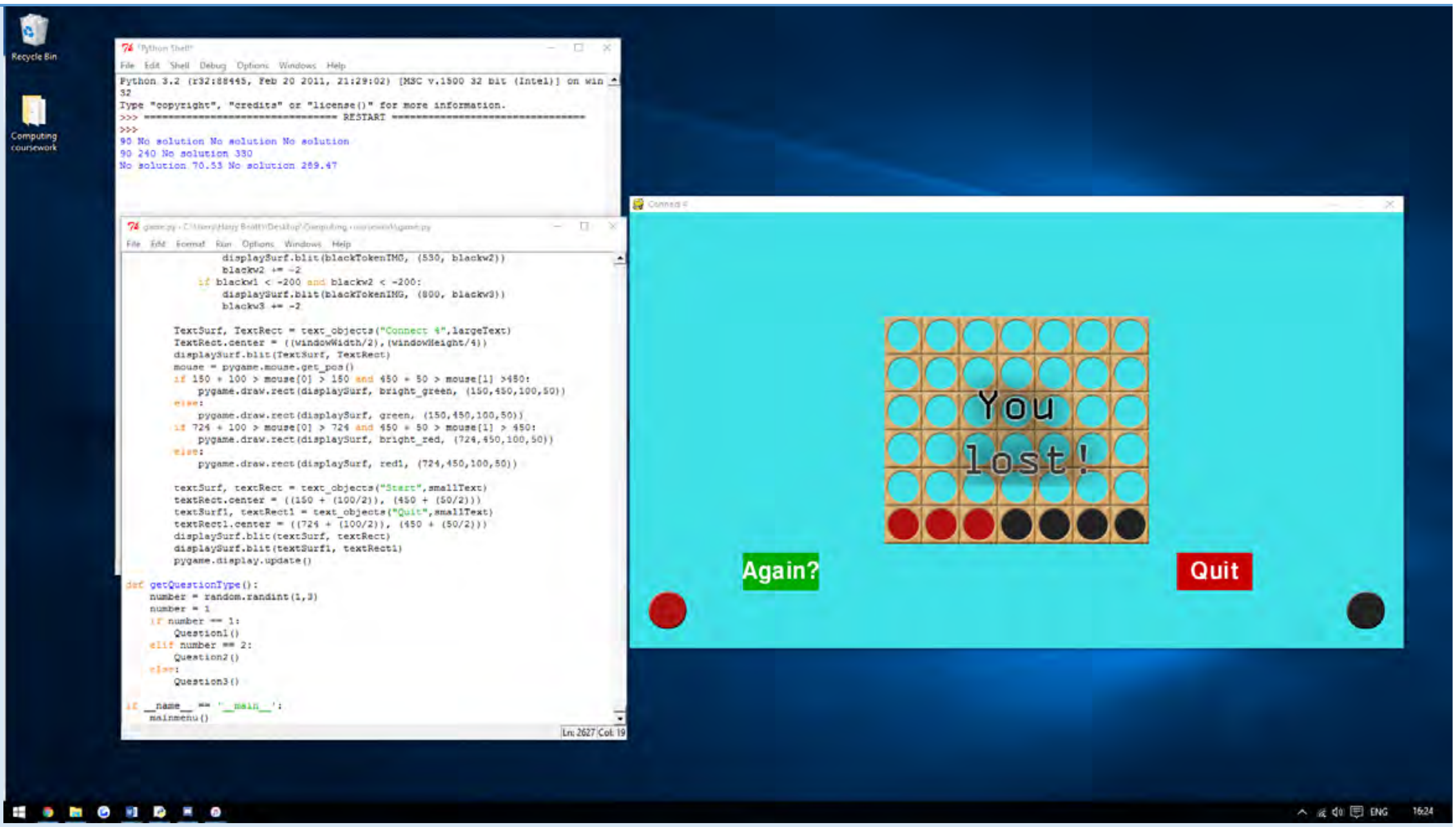
Recycle Bin
Computing coursework
Python Shell
File Edit Shell Debug Options Windows Help
File Edit Format Run Options Windows Help
Ln: 2627 Col: 19
Connect 4
Drag token over top of board
ENG
16:21

```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
90 No solution No solution No solution
```

```
                displaySurf.blit(blackTokenIMG, (530, blackw2))
                blackw2 += -2
            if blackw1 < -200 and blackw2 < -200:
                displaySurf.blit(blackTokenIMG, (800, blackw3))
                blackw3 += -2

        TextSurf, TextRect = text_objects("Connect 4",largeText)
        TextRect.center = ((windowWidth/2),(windowHeight/4))
        displaySurf.blit(TextSurf, TextRect)
        mouse = pygame.mouse.get_pos()
        if 150 + 100 > mouse[0] > 150 and 450 + 50 > mouse[1] >450:
            pygame.draw.rect(displaySurf, bright_green, (150,450,100,50))
        else:
            pygame.draw.rect(displaySurf, green, (150,450,100,50))
        if 724 + 100 > mouse[0] > 724 and 450 + 50 > mouse[1] > 450:
            pygame.draw.rect(displaySurf, bright_red, (724,450,100,50))
        else:
            pygame.draw.rect(displaySurf, red1, (724,450,100,50))

        textSurf, textRect = text_objects("Start",smallText)
        textRect.center = ((150 + (100/2)), (450 + (50/2)))
        textSurf1, textRect1 = text_objects("Quit",smallText)
        textRect1.center = ((724 + (100/2)), (450 + (50/2)))
        displaySurf.blit(textSurf, textRect)
        displaySurf.blit(textSurf1, textRect1)
        pygame.display.update()

def getQuestionType():
    number = random.randint(1,3)
    number = 1
    if number == 1:
        Question1()
    elif number == 2:
        Question2()
    else:
        Question3()

if __name__ == '__main__':
    mainmenu()
```

A help message is displayed when the user has a turn for the first time.

15.

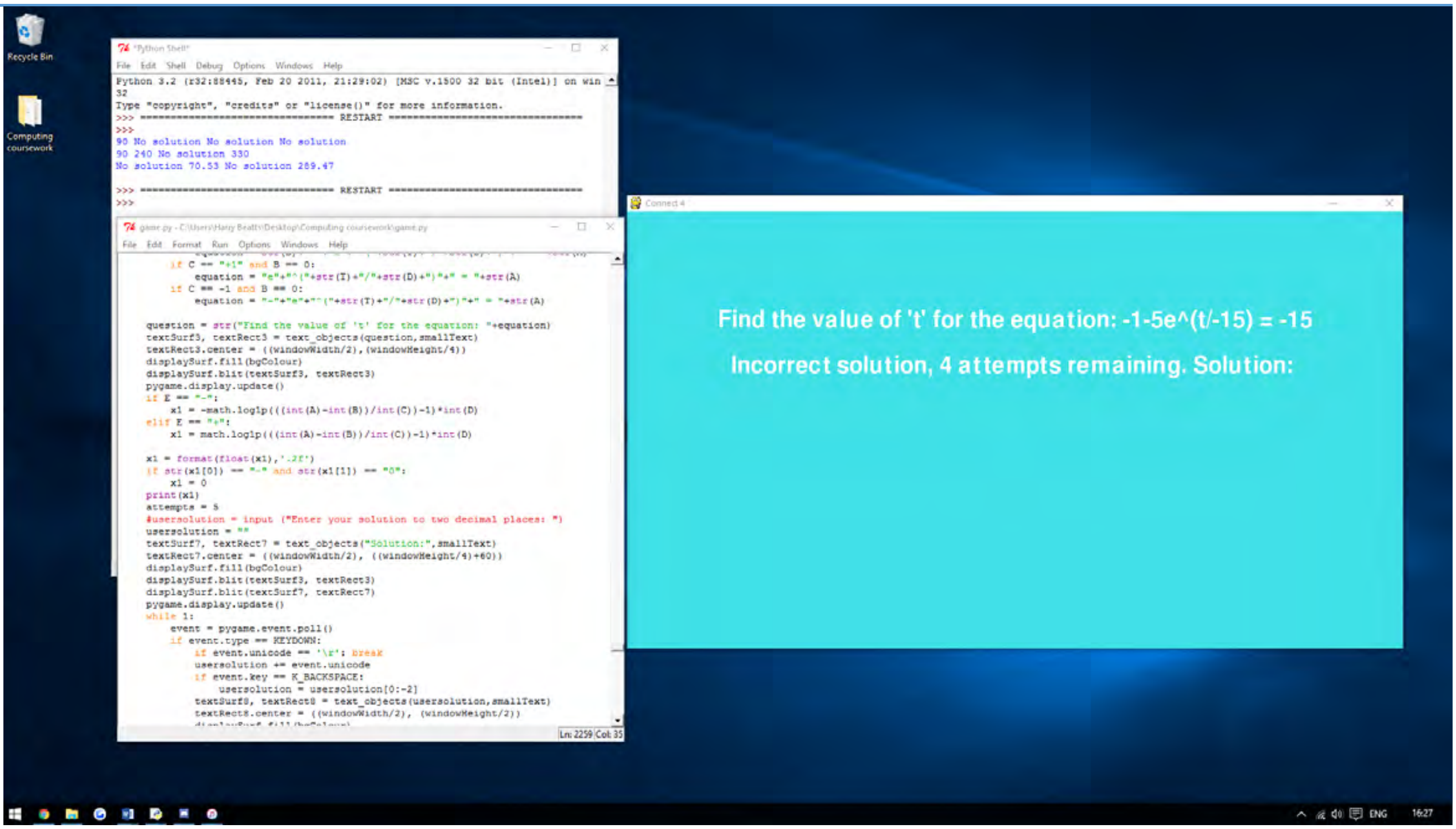


```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
90 No solution No solution No solution
90 240 No solution 330
No solution 70.53 No solution 289.47
```

```
                displaySurf.blit(blackTokenIMG, (530, blackw2))
                blackw2 += -2
            if blackw1 < -200 and blackw2 < -200:
                displaySurf.blit(blackTokenIMG, (800, blackw3))
                blackw3 += -2

        TextSurf, TextRect = text_objects("Connect 4",largeText)
        TextRect.center = ((windowWidth/2),(windowHeight/4))
        displaySurf.blit(TextSurf, TextRect)
        mouse = pygame.mouse.get_pos()
        if 150 + 100 > mouse[0] > 150 and 450 + 50 > mouse[1] >450:
            pygame.draw.rect(displaySurf, bright_green, (150,450,100,50))
        else:
            pygame.draw.rect(displaySurf, green, (150,450,100,50))
        if 724 + 100 > mouse[0] > 724 and 450 + 50 > mouse[1] > 450:
            pygame.draw.rect(displaySurf, bright_red, (724,450,100,50))
        else:
            pygame.draw.rect(displaySurf, red1, (724,450,100,50))

        textSurf, textRect = text_objects("Start",smallText)
        textRect.center = ((150 + (100/2)), (450 + (50/2)))
        textSurf1, textRect1 = text_objects("Quit",smallText)
        textRect1.center = ((724 + (100/2)), (450 + (50/2)))
        displaySurf.blit(textSurf, textRect)
        displaySurf.blit(textSurf1, textRect1)
        pygame.display.update()

def getQuestionType():
    number = random.randint(1,3)
    number = 1
    if number == 1:
        Question1()
    elif number == 2:
        Question2()
    else:
        Question3()

if __name__ == '__main__':
    mainmenu()
```

Game is shown to be won by the computer when there is four counters in a row.

16.



Recycle Bin
Computing coursework
Python Shell
File Edit Shell Debug Options Windows Help

```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
90 No solution No solution No solution
90 240 No solution 330
No solution 70.53 No solution 289.47

>>> ================================ RESTART ================================
>>>
```

game.py - C:\Users\Harry Beatt\Desktop\Computing coursework\game.py
File Edit Format Run Options Windows Help

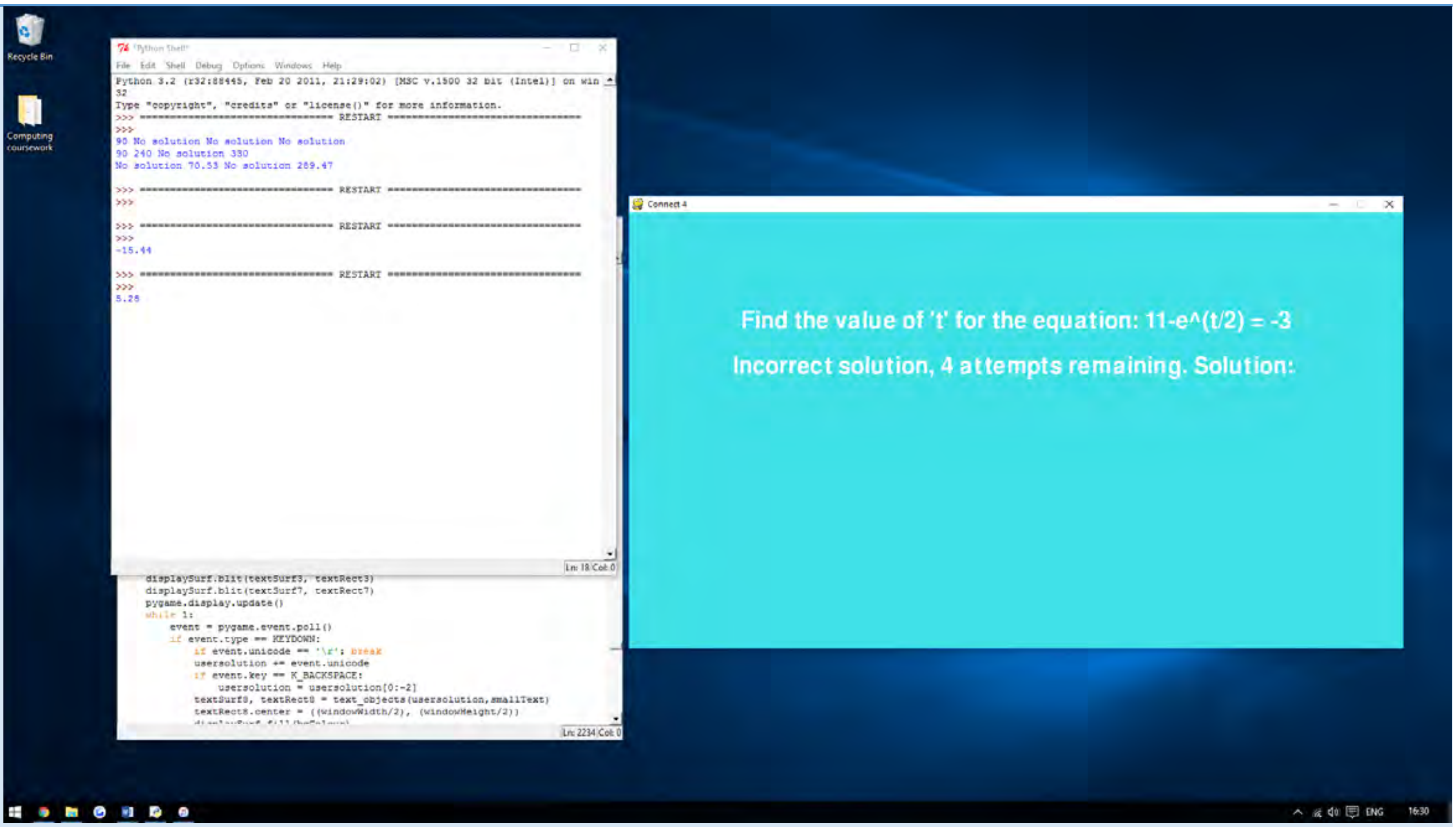
```
        if C == "+1" and B == 0:
            equation = "e"+"^("+str(T)+"/"+str(D)+")"+" = "+str(A)
        if C == -1 and B == 0:
            equation = "-"+"e"+"^("+str(T)+"/"+str(D)+")"+" = "+str(A)

    question = str("Find the value of 't' for the equation: "+equation)
    textSurf3, textRect3 = text_objects(question,smallText)
    textRect3.center = ((windowWidth/2),(windowHeight/4))
    displaySurf.fill(bgColour)
    displaySurf.blit(textSurf3, textRect3)
    pygame.display.update()
    if E == "-":
        x1 = -math.log1p(((int(A)-int(B))/int(C))-1)*int(D)
    elif E == "+":
        x1 = math.log1p(((int(A)-int(B))/int(C))-1)*int(D)

    x1 = format(float(x1),'.2f')
    if str(x1[0]) == "-" and str(x1[1]) == "0":
        x1 = 0
    print(x1)
    attempts = 5
    #usersolution = input ("Enter your solution to two decimal places: ")
    usersolution = ""
    textSurf7, textRect7 = text_objects("Solution:",smallText)
    textRect7.center = ((windowWidth/2), ((windowHeight/4)+60))
    displaySurf.fill(bgColour)
    displaySurf.blit(textSurf3, textRect3)
    displaySurf.blit(textSurf7, textRect7)
    pygame.display.update()
    while 1:
        event = pygame.event.poll()
        if event.type == KEYDOWN:
            if event.unicode == '\r': break
            usersolution += event.unicode
            if event.key == K_BACKSPACE:
                usersolution = usersolution[0:-2]
            textSurf8, textRect8 = text_objects(usersolution,smallText)
            textRect8.center = ((windowWidth/2), (windowHeight/2))
```

Find the value of 't' for the equation: -1-5e^(t/-15) = -15

Incorrect solution, 4 attempts remaining. Solution:

Any value which is not to two decimal places is shown to be incorrect.

17.



```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
90 No solution No solution No solution
90 240 No solution 330
No solution 70.53 No solution 289.47

>>> ================================ RESTART ================================
>>>

>>> ================================ RESTART ================================
>>>
-15.44

>>> ================================ RESTART ================================
>>>
5.28
```

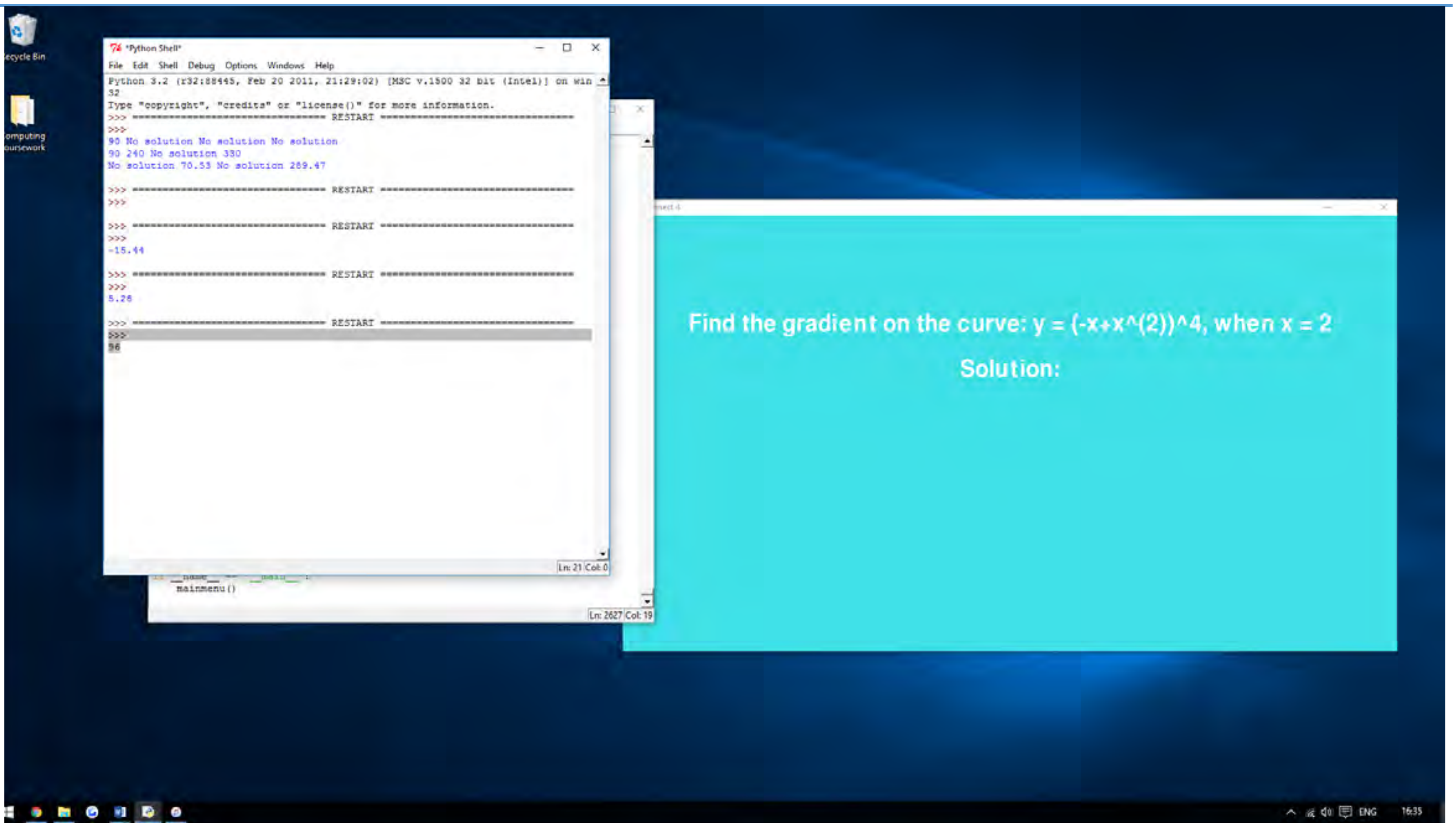
```
    displaySurf.blit(textSurf3, textRect3)
    displaySurf.blit(textSurf7, textRect7)
    pygame.display.update()
    while 1:
        event = pygame.event.poll()
        if event.type == KEYDOWN:
            if event.unicode == '\r': break
            usersolution += event.unicode
            if event.key == K_BACKSPACE:
                usersolution = usersolution[0:-2]
            textSurf8, textRect8 = text_objects(usersolution,smallText)
            textRect8.center = ((windowWidth/2), (windowHeight/2))
            displaySurf.fill(bgColour)
```

Find the value of 't' for the equation: 11-e^(t/2) = -3

Incorrect solution, 4 attempts remaining. Solution:

Error message when a float is not entered.

18.



```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
90 No solution No solution No solution
90 240 No solution 330
No solution 70.53 No solution 289.47

>>> ================================ RESTART ================================
>>>

>>> ================================ RESTART ================================
>>>
-15.44

>>> ================================ RESTART ================================
>>>
5.26

>>> ================================ RESTART ================================
>>>
96
```

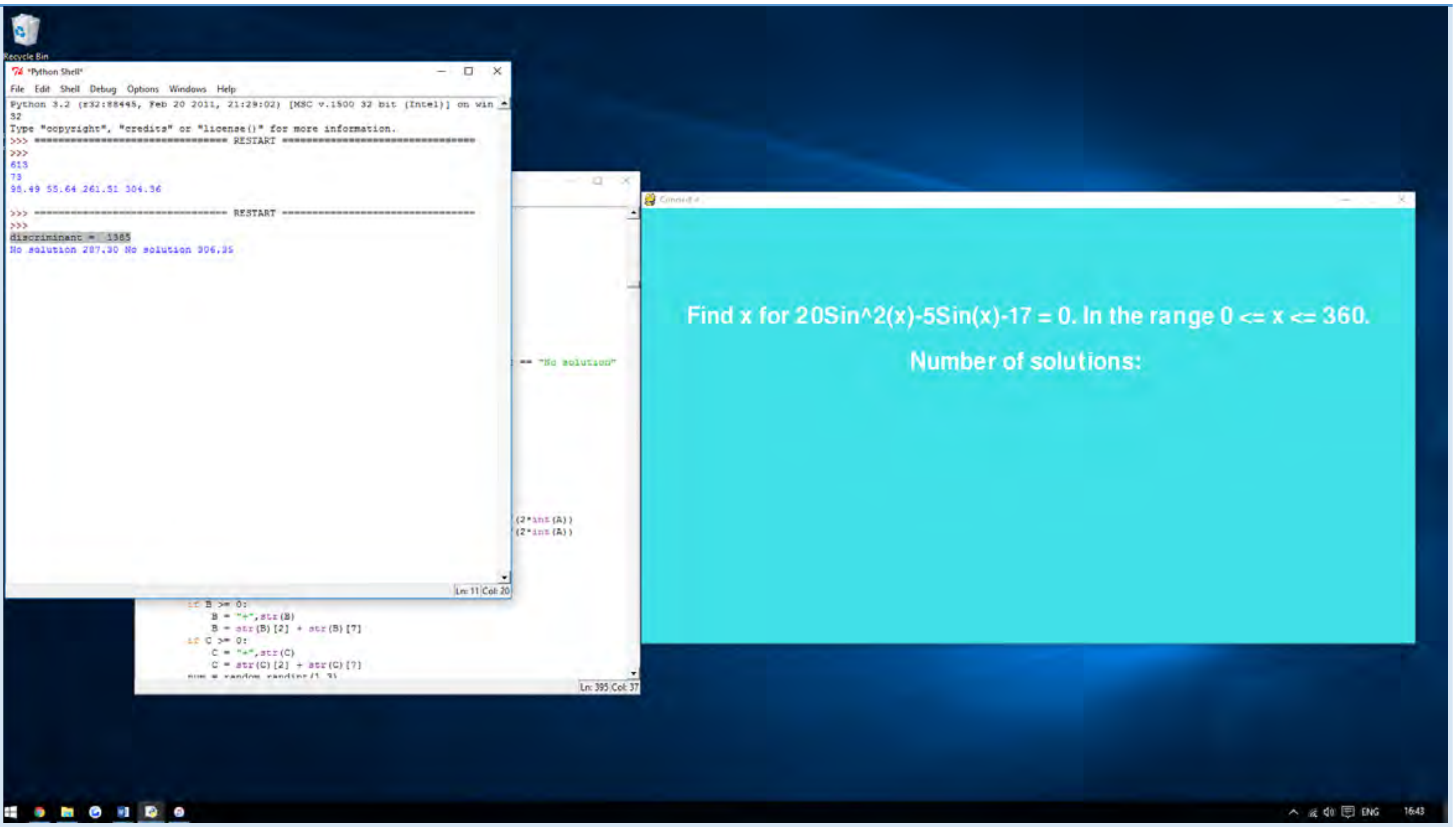
```
mainmenu()
```

Find the gradient on the curve: y = (-x+x^(2))^4, when x = 2

Solution:

Answer needed is shown highlighted in python shell window. The value needed is an integer.

19.



```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
613
73
98.49 55.64 261.51 304.36

>>> ================================ RESTART ================================
>>>
discriminant =  1385
No solution 287.30 No solution 306.25
```

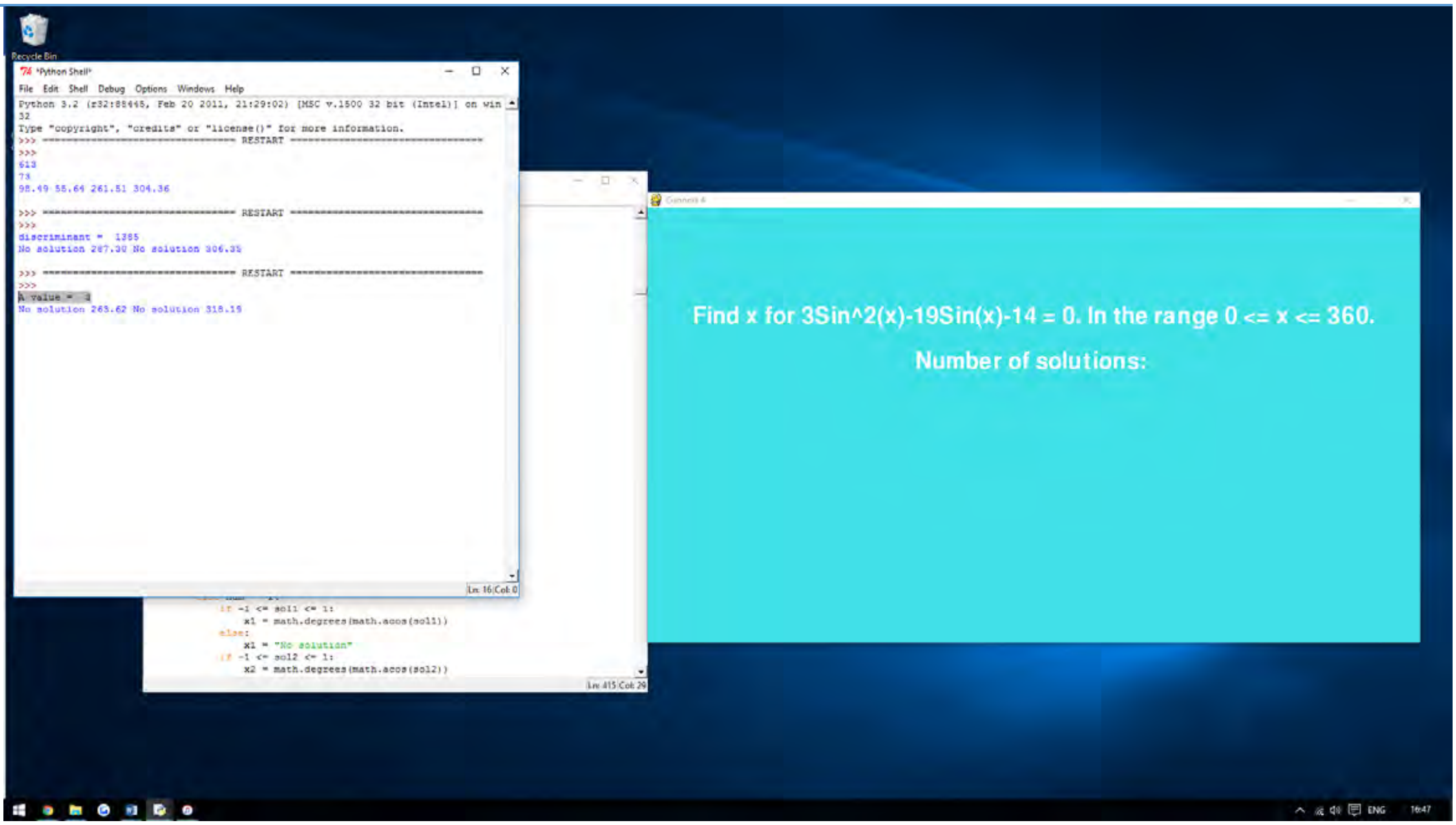
```
== "No solution"
(2*int(A))
(2*int(A))
    if B >= 0:
        B = "+",str(B)
        B = str(B)[2] + str(B)[7]
    if C >= 0:
        C = "+",str(C)
        C = str(C)[2] + str(C)[7]
    num = random.randint(1,3)
```

Find x for 20Sin^2(x)-5Sin(x)-17 = 0. In the range 0 <= x <= 360.

Number of solutions:

The discriminant is shown to be a positive value in the python shell window.

20.



```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
613
73
98.49 55.64 261.51 304.36

>>> ================================ RESTART ================================
>>>
discriminant =  1385
No solution 287.30 No solution 306.35

>>> ================================ RESTART ================================
>>>
A value =  3
No solution 263.62 No solution 318.15
```

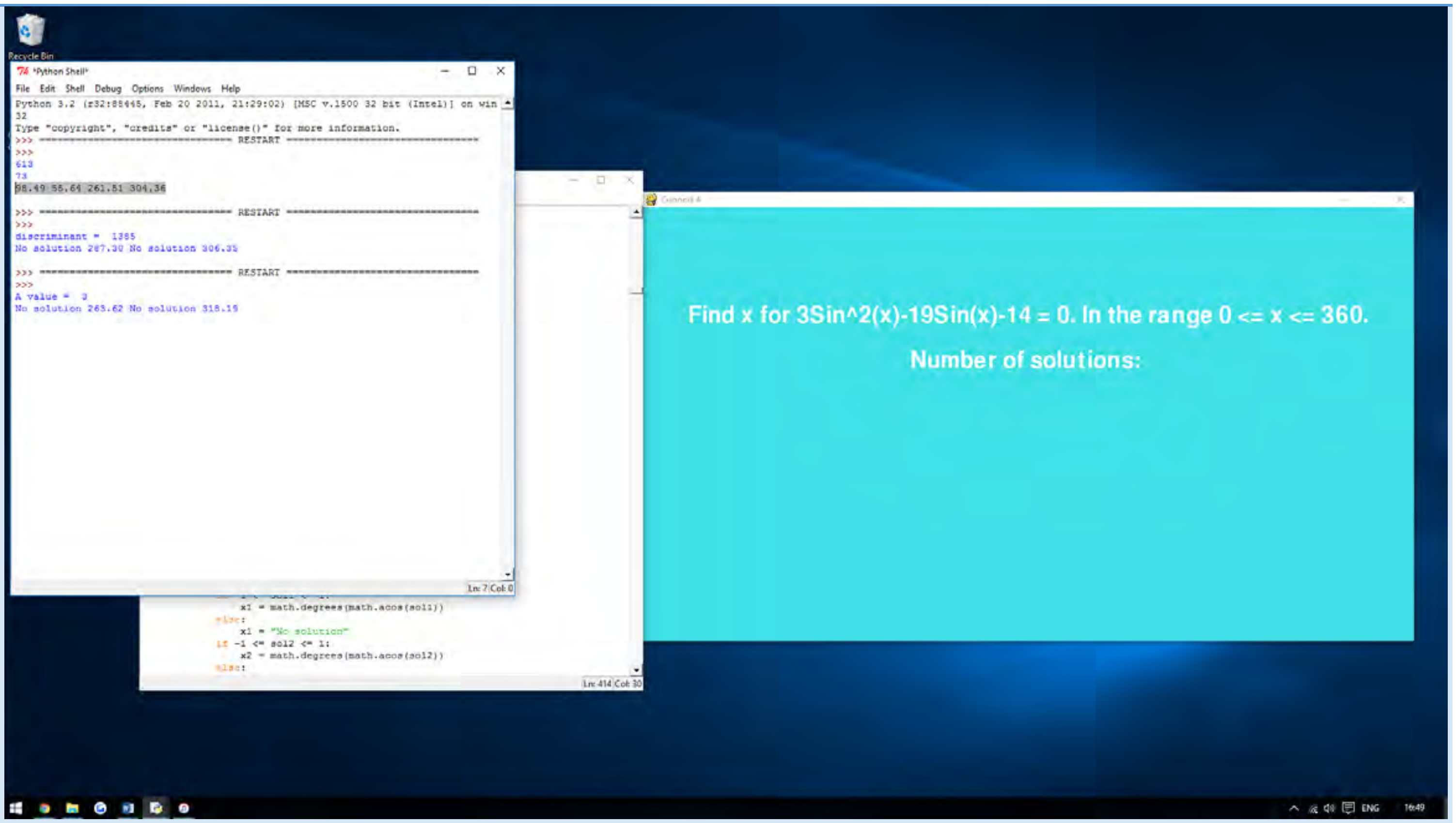
```
        if -1 <= sol1 <= 1:
            x1 = math.degrees(math.acos(sol1))
        else:
            x1 = "No solution"
        if -1 <= sol2 <= 1:
            x2 = math.degrees(math.acos(sol2))
```

Find x for 3Sin^2(x)-19Sin(x)-14 = 0. In the range 0 <= x <= 360.

Number of solutions:

The value of “A” is shown to be 3, which is in the range needed.

21.



```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
613
73
98.49 55.64 261.51 304.36

>>> ================================ RESTART ================================
>>>
discriminant =  1385
No solution 287.30 No solution 306.35

>>> ================================ RESTART ================================
>>>
A value =  3
No solution 263.62 No solution 318.15
```

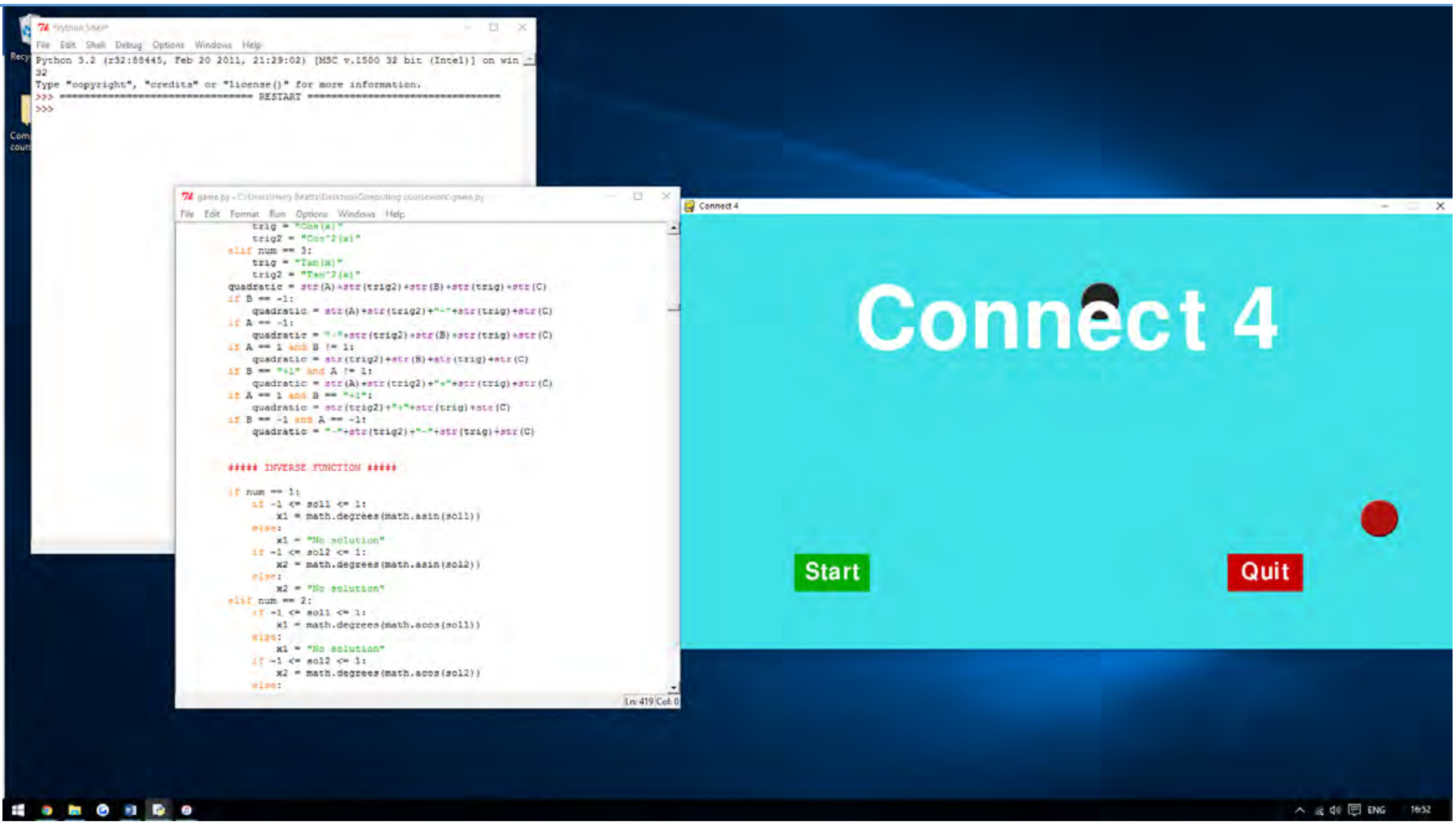
```
        x1 = math.degrees(math.acos(sol1))
    else:
        x1 = "No solution"
    if -1 <= sol2 <= 1:
        x2 = math.degrees(math.acos(sol2))
    else:
```

Find x for 3Sin^2(x)-19Sin(x)-14 = 0. In the range 0 <= x <= 360.

Number of solutions:

The values needed in question one are all in the range specified by the question.

22.

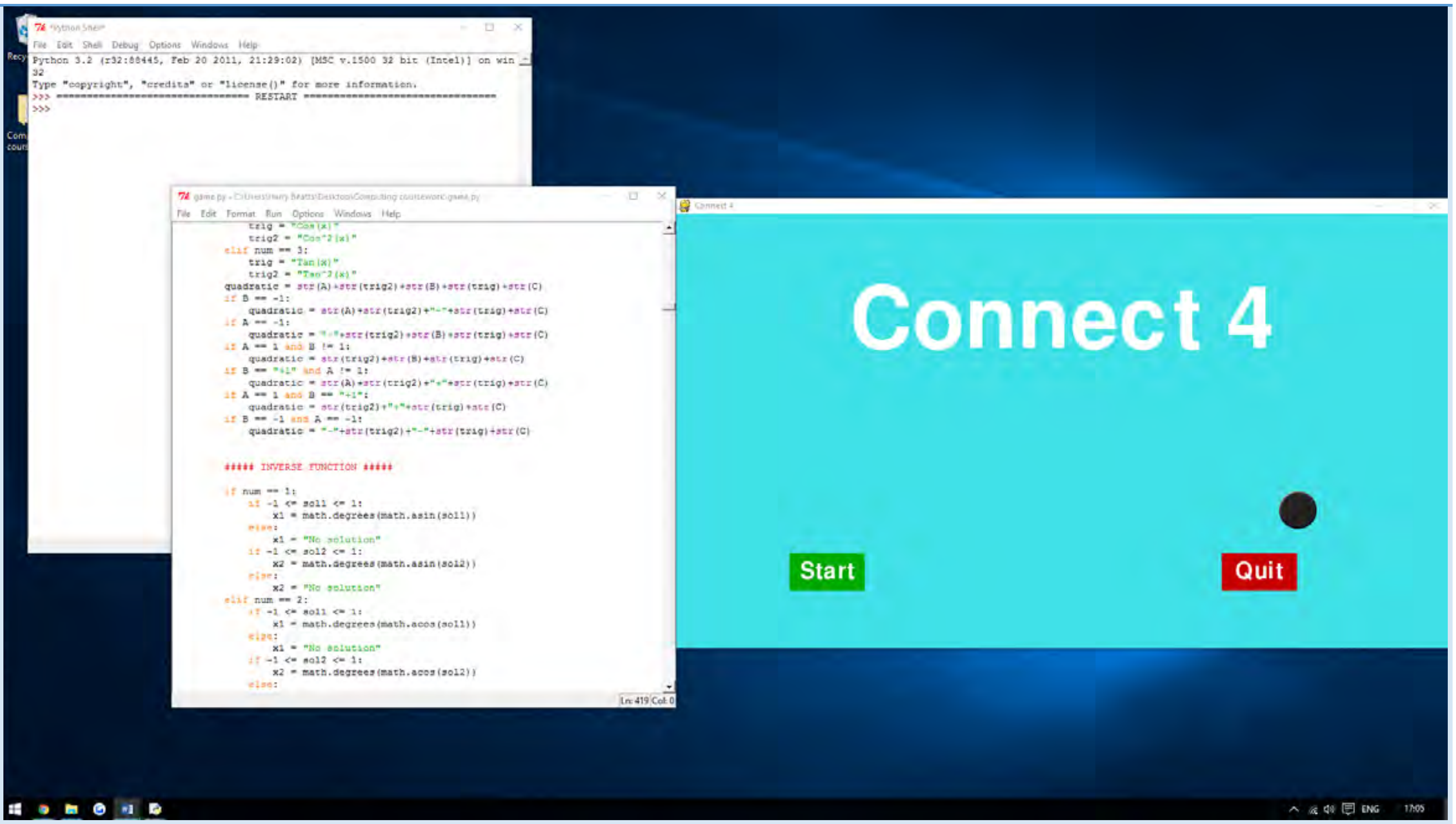

Python Shell
File Edit Shell Debug Options Windows Help
game.py - C:\Users\Harry Beatts\Desktop\Computing coursework\game.py
File Edit Format Run Options Windows Help
Ln: 419 Col: 0
Connect 4
Connect 4
Start
Quit
ENG
16:52

```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
```

```
            trig = "Cos(x)"
            trig2 = "Cos^2(x)"
        elif num == 3:
            trig = "Tan(x)"
            trig2 = "Tan^2(x)"
        quadratic = str(A)+str(trig2)+str(B)+str(trig)+str(C)
        if B == -1:
            quadratic = str(A)+str(trig2)+"-"+str(trig)+str(C)
        if A == -1:
            quadratic = "-"+str(trig2)+str(B)+str(trig)+str(C)
        if A == 1 and B != 1:
            quadratic = str(trig2)+str(B)+str(trig)+str(C)
        if B == "+1" and A != 1:
            quadratic = str(A)+str(trig2)+"+"+str(trig)+str(C)
        if A == 1 and B == "+1":
            quadratic = str(trig2)+"+"+str(trig)+str(C)
        if B == -1 and A == -1:
            quadratic = "-"+str(trig2)+"-"+str(trig)+str(C)


        ##### INVERSE FUNCTION #####

        if num == 1:
            if -1 <= sol1 <= 1:
                x1 = math.degrees(math.asin(sol1))
            else:
                x1 = "No solution"
            if -1 <= sol2 <= 1:
                x2 = math.degrees(math.asin(sol2))
            else:
                x2 = "No solution"
        elif num == 2:
            if -1 <= sol1 <= 1:
                x1 = math.degrees(math.acos(sol1))
            else:
                x1 = "No solution"
            if -1 <= sol2 <= 1:
                x2 = math.degrees(math.acos(sol2))
            else:
```

The menu tokens are shown to be off set.

23.



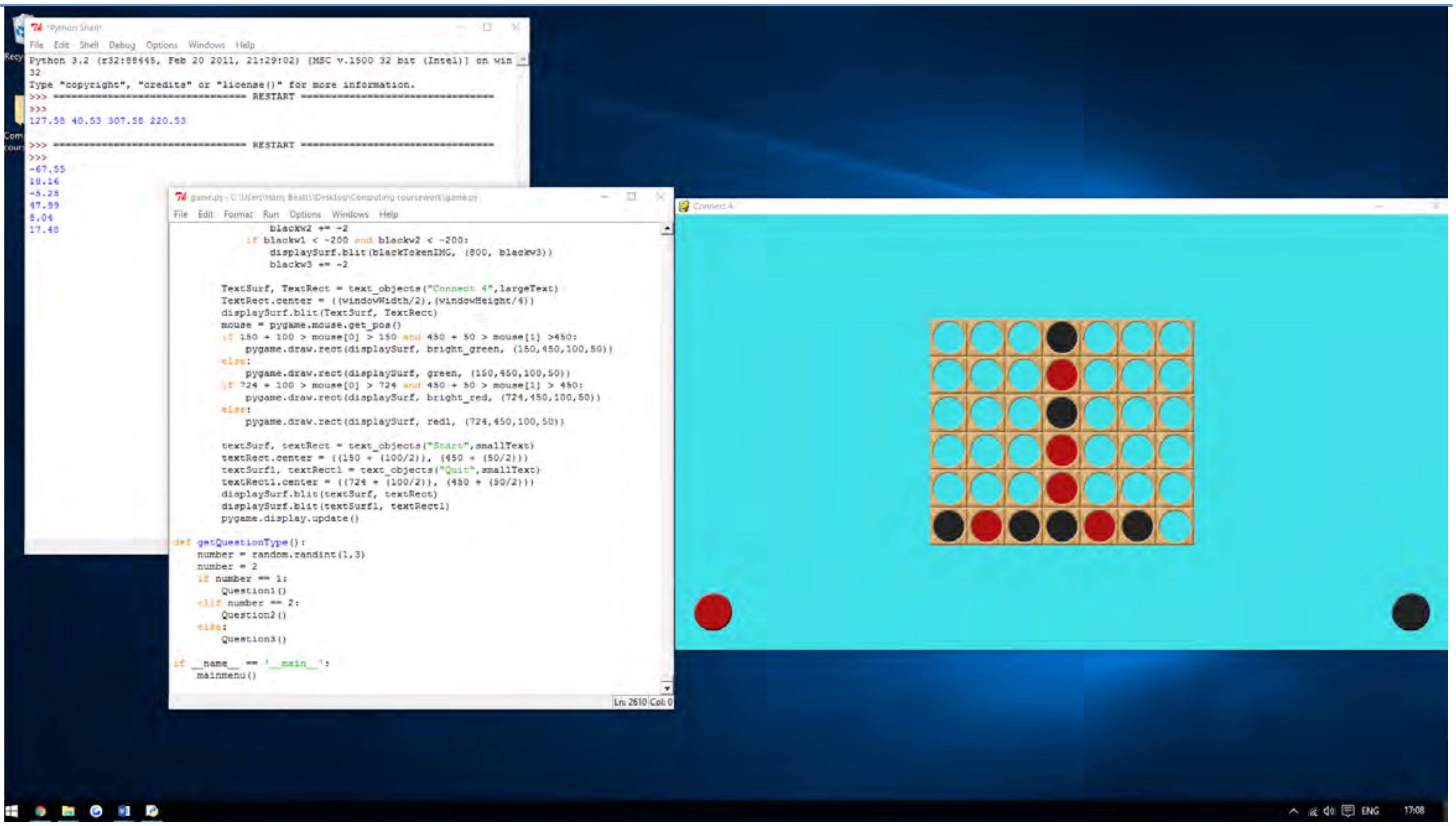
Python Shell
File Edit Shell Debug Options Windows Help
game.py - C:\Users\Harry Beatts\Desktop\Computing coursework\game.py
File Edit Format Run Options Windows Help
Ln: 419 Col: 0
Connect 4
Connect 4
Start
Quit
ENG
17:05

```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
```

```
            trig = "Cos(x)"
            trig2 = "Cos^2(x)"
        elif num == 3:
            trig = "Tan(x)"
            trig2 = "Tan^2(x)"
        quadratic = str(A)+str(trig2)+str(B)+str(trig)+str(C)
        if B == -1:
            quadratic = str(A)+str(trig2)+"-"+str(trig)+str(C)
        if A == -1:
            quadratic = "-"+str(trig2)+str(B)+str(trig)+str(C)
        if A == 1 and B != 1:
            quadratic = str(trig2)+str(B)+str(trig)+str(C)
        if B == "+1" and A != 1:
            quadratic = str(A)+str(trig2)+"+"+str(trig)+str(C)
        if A == 1 and B == "+1":
            quadratic = str(trig2)+"+"+str(trig)+str(C)
        if B == -1 and A == -1:
            quadratic = "-"+str(trig2)+"-"+str(trig)+str(C)


        ##### INVERSE FUNCTION #####

        if num == 1:
            if -1 <= sol1 <= 1:
                x1 = math.degrees(math.asin(sol1))
            else:
                x1 = "No solution"
            if -1 <= sol2 <= 1:
                x2 = math.degrees(math.asin(sol2))
            else:
                x2 = "No solution"
        elif num == 2:
            if -1 <= sol1 <= 1:
                x1 = math.degrees(math.acos(sol1))
            else:
                x1 = "No solution"
            if -1 <= sol2 <= 1:
                x2 = math.degrees(math.acos(sol2))
            else:
```

Menu tokens continue to look when an option has not been selected.

24.

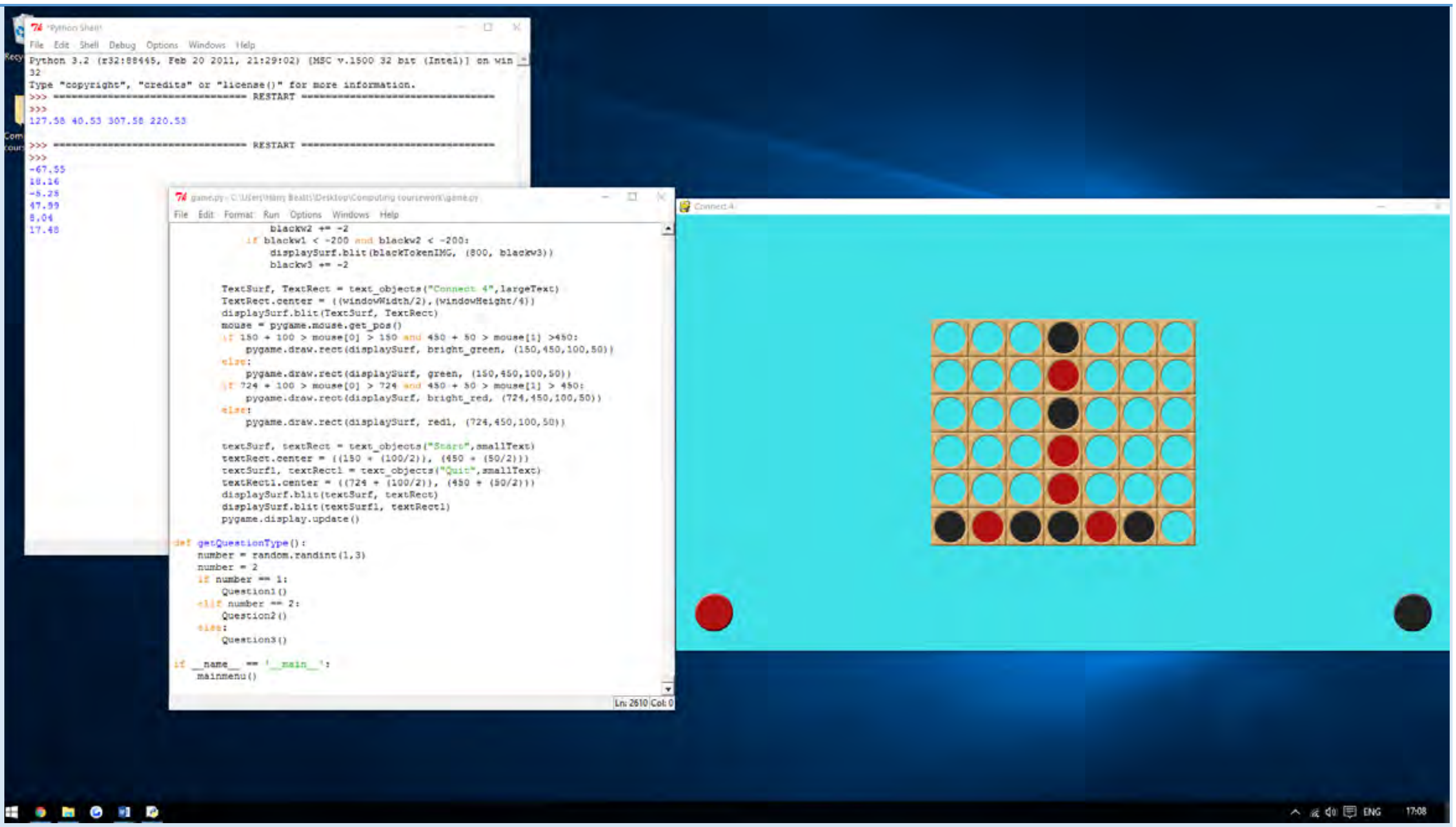

Python Shell
File Edit Shell Debug Options Windows Help
game.py - C:\Users\Harry Beatt\Desktop\Computing coursework\game.py
File Edit Format Run Options Windows Help
Ln: 2610 Col: 0
Connect 4
ENG
17:08

```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
127.58 40.53 307.58 220.53

>>> ================================ RESTART ================================
>>>
-67.55
18.16
-5.28
47.99
8.04
17.48
```

```
                blackw2 += -2
            if blackw1 < -200 and blackw2 < -200:
                displaySurf.blit(blackTokenIMG, (800, blackw3))
                blackw3 += -2

        TextSurf, TextRect = text_objects("Connect 4",largeText)
        TextRect.center = ((windowWidth/2),(windowHeight/4))
        displaySurf.blit(TextSurf, TextRect)
        mouse = pygame.mouse.get_pos()
        if 150 + 100 > mouse[0] > 150 and 450 + 50 > mouse[1] >450:
            pygame.draw.rect(displaySurf, bright_green, (150,450,100,50))
        else:
            pygame.draw.rect(displaySurf, green, (150,450,100,50))
        if 724 + 100 > mouse[0] > 724 and 450 + 50 > mouse[1] > 450:
            pygame.draw.rect(displaySurf, bright_red, (724,450,100,50))
        else:
            pygame.draw.rect(displaySurf, red1, (724,450,100,50))

        textSurf, textRect = text_objects("Start",smallText)
        textRect.center = ((150 + (100/2)), (450 + (50/2)))
        textSurf1, textRect1 = text_objects("Quit",smallText)
        textRect1.center = ((724 + (100/2)), (450 + (50/2)))
        displaySurf.blit(textSurf, textRect)
        displaySurf.blit(textSurf1, textRect1)
        pygame.display.update()

def getQuestionType():
    number = random.randint(1,3)
    number = 2
    if number == 1:
        Question1()
    elif number == 2:
        Question2()
    else:
        Question3()

if __name__ == '__main__':
    mainmenu()
```

The computer is not trying to place a move in a column which is full.

25.



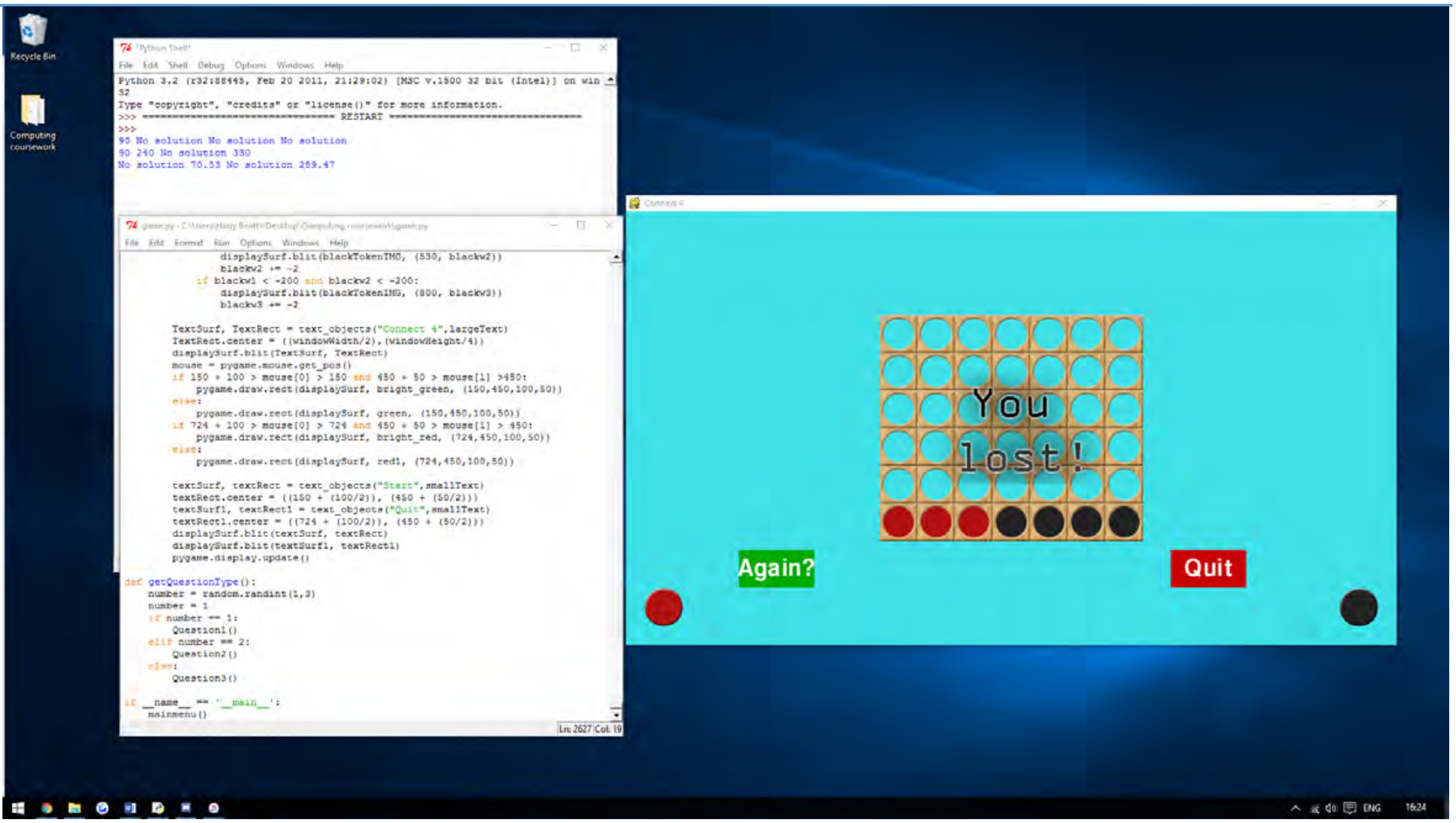
```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>> 
127.58 40.53 307.58 220.53

>>> ================================ RESTART ================================
>>> 
-67.55
18.16
-5.28
47.99
8.04
17.48
```

```
                blackw2 += -2
            if blackw1 < -200 and blackw2 < -200:
                displaySurf.blit(blackTokenIMG, (800, blackw3))
                blackw3 += -2

        TextSurf, TextRect = text_objects("Connect 4",largeText)
        TextRect.center = ((windowWidth/2),(windowHeight/4))
        displaySurf.blit(TextSurf, TextRect)
        mouse = pygame.mouse.get_pos()
        if 150 + 100 > mouse[0] > 150 and 450 + 50 > mouse[1] >450:
            pygame.draw.rect(displaySurf, bright_green, (150,450,100,50))
        else:
            pygame.draw.rect(displaySurf, green, (150,450,100,50))
        if 724 + 100 > mouse[0] > 724 and 450 + 50 > mouse[1] > 450:
            pygame.draw.rect(displaySurf, bright_red, (724,450,100,50))
        else:
            pygame.draw.rect(displaySurf, red1, (724,450,100,50))

        textSurf, textRect = text_objects("Start",smallText)
        textRect.center = ((150 + (100/2)), (450 + (50/2)))
        textSurf1, textRect1 = text_objects("Quit",smallText)
        textRect1.center = ((724 + (100/2)), (450 + (50/2)))
        displaySurf.blit(textSurf, textRect)
        displaySurf.blit(textSurf1, textRect1)
        pygame.display.update()

def getQuestionType():
    number = random.randint(1,3)
    number = 2
    if number == 1:
        Question1()
    elif number == 2:
        Question2()
    else:
        Question3()

if __name__ == '__main__':
    mainmenu()
```

Python Shell
File Edit Shell Debug Options Windows Help
game.py - C:\Users\Harry Beatt\Desktop\Computing coursework\game.py
File Edit Format Run Options Windows Help
Ln: 2610 Col: 0
Connect 4
ENG
17:08

It is not possible for the user to place a token in a column which is full.

26.

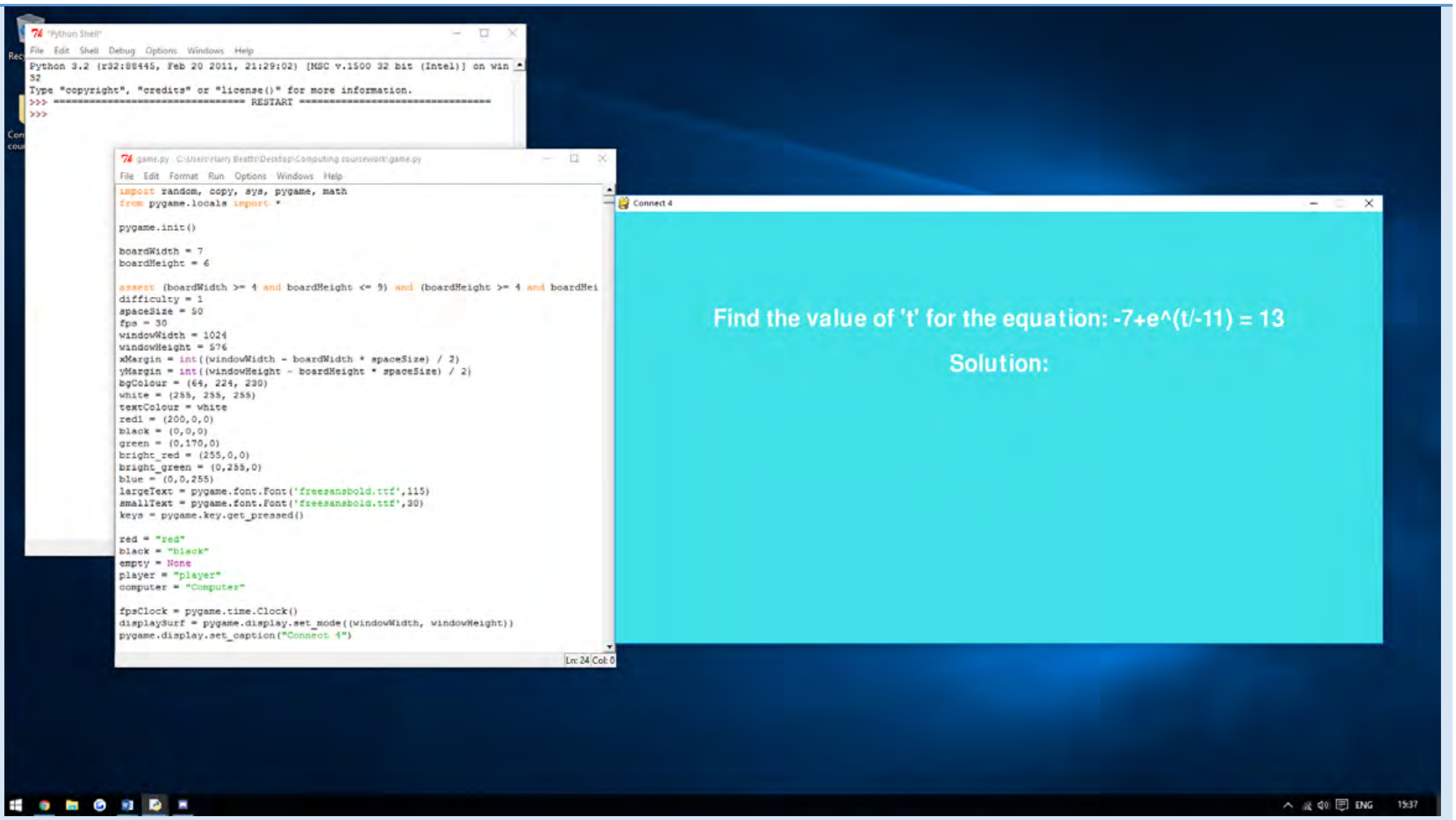

Recycle Bin
Computing coursework
Python Shell
File Edit Shell Debug Options Windows Help
game.py - C:\Users\Harry Bratt\Desktop\Computing coursework\game.py
File Edit Format Run Options Windows Help
Connect 4
You lost!
Again?
Quit
Ln: 2627 Col: 19
ENG
16:24

```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
90 No solution No solution No solution
90 240 No solution 330
No solution 70.53 No solution 289.47
```

```
            displaySurf.blit(blackTokenIMG, (530, blackw2))
            blackw2 += -2
        if blackw1 < -200 and blackw2 < -200:
            displaySurf.blit(blackTokenIMG, (800, blackw3))
            blackw3 += -2

    TextSurf, TextRect = text_objects("Connect 4",largeText)
    TextRect.center = ((windowWidth/2),(windowHeight/4))
    displaySurf.blit(TextSurf, TextRect)
    mouse = pygame.mouse.get_pos()
    if 150 + 100 > mouse[0] > 150 and 450 + 50 > mouse[1] >450:
        pygame.draw.rect(displaySurf, bright_green, (150,450,100,50))
    else:
        pygame.draw.rect(displaySurf, green, (150,450,100,50))
    if 724 + 100 > mouse[0] > 724 and 450 + 50 > mouse[1] > 450:
        pygame.draw.rect(displaySurf, bright_red, (724,450,100,50))
    else:
        pygame.draw.rect(displaySurf, red1, (724,450,100,50))

    textSurf, textRect = text_objects("Start",smallText)
    textRect.center = ((150 + (100/2)), (450 + (50/2)))
    textSurf1, textRect1 = text_objects("Quit",smallText)
    textRect1.center = ((724 + (100/2)), (450 + (50/2)))
    displaySurf.blit(textSurf, textRect)
    displaySurf.blit(textSurf1, textRect1)
    pygame.display.update()

def getQuestionType():
    number = random.randint(1,3)
    number = 1
    if number == 1:
        Question1()
    elif number == 2:
        Question2()
    else:
        Question3()

if __name__ == '__main__':
    mainmenu()
```

There is no change when the user clicks close to the start button.

27.



```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
```

```
import random, copy, sys, pygame, math
from pygame.locals import *

pygame.init()

boardWidth = 7
boardHeight = 6

assert (boardWidth >= 4 and boardHeight <= 9) and (boardHeight >= 4 and boardHei
difficulty = 1
spaceSize = 50
fps = 30
windowWidth = 1024
windowHeight = 576
xMargin = int((windowWidth - boardWidth * spaceSize) / 2)
yMargin = int((windowHeight - boardHeight * spaceSize) / 2)
bgColour = (64, 224, 230)
white = (255, 255, 255)
textColour = white
red1 = (200,0,0)
black = (0,0,0)
green = (0,170,0)
bright_red = (255,0,0)
bright_green = (0,255,0)
blue = (0,0,255)
largeText = pygame.font.Font('freesansbold.ttf',115)
smallText = pygame.font.Font('freesansbold.ttf',30)
keys = pygame.key.get_pressed()

red = "red"
black = "black"
empty = None
player = "player"
computer = "Computer"

fpsClock = pygame.time.Clock()
displaySurf = pygame.display.set_mode((windowWidth, windowHeight))
pygame.display.set_caption("Connect 4")
```

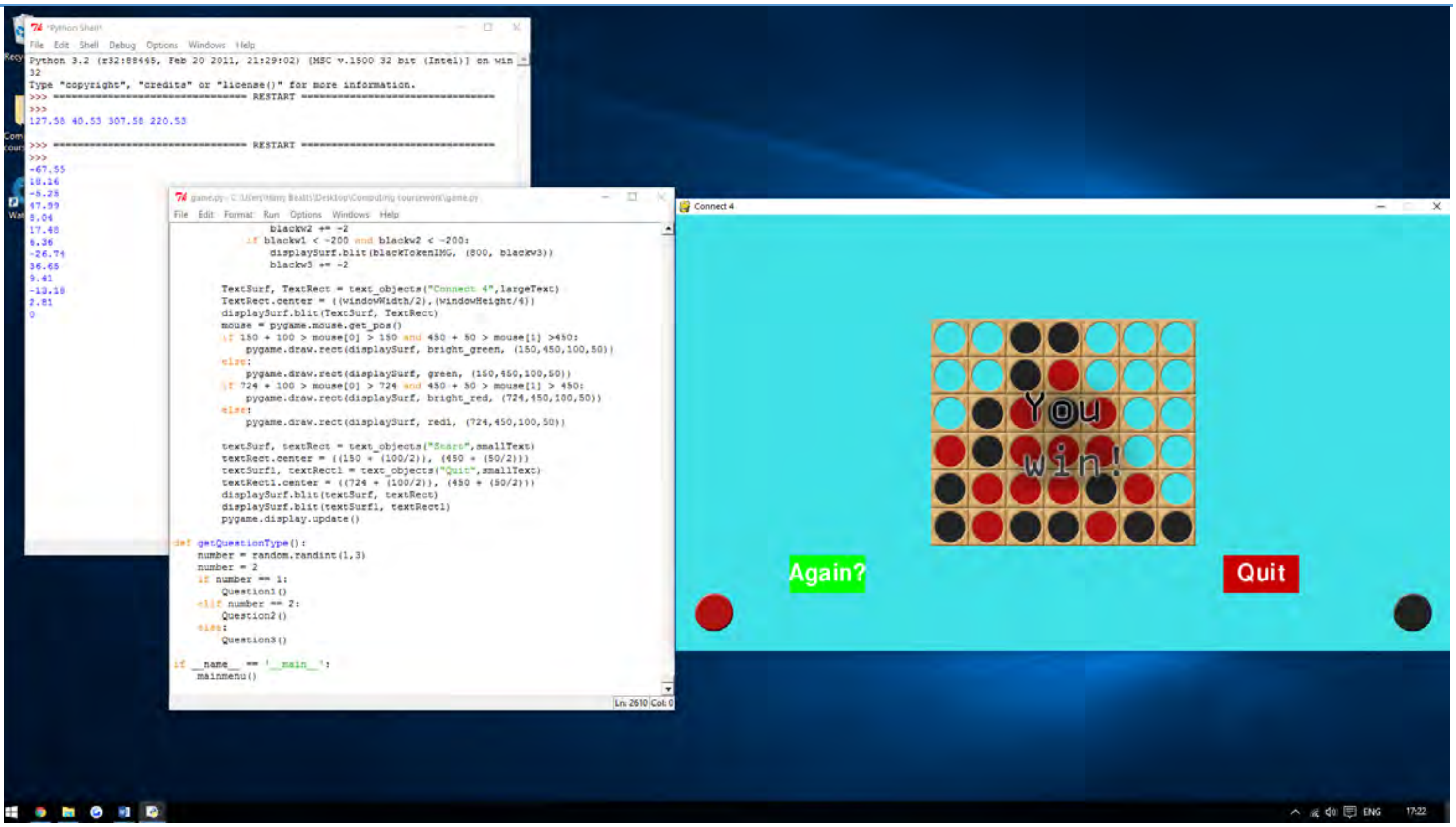
Find the value of 't' for the equation: -7+e^(t/-11) = 13

Solution:

Python Shell
File Edit Shell Debug Options Windows Help
game.py - C:\Users\Harry Beatts\Desktop\Computing coursework\game.py
File Edit Format Run Options Windows Help
Ln: 24 Col: 0
Connect 4
ENG
15:37

The start button will make the game restart, and will go back to the question screen once

28.

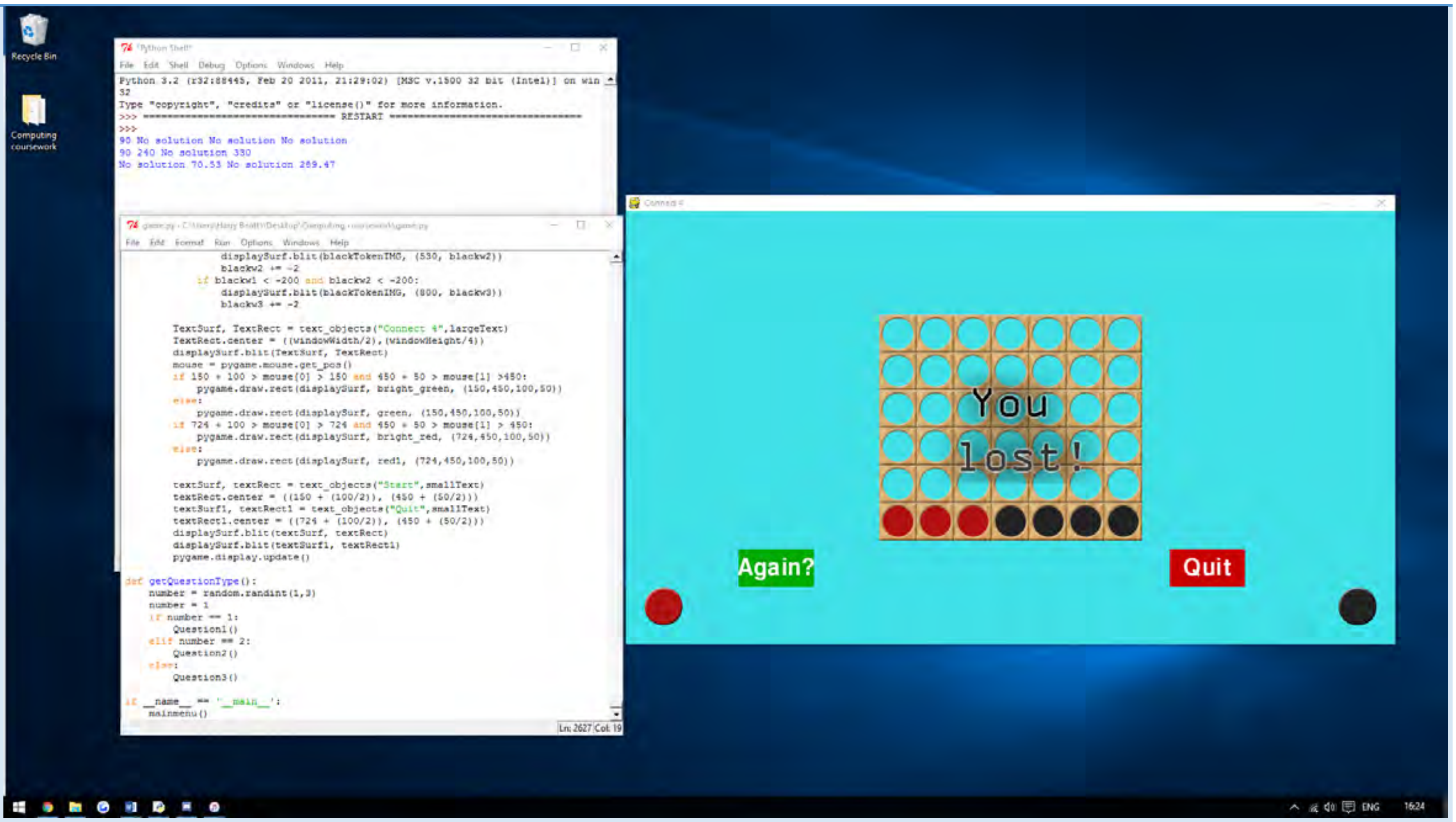

Python Shell
File Edit Shell Debug Options Windows Help
game.py - C:\Users\Harry Beatt\Desktop\Computing coursework\game.py
File Edit Format Run Options Windows Help
Ln: 2610 Col: 0
Connect 4
You win!
Again?
Quit
ENG
17:22

```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
127.58 40.53 307.58 220.53

>>> ================================ RESTART ================================
>>>
-67.55
18.16
-5.28
47.99
8.04
17.48
6.36
-26.74
36.65
9.41
-13.18
2.81
0
```

```
                blackw2 += -2
            if blackw1 < -200 and blackw2 < -200:
                displaySurf.blit(blackTokenIMG, (800, blackw3))
                blackw3 += -2

        TextSurf, TextRect = text_objects("Connect 4",largeText)
        TextRect.center = ((windowWidth/2),(windowHeight/4))
        displaySurf.blit(TextSurf, TextRect)
        mouse = pygame.mouse.get_pos()
        if 150 + 100 > mouse[0] > 150 and 450 + 50 > mouse[1] >450:
            pygame.draw.rect(displaySurf, bright_green, (150,450,100,50))
        else:
            pygame.draw.rect(displaySurf, green, (150,450,100,50))
        if 724 + 100 > mouse[0] > 724 and 450 + 50 > mouse[1] > 450:
            pygame.draw.rect(displaySurf, bright_red, (724,450,100,50))
        else:
            pygame.draw.rect(displaySurf, red1, (724,450,100,50))

        textSurf, textRect = text_objects("Start",smallText)
        textRect.center = ((150 + (100/2)), (450 + (50/2)))
        textSurf1, textRect1 = text_objects("Quit",smallText)
        textRect1.center = ((724 + (100/2)), (450 + (50/2)))
        displaySurf.blit(textSurf, textRect)
        displaySurf.blit(textSurf1, textRect1)
        pygame.display.update()

def getQuestionType():
    number = random.randint(1,3)
    number = 2
    if number == 1:
        Question1()
    elif number == 2:
        Question2()
    else:
        Question3()

if __name__ == '__main__':
    mainmenu()
```

Start button is shown to become brighter hen the mouse is hovered over it.

39.

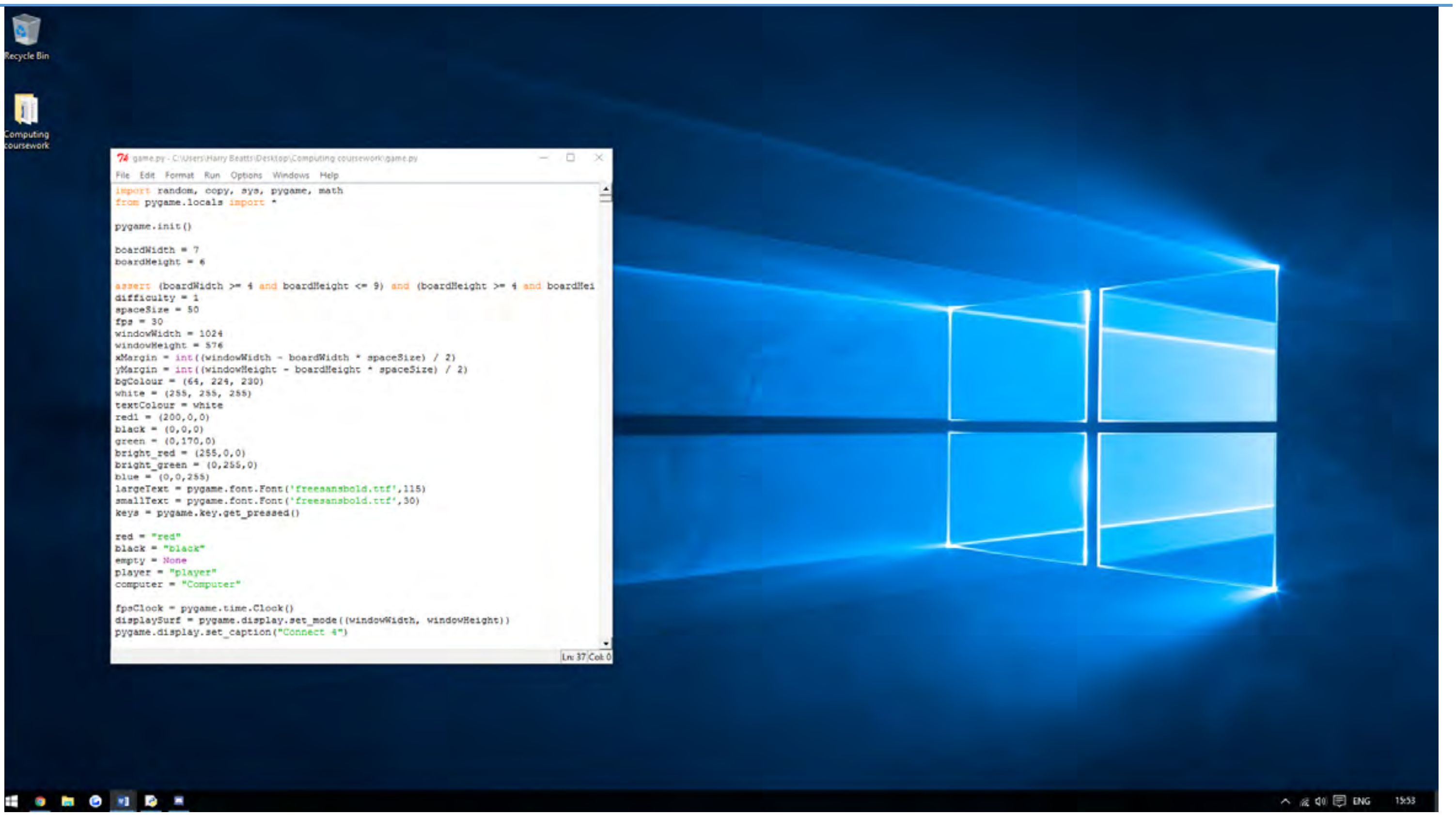

Recycle Bin
Computing coursework
Python Shell
File Edit Shell Debug Options Windows Help
game.py - C:\Users\Harry Bratt\Desktop\Computing coursework\game.py
File Edit Format Run Options Windows Help
Ln: 2627 Col: 19
Connect 4
You lost!
Again?
Quit
ENG
16:24

```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
90 No solution No solution No solution
90 240 No solution 330
No solution 70.53 No solution 289.47
```

```
                displaySurf.blit(blackTokenIMG, (530, blackw2))
                blackw2 += -2
            if blackw1 < -200 and blackw2 < -200:
                displaySurf.blit(blackTokenIMG, (800, blackw3))
                blackw3 += -2

        TextSurf, TextRect = text_objects("Connect 4",largeText)
        TextRect.center = ((windowWidth/2),(windowHeight/4))
        displaySurf.blit(TextSurf, TextRect)
        mouse = pygame.mouse.get_pos()
        if 150 + 100 > mouse[0] > 150 and 450 + 50 > mouse[1] >450:
            pygame.draw.rect(displaySurf, bright_green, (150,450,100,50))
        else:
            pygame.draw.rect(displaySurf, green, (150,450,100,50))
        if 724 + 100 > mouse[0] > 724 and 450 + 50 > mouse[1] > 450:
            pygame.draw.rect(displaySurf, bright_red, (724,450,100,50))
        else:
            pygame.draw.rect(displaySurf, red1, (724,450,100,50))

        textSurf, textRect = text_objects("Start",smallText)
        textRect.center = ((150 + (100/2)), (450 + (50/2)))
        textSurf1, textRect1 = text_objects("Quit",smallText)
        textRect1.center = ((724 + (100/2)), (450 + (50/2)))
        displaySurf.blit(textSurf, textRect)
        displaySurf.blit(textSurf1, textRect1)
        pygame.display.update()

def getQuestionType():
    number = random.randint(1,3)
    number = 1
    if number == 1:
        Question1()
    elif number == 2:
        Question2()
    else:
        Question3()

if __name__ == '__main__':
    mainmenu()
```

There is no change when the user clicks close to the quit button.

30.

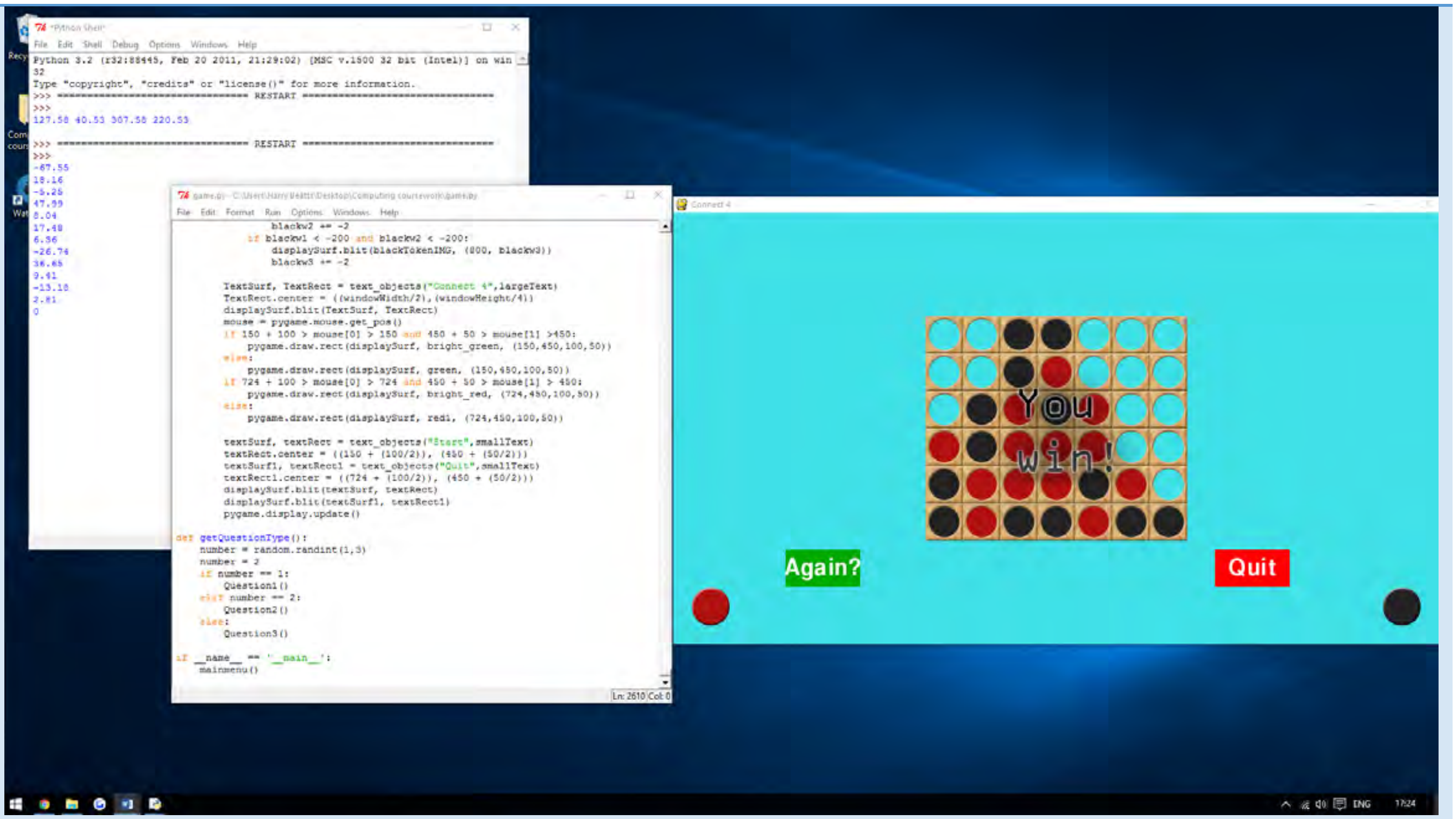


Recycle Bin
Computing coursework
game.py - C:\Users\Harry Beatts\Desktop\Computing coursework\game.py
File Edit Format Run Options Windows Help
Ln: 37 Col: 0
ENG
15:53

```
import random, copy, sys, pygame, math
from pygame.locals import *

pygame.init()

boardWidth = 7
boardHeight = 6

assert (boardWidth >= 4 and boardHeight <= 9) and (boardHeight >= 4 and boardHei
difficulty = 1
spaceSize = 50
fps = 30
windowWidth = 1024
windowHeight = 576
xMargin = int((windowWidth - boardWidth * spaceSize) / 2)
yMargin = int((windowHeight - boardHeight * spaceSize) / 2)
bgColour = (64, 224, 230)
white = (255, 255, 255)
textColour = white
red1 = (200,0,0)
black = (0,0,0)
green = (0,170,0)
bright_red = (255,0,0)
bright_green = (0,255,0)
blue = (0,0,255)
largeText = pygame.font.Font('freesansbold.ttf',115)
smallText = pygame.font.Font('freesansbold.ttf',30)
keys = pygame.key.get_pressed()

red = "red"
black = "black"
empty = None
player = "player"
computer = "Computer"

fpsClock = pygame.time.Clock()
displaySurf = pygame.display.set_mode((windowWidth, windowHeight))
pygame.display.set_caption("Connect 4")
```

The game window has close when the quit button was clicked on.

31.


Python Shell
File Edit Shell Debug Options Windows Help
game.py - C:\Users\Harry Beattie\Desktop\Computing coursework\game.py
File Edit Format Run Options Windows Help
Ln: 2610 Col: 0
Connect 4
You win!
Again?
Quit
ENG
17:24

```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
127.58 40.53 307.58 220.53

>>> ================================ RESTART ================================
>>>
-67.55
18.16
-5.25
47.99
8.04
17.48
6.36
-26.74
36.65
9.41
-13.18
2.81
0
```

```
                blackw2 += -2
            if blackw1 < -200 and blackw2 < -200:
                displaySurf.blit(blackTokenIMG, (800, blackw3))
                blackw3 += -2

        TextSurf, TextRect = text_objects("Connect 4",largeText)
        TextRect.center = ((windowWidth/2),(windowHeight/4))
        displaySurf.blit(TextSurf, TextRect)
        mouse = pygame.mouse.get_pos()
        if 150 + 100 > mouse[0] > 150 and 450 + 50 > mouse[1] >450:
            pygame.draw.rect(displaySurf, bright_green, (150,450,100,50))
        else:
            pygame.draw.rect(displaySurf, green, (150,450,100,50))
        if 724 + 100 > mouse[0] > 724 and 450 + 50 > mouse[1] > 450:
            pygame.draw.rect(displaySurf, bright_red, (724,450,100,50))
        else:
            pygame.draw.rect(displaySurf, red1, (724,450,100,50))

        textSurf, textRect = text_objects("Start",smallText)
        textRect.center = ((150 + (100/2)), (450 + (50/2)))
        textSurf1, textRect1 = text_objects("Quit",smallText)
        textRect1.center = ((724 + (100/2)), (450 + (50/2)))
        displaySurf.blit(textSurf, textRect)
        displaySurf.blit(textSurf1, textRect1)
        pygame.display.update()

def getQuestionType():
    number = random.randint(1,3)
    number = 2
    if number == 1:
        Question1()
    elif number == 2:
        Question2()
    else:
        Question3()

if __name__ == '__main__':
    mainmenu()
```

The quit button becomes a brighter red when the mouse is hovering over it.

Technical Solution

Key Variables

Field name	Data type	Data length	Example	Description
difficulty	Integer	1	1	How many moves the computer calculates in advance. Options being either “1” or “2.”
spaceSize	Integer	3	50	Used to scale counters and board to the set resolution. Can be left at 50 for the resolution set.
fps	Integer	2	60	Frame rate that the program is run at.
windowWidth	Integer	4	1024	Width of the game window. In pixels.
windowHeight	Integer	3	576	Height of the game window. In pixels.
xMargin	Integer	3	337	The amount of pixels left at each size of the connect four board. Centres the board on the horizontally. Automatically adjusts depending on resolution set.
yMargin	Integer	3	138	The amount of pixels left at the top and bottom of the connect four board. Centres the board on the vertically. Automatically adjusts depending on resolution set.
bgColour	Tuple	12	(64, 224, 230)	Each number represents the RGB values to create a colour.
white	Tuple	13	(255, 255, 255)	Each number represents the RGB values to create a colour.
textColour	Tuple	13	white	Each number represents the RGB values to create a colour. Size dependant on colour used, for example if white is used size = 13.
red1	Tuple	9	(200, 0, 0)	Each number represents the RGB values to create a colour.
black	Tuple	7	(0,0,0)	Each number represents the RGB values to create a colour.
green	Tuple	9	(0,170,0)	Each number represents the RGB values to create a colour.
bright_red	Tuple	9	(255,0,0)	Each number represents the RGB values to create a colour.

bright_green	Tuple	9	(0,255,0)	Each number represents the RGB values to create a colour.
blue	Tuple	9	(0,0,255)	Each number represents the RGB values to create a colour.
largeText	Class	40	pygame.font.Font('freesansbold.ttf',115)	Used to automatically change a string to the font size “115,” and font type to “freesansbold.ttf”.
smallText	Class	39	pygame.font.Font('freesansbold.ttf',30)	Used to automatically change a string to the font size “30,” and font type to “freesansbold.ttf”.
keys	Class	24	pygame.key.get_pressed()	Class which gets the get pressed by the user.
red	String	5	"red"	Used in subroutines to find if a player has won the game.
black	String	7	"black"	Used in subroutines to find if a player has won the game.
empty	None	0	None	Used to find the lowest free space on a column.
player	String	8	"player"	Used in subroutines to find which players turn, player or computer.
computer	String	10	"Computer"	Used in subroutines to find which players turn, player or computer.
fpsClock	Class	20	pygame.time.Clock()	Used to track and control game framerate.
displaySurf	Class	52	pygame.display.set_mode((windowWidth, windowHeight))	Used to set the resolution of the game window.
redPileRect	Class	100	pygame.Rect(int(spaceSize/2), windowHeight - int(3 * spaceSize / 2), spaceSize, spaceSize)	Used to creates the range where the user can click to drag the token.
blackPileRect	class	100	pygame.Rect(windowWidth - int(3 * spaceSize / 2), windowHeight - int(3 * spaceSize / 2), spaceSize, spaceSize)	Used to creates the range where the user can click to drag the token.

redTokenIMG	Class	120	pygame.image.load('4row_red.png')
			pygame.transform.smoothscale(redTokenIMG, (spaceSize, spaceSize))
				Saves the red token image under the variable.
				Enables smooth scaling, to help stop image becoming pixelated as resolution increases

blackTokenIMG	Class	120	pygame.image.load('4row_black.png')
			pygame.transform.smoothscale(blackTokenIMG, (spaceSize, spaceSize))
				Saves the black token image under the variable.
				Enables smooth scaling, to help stop image becoming pixelated as resolution increases

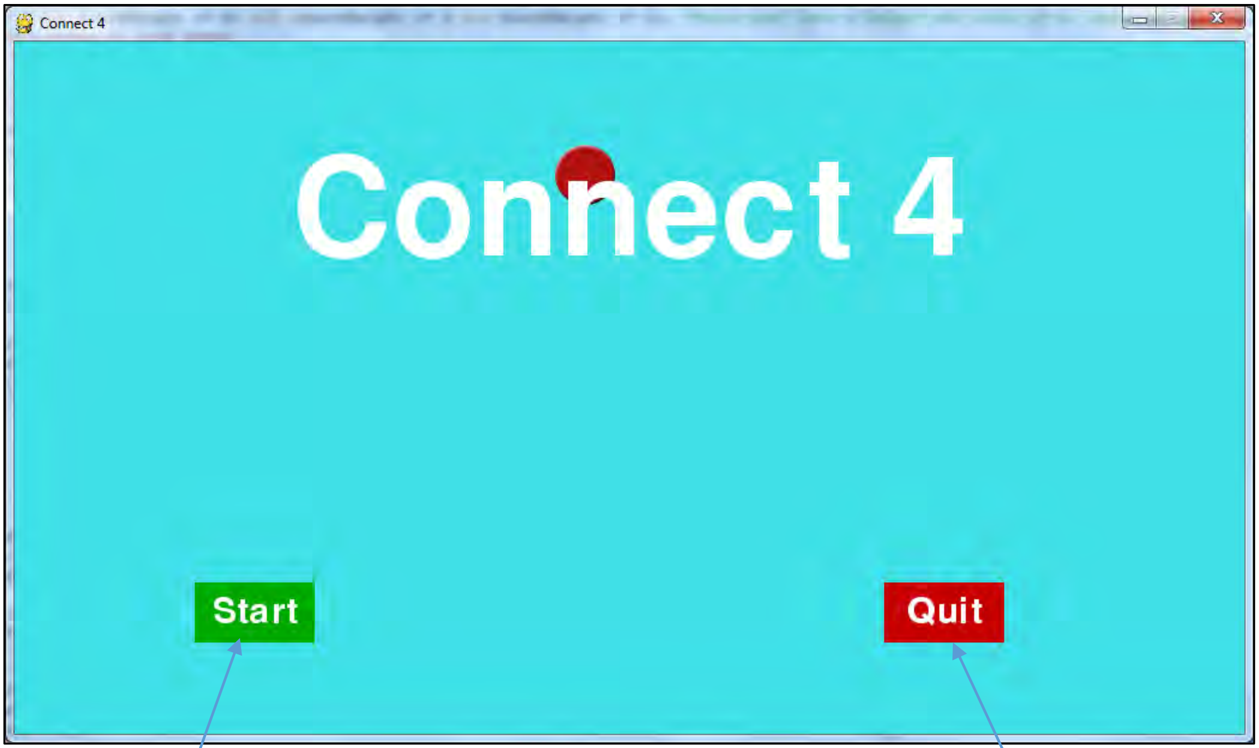
boardIMG	Class	120	pygame.image.load('4row_board.png')
			pygame.transform.smoothscale(boardIMG, (spaceSize, spaceSize))
				Saves the board image under the variable.
				Enables smooth scaling, to help stop image becoming pixelated as resolution increases

playerWinnerIMG	Class	40	pygame.image.load('4row_playerwinner.png')	Saves the player winning image to the variable.
computerWinnerIMG	Class	40	pygame.image.load('4row_computerwinner.png')	Saves the computer winning image to the variable.
tieWinnerIMG	Class	40	pygame.image.load('4row_tie.png')	Saves the tie image to the variable.
winnerRect	Class	40	playerWinnerIMG.get_rect()	Centre's the winning image.
arrowIMG	Class	40	pygame.image.load('4row_arrow.png')	Saves the arrow/help image to the variable.
arrowRect	Class	20	arrowIMG.get_rect()	Finds the range of the image.
SurfaceText	Class	20	font.render(text, True, white)	Sets the string and font.
firstGame	Boolean	5	True	Used to see if its the users first game.
mouse	Class	22	pygame.mouse.get_pos()	Used to store the mouse position.
lowest	Class	35	getLowestEmptySpace(board, column)	Used to find the lowest available space in each column.
spaceRect	Class	50	pygame.Rect(0, 0, spaceSize, spaceSize)	Used to scale the size of the board spaces, depending on the game resolution.
UserAnswer	String	11	"Incorrect"	Used to store if the user got the question correct.
X	Integer	4	xMargin + column * spaceSize	Used to align the token with the board column when animating dropping.
y	Integer	4	yMargin - spaceSize	Used to align the y axis when animating dropping.
dropSpeed	Float	3	0.5	Increased as the token falls, gives a gravity effect to the token dropping.
potentialMoves	Class	70	getPotentialMoves(board, black, difficulty)	Finds the moves available for the computer.
lookAhead	Integer	1	1	The amount of moves for the computer which are calculated in advanced.
dupeBoard	List	20	copy.deepcopy(board)	Saves a copy of the board to calculate the possible moves from a list.
dupeBoard	List	20	copy.deepcopy(dupeBoard)	Creates another copy if a move is not found first time.
A	Integer	2	random.randint(-20,20)	Random number.

B	Integer	2	random.randint(-20,20)	Random number.
C	Integer	2	random.randint(-20,20)	Random number.
D	Integer	2	random.randint(-20,20)	Random number.
E	Integer	2	random.randint(-20,20)	Random number.
T	“String”	3	“t”	Used to represent “t”.
X	Integer	2	random.randint(-20,20)	Random number.
F	Integer	2	random.randint(-20,20)	Random number.
G	Integer	2	random.randint(-20,20)	Random number.
H	Integer	2	random.randint(-20,20)	Random number.
I	Integer	2	random.randint(-20,20)	Random number.
Complex	Boolean	5	False	Used to see if a number is a complex or real number.
discriminant	Integer	1000	int(B)**2-4*int(A)*int(C)	Used to stop the program from attempting to square root a negative number.
sol1	Float	1000	(-int(B)+(int(B)**2-4*int(A)*int(C))**0.5)/(2*int(A))	Quadratic formula to calculate the first potential answers.
Sol2	Float	1000	(-int(B)-(int(B)**2-4*int(A)*int(C))**0.5)/(2*int(A))	Quadratic formula to calculate the first potential answers.
quadratic	String	41	str(A)+str(trig2)+str(B)+str(trig)+str(C)	Used to create a quadratic equation which can be printed onto the screen.
x1	Float	6	math.degrees(math.asin(sol1))	Used to find the answers for the question.
x2	Float	6	math.degrees(math.acos(sol1))	Used to find the answers for the question.
x2_2	Float	6	str(360+float(x2))	Used to find the answers for the question.
x1_2	Float	6	str(360-float(x1))	Used to find the answers for the question.
Solutions	Integer	1	4	The number of solutions to the question.
usersolutionsno	Integer	1	2	The number of solutions the user thinks there is.
attempts	Integer	1	5	The number of attempts the user has at a question.
equation	String	100	str(B)+"-"+"e"+"^("+str(E)+str(T)+"/"+str(D)+")"+" = "+str(A)	The equation which is printed to the screen.
menu	Boolean	5	True	Used to initiate the animation of the counters in the main menu.
redw1	Integer	4	-200	The position of the counter is changed each time the screen is

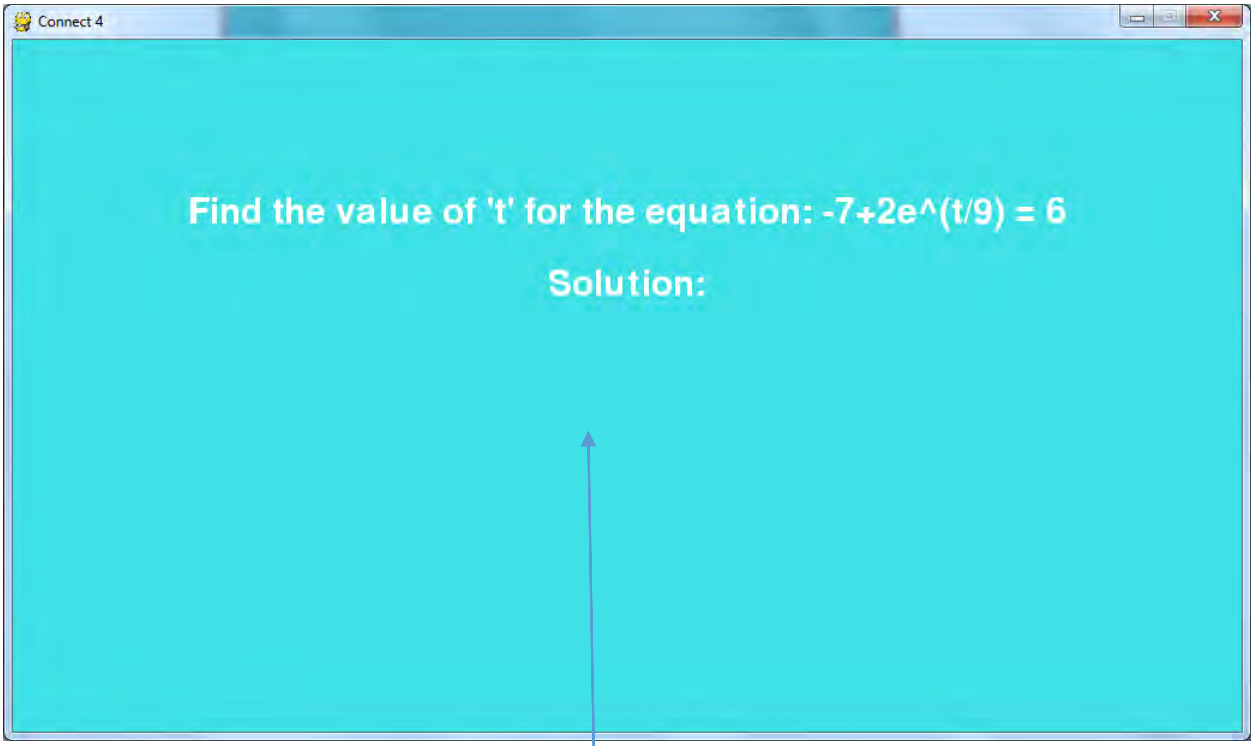
				updated, giving an animation of the counters moving.
redw2	Integer	4	-200	The position of the counter is changed each time the screen is updated, giving an animation of the counters moving.
redw3	Integer	4	-200	The position of the counter is changed each time the screen is updated, giving an animation of the counters moving.
blackw1	Integer	4	776	The position of the counter is changed each time the screen is updated, giving an animation of the counters moving.
blackw2	Integer	4	776	The position of the counter is changed each time the screen is updated, giving an animation of the counters moving.
blackw3	Integer	4	776	The position of the counter is changed each time the screen is updated, giving an animation of the counters moving.
number	Integer	12	random.randint(1,3)	A random number between one and three, used to decide the type of question asked to the user.

Overview of Interface



The users will start the game by clicking here

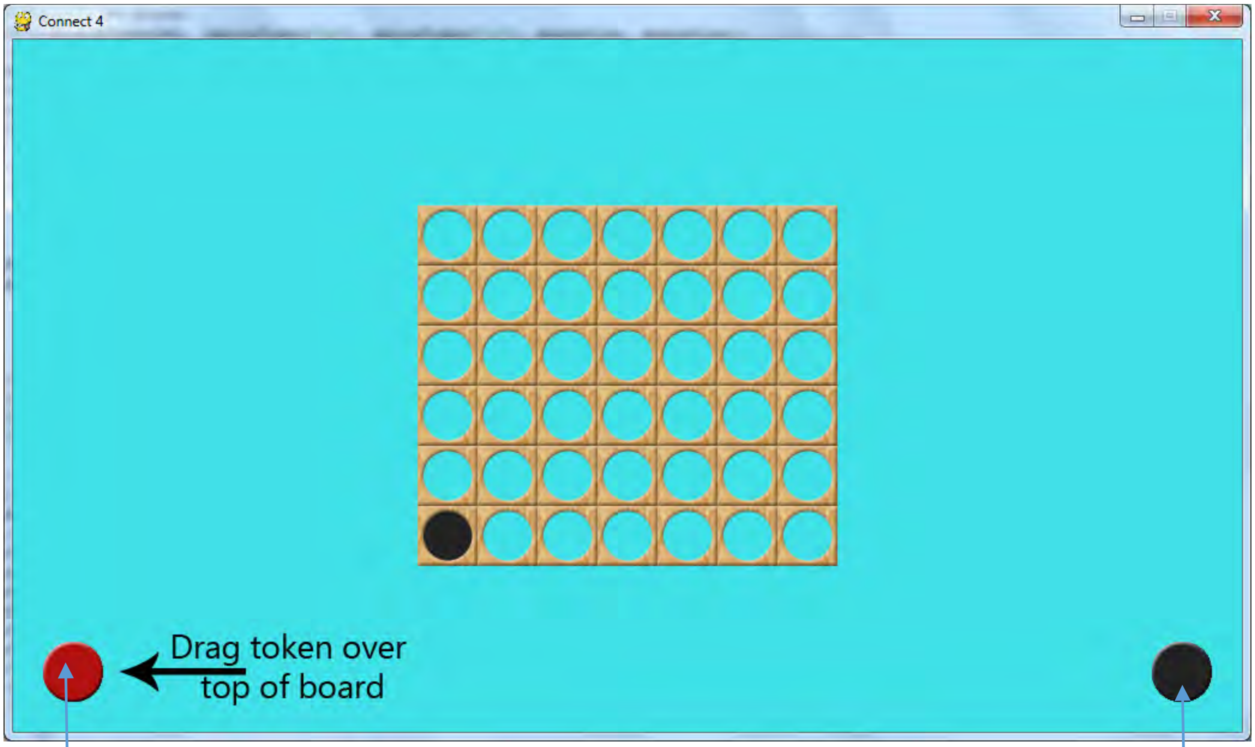
The users will quit the game by clicking here



Find the value of 't' for the equation: -7+2e^(t/9) = 6

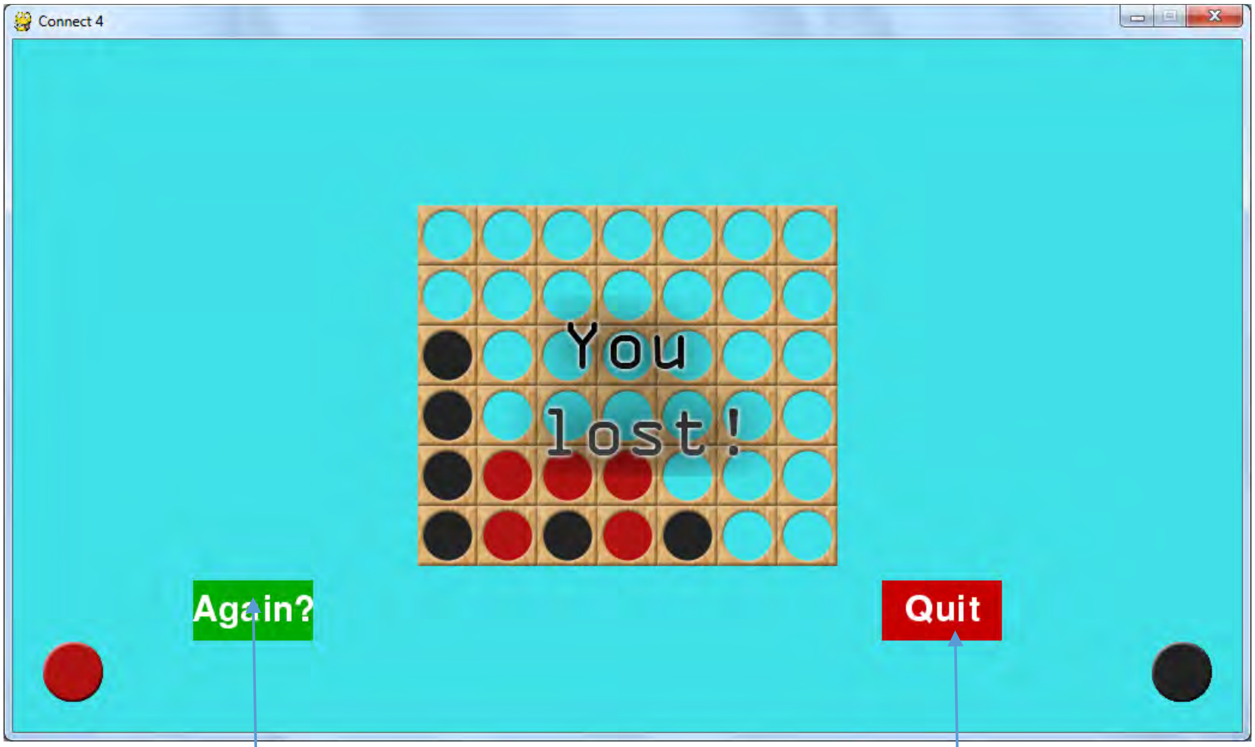
Solution:

When the user types their answer it will appear here



The user will drag a token from here.

The computer will drag a token from here.



The game will begin again if the user click here

The game will close if the user click here

This is the same for the winning game

Program code

```
import random, copy, sys, pygame, math
from pygame.locals import *

pygame.init()

boardWidth = 7 #how wide the board is
boardHeight = 6 # how tall the board is

assert (boardWidth >= 4 and boardHeight <= 9) and (boardHeight >= 4 and boardHeight <= 9), "Board
must have a height and width of at least 4."
difficulty = 1# how many moves to look ahead
spaceSize = 50# size of the tokens and individual board spaces in pixels
fps = 30# frames per second to update the screen
windowWidth = 1024# width of the program's window, in pixels
windowHeight = 576 # height in pixels
xMargin = int((windowWidth - boardWidth * spaceSize) / 2)
yMargin = int((windowHeight - boardHeight * spaceSize) / 2)
bgColour = (64, 224, 230)
white = (255, 255, 255)
textColour = white
red1 = (200,0,0)
black = (0,0,0)
green = (0,170,0)
bright_red = (255,0,0)
bright_green = (0,255,0)
blue = (0,0,255)
largeText = pygame.font.Font('freesansbold.ttf',115)
smallText = pygame.font.Font('freesansbold.ttf',30)
keys = pygame.key.get_pressed()

red = "red"
black = "black"
empty = None
player = "player"
computer = "Computer"

fpsClock = pygame.time.Clock()
displaySurf = pygame.display.set_mode((windowWidth, windowHeight))
pygame.display.set_caption("Connect 4")

redPileRect = pygame.Rect(int(spaceSize/2), windowHeight - int(3 * spaceSize / 2), spaceSize,
spaceSize)
blackPileRect = pygame.Rect(windowWidth - int(3 * spaceSize / 2), windowHeight - int(3 * spaceSize /
2), spaceSize, spaceSize)
redTokenIMG = pygame.image.load('4row_red.png')
redTokenIMG = pygame.transform.smoothscale(redTokenIMG, (spaceSize, spaceSize))
blackTokenIMG = pygame.image.load('4row_black.png')
blackTokenIMG = pygame.transform.smoothscale(blackTokenIMG, (spaceSize, spaceSize))
boardIMG = pygame.image.load('4row_board.png')
boardIMG = pygame.transform.smoothscale(boardIMG, (spaceSize, spaceSize))

playerWinnerIMG = pygame.image.load('4row_playerwinner.png')
computerWinnerIMG = pygame.image.load('4row_computerwinner.png')
tieWinnerIMG = pygame.image.load('4row_tie.png')
winnerRect = playerWinnerIMG.get_rect()
winnerRect.center = (int(windowWidth / 2), int(windowHeight / 2))

arrowIMG = pygame.image.load('4row_arrow.png')
arrowRect = arrowIMG.get_rect()
arrowRect.left = redPileRect.right + 10
arrowRect.centery = redPileRect.centery



def text_objects(text, font): # used to create text to display on the screen
    SurfaceText = font.render(text, True, white)
    return SurfaceText, SurfaceText.get_rect()

def main():
    firstGame = True

    while True:
        runGame(firstGame)
        firstGame = False

def runGame(firstGame):
    if firstGame: # The computer has the first move, to shpw the player what to do.
        turn = computer
        showHelp = True
    else:
        if random.randint(0, 1) == 0:
            turn = computer
        else: # randomly chooses who goes first
            turn = player
        showHelp = False

    mainBoard = getNewBoard() #get a new, blank game board.

    while True: #main game loop
        if turn == player: # players turn
            getPlayerMove(mainBoard, showHelp)
            if showHelp:
                showHelp = False # turn off help arrow after the first move.
            if isWinner(mainBoard, red):
                winnerIMG = playerWinnerIMG
                break
            turn = computer # switch to other players turn.
        else:# computers turn.
            column = getComputerMove(mainBoard)
            animateComputerMoving(mainBoard, column)
            makeMove(mainBoard, black, column)
            if isWinner(mainBoard, black):
                winnerIMG = computerWinnerIMG
                break
            turn = player #switch players turn
        if isBoardFull(mainBoard): # tie function.
            winnerIMG = tieWinnerIMG
            break
    while True: #loop until the player clicks a button.
        drawBoard(mainBoard)
        displaySurf.blit(winnerIMG, winnerRect)
        mouse = pygame.mouse.get_pos()
        if 150 + 100 > mouse[0] > 150 and 450 + 50 > mouse[1] >450:
            pygame.draw.rect(displaySurf, bright_green, (150,450,100,50))
        else:
            pygame.draw.rect(displaySurf, green, (150,450,100,50))
        if 724 + 100 > mouse[0] > 724 and 450 + 50 > mouse[1] > 450:
            pygame.draw.rect(displaySurf, bright_red, (724,450,100,50))
        else:
            pygame.draw.rect(displaySurf, red1, (724,450,100,50))
        textSurf, textRect = text_objects("Again?",smallText)
        textRect.center = ((150 + (100/2)), (450 + (50/2)))
        textSurf1, textRect1 = text_objects("Quit",smallText)
        textRect1.center = ((724 + (100/2)), (450 + (50/2)))
        displaySurf.blit(textSurf, textRect)
        displaySurf.blit(textSurf1, textRect1)
        for event in pygame.event.get(): # event handling
            if event.type == QUIT:
                pygame.quit()
                quit()
            if event.type == MOUSEBUTTONDOWN:
                if 150 + 100 > mouse[0] > 150 and 450 + 50 > mouse[1] >450:
                    firstGame = False
                    runGame(firstGame)
                if 724 + 100 > mouse[0] > 724 and 450 + 50 > mouse[1] > 450:
                    pygame.quit()
                    quit()
        pygame.display.update()
        fpsClock.tick()

def makeMove(board, player, column): #making a move function
```

```
    lowest = getLowestEmptySpace (board, column)
    if lowest != -1:
        board[column][lowest] = player

def drawBoard(board, extraToken = None):
    displaySurf.fill(bgColour)
    #display tokens
    spaceRect = pygame.Rect(0, 0, spaceSize, spaceSize)
    for x in range(boardWidth):
        for y in range(boardHeight):
            spaceRect.topleft = (xMargin + (x * spaceSize), yMargin + (y * spaceSize))
            if board[x][y] == red:
                displaySurf.blit(redTokenIMG, spaceRect)
            elif board[x][y] == black:
                displaySurf.blit(blackTokenIMG, spaceRect)
    #display an extra token
    if extraToken != None:
        if extraToken["colour"] == red:
            displaySurf.blit(redTokenIMG, (extraToken["x"], extraToken["y"], spaceSize, spaceSize))
        elif extraToken["colour"] == black:
            displaySurf.blit(blackTokenIMG, (extraToken["x"], extraToken["y"], spaceSize, spaceSize))
    #display board over tokens
    for x in range(boardWidth):
        for y in range(boardHeight):
            spaceRect.topleft = (xMargin + (x * spaceSize), yMargin + (y * spaceSize))
            displaySurf.blit(boardIMG, spaceRect)

        displaySurf.blit(redTokenIMG, redPileRect)#puts red tokens on left side
        displaySurf.blit(blackTokenIMG, blackPileRect)#puts black tokens on right side

def getNewBoard():
    board = []
    for x in range(boardWidth):
        board.append([empty] * boardHeight)
    return board

def getPlayerMove(board, isFirstMove):
    getQuestionType()
    draggingToken = False
    tokenX, tokenY = None, None
    if UserAnswer == "Correct": # has the user got the question right
        while True:
            for event in pygame.event.get(): # event handling
                if event.type == QUIT:
                    pygame.quit()
                    quit()
                elif event.type == MOUSEBUTTONDOWN and not draggingToken and
redPileRect.collidepoint(event.pos):
```

```
                    draggingToken = True # start of dragging on red token pile.
                    tokenX, tokenY = event.pos
                elif event.type == MOUSEMOTION and draggingToken:
                    tokenX, tokenY = event.pos # update the position of the red token being dragged
                elif event.type == MOUSEBUTTONUP and draggingToken:# let go of the token being dragged
                    if tokenY < yMargin and tokenX > xMargin and tokenX < windowWidth - xMargin: #let go of token at top of the screen
                        column = int((tokenX - xMargin) / spaceSize)
                        if isValidMove(board, column):
                            animateDroppingToken(board, column, red)
                            board[column][getLowestEmptySpace(board, column)] = red
                            drawBoard(board)
                            pygame.display.update()
                            return
                    tokenX, tokenY = None, None
                    draggingToken = False
            if tokenX != None and tokenY != None:
                drawBoard(board, {'x':tokenX - int(spaceSize / 2), 'y':tokenY - int(spaceSize / 2), 'colour':red})
            else:
                drawBoard(board)

            if isFirstMove: # help arrow
                displaySurf.blit(arrowIMG, arrowRect)

            pygame.display.update()
            fpsClock.tick()
    if UserAnswer == "Incorrect": # no turn if question was incorrectly answered
        drawBoard(board)
        pygame.display.update()
        fpsClock.tick()

def animateDroppingToken(board, column, colour):
    x = xMargin + column * spaceSize
    y = yMargin - spaceSize
    dropSpeed = 1.0

    lowestEmptySpace = getLowestEmptySpace(board, column)

    while True:
        y += int(dropSpeed)
        dropSpeed += 0.5
        if int((y - yMargin) / spaceSize) >= lowestEmptySpace:
            return
        drawBoard(board, {'x':x, 'y':y, 'colour':colour})
        pygame.display.update()
        fpsClock.tick()

def animateComputerMoving(board, column):
    x = blackPileRect.left
    y = blackPileRect.top
    speed = 1.0
    #moving the black token up
    while y > (yMargin - spaceSize):
        y -= int(speed)
        speed += 0.5
        drawBoard(board, {'x':x, 'y':y, 'colour':black})
        pygame.display.update()
        fpsClock.tick()
    #moving the black token over
    y = yMargin - spaceSize
    speed = 1.0
    while x > (xMargin + column * spaceSize):
        x -= int(speed)
        speed += 0.5
        drawBoard(board, {'x':x, 'y':y, 'colour':black})
        pygame.display.update()
        fpsClock.tick()
    #dropping the black token
    animateDroppingToken(board, column, black)

def getComputerMove(board):
    potentialMoves = getPotentialMoves(board, black, difficulty)
    bestMoveFitness = -1
    #find the best potential move
    for a in range(boardWidth):
        if potentialMoves[a] > bestMoveFitness and isValidMove(board, a):
            bestMoveFitness = potentialMoves[a]
    # finds all the potential moves
    bestMoves = []
    for a in range(len(potentialMoves)):
        if potentialMoves[a] == bestMoveFitness and isValidMove(board, a):
            bestMoves.append(a)
    return random.choice(bestMoves)

def getPotentialMoves(board, tile, lookAhead):
    if lookAhead == 0 or isBoardFull(board):
        return [0] * boardWidth

    if tile == red:
        enemyTile = black
    else:
        enemyTile = red
    #find best move to take
    potentialMoves = [0] * boardWidth
    for firstMove in range(boardWidth):
        dupeBoard = copy.deepcopy(board)
        if not isValidMove(dupeBoard, firstMove):
            continue
        makeMove(dupeBoard, tile, firstMove)
        if isWinner(dupeBoard, tile):
            #winning move will always be the best move
            potentialMoves[firstMove] = 1
            break # doesnt calculate more moves when the winning move is calculated
        else:
            if isBoardFull(dupeBoard):
                potentialMoves[firstMove] = 0
            else: # do other players counter moves
                for counterMove in range(boardWidth):
                    dupeBoard2 = copy.deepcopy(dupeBoard)
                    if not isValidMove(dupeBoard2, counterMove):
                        continue
                    makeMove(dupeBoard2, enemyTile, counterMove)
                    if isWinner(dupeBoard2, enemyTile):
                        # a losing move automatically gets lowest priority
                        potentialMoves[firstMove] = -1
                        break
                    else: # recursive call to get more moves.
                        results = getPotentialMoves(dupeBoard2, tile, lookAhead - 1)
                        potentialMoves[firstMove] += (sum(results) / boardWidth) / boardWidth
    return potentialMoves

def getLowestEmptySpace(board, column):
    # return row number of lowest space in a column
    for y in range(boardHeight-1, -1, -1):
        if board[column][y] == empty:
            return y
    return -1

def isValidMove(board, column):
    #returns true if there is an empty space in a column.
    if column < 0 or column >= (boardWidth) or board[column][0] != empty:
        return False
    return True

def isBoardFull(board):
    # finds if the board is full.
    for x in range(boardWidth):
        for y in range(boardHeight):
            if board[x][y] == empty:
                return False
    return True

def isWinner(board, tile):
    #check horizontal spaces
    for x in range(boardWidth - 3):
        for y in range(boardHeight):
            if board[x][y] == tile and board[x+1][y] == tile and board[x+2][y] == tile and board[x+3][y] == tile:
                return True
    #check veritcal spaces
    for x in range(boardWidth):
        for y in range(boardHeight - 3):
            if board[x][y] == tile and board[x][y+1] == tile and board[x][y+2] == tile and board[x][y+3] == tile:
                return True
    #check diagonal spaces
    for x in range(boardWidth - 3):
        for y in range(3, boardHeight):
            if board[x][y] == tile and board[x+1][y-1] == tile and board[x+2][y-2] == tile and board[x+3][y-3]
== tile:
                return True
    #check diagonal spaces
    for x in range(boardWidth - 3):
        for y in range(boardHeight - 3):
            if board[x][y] == tile and board[x+1][y+1] == tile and board[x+2][y+2] == tile and board[x+3][y+3]
== tile:
                return True
    return False

def menuCounters(redw1, redw2, redw3, blackw1, blackw2, blackw3):

    runs = 0
    while runs < 4: # animates tokens moving to find initial starting positions
        if redw1 > 700 and redw2 > 700 and redw3 > 700:
                redw1 = -200
                redw2 = -200
                redw3 = -200
        else:

            if redw1 < 710:
                redw1 += 2
            if redw1 > 700:

                redw2 += 2
            if redw1 > 700 and redw2 > 700:

                redw3 += 2
        if blackw1 < -200 and blackw2 < -200 and blackw3 < -200:
            blackw1 = 776
            blackw2 = 776
            blackw3 = 776
            blackw4 = 776
        else:
            if blackw1 > -210:
                blackw1 += -2
            if blackw1 < -200:

                blackw2 += -2
            if blackw1 < -200 and blackw2 < -200:

                blackw3 += -2
        runs = runs + 1
        return redw1, redw2, redw3, blackw1, blackw2, blackw3

def Question1():
    global UserAnswer
    UserAnswer = "Incorrect"
    x1 = "No solution"
    x2 = "No solution"
    x1_2 = "No solution"
    x2_2 = "No solution"

    while x1 == "No solution" and x2 == "No solution" and x1_2 == "No solution" and x2_2 == "No
solution":
        ###### CREATE EQUATION #####
        A = random.randint(-20,20)
        B = random.randint(-20,20)
        C = random.randint(-20,20)
        discriminant = (int(B)**2-4*int(A)*int(C))
        Complex = True
        while Complex == True:
            while discriminant < 0 or A == 0 or B == 0:
                A = random.randint(-20,20)
                B = random.randint(-20,20)
                C = random.randint(-20,20)
                discriminant = (int(B)**2-4*int(A)*int(C))
            sol1 = (-int(B)+(int(B)**2-4*int(A)*int(C))**0.5)/(2*int(A))
            sol2 = (-int(B)-(int(B)**2-4*int(A)*int(C))**0.5)/(2*int(A))
            if type(sol1) == complex or type(sol2) == complex:
                Complex = True
            else:
                Complex = False
        if B >= 0:
            B = "+",str(B)
            B = str(B)[2] + str(B)[7]
        if C >= 0:
            C = "+",str(C)
            C = str(C)[2] + str(C)[7]
        num = random.randint(1,3)
        if num == 1:
            trig = "Sin(x)"
            trig2 = "Sin^2(x)"
        elif num == 2:
            trig = "Cos(x)"
            trig2 = "Cos^2(x)"
        elif num == 3:
            trig = "Tan(x)"
            trig2 = "Tan^2(x)"
        quadratic = str(A)+str(trig2)+str(B)+str(trig)+str(C)
        if B == -1:
            quadratic = str(A)+str(trig2)+"-"+str(trig)+str(C)
        if A == -1:
            quadratic = "-"+str(trig2)+str(B)+str(trig)+str(C)
        if A == 1 and B != 1:
            quadratic = str(trig2)+str(B)+str(trig)+str(C)
        if B == "+1" and A != 1:
            quadratic = str(A)+str(trig2)+"+"+str(trig)+str(C)
        if A == 1 and B == "+1":
            quadratic = str(trig2)+"+"+str(trig)+str(C)
        if B == -1 and A == -1:
            quadratic = "-"+str(trig2)+"-"+str(trig)+str(C)


        ##### INVERSE FUNCTION #####

        if num == 1:
            if -1 <= sol1 <= 1:
                x1 = math.degrees(math.asin(sol1))
            else:
                x1 = "No solution"
            if -1 <= sol2 <= 1:
                x2 = math.degrees(math.asin(sol2))
            else:
                x2 = "No solution"
        elif num == 2:
            if -1 <= sol1 <= 1:
                x1 = math.degrees(math.acos(sol1))
            else:
                x1 = "No solution"
            if -1 <= sol2 <= 1:
                x2 = math.degrees(math.acos(sol2))
            else:
                x2 = "No solution"
        elif num == 3:
            x1 = math.degrees(math.atan(sol1))
            x2 = math.degrees(math.atan(sol2))

        ##### SOLUTIONS IN GIVEN RANGE #####
```

```
## SIN##

if num == 1:
    if x1 != "No solution":
        if float(x1) < 180 and float(x1) > 0 and float(x1) != 90:
            x1_2 = str(180 - float(x1))
        elif float(x1) > 180 and float(x1) < 360 and float(x1) != 270:
            x1_2 = str(360-float(x1))
        elif float(x1) > -180 and float(x1) < 0 and float(x1) != -90:
            x1_2 = str(360-(float(x1)*-1))
            x1 = str(180 + (float(x1)*-1))
        elif float(x1) < -180 and float(x1) > -360 and float(x1) != -270:
            x1 = str((float(x1)*-1) -180)
            x1_2 = str(180 - float(x1))
    if x2 != "No solution":
        if float(x2) < 180 and float(x2) > 0 and float(x2) != 90:
            x2_2 = str(180 - float(x2))
        elif float(x2) > 180 and float(x2) < 360 and float(x2) != 270:
            x2_2 = str(360-float(x2))
        elif float(x2) > -180 and float(x2) < 0 and float(x2) != -90:
            x2_2 = str(360+float(x2))
            x2 = str((float(x2)*-2) + 180)
        elif float(x2) < -180 and float(x2) > -360 and float(x2) != -270:
            x2 = str((float(x2)*-1) -180)
            x2_2 = str(180 - float(x2))

## COS ##

elif num == 2:
    if x1 != "No solution":
        if float(x1) < 180 and float(x1) > 0:
            x1_2 = 360 - float(x1)
        elif float(x1) > 180 and float(x1) < 360:
            x1_2 = 360 - float(x1)
        elif float(x1) > -180 and float(x1) < 0:
            x1 = 0-float(x1)
            x1_2 = 360 - float(x1)
        elif float(x1) < -180 and float(x1) > -360:
            x1_2 = float(str(x1[1:-1]))
            x1 = 360 - float(x1)
    if x2 != "No solution":
        if float(x2) < 180 and float(x2) > 0:
            x2_2 = 360 - float(x2)
        elif float(x2) > 180 and float(x2) < 360:
            x2_2 = 360 - float(x2)
        elif float(x2) > -180 and float(x2) < 0:
            x2 = 0-float(x2)
            x2_2 = 360 - float(x2)
        elif float(x2) < -180 and float(x2) > -360:
            x2_2 = float(str(x2[1:-1]))
            x2 = 360 - float(x2)
        elif float(x2) == 0.00:
            x2_2 = 360
        elif float(x2) == -180:
            x2 = 180

## TAN ##

else:
    if x1 != "No solution":
        if float(x1) >= 0 and float(x1) <= 90:
            x1_2 = float(x1) + 180
        if float(x1) >= -180 and float(x1) < -90:
            x1 = float(x1) + 180
            x1_2 = float(x1) + 180
        if float(x1) >= -360 and float(x1) <= -270:
            x1 = float(x1) + 360
            x1_2 = float(x1) + 180
        if float(x1) <= -180 and float(x1) >=-270:
            x1 = float(x1) + 360
            x1_2 = float(x1) + 180
        if float(x1) <= 0 and float(x1) >= -90:
            x1 = float(x1) + 180
            x1_2 = float(x1) + 180
        if float(x1) <= 180 and float(x1) >= 90:
            x1_2 = float(x1) + 180
    if x2 != "No solution":
        if float(x2) >= 0 and float(x2) <= 90:
            x2_2 = float(x2) + 180
        if float(x2) >= -180 and float(x2) < -90:
            x2 = float(x2) + 180
            x2_2 = float(x2) + 180
        if float(x2) >= -360 and float(x2) <= -270:
            x2 = float(x2) + 360
            x2_2 = float(x2) + 180
        if float(x2) <= -180 and float(x2) >=-270:
            x2 = float(x2) + 360
            x2_2 = float(x2) + 180
        if float(x2) <= 0 and float(x2) >= -90:
            x2 = float(x2) + 180
            x2_2 = float(x2) + 180
        if float(x2) <= 180 and float(x2) >= 90:
            x2_2 = float(x2) + 180

##### FORMATTING SOLUTIONS #####
```

```
    if x1 != "No solution":
        x1 = format(float(x1),'.2f')
    if x2 != "No solution":
        x2 = format(float(x2),'.2f')
    if x1_2 != "No solution":
        x1_2 = format(float(x1_2),'.2f')
    if x2_2 != "No solution":
        x2_2 = format(float(x2_2),'.2f')
    if str(x1[0]) == "-" and str(x1[1]) == "0":
        x1 = format(float(x1[1:-1]),'.2f')
    if str(x2[0]) == "-" and str(x2[1]) == "0":
        x2 = format(float(x2[1:-1]),'.2f')
    if str(x1_2[0]) == "-" and str(x1_2[1]) == "0":
        x1 = format(float(x1_2[1:-1]),'.2f')
    if str(x2_2[0]) == "-" and str(x2_2[1]) == "0":
        x1 = format(float(x1_2[1:-1]),'.2f')

    if str(x1[-3] + x1[-2] + x1[-1]) == ".00":
        x1 = int(float(x1))
    if str(x1_2[-3] + x1_2[-2] + x1_2[-1]) == ".00":
        x1_2 = int(float(x1_2))
    if str(x2[-3] + x2[-2] + x2[-1]) == ".00":
        x2 = int(float(x2))
    if str(x2_2[-3] + x2_2[-2] + x2_2[-1]) == ".00":
        x2_2 = int(float(x2_2))

question = str("Find x for "+ quadratic+ " = 0. In the range 0 <= x <= 360.")
textSurf3, textRect3 = text_objects(question,smallText)
textRect3.center = ((windowWidth/2),(windowHeight/4))
displaySurf.fill(bgColour)
displaySurf.blit(textSurf3, textRect3)
pygame.display.update()
solutions = 0
#finds number of solutions
if x1 != "No solution":
    solutions = solutions + 1
if x1_2 != "No solution":
    solutions = solutions + 1
if x2 != "No solution":
    solutions = solutions + 1
if x2_2 != "No solution":
    solutions = solutions + 1
usersolutionsno = ""
textSurf5, textRect5 = text_objects("Number of solutions: ",smallText)
textRect5.center = ((windowWidth/2), ((windowHeight/4)+60))
displaySurf.blit(textSurf5, textRect5)
pygame.display.update()
while 1: # gets text from the user and displays on screen (repeated throughout program).
```

```
    event = pygame.event.poll()
    if event.type == KEYDOWN:
      if event.unicode == '\r': break
      usersolutionsno += event.unicode
      if event.key == K_BACKSPACE:
        usersolutionsno = usersolutionsno[0:-2]
      textSurf4, textRect4 = text_objects(usersolutionsno,smallText)
      textRect4.center = ((windowWidth/2), (windowHeight/2))
      displaySurf.fill(bgColour)
      displaySurf.blit(textSurf3, textRect3)
      displaySurf.blit(textSurf4, textRect4)
      displaySurf.blit(textSurf5, textRect5)
      pygame.display.update()
  attempts  = 5
  while (usersolutionsno != str(solutions)) and attempts > 1: # user has number of solutions wrong.
    attempts = attempts - 1
    usersolutionsno = ""
    textSurf4, textRect4 = text_objects(("Incorrect. "+str(attempts)+" attempts remaining, type to try
again."),smallText)
    textRect4.center = ((windowWidth/2), (windowHeight/2))
    displaySurf.fill(bgColour)
    displaySurf.blit(textSurf3, textRect3)
    displaySurf.blit(textSurf4, textRect4)
    displaySurf.blit(textSurf5, textRect5)
    pygame.display.update()
    while 1:
      event = pygame.event.poll()
      if event.type == KEYDOWN:
        if event.unicode == '\r': break
        usersolutionsno += event.unicode
        if event.key == K_BACKSPACE:
          usersolutionsno = usersolutionsno[0:-2]
        textSurf4, textRect4 = text_objects(usersolutionsno,smallText)
        textRect4.center = ((windowWidth/2), (windowHeight/2))
        displaySurf.fill(bgColour)
        displaySurf.blit(textSurf3, textRect3)
        displaySurf.blit(textSurf4, textRect4)
        displaySurf.blit(textSurf5, textRect5)
        pygame.display.update()

  if usersolutionsno != str(solutions):
    UserAnswer = "Incorrect"
    textSurf4, textRect4 = text_objects(UserAnswer,smallText)
    textRect4.center = ((windowWidth/2), (windowHeight/2))
    displaySurf.fill(bgColour)
    displaySurf.blit(textSurf3, textRect3)
    displaySurf.blit(textSurf4, textRect4)
    displaySurf.blit(textSurf5, textRect5)
```

```
        pygame.display.update()
        pygame.time.wait(2000)
    else:
        displaySurf.fill(bgColour)
        displaySurf.blit(textSurf3, textRect3)
        textSurf6, textRect6 = text_objects("Correct number of solutions!",smallText)
        textRect6.center = ((windowWidth/2), ((windowHeight/4)+60))
        displaySurf.blit(textSurf6, textRect6)
        pygame.display.update()
        pygame.time.wait(2000)
    #resets given solutions from previous questions.
    usersolution1 = ""
    usersolution2 = ""
    usersolution3 = ""
    usersolution4 = ""
    Correct = False
    ##### Getting solutions from the user #######
   ## One solution ##

    if solutions == 1 and usersolutionsno == str(solutions):
        if x1 != "No solution":
            textSurf7, textRect7 = text_objects("Solution: ",smallText)
            textRect7.center = ((windowWidth/2), ((windowHeight/4)+60))
            displaySurf.fill(bgColour)
            displaySurf.blit(textSurf3, textRect3)
            displaySurf.blit(textSurf7, textRect7)
            pygame.display.update()
            while 1:
                event = pygame.event.poll()
                if event.type == KEYDOWN:
                    if event.unicode == '\r': break
                    usersolution1 += event.unicode
                    if event.key == K_BACKSPACE:
                        usersolution1 = usersolution1[0:-2]
                    textSurf8, textRect8 = text_objects(usersolution1,smallText)
                    textRect8.center = ((windowWidth/2), (windowHeight/2))
                    displaySurf.fill(bgColour)
                    displaySurf.blit(textSurf3, textRect3)
                    displaySurf.blit(textSurf7, textRect7)
                    displaySurf.blit(textSurf8, textRect8)
                    pygame.display.update()
            while usersolution1 != str(x1) and attempts > 1:# answers were incorrect
                attempts = attempts - 1

                textSurf7, textRect7 = text_objects("Incorrect solutions, "+str(attempts)+" attempts
remaining. Solution: ",smallText)
                textRect7.center = ((windowWidth/2), ((windowHeight/4)+60))
                displaySurf.fill(bgColour)
        displaySurf.blit(textSurf3, textRect3)
        displaySurf.blit(textSurf7, textRect7)
        pygame.display.update()
        while 1:
          event = pygame.event.poll()
          if event.type == KEYDOWN:
            if event.unicode == '\r': break
            usersolution1 += event.unicode
            if event.key == K_BACKSPACE:
              usersolution1 = usersolution1[0:-2]
            textSurf8, textRect8 = text_objects(usersolution1,smallText)
            textRect8.center = ((windowWidth/2), (windowHeight/2))
            displaySurf.fill(bgColour)
            displaySurf.blit(textSurf3, textRect3)
            displaySurf.blit(textSurf7, textRect7)
            displaySurf.blit(textSurf8, textRect8)
            pygame.display.update()
      if x1 != "No solution" and usersolution1 == str(x1):
        UserAnswer = "Correct"
      else:
        UserAnswer = "Incorrect"
    if x2 != "No solution":
      textSurf10, textRect10 = text_objects("Solution: ",smallText)
      textRect10.center = ((windowWidth/2), ((windowHeight/4)+60))
      displaySurf.fill(bgColour)
      displaySurf.blit(textSurf3, textRect3)
      displaySurf.blit(textSurf10, textRect10)
      pygame.display.update()
      while 1:
        event = pygame.event.poll()
        if event.type == KEYDOWN:
          if event.unicode == '\r': break
          usersolution3 += event.unicode
          if event.key == K_BACKSPACE:
            usersolution3 = usersolution3[0:-2]
          textSurf8, textRect8 = text_objects(usersolution3,smallText)
          textRect8.center = ((windowWidth/2), (windowHeight/2))
          displaySurf.fill(bgColour)
          displaySurf.blit(textSurf3, textRect3)
          displaySurf.blit(textSurf10, textRect10)
          displaySurf.blit(textSurf8, textRect8)
          pygame.display.update()
      while usersolution3 != str(x2) and attempts > 1:# answers were incorrect
        attempts = attempts - 1
        textSurf10, textRect10 = text_objects("Incorrect solutions, "+str(attempts)+" attempts
remaining. Solution: ",smallText)
        textRect10.center = ((windowWidth/2), ((windowHeight/4)+60))
        displaySurf.fill(bgColour)
```

```
                displaySurf.blit(textSurf3, textRect3)
                displaySurf.blit(textSurf10, textRect10)
                pygame.display.update()
                while 1:
                    event = pygame.event.poll()
                    if event.type == KEYDOWN:
                        if event.unicode == '\r': break
                        usersolution3 += event.unicode
                        if event.key == K_BACKSPACE:
                            usersolution3 = usersolution3[0:-2]
                        textSurf8, textRect8 = text_objects(usersolution3,smallText)
                        textRect8.center = ((windowWidth/2), (windowHeight/2))
                        displaySurf.fill(bgColour)
                        displaySurf.blit(textSurf3, textRect3)
                        displaySurf.blit(textSurf10, textRect10)
                        displaySurf.blit(textSurf8, textRect8)
                        pygame.display.update()
            if x2 != "No solution" and usersolution3 == str(x2):
                UserAnswer = "Correct"
            else:
                UserAnswer = "Incorrect"
        if x1_2 != "No solution":
            textSurf9, textRect9 = text_objects("Solution: ",smallText)
            textRect9.center = ((windowWidth/2), ((windowHeight/4)+60))
            displaySurf.fill(bgColour)
            displaySurf.blit(textSurf3, textRect3)
            displaySurf.blit(textSurf9, textRect9)
            pygame.display.update()
            while 1:
                event = pygame.event.poll()
                if event.type == KEYDOWN:
                    if event.unicode == '\r': break
                    usersolution2 += event.unicode
                    if event.key == K_BACKSPACE:
                        usersolution2 = usersolution2[0:-2]
                    textSurf8, textRect8 = text_objects(usersolution2,smallText)
                    textRect8.center = ((windowWidth/2), (windowHeight/2))
                    displaySurf.fill(bgColour)
                    displaySurf.blit(textSurf3, textRect3)
                    displaySurf.blit(textSurf9, textRect9)
                    displaySurf.blit(textSurf8, textRect8)
                    pygame.display.update()
            while usersolution2 != str(x1_2) and attempts > 1:# answers were incorrect
                attempts = attempts - 1
                textSurf9, textRect9 = text_objects("Incorrect solutions, "+str(attempts)+" attempts
remaining. Solution: ",smallText)
                textRect9.center = ((windowWidth/2), ((windowHeight/4)+60))
                displaySurf.fill(bgColour)
                displaySurf.blit(textSurf3, textRect3)
                displaySurf.blit(textSurf9, textRect9)
                pygame.display.update()
                while 1:
                    event = pygame.event.poll()
                    if event.type == KEYDOWN:
                        if event.unicode == '\r': break
                        usersolution2 += event.unicode
                        if event.key == K_BACKSPACE:
                            usersolution2 = usersolution2[0:-2]
                        textSurf8, textRect8 = text_objects(usersolution2,smallText)
                        textRect8.center = ((windowWidth/2), (windowHeight/2))
                        displaySurf.fill(bgColour)
                        displaySurf.blit(textSurf3, textRect3)
                        displaySurf.blit(textSurf9, textRect9)
                        displaySurf.blit(textSurf8, textRect8)
                        pygame.display.update()
            if x1_2 != "No solution" and usersolution2 == str(x1_2):
                UserAnswer = "Correct"
            else:
                UserAnswer = "Incorrect"
        if x2_2 != "No solution":
            textSurf11, textRect11 = text_objects("Solution: ",smallText)
            textRect11.center = ((windowWidth/2), ((windowHeight/4)+60))
            displaySurf.fill(bgColour)
            displaySurf.blit(textSurf3, textRect3)
            displaySurf.blit(textSurf11, textRect11)
            pygame.display.update()
            while 1:
                event = pygame.event.poll()
                if event.type == KEYDOWN:
                    if event.unicode == '\r': break
                    usersolution4 += event.unicode
                    if event.key == K_BACKSPACE:
                        usersolution4 = usersolution4[0:-2]
                    textSurf8, textRect8 = text_objects(usersolution4,smallText)
                    textRect8.center = ((windowWidth/2), (windowHeight/2))
                    displaySurf.fill(bgColour)
                    displaySurf.blit(textSurf3, textRect3)
                    displaySurf.blit(textSurf11, textRect11)
                    displaySurf.blit(textSurf8, textRect8)
                    pygame.display.update()
            while usersolution4 != str(x2_2) and attempts > 1:# answers were incorrect
                attempts = attempts - 1
                textSurf11, textRect11 = text_objects("Incorrect solutions, "+str(attempts)+" attempts
remaining. Solution one: ",smallText)
                textRect11.center = ((windowWidth/2), ((windowHeight/4)+60))
                displaySurf.fill(bgColour)
                displaySurf.blit(textSurf3, textRect3)
                displaySurf.blit(textSurf11, textRect11)
                pygame.display.update()
                while 1:
                    event = pygame.event.poll()
                    if event.type == KEYDOWN:
                        if event.unicode == '\r': break
                        usersolution4 += event.unicode
                        if event.key == K_BACKSPACE:
                            usersolution4 = usersolution4[0:-2]
                        textSurf8, textRect8 = text_objects(usersolution4,smallText)
                        textRect8.center = ((windowWidth/2), (windowHeight/2))
                        displaySurf.fill(bgColour)
                        displaySurf.blit(textSurf3, textRect3)
                        displaySurf.blit(textSurf11, textRect11)
                        displaySurf.blit(textSurf8, textRect8)
                        pygame.display.update()
            if x2_2 != "No solution" and usersolution4 == str(x2_2):
                UserAnswer = "Correct"
            else:
                UserAnswer = "Incorrect"
    ## Two solutions ##

    if solutions == 2 and usersolutionsno == str(solutions):
        if x1 != "No solution" and x2 != "No solution": #finds the variables which are answers
            textSurf7, textRect7 = text_objects("Solution one: ",smallText)
            textRect7.center = ((windowWidth/2), ((windowHeight/4)+60))
            displaySurf.fill(bgColour)
            displaySurf.blit(textSurf3, textRect3)
            displaySurf.blit(textSurf7, textRect7)
            pygame.display.update()
            while 1:
                event = pygame.event.poll()
                if event.type == KEYDOWN:
                    if event.unicode == '\r': break
                    usersolution1 += event.unicode
                    if event.key == K_BACKSPACE:
                        usersolution1 = usersolution1[0:-2]
                    textSurf8, textRect8 = text_objects(usersolution1,smallText)
                    textRect8.center = ((windowWidth/2), (windowHeight/2))
                    displaySurf.fill(bgColour)
                    displaySurf.blit(textSurf3, textRect3)
                    displaySurf.blit(textSurf7, textRect7)
                    displaySurf.blit(textSurf8, textRect8)
                    pygame.display.update()
            textSurf10, textRect10 = text_objects("Solution two: ",smallText)
            textRect10.center = ((windowWidth/2), ((windowHeight/4)+60))
            displaySurf.fill(bgColour)
            displaySurf.blit(textSurf3, textRect3)
            displaySurf.blit(textSurf10, textRect10)
            pygame.display.update()
            while 1:
              event = pygame.event.poll()
              if event.type == KEYDOWN:
                if event.unicode == '\r': break
                usersolution3 += event.unicode
                if event.key == K_BACKSPACE:
                  usersolution3 = usersolution3[0:-2]
                textSurf8, textRect8 = text_objects(usersolution3,smallText)
                textRect8.center = ((windowWidth/2), (windowHeight/2))
                displaySurf.fill(bgColour)
                displaySurf.blit(textSurf3, textRect3)
                displaySurf.blit(textSurf10, textRect10)
                displaySurf.blit(textSurf8, textRect8)
                pygame.display.update()
            while attempts > 1:
              if (usersolution1 == str(x1) and usersolution3 == str(x2)) or (usersolution1 == str(x2) and 
    usersolution3 == str(x1)):
                UserAnswer = "Correct"
                attempts = 0
              else:# answers were incorrect
                attempts = attempts - 1
                textSurf7, textRect7 = text_objects("Incorrect solutions, "+str(attempts)+" attempts 
    remaining. Solution one: ",smallText)
                textRect7.center = ((windowWidth/2), ((windowHeight/4)+60))
                displaySurf.fill(bgColour)
                displaySurf.blit(textSurf3, textRect3)
                displaySurf.blit(textSurf7, textRect7)
                pygame.display.update()
                while 1:
                  event = pygame.event.poll()
                  if event.type == KEYDOWN:
                    if event.unicode == '\r': break
                    usersolution1 += event.unicode
                    if event.key == K_BACKSPACE:
                      usersolution1 = usersolution1[0:-2]
                    textSurf8, textRect8 = text_objects(usersolution1,smallText)
                    textRect8.center = ((windowWidth/2), (windowHeight/2))
                    displaySurf.fill(bgColour)
                    displaySurf.blit(textSurf3, textRect3)
                    displaySurf.blit(textSurf7, textRect7)
                    displaySurf.blit(textSurf8, textRect8)
                    pygame.display.update()
                textSurf10, textRect10 = text_objects("Solution two: ",smallText)
                textRect10.center = ((windowWidth/2), ((windowHeight/4)+60))
                displaySurf.fill(bgColour)
```

```
      displaySurf.blit(textSurf3, textRect3)
      displaySurf.blit(textSurf10, textRect10)
      pygame.display.update()
      while 1:
        event = pygame.event.poll()
        if event.type == KEYDOWN:
          if event.unicode == '\r': break
          usersolution3 += event.unicode
          if event.key == K_BACKSPACE:
            usersolution3 = usersolution2[0:-2]
          textSurf8, textRect8 = text_objects(usersolution3,smallText)
          textRect8.center = ((windowWidth/2), (windowHeight/2))
          displaySurf.fill(bgColour)
          displaySurf.blit(textSurf3, textRect3)
          displaySurf.blit(textSurf10, textRect10)
          displaySurf.blit(textSurf8, textRect8)
          pygame.display.update()

if x1 != "No solution" and x1_2 != "No solution":#finds the variables which are answers
  textSurf7, textRect7 = text_objects("Solution one: ",smallText)
  textRect7.center = ((windowWidth/2), ((windowHeight/4)+60))
  displaySurf.fill(bgColour)
  displaySurf.blit(textSurf3, textRect3)
  displaySurf.blit(textSurf7, textRect7)
  pygame.display.update()
  while 1:
    event = pygame.event.poll()
    if event.type == KEYDOWN:
      if event.unicode == '\r': break
      usersolution1 += event.unicode
      if event.key == K_BACKSPACE:
        usersolution1 = usersolution1[0:-2]
      textSurf8, textRect8 = text_objects(usersolution1,smallText)
      textRect8.center = ((windowWidth/2), (windowHeight/2))
      displaySurf.fill(bgColour)
      displaySurf.blit(textSurf3, textRect3)
      displaySurf.blit(textSurf7, textRect7)
      displaySurf.blit(textSurf8, textRect8)
      pygame.display.update()
  textSurf9, textRect9 = text_objects("Solution two: ",smallText)
  textRect9.center = ((windowWidth/2), ((windowHeight/4)+60))
  displaySurf.fill(bgColour)
  displaySurf.blit(textSurf3, textRect3)
  displaySurf.blit(textSurf9, textRect9)
  pygame.display.update()
  while 1:
    event = pygame.event.poll()
    if event.type == KEYDOWN:
            if event.unicode == '\r': break
            usersolution2 += event.unicode
            if event.key == K_BACKSPACE:
                usersolution2 = usersolution2[0:-2]
            textSurf8, textRect8 = text_objects(usersolution2,smallText)
            textRect8.center = ((windowWidth/2), (windowHeight/2))
            displaySurf.fill(bgColour)
            displaySurf.blit(textSurf3, textRect3)
            displaySurf.blit(textSurf9, textRect9)
            displaySurf.blit(textSurf8, textRect8)
            pygame.display.update()
    while attempts > 1:
        if (usersolution1 == str(x1) and usersolution2 == str(x1_2)) or (usersolution1 == str(x1_2) and
usersolution2 == str(x1)):
            UserAnswer = "Correct"
            attempts = 0
        else:# answers were incorrect
            attempts = attempts - 1
            textSurf7, textRect7 = text_objects("Incorrect solutions, "+str(attempts)+" attempts
remaining. Solution one: ",smallText)
            textRect7.center = ((windowWidth/2), ((windowHeight/4)+60))
            displaySurf.fill(bgColour)
            displaySurf.blit(textSurf3, textRect3)
            displaySurf.blit(textSurf7, textRect7)
            pygame.display.update()
            while 1:
                event = pygame.event.poll()
                if event.type == KEYDOWN:
                    if event.unicode == '\r': break
                    usersolution1 += event.unicode
                    if event.key == K_BACKSPACE:
                        usersolution1 = usersolution1[0:-2]
                    textSurf8, textRect8 = text_objects(usersolution1,smallText)
                    textRect8.center = ((windowWidth/2), (windowHeight/2))
                    displaySurf.fill(bgColour)
                    displaySurf.blit(textSurf3, textRect3)
                    displaySurf.blit(textSurf7, textRect7)
                    displaySurf.blit(textSurf8, textRect8)
                    pygame.display.update()
            textSurf9, textRect9 = text_objects("Solution two: ",smallText)
            textRect9.center = ((windowWidth/2), ((windowHeight/4)+60))
            displaySurf.fill(bgColour)
            displaySurf.blit(textSurf3, textRect3)
            displaySurf.blit(textSurf9, textRect9)
            pygame.display.update()
            while 1:
                event = pygame.event.poll()
                if event.type == KEYDOWN:
                    if event.unicode == '\r': break
                    usersolution2 += event.unicode
                    if event.key == K_BACKSPACE:
                        usersolution2 = usersolution2[0:-2]
                    textSurf8, textRect8 = text_objects(usersolution2,smallText)
                    textRect8.center = ((windowWidth/2), (windowHeight/2))
                    displaySurf.fill(bgColour)
                    displaySurf.blit(textSurf3, textRect3)
                    displaySurf.blit(textSurf9, textRect9)
                    displaySurf.blit(textSurf8, textRect8)
                    pygame.display.update()

if x1 != "No solution" and x2_2 != "No solution":#finds the variables which are answers
    textSurf7, textRect7 = text_objects("Solution one: ",smallText)
    textRect7.center = ((windowWidth/2), ((windowHeight/4)+60))
    displaySurf.fill(bgColour)
    displaySurf.blit(textSurf3, textRect3)
    displaySurf.blit(textSurf7, textRect7)
    pygame.display.update()
    while 1:
        event = pygame.event.poll()
        if event.type == KEYDOWN:
            if event.unicode == '\r': break
            usersolution1 += event.unicode
            if event.key == K_BACKSPACE:
                usersolution1 = usersolution1[0:-2]
            textSurf8, textRect8 = text_objects(usersolution1,smallText)
            textRect8.center = ((windowWidth/2), (windowHeight/2))
            displaySurf.fill(bgColour)
            displaySurf.blit(textSurf3, textRect3)
            displaySurf.blit(textSurf7, textRect7)
            displaySurf.blit(textSurf8, textRect8)
            pygame.display.update()
    textSurf11, textRect11 = text_objects("Solution two: ",smallText)
    textRect11.center = ((windowWidth/2), ((windowHeight/4)+60))
    displaySurf.fill(bgColour)
    displaySurf.blit(textSurf3, textRect3)
    displaySurf.blit(textSurf11, textRect11)
    pygame.display.update()
    while 1:
        event = pygame.event.poll()
        if event.type == KEYDOWN:
            if event.unicode == '\r': break
            usersolution4 += event.unicode
            if event.key == K_BACKSPACE:
                usersolution4 = usersolution4[0:-2]
            textSurf8, textRect8 = text_objects(usersolution4,smallText)
            textRect8.center = ((windowWidth/2), (windowHeight/2))
```

```
                displaySurf.fill(bgColour)
                displaySurf.blit(textSurf3, textRect3)
                displaySurf.blit(textSurf11, textRect11)
                displaySurf.blit(textSurf8, textRect8)
                pygame.display.update()
        while attempts > 1:
            if (usersolution1 == str(x1) and usersolution4 == str(x2_2)) or (usersolution1 == str(x2_2) and
usersolution4 == str(x1)):
                UserAnswer = "Correct"
                attempts = 0
            else:# answers were incorrect
                attempts = attempts - 1
                textSurf7, textRect7 = text_objects("Incorrect solutions, "+str(attempts)+" attempts
remaining. Solution one: ",smallText)
                textRect7.center = ((windowWidth/2), ((windowHeight/4)+60))
                displaySurf.fill(bgColour)
                displaySurf.blit(textSurf3, textRect3)
                displaySurf.blit(textSurf7, textRect7)
                pygame.display.update()
                while 1:
                    event = pygame.event.poll()
                    if event.type == KEYDOWN:
                        if event.unicode == '\r': break
                        usersolution1 += event.unicode
                        if event.key == K_BACKSPACE:
                            usersolution1 = usersolution1[0:-2]
                        textSurf8, textRect8 = text_objects(usersolution1,smallText)
                        textRect8.center = ((windowWidth/2), (windowHeight/2))
                        displaySurf.fill(bgColour)
                        displaySurf.blit(textSurf3, textRect3)
                        displaySurf.blit(textSurf7, textRect7)
                        displaySurf.blit(textSurf8, textRect8)
                        pygame.display.update()
                textSurf11, textRect11 = text_objects("Solution two: ",smallText)
                textRect11.center = ((windowWidth/2), ((windowHeight/4)+60))
                displaySurf.fill(bgColour)
                displaySurf.blit(textSurf3, textRect3)
                displaySurf.blit(textSurf11, textRect11)
                pygame.display.update()
                while 1:
                    event = pygame.event.poll()
                    if event.type == KEYDOWN:
                        if event.unicode == '\r': break
                        usersolution4 += event.unicode
                        if event.key == K_BACKSPACE:
                            usersolution4 = usersolution2[0:-2]
                        textSurf8, textRect8 = text_objects(usersolution2,smallText)
                        textRect8.center = ((windowWidth/2), (windowHeight/2))
```

```
              displaySurf.fill(bgColour)
              displaySurf.blit(textSurf3, textRect3)
              displaySurf.blit(textSurf11, textRect11)
              displaySurf.blit(textSurf8, textRect8)
              pygame.display.update()
    if x1_2 != "No solution" and x2 != "No solution":#finds the variables which are answers
      textSurf9, textRect9 = text_objects("Solution one: ",smallText)
      textRect9.center = ((windowWidth/2), ((windowHeight/4)+60))
      displaySurf.fill(bgColour)
      displaySurf.blit(textSurf3, textRect3)
      displaySurf.blit(textSurf9, textRect9)
      pygame.display.update()
      while 1:
        event = pygame.event.poll()
        if event.type == KEYDOWN:
          if event.unicode == '\r': break
          usersolution2 += event.unicode
          if event.key == K_BACKSPACE:
            usersolution2 = usersolution2[0:-2]
          textSurf8, textRect8 = text_objects(usersolution2,smallText)
          textRect8.center = ((windowWidth/2), (windowHeight/2))
          displaySurf.fill(bgColour)
          displaySurf.blit(textSurf3, textRect3)
          displaySurf.blit(textSurf9, textRect9)
          displaySurf.blit(textSurf8, textRect8)
          pygame.display.update()
      textSurf10, textRect10 = text_objects("Solution two: ",smallText)
      textRect10.center = ((windowWidth/2), ((windowHeight/4)+60))
      displaySurf.fill(bgColour)
      displaySurf.blit(textSurf3, textRect3)
      displaySurf.blit(textSurf10, textRect10)
      pygame.display.update()
      while 1:
        event = pygame.event.poll()
        if event.type == KEYDOWN:
          if event.unicode == '\r': break
          usersolution3 += event.unicode
          if event.key == K_BACKSPACE:
            usersolution3 = usersolution3[0:-2]
          textSurf8, textRect8 = text_objects(usersolution3,smallText)
          textRect8.center = ((windowWidth/2), (windowHeight/2))
          displaySurf.fill(bgColour)
          displaySurf.blit(textSurf3, textRect3)
          displaySurf.blit(textSurf10, textRect10)
          displaySurf.blit(textSurf8, textRect8)
          pygame.display.update()
      while attempts > 1:
```

```
    if (usersolution2 == str(x1_2) and usersolution3 == str(x2)) or (usersolution2 == str(x2) and usersolution3 == str(x1_2)):
        UserAnswer = "Correct"
        attempts = 0
    else:# answers were incorrect
        attempts = attempts - 1
        textSurf9, textRect9 = text_objects("Incorrect solutions, "+str(attempts)+" attempts remaining. Solution one: ",smallText)
        textRect9.center = ((windowWidth/2), ((windowHeight/4)+60))
        displaySurf.fill(bgColour)
        displaySurf.blit(textSurf3, textRect3)
        displaySurf.blit(textSurf9, textRect9)
        pygame.display.update()
        while 1:
            event = pygame.event.poll()
            if event.type == KEYDOWN:
                if event.unicode == '\r': break
                usersolution2 += event.unicode
                if event.key == K_BACKSPACE:
                    usersolution2 = usersolution2[0:-2]
                textSurf8, textRect8 = text_objects(usersolution2,smallText)
                textRect8.center = ((windowWidth/2), (windowHeight/2))
                displaySurf.fill(bgColour)
                displaySurf.blit(textSurf3, textRect3)
                displaySurf.blit(textSurf9, textRect9)
                displaySurf.blit(textSurf8, textRect8)
                pygame.display.update()
        textSurf10, textRect10 = text_objects("Solution two: ",smallText)
        textRect10.center = ((windowWidth/2), ((windowHeight/4)+60))
        displaySurf.fill(bgColour)
        displaySurf.blit(textSurf3, textRect3)
        displaySurf.blit(textSurf10, textRect10)
        pygame.display.update()
        while 1:
            event = pygame.event.poll()
            if event.type == KEYDOWN:
                if event.unicode == '\r': break
                usersolution3 += event.unicode
                if event.key == K_BACKSPACE:
                    usersolution3 = usersolution2[0:-2]
                textSurf8, textRect8 = text_objects(usersolution3,smallText)
                textRect8.center = ((windowWidth/2), (windowHeight/2))
                displaySurf.fill(bgColour)
                displaySurf.blit(textSurf3, textRect3)
                displaySurf.blit(textSurf10, textRect10)
                displaySurf.blit(textSurf8, textRect8)
                pygame.display.update()
        if x1_2 != "No solution" and x2_2 != "No solution":#finds the variables which are answers
            textSurf9, textRect9 = text_objects("Solution one: ",smallText)
            textRect9.center = ((windowWidth/2), ((windowHeight/4)+60))
            displaySurf.fill(bgColour)
            displaySurf.blit(textSurf3, textRect3)
            displaySurf.blit(textSurf9, textRect9)
            pygame.display.update()
            while 1:
                event = pygame.event.poll()
                if event.type == KEYDOWN:
                    if event.unicode == '\r': break
                    usersolution2 += event.unicode
                    if event.key == K_BACKSPACE:
                        usersolution2 = usersolution2[0:-2]
                    textSurf8, textRect8 = text_objects(usersolution2,smallText)
                    textRect8.center = ((windowWidth/2), (windowHeight/2))
                    displaySurf.fill(bgColour)
                    displaySurf.blit(textSurf3, textRect3)
                    displaySurf.blit(textSurf9, textRect9)
                    displaySurf.blit(textSurf8, textRect8)
                    pygame.display.update()
            textSurf11, textRect11 = text_objects("Solution two: ",smallText)
            textRect11.center = ((windowWidth/2), ((windowHeight/4)+60))
            displaySurf.fill(bgColour)
            displaySurf.blit(textSurf3, textRect3)
            displaySurf.blit(textSurf11, textRect11)
            pygame.display.update()
            while 1:
                event = pygame.event.poll()
                if event.type == KEYDOWN:
                    if event.unicode == '\r': break
                    usersolution4 += event.unicode
                    if event.key == K_BACKSPACE:
                        usersolution4 = usersolution4[0:-2]
                    textSurf8, textRect8 = text_objects(usersolution4,smallText)
                    textRect8.center = ((windowWidth/2), (windowHeight/2))
                    displaySurf.fill(bgColour)
                    displaySurf.blit(textSurf3, textRect3)
                    displaySurf.blit(textSurf11, textRect11)
                    displaySurf.blit(textSurf8, textRect8)
                    pygame.display.update()
            while attempts > 1:
                if (usersolution2 == str(x1_2) and usersolution4 == str(x2_2)) or (usersolution2 == str(x2_2)
and usersolution4 == str(x1_2)):
                    UserAnswer = "Correct"
                    attempts = 0
                else:
                    attempts = attempts - 1# answers were incorrect
```

```
          textSurf9, textRect9 = text_objects("Incorrect solutions, "+str(attempts)+" attempts
remaining. Solution one: ",smallText)
          textRect9.center = ((windowWidth/2), ((windowHeight/4)+60))
          displaySurf.fill(bgColour)
          displaySurf.blit(textSurf3, textRect3)
          displaySurf.blit(textSurf9, textRect9)
          pygame.display.update()
          while 1:
            event = pygame.event.poll()
            if event.type == KEYDOWN:
              if event.unicode == '\r': break
              usersolution2 += event.unicode
              if event.key == K_BACKSPACE:
                usersolution2 = usersolution2[0:-2]
              textSurf8, textRect8 = text_objects(usersolution2,smallText)
              textRect8.center = ((windowWidth/2), (windowHeight/2))
              displaySurf.fill(bgColour)
              displaySurf.blit(textSurf3, textRect3)
              displaySurf.blit(textSurf9, textRect9)
              displaySurf.blit(textSurf8, textRect8)
              pygame.display.update()
          textSurf11, textRect11 = text_objects("Solution two: ",smallText)
          textRect11.center = ((windowWidth/2), ((windowHeight/4)+60))
          displaySurf.fill(bgColour)
          displaySurf.blit(textSurf3, textRect3)
          displaySurf.blit(textSurf11, textRect11)
          pygame.display.update()
          while 1:
            event = pygame.event.poll()
            if event.type == KEYDOWN:
              if event.unicode == '\r': break
              usersolution4 += event.unicode
              if event.key == K_BACKSPACE:
                usersolution4 = usersolution2[0:-2]
              textSurf8, textRect8 = text_objects(usersolution2,smallText)
              textRect8.center = ((windowWidth/2), (windowHeight/2))
              displaySurf.fill(bgColour)
              displaySurf.blit(textSurf3, textRect3)
              displaySurf.blit(textSurf11, textRect11)
              displaySurf.blit(textSurf8, textRect8)
              pygame.display.update()

    if x2_2 != "No solution" and x2 != "No solution":#finds the variables which are answers
      textSurf11, textRect11 = text_objects("Solution one: ",smallText)
      textRect11.center = ((windowWidth/2), ((windowHeight/4)+60))
      displaySurf.fill(bgColour)
      displaySurf.blit(textSurf3, textRect3)
      displaySurf.blit(textSurf11, textRect11)
```

```
        pygame.display.update()
        while 1:
            event = pygame.event.poll()
            if event.type == KEYDOWN:
                if event.unicode == '\r': break
                usersolution4 += event.unicode
                if event.key == K_BACKSPACE:
                    usersolution4 = usersolution4[0:-2]
                textSurf8, textRect8 = text_objects(usersolution4,smallText)
                textRect8.center = ((windowWidth/2), (windowHeight/2))
                displaySurf.fill(bgColour)
                displaySurf.blit(textSurf3, textRect3)
                displaySurf.blit(textSurf11, textRect11)
                displaySurf.blit(textSurf8, textRect8)
                pygame.display.update()
        textSurf10, textRect10 = text_objects("Solution two: ",smallText)
        textRect10.center = ((windowWidth/2), ((windowHeight/4)+60))
        displaySurf.fill(bgColour)
        displaySurf.blit(textSurf3, textRect3)
        displaySurf.blit(textSurf10, textRect10)
        pygame.display.update()
        while 1:
            event = pygame.event.poll()
            if event.type == KEYDOWN:
                if event.unicode == '\r': break
                usersolution3 += event.unicode
                if event.key == K_BACKSPACE:
                    usersolution3 = usersolution3[0:-2]
                textSurf8, textRect8 = text_objects(usersolution3,smallText)
                textRect8.center = ((windowWidth/2), (windowHeight/2))
                displaySurf.fill(bgColour)
                displaySurf.blit(textSurf3, textRect3)
                displaySurf.blit(textSurf10, textRect10)
                displaySurf.blit(textSurf8, textRect8)
                pygame.display.update()
        while attempts > 1:
            if (usersolution4 == str(x2_2) and usersolution3 == str(x2)) or (usersolution4 == str(x2) and
usersolution3 == str(x2_2)):
                UserAnswer = "Correct"
                attempts = 0
            else:# answers were incorrect
                attempts = attempts - 1
                textSurf11, textRect11 = text_objects("Incorrect solutions, "+str(attempts)+" attempts
remaining. Solution one: ",smallText)
                textRect11.center = ((windowWidth/2), ((windowHeight/4)+60))
                displaySurf.fill(bgColour)
                displaySurf.blit(textSurf3, textRect3)
                displaySurf.blit(textSurf11, textRect11)
```

```
             pygame.display.update()
             while 1:
                event = pygame.event.poll()
                if event.type == KEYDOWN:
                   if event.unicode == '\r': break
                   usersolution4 += event.unicode
                   if event.key == K_BACKSPACE:
                      usersolution4 = usersolution2[0:-2]
                   textSurf8, textRect8 = text_objects(usersolution2,smallText)
                   textRect8.center = ((windowWidth/2), (windowHeight/2))
                   displaySurf.fill(bgColour)
                   displaySurf.blit(textSurf3, textRect3)
                   displaySurf.blit(textSurf11, textRect11)
                   displaySurf.blit(textSurf8, textRect8)
                   pygame.display.update()
             textSurf10, textRect10 = text_objects("Solution two: ",smallText)
             textRect10.center = ((windowWidth/2), ((windowHeight/4)+60))
             displaySurf.fill(bgColour)
             displaySurf.blit(textSurf3, textRect3)
             displaySurf.blit(textSurf10, textRect10)
             pygame.display.update()
             while 1:
                event = pygame.event.poll()
                if event.type == KEYDOWN:
                   if event.unicode == '\r': break
                   usersolution3 += event.unicode
                   if event.key == K_BACKSPACE:
                      usersolution3 = usersolution2[0:-2]
                   textSurf8, textRect8 = text_objects(usersolution3,smallText)
                   textRect8.center = ((windowWidth/2), (windowHeight/2))
                   displaySurf.fill(bgColour)
                   displaySurf.blit(textSurf3, textRect3)
                   displaySurf.blit(textSurf10, textRect10)
                   displaySurf.blit(textSurf8, textRect8)
                   pygame.display.update()
## Three solutions ##

 if solutions == 3 and usersolutionsno == str(solutions):#finds the variables which are answers
    if x1 != "No solution" and x1_2  != "No solution" and x2 != "No solution":
       textSurf7, textRect7 = text_objects("Solution one: ",smallText)
       textRect7.center = ((windowWidth/2), ((windowHeight/4)+60))
       displaySurf.fill(bgColour)
       displaySurf.blit(textSurf3, textRect3)
       displaySurf.blit(textSurf7, textRect7)
       pygame.display.update()
       while 1:
          event = pygame.event.poll()
          if event.type == KEYDOWN:
```

```
        if event.unicode == '\r': break
        usersolution1 += event.unicode
        if event.key == K_BACKSPACE:
            usersolution1 = usersolution1[0:-2]
        textSurf8, textRect8 = text_objects(usersolution1,smallText)
        textRect8.center = ((windowWidth/2), (windowHeight/2))
        displaySurf.fill(bgColour)
        displaySurf.blit(textSurf3, textRect3)
        displaySurf.blit(textSurf7, textRect7)
        displaySurf.blit(textSurf8, textRect8)
        pygame.display.update()
textSurf9, textRect9 = text_objects("Solution two: ",smallText)
textRect9.center = ((windowWidth/2), ((windowHeight/4)+60))
displaySurf.fill(bgColour)
displaySurf.blit(textSurf3, textRect3)
displaySurf.blit(textSurf9, textRect9)
pygame.display.update()
while 1:
    event = pygame.event.poll()
    if event.type == KEYDOWN:
        if event.unicode == '\r': break
        usersolution2 += event.unicode
        if event.key == K_BACKSPACE:
            usersolution2 = usersolution2[0:-2]
        textSurf8, textRect8 = text_objects(usersolution2,smallText)
        textRect8.center = ((windowWidth/2), (windowHeight/2))
        displaySurf.fill(bgColour)
        displaySurf.blit(textSurf3, textRect3)
        displaySurf.blit(textSurf9, textRect9)
        displaySurf.blit(textSurf8, textRect8)
        pygame.display.update()

textSurf10, textRect10 = text_objects("Solution three: ",smallText)
textRect10.center = ((windowWidth/2), ((windowHeight/4)+60))
displaySurf.fill(bgColour)
displaySurf.blit(textSurf3, textRect3)
displaySurf.blit(textSurf10, textRect10)
pygame.display.update()
while 1:
    event = pygame.event.poll()
    if event.type == KEYDOWN:
        if event.unicode == '\r': break
        usersolution3 += event.unicode
        if event.key == K_BACKSPACE:
            usersolution3 = usersolution3[0:-2]
        textSurf8, textRect8 = text_objects(usersolution3,smallText)
        textRect8.center = ((windowWidth/2), (windowHeight/2))
        displaySurf.fill(bgColour)
```

```
            displaySurf.blit(textSurf3, textRect3)
            displaySurf.blit(textSurf10, textRect10)
            displaySurf.blit(textSurf8, textRect8)
            pygame.display.update()
    while attempts > 1:
        if (usersolution1 == str(x1) and usersolution2 == str(x1_2) and usersolution3 == str(x2))\
           or (usersolution1 == str(x1) and usersolution2 == str(x2) and userslolution3 == str(x1_2))\
           or (usersolution1 == str(x1_2) and usersolution2 == str(x1) and usersolution3 == str(x2))\
           or (usersolution1 == str(x1_2) and usersolution2 == str(x2) and usersolution3 == str(x1))\
           or (usersolution1 == str(x2) and usersolution2 == str(x1) and usersolution3 == str(x1_2))\
           or (usersolution1 == str(x2) and usersolution2 == str(x1_2) and usersolution3 == str(x1)):
                UserAnswer = "Correct"
                attempts = 0
        else:# answers were incorrect
            attempts = attempts - 1
            textSurf7, textRect7 = text_objects("Incorrect solutions, "+str(attempts)+" attempts
remaining. Solution one: ",smallText)
            textRect7.center = ((windowWidth/2), ((windowHeight/4)+60))
            displaySurf.fill(bgColour)
            displaySurf.blit(textSurf3, textRect3)
            displaySurf.blit(textSurf7, textRect7)
            pygame.display.update()
            while 1:
                event = pygame.event.poll()
                if event.type == KEYDOWN:
                    if event.unicode == '\r': break
                    usersolution1 += event.unicode
                    if event.key == K_BACKSPACE:
                        usersolution1 = usersolution1[0:-2]
                    textSurf8, textRect8 = text_objects(usersolution1,smallText)
                    textRect8.center = ((windowWidth/2), (windowHeight/2))
                    displaySurf.fill(bgColour)
                    displaySurf.blit(textSurf3, textRect3)
                    displaySurf.blit(textSurf7, textRect7)
                    displaySurf.blit(textSurf8, textRect8)
                    pygame.display.update()
            textSurf9, textRect9 = text_objects("Solution two: ",smallText)
            textRect9.center = ((windowWidth/2), ((windowHeight/4)+60))
            displaySurf.fill(bgColour)
            displaySurf.blit(textSurf3, textRect3)
            displaySurf.blit(textSurf9, textRect9)
            pygame.display.update()
            while 1:
                event = pygame.event.poll()
                if event.type == KEYDOWN:
                    if event.unicode == '\r': break
                    usersolution2 += event.unicode
                    if event.key == K_BACKSPACE:
                        usersolution2 = usersolution2[0:-2]
                    textSurf8, textRect8 = text_objects(usersolution2,smallText)
                    textRect8.center = ((windowWidth/2), (windowHeight/2))
                    displaySurf.fill(bgColour)
                    displaySurf.blit(textSurf3, textRect3)
                    displaySurf.blit(textSurf9, textRect9)
                    displaySurf.blit(textSurf8, textRect8)
                    pygame.display.update()
            textSurf10, textRect10 = text_objects("Solution three: ",smallText)
            textRect10.center = ((windowWidth/2), ((windowHeight/4)+60))
            displaySurf.fill(bgColour)
            displaySurf.blit(textSurf3, textRect3)
            displaySurf.blit(textSurf10, textRect10)
            pygame.display.update()
            while 1:
                event = pygame.event.poll()
                if event.type == KEYDOWN:
                    if event.unicode == '\r': break
                    usersolution3 += event.unicode
                    if event.key == K_BACKSPACE:
                        usersolution3 = usersolution3[0:-2]
                    textSurf8, textRect8 = text_objects(usersolution3,smallText)
                    textRect8.center = ((windowWidth/2), (windowHeight/2))
                    displaySurf.fill(bgColour)
                    displaySurf.blit(textSurf3, textRect3)
                    displaySurf.blit(textSurf10, textRect10)
                    displaySurf.blit(textSurf8, textRect8)
                    pygame.display.update()

    if x1 != "No solution" and x1_2  != "No solution" and x2_2 != "No solution":#finds the variables which are answers
        textSurf7, textRect7 = text_objects("Solution one: ",smallText)
        textRect7.center = ((windowWidth/2), ((windowHeight/4)+60))
        displaySurf.fill(bgColour)
        displaySurf.blit(textSurf3, textRect3)
        displaySurf.blit(textSurf7, textRect7)
        pygame.display.update()
        while 1:
        event = pygame.event.poll()
        if event.type == KEYDOWN:
            if event.unicode == '\r': break
            usersolution1 += event.unicode
            if event.key == K_BACKSPACE:
                usersolution1 = usersolution1[0:-2]
            textSurf8, textRect8 = text_objects(usersolution1,smallText)
            textRect8.center = ((windowWidth/2), (windowHeight/2))
            displaySurf.fill(bgColour)
            displaySurf.blit(textSurf3, textRect3)
```

```
    displaySurf.blit(textSurf7, textRect7)
    displaySurf.blit(textSurf8, textRect8)
    pygame.display.update()
textSurf9, textRect9 = text_objects("Solution two: ",smallText)
textRect9.center = ((windowWidth/2), ((windowHeight/4)+60))
displaySurf.fill(bgColour)
displaySurf.blit(textSurf3, textRect3)
displaySurf.blit(textSurf9, textRect9)
pygame.display.update()
while 1:
  event = pygame.event.poll()
  if event.type == KEYDOWN:
    if event.unicode == '\r': break
    usersolution2 += event.unicode
    if event.key == K_BACKSPACE:
      usersolution2 = usersolution2[0:-2]
    textSurf8, textRect8 = text_objects(usersolution2,smallText)
    textRect8.center = ((windowWidth/2), (windowHeight/2))
    displaySurf.fill(bgColour)
    displaySurf.blit(textSurf3, textRect3)
    displaySurf.blit(textSurf9, textRect9)
    displaySurf.blit(textSurf8, textRect8)
    pygame.display.update()
textSurf11, textRect11 = text_objects("Solution three: ",smallText)
textRect11.center = ((windowWidth/2), ((windowHeight/4)+60))
displaySurf.fill(bgColour)
displaySurf.blit(textSurf3, textRect3)
displaySurf.blit(textSurf11, textRect11)
pygame.display.update()
while 1:
  event = pygame.event.poll()
  if event.type == KEYDOWN:
    if event.unicode == '\r': break
    usersolution4 += event.unicode
    if event.key == K_BACKSPACE:
      usersolution4 = usersolution4[0:-2]
    textSurf8, textRect8 = text_objects(usersolution4,smallText)
    textRect8.center = ((windowWidth/2), (windowHeight/2))
    displaySurf.fill(bgColour)
    displaySurf.blit(textSurf3, textRect3)
    displaySurf.blit(textSurf11, textRect11)
    displaySurf.blit(textSurf8, textRect8)
    pygame.display.update()
while attempts > 1:
  if (usersolution1 == str(x1) and usersolution2 == str(x1_2) and usersolution4 == str(x2_2))\
     or (usersolution1 == str(x1) and usersolution2 == str(x2_2) and usersloution4 == str(x1_2))\
     or (usersolution1 == str(x1_2) and usersolution2 == str(x1) and usersolution4 == str(x2_2))\
     or (usersolution1 == str(x1_2) and usersolution2 == str(x2_2) and usersolution4 == str(x1))\
                   or (usersolution1 == str(x2_2) and usersolution2 == str(x1) and usersolution4 == str(x1_2))\
                   or (usersolution1 == str(x2_2) and usersolution2 == str(x1_2) and usersolution4 == str(x1)):
                        UserAnswer = "Correct"
                        attempts = 0
                else:# answers were incorrect
                    attempts = attempts - 1
                    textSurf7, textRect7 = text_objects("Incorrect solutions, "+str(attempts)+" attempts
remaining. Solution one: ",smallText)
                    textRect7.center = ((windowWidth/2), ((windowHeight/4)+60))
                    displaySurf.fill(bgColour)
                    displaySurf.blit(textSurf3, textRect3)
                    displaySurf.blit(textSurf7, textRect7)
                    pygame.display.update()
                    while 1:
                        event = pygame.event.poll()
                        if event.type == KEYDOWN:
                            if event.unicode == '\r': break
                            usersolution1 += event.unicode
                            if event.key == K_BACKSPACE:
                                usersolution1 = usersolution1[0:-2]
                            textSurf8, textRect8 = text_objects(usersolution1,smallText)
                            textRect8.center = ((windowWidth/2), (windowHeight/2))
                            displaySurf.fill(bgColour)
                            displaySurf.blit(textSurf3, textRect3)
                            displaySurf.blit(textSurf7, textRect7)
                            displaySurf.blit(textSurf8, textRect8)
                            pygame.display.update()
                    textSurf9, textRect9 = text_objects("Solution two: ",smallText)
                    textRect9.center = ((windowWidth/2), ((windowHeight/4)+60))
                    displaySurf.fill(bgColour)
                    displaySurf.blit(textSurf3, textRect3)
                    displaySurf.blit(textSurf9, textRect9)
                    pygame.display.update()
                    while 1:
                        event = pygame.event.poll()
                        if event.type == KEYDOWN:
                            if event.unicode == '\r': break
                            usersolution2 += event.unicode
                            if event.key == K_BACKSPACE:
                                usersolution2 = usersolution2[0:-2]
                            textSurf8, textRect8 = text_objects(usersolution2,smallText)
                            textRect8.center = ((windowWidth/2), (windowHeight/2))
                            displaySurf.fill(bgColour)
                            displaySurf.blit(textSurf3, textRect3)
                            displaySurf.blit(textSurf9, textRect9)
                            displaySurf.blit(textSurf8, textRect8)
                            pygame.display.update()
                    textSurf11, textRect11 = text_objects("Solution three: ",smallText)
```

```
          textRect11.center = ((windowWidth/2), ((windowHeight/4)+60))
          displaySurf.fill(bgColour)
          displaySurf.blit(textSurf3, textRect3)
          displaySurf.blit(textSurf11, textRect11)
          pygame.display.update()
          while 1:
            event = pygame.event.poll()
            if event.type == KEYDOWN:
              if event.unicode == '\r': break
              usersolution4 += event.unicode
              if event.key == K_BACKSPACE:
                usersolution4 = usersolution2[0:-2]
              textSurf8, textRect8 = text_objects(usersolution2,smallText)
              textRect8.center = ((windowWidth/2), (windowHeight/2))
              displaySurf.fill(bgColour)
              displaySurf.blit(textSurf3, textRect3)
              displaySurf.blit(textSurf11, textRect11)
              displaySurf.blit(textSurf8, textRect8)
              pygame.display.update()

    if x1 != "No solution" and x2  != "No solution" and x2_2 != "No solution":#finds the variables which
are answers
      textSurf7, textRect7 = text_objects("Solution one: ",smallText)
      textRect7.center = ((windowWidth/2), ((windowHeight/4)+60))
      displaySurf.fill(bgColour)
      displaySurf.blit(textSurf3, textRect3)
      displaySurf.blit(textSurf7, textRect7)
      pygame.display.update()
      while 1:
        event = pygame.event.poll()
        if event.type == KEYDOWN:
          if event.unicode == '\r': break
          usersolution1 += event.unicode
          if event.key == K_BACKSPACE:
            usersolution1 = usersolution1[0:-2]
          textSurf8, textRect8 = text_objects(usersolution1,smallText)
          textRect8.center = ((windowWidth/2), (windowHeight/2))
          displaySurf.fill(bgColour)
          displaySurf.blit(textSurf3, textRect3)
          displaySurf.blit(textSurf7, textRect7)
          displaySurf.blit(textSurf8, textRect8)
          pygame.display.update()
      textSurf10, textRect10 = text_objects("Solution two: ",smallText)
      textRect10.center = ((windowWidth/2), ((windowHeight/4)+60))
      displaySurf.fill(bgColour)
      displaySurf.blit(textSurf3, textRect3)
      displaySurf.blit(textSurf10, textRect10)
      pygame.display.update()
```

```
        while 1:
            event = pygame.event.poll()
            if event.type == KEYDOWN:
                if event.unicode == '\r': break
                usersolution3 += event.unicode
                if event.key == K_BACKSPACE:
                    usersolution3 = usersolution3[0:-2]
                textSurf8, textRect8 = text_objects(usersolution3,smallText)
                textRect8.center = ((windowWidth/2), (windowHeight/2))
                displaySurf.fill(bgColour)
                displaySurf.blit(textSurf3, textRect3)
                displaySurf.blit(textSurf10, textRect10)
                displaySurf.blit(textSurf8, textRect8)
                pygame.display.update()
        textSurf11, textRect11 = text_objects("Solution three: ",smallText)
        textRect11.center = ((windowWidth/2), ((windowHeight/4)+60))
        displaySurf.fill(bgColour)
        displaySurf.blit(textSurf3, textRect3)
        displaySurf.blit(textSurf11, textRect11)
        pygame.display.update()
        while 1:
            event = pygame.event.poll()
            if event.type == KEYDOWN:
                if event.unicode == '\r': break
                usersolution4 += event.unicode
                if event.key == K_BACKSPACE:
                    usersolution4 = usersolution4[0:-2]
                textSurf8, textRect8 = text_objects(usersolution4,smallText)
                textRect8.center = ((windowWidth/2), (windowHeight/2))
                displaySurf.fill(bgColour)
                displaySurf.blit(textSurf3, textRect3)
                displaySurf.blit(textSurf11, textRect11)
                displaySurf.blit(textSurf8, textRect8)
                pygame.display.update()
        while attempts > 1:
            if (usersolution1 == str(x1) and usersolution3 == str(x2) and usersolution4 == str(x2_2))\
              or (usersolution1 == str(x1) and usersolution3 == str(x2_2) and userslouti0n4 == str(x2))\
              or (usersolution1 == str(x2) and usersolution3 == str(x1) and usersolution4 == str(x2_2))\
              or (usersolution1 == str(x2) and usersolution3 == str(x2_2) and usersolution4 == str(x1))\
              or (usersolution1 == str(x2_2) and usersolution3 == str(x1) and usersolution4 == str(x2))\
              or (usersolution1 == str(x2_2) and usersolution3 == str(x2) and usersolution4 == str(x1)):
                    UserAnswer = "Correct"
                    attempts = 0
            else:# answers were incorrect
                attempts = attempts - 1
                textSurf7, textRect7 = text_objects("Incorrect solutions, "+str(attempts)+" attempts
remaining. Solution one: ",smallText)
                textRect7.center = ((windowWidth/2), ((windowHeight/4)+60))
```

```
displaySurf.fill(bgColour)
displaySurf.blit(textSurf3, textRect3)
displaySurf.blit(textSurf7, textRect7)
pygame.display.update()
while 1:
    event = pygame.event.poll()
    if event.type == KEYDOWN:
        if event.unicode == '\r': break
        usersolution1 += event.unicode
        if event.key == K_BACKSPACE:
            usersolution1 = usersolution1[0:-2]
        textSurf8, textRect8 = text_objects(usersolution1,smallText)
        textRect8.center = ((windowWidth/2), (windowHeight/2))
        displaySurf.fill(bgColour)
        displaySurf.blit(textSurf3, textRect3)
        displaySurf.blit(textSurf7, textRect7)
        displaySurf.blit(textSurf8, textRect8)
        pygame.display.update()
textSurf10, textRect10 = text_objects("Solution two: ",smallText)
textRect10.center = ((windowWidth/2), ((windowHeight/4)+60))
displaySurf.fill(bgColour)
displaySurf.blit(textSurf3, textRect3)
displaySurf.blit(textSurf10, textRect10)
pygame.display.update()
while 1:
    event = pygame.event.poll()
    if event.type == KEYDOWN:
        if event.unicode == '\r': break
        usersolution3 += event.unicode
        if event.key == K_BACKSPACE:
            usersolution3 = usersolution2[0:-2]
        textSurf8, textRect8 = text_objects(usersolution3,smallText)
        textRect8.center = ((windowWidth/2), (windowHeight/2))
        displaySurf.fill(bgColour)
        displaySurf.blit(textSurf3, textRect3)
        displaySurf.blit(textSurf10, textRect10)
        displaySurf.blit(textSurf8, textRect8)
        pygame.display.update()
textSurf11, textRect11 = text_objects("Solution three: ",smallText)
textRect11.center = ((windowWidth/2), ((windowHeight/4)+60))
displaySurf.fill(bgColour)
displaySurf.blit(textSurf3, textRect3)
displaySurf.blit(textSurf11, textRect11)
pygame.display.update()
while 1:
    event = pygame.event.poll()
    if event.type == KEYDOWN:
        if event.unicode == '\r': break
```

```
              usersolution4 += event.unicode
              if event.key == K_BACKSPACE:
                usersolution4 = usersolution2[0:-2]
              textSurf8, textRect8 = text_objects(usersolution2,smallText)
              textRect8.center = ((windowWidth/2), (windowHeight/2))
              displaySurf.fill(bgColour)
              displaySurf.blit(textSurf3, textRect3)
              displaySurf.blit(textSurf11, textRect11)
              displaySurf.blit(textSurf8, textRect8)
              pygame.display.update()


    if x1_2 != "No solution" and x2  != "No solution" and x2_2 != "No solution":#finds the variables
which are answers
      textSurf9, textRect9 = text_objects("Solution one: ",smallText)
      textRect9.center = ((windowWidth/2), ((windowHeight/4)+60))
      displaySurf.fill(bgColour)
      displaySurf.blit(textSurf3, textRect3)
      displaySurf.blit(textSurf9, textRect9)
      pygame.display.update()
      while 1:
        event = pygame.event.poll()
        if event.type == KEYDOWN:
          if event.unicode == '\r': break
          usersolution2 += event.unicode
          if event.key == K_BACKSPACE:
            usersolution2 = usersolution2[0:-2]
          textSurf8, textRect8 = text_objects(usersolution2,smallText)
          textRect8.center = ((windowWidth/2), (windowHeight/2))
          displaySurf.fill(bgColour)
          displaySurf.blit(textSurf3, textRect3)
          displaySurf.blit(textSurf9, textRect9)
          displaySurf.blit(textSurf8, textRect8)
          pygame.display.update()
      textSurf10, textRect10 = text_objects("Solution two: ",smallText)
      textRect10.center = ((windowWidth/2), ((windowHeight/4)+60))
      displaySurf.fill(bgColour)
      displaySurf.blit(textSurf3, textRect3)
      displaySurf.blit(textSurf10, textRect10)
      pygame.display.update()
      while 1:
        event = pygame.event.poll()
        if event.type == KEYDOWN:
          if event.unicode == '\r': break
          usersolution3 += event.unicode
          if event.key == K_BACKSPACE:
            usersolution3 = usersolution3[0:-2]
          textSurf8, textRect8 = text_objects(usersolution3,smallText)
```

```
            textRect8.center = ((windowWidth/2), (windowHeight/2))
            displaySurf.fill(bgColour)
            displaySurf.blit(textSurf3, textRect3)
            displaySurf.blit(textSurf10, textRect10)
            displaySurf.blit(textSurf8, textRect8)
            pygame.display.update()
        textSurf11, textRect11 = text_objects("Solution three: ",smallText)
        textRect11.center = ((windowWidth/2), ((windowHeight/4)+60))
        displaySurf.fill(bgColour)
        displaySurf.blit(textSurf3, textRect3)
        displaySurf.blit(textSurf11, textRect11)
        pygame.display.update()
        while 1:
          event = pygame.event.poll()
          if event.type == KEYDOWN:
            if event.unicode == '\r': break
            usersolution4 += event.unicode
            if event.key == K_BACKSPACE:
              usersolution4 = usersolution4[0:-2]
            textSurf8, textRect8 = text_objects(usersolution4,smallText)
            textRect8.center = ((windowWidth/2), (windowHeight/2))
            displaySurf.fill(bgColour)
            displaySurf.blit(textSurf3, textRect3)
            displaySurf.blit(textSurf11, textRect11)
            displaySurf.blit(textSurf8, textRect8)
            pygame.display.update()
        while attempts > 1: # answers were incorrect
          if (usersolution1 == str(x1_2) and usersolution3 == str(x2) and usersolution4 == str(x2_2))\
           or (usersolution2 == str(x1_2) and usersolution3 == str(x2_2) and userslution4 == str(x2))\
           or (usersolution2 == str(x2) and usersolution3 == str(x1_2) and usersolution4 == str(x2_2))\
           or (usersolution2 == str(x2) and usersolution3 == str(x2_2) and usersolution4 == str(x1_2))\
           or (usersolution2 == str(x2_2) and usersolution3 == str(x1_2) and usersolution4 == str(x2))\
           or (usersolution2 == str(x2_2) and usersolution3 == str(x2) and usersolution4 == str(x1_2)):
              UserAnswer = "Correct"
              attempts = 0
          else:# answers were incorrect
            attempts = attempts - 1
            textSurf9, textRect9 = text_objects("Incorrect solutions, "+str(attempts)+" attempts
remaining. Solution one: ",smallText)
            textRect9.center = ((windowWidth/2), ((windowHeight/4)+60))
            displaySurf.fill(bgColour)
            displaySurf.blit(textSurf3, textRect3)
            displaySurf.blit(textSurf9, textRect9)
            pygame.display.update()
            while 1:
              event = pygame.event.poll()
              if event.type == KEYDOWN:
                if event.unicode == '\r': break
                usersolution2 += event.unicode
                if event.key == K_BACKSPACE:
                    usersolution2 = usersolution2[0:-2]
                textSurf8, textRect8 = text_objects(usersolution2,smallText)
                textRect8.center = ((windowWidth/2), (windowHeight/2))
                displaySurf.fill(bgColour)
                displaySurf.blit(textSurf3, textRect3)
                displaySurf.blit(textSurf9, textRect9)
                displaySurf.blit(textSurf8, textRect8)
                pygame.display.update()
            textSurf10, textRect10 = text_objects("Solution two: ",smallText)
            textRect10.center = ((windowWidth/2), ((windowHeight/4)+60))
            displaySurf.fill(bgColour)
            displaySurf.blit(textSurf3, textRect3)
            displaySurf.blit(textSurf10, textRect10)
            pygame.display.update()
            while 1:
              event = pygame.event.poll()
              if event.type == KEYDOWN:
                if event.unicode == '\r': break
                usersolution3 += event.unicode
                if event.key == K_BACKSPACE:
                    usersolution3 = usersolution2[0:-2]
                textSurf8, textRect8 = text_objects(usersolution3,smallText)
                textRect8.center = ((windowWidth/2), (windowHeight/2))
                displaySurf.fill(bgColour)
                displaySurf.blit(textSurf3, textRect3)
                displaySurf.blit(textSurf10, textRect10)
                displaySurf.blit(textSurf8, textRect8)
                pygame.display.update()
            textSurf11, textRect11 = text_objects("Solution three: ",smallText)
            textRect11.center = ((windowWidth/2), ((windowHeight/4)+60))
            displaySurf.fill(bgColour)
            displaySurf.blit(textSurf3, textRect3)
            displaySurf.blit(textSurf11, textRect11)
            pygame.display.update()
            while 1:
              event = pygame.event.poll()
              if event.type == KEYDOWN:
                if event.unicode == '\r': break
                usersolution4 += event.unicode
                if event.key == K_BACKSPACE:
                    usersolution4 = usersolution2[0:-2]
                textSurf8, textRect8 = text_objects(usersolution2,smallText)
                textRect8.center = ((windowWidth/2), (windowHeight/2))
                displaySurf.fill(bgColour)
                displaySurf.blit(textSurf3, textRect3)
                displaySurf.blit(textSurf11, textRect11)
```

```
                        displaySurf.blit(textSurf8, textRect8)
                        pygame.display.update()
## Four solutions ##

 if solutions == 4 and usersolutionsno == str(solutions): # gets four answers from the user.
    textSurf7, textRect7 = text_objects("Solution one: ",smallText)
    textRect7.center = ((windowWidth/2), ((windowHeight/4)+60))
    displaySurf.fill(bgColour)
    displaySurf.blit(textSurf3, textRect3)
    displaySurf.blit(textSurf7, textRect7)
    pygame.display.update()
    while 1:
        event = pygame.event.poll()
        if event.type == KEYDOWN:
            if event.unicode == '\r': break
            usersolution1 += event.unicode
            if event.key == K_BACKSPACE:
                usersolution1 = usersolution1[0:-2]
            textSurf8, textRect8 = text_objects(usersolution1,smallText)
            textRect8.center = ((windowWidth/2), (windowHeight/2))
            displaySurf.fill(bgColour)
            displaySurf.blit(textSurf3, textRect3)
            displaySurf.blit(textSurf7, textRect7)
            displaySurf.blit(textSurf8, textRect8)
            pygame.display.update()

    textSurf9, textRect9 = text_objects("Solution two: ",smallText)
    textRect9.center = ((windowWidth/2), ((windowHeight/4)+60))
    displaySurf.fill(bgColour)
    displaySurf.blit(textSurf3, textRect3)
    displaySurf.blit(textSurf9, textRect9)
    pygame.display.update()
    while 1:
        event = pygame.event.poll()
        if event.type == KEYDOWN:
            if event.unicode == '\r': break
            usersolution2 += event.unicode
            if event.key == K_BACKSPACE:
                usersolution2 = usersolution2[0:-2]
            textSurf8, textRect8 = text_objects(usersolution2,smallText)
            textRect8.center = ((windowWidth/2), (windowHeight/2))
            displaySurf.fill(bgColour)
            displaySurf.blit(textSurf3, textRect3)
            displaySurf.blit(textSurf9, textRect9)
            displaySurf.blit(textSurf8, textRect8)
            pygame.display.update()
    textSurf10, textRect10 = text_objects("Solution three: ",smallText)
    textRect10.center = ((windowWidth/2), ((windowHeight/4)+60))
    displaySurf.fill(bgColour)
    displaySurf.blit(textSurf3, textRect3)
    displaySurf.blit(textSurf10, textRect10)
    pygame.display.update()
    while 1:
      event = pygame.event.poll()
      if event.type == KEYDOWN:
        if event.unicode == '\r': break
        usersolution3 += event.unicode
        if event.key == K_BACKSPACE:
          usersolution3 = usersolution3[0:-2]
        textSurf8, textRect8 = text_objects(usersolution3,smallText)
        textRect8.center = ((windowWidth/2), (windowHeight/2))
        displaySurf.fill(bgColour)
        displaySurf.blit(textSurf3, textRect3)
        displaySurf.blit(textSurf10, textRect10)
        displaySurf.blit(textSurf8, textRect8)
        pygame.display.update()

    textSurf11, textRect11 = text_objects("Solution four: ",smallText)
    textRect11.center = ((windowWidth/2), ((windowHeight/4)+60))
    displaySurf.fill(bgColour)
    displaySurf.blit(textSurf3, textRect3)
    displaySurf.blit(textSurf11, textRect11)
    pygame.display.update()
    while 1:
      event = pygame.event.poll()
      if event.type == KEYDOWN:
        if event.unicode == '\r': break
        usersolution4 += event.unicode
        if event.key == K_BACKSPACE:
          usersolution4 = usersolution4[0:-2]
        textSurf8, textRect8 = text_objects(usersolution4,smallText)
        textRect8.center = ((windowWidth/2), (windowHeight/2))
        displaySurf.fill(bgColour)
        displaySurf.blit(textSurf3, textRect3)
        displaySurf.blit(textSurf11, textRect11)
        displaySurf.blit(textSurf8, textRect8)
        pygame.display.update()
    while attempts > 1:#checks if the answers were correct, order which answers were given does not
matter.
      if (usersolution1 == str(x1) and usersolution2 == str(x1_2) and usersolution3 == str(x2) and
usersolution4 == str(x2_2))\
        or (usersolution1 == str(x1) and usersolution2 == str(x1_2) and usersolution3 == str(x2_2) and
usersolution4 == str(x2))\
            or (usersolution1 == str(x1) and usersolution2 == str(x2) and usersolution3 == str(x1_2) and 
usersolution4 == str(x2_2))\
            or (usersolution1 == str(x1) and usersolution2 == str(x2) and usersolution3 == str(x2_2) and 
usersolution4 == str(x1_2))\
            or (usersolution1 == str(x1) and usersolution2 == str(x2_2) and usersolution3 == str(x1_2) and 
usersolution4 == str(x2))\
            or (usersolution1 == str(x1) and usersolution2 == str(x2_2) and usersolution3 == str(x2) and 
usersolution4 == str(x1_2))\
            or (usersolution1 == str(x1_2) and usersolution2 == str(x1) and usersolution3 == str(x2) and 
usersolution4 == str(x2_2))\
            or (usersolution1 == str(x1_2) and usersolution2 == str(x1) and usersolution3 == str(x2_2) and 
usersolution4 == str(x2))\
            or (usersolution1 == str(x1_2) and usersolution2 == str(x2) and usersolution3 == str(x1) and 
usersolution4 == str(x2_2))\
            or (usersolution1 == str(x1_2) and usersolution2 == str(x2) and usersolution3 == str(x2_2) and 
usersolution4 == str(x1))\
            or (usersolution1 == str(x1_2) and usersolution2 == str(x2_2) and usersolution3 == str(x1) and 
usersolution4 == str(x2))\
            or (usersolution1 == str(x1_2) and usersolution2 == str(x2_2) and usersolution3 == str(x2) and 
usersolution4 == str(x1))\
            or (usersolution1 == str(x2) and usersolution2 == str(x1) and usersolution3 == str(x1_2) and 
usersolution4 == str(x2_2))\
            or (usersolution1 == str(x2) and usersolution2 == str(x1) and usersolution3 == str(x2_2) and 
usersolution4 == str(x1_2))\
            or (usersolution1 == str(x2) and usersolution2 == str(x1_2) and usersolution3 == str(x1) and 
usersolution4 == str(x2_2))\
            or (usersolution1 == str(x2) and usersolution2 == str(x1_2) and usersolution3 == str(x2_2) and 
usersolution4 == str(x1))\
            or (usersolution1 == str(x2) and usersolution2 == str(x2_2) and usersolution3 == str(x1) and 
usersolution4 == str(x1_2))\
            or (usersolution1 == str(x2) and usersolution2 == str(x2_2) and usersolution3 == str(x1_2) and 
usersolution4 == str(x1))\
            or (usersolution1 == str(x2_2) and usersolution2 == str(x1) and usersolution3 == str(x1_2) and 
usersolution4 == str(x2))\
            or (usersolution1 == str(x2_2) and usersolution2 == str(x1) and usersolution3 == str(x2) and 
usersolution4 == str(x1_2))\
            or (usersolution1 == str(x2_2) and usersolution2 == str(x1_2) and usersolution3 == str(x1) and 
usersolution4 == str(x2))\
            or (usersolution1 == str(x2_2) and usersolution2 == str(x1_2) and usersolution3 == str(x2) and 
usersolution4 == str(x1))\
            or (usersolution1 == str(x2_2) and usersolution2 == str(x2) and usersolution3 == str(x1) and 
usersolution4 == str(x1_2))\
            or (usersolution1 == str(x2_2) and usersolution2 == str(x2) and usersolution3 == str(x1_2) and 
usersolution4 == str(x1)):
                UserAnswer = "Correct"
                attempts = 0
          else: #gets answers again if they were incorrect.
            attempts = attempts - 1
```

```
        textSurf7, textRect7 = text_objects("Incorrect solutions, "+str(attempts)+" attempts remaining. Solution one: ",smallText)
        textRect7.center = ((windowWidth/2), ((windowHeight/4)+60))
        displaySurf.fill(bgColour)
        displaySurf.blit(textSurf3, textRect3)
        displaySurf.blit(textSurf7, textRect7)
        pygame.display.update()
        while 1:
          event = pygame.event.poll()
          if event.type == KEYDOWN:
            if event.unicode == '\r': break
            usersolution1 += event.unicode
            if event.key == K_BACKSPACE:
              usersolution1 = usersolution1[0:-2]
            textSurf8, textRect8 = text_objects(usersolution1,smallText)
            textRect8.center = ((windowWidth/2), (windowHeight/2))
            displaySurf.fill(bgColour)
            displaySurf.blit(textSurf3, textRect3)
            displaySurf.blit(textSurf7, textRect7)
            displaySurf.blit(textSurf8, textRect8)
            pygame.display.update()
        textSurf9, textRect9 = text_objects("Solution two: ",smallText)
        textRect9.center = ((windowWidth/2), ((windowHeight/4)+60))
        displaySurf.fill(bgColour)
        displaySurf.blit(textSurf3, textRect3)
        displaySurf.blit(textSurf9, textRect9)
        pygame.display.update()
        while 1:
          event = pygame.event.poll()
          if event.type == KEYDOWN:
            if event.unicode == '\r': break
            usersolution2 += event.unicode
            if event.key == K_BACKSPACE:
              usersolution2 = usersolution2[0:-2]
            textSurf8, textRect8 = text_objects(usersolution2,smallText)
            textRect8.center = ((windowWidth/2), (windowHeight/2))
            displaySurf.fill(bgColour)
            displaySurf.blit(textSurf3, textRect3)
            displaySurf.blit(textSurf9, textRect9)
            displaySurf.blit(textSurf8, textRect8)
            pygame.display.update()
        textSurf10, textRect10 = text_objects("Solution three: ",smallText)
        textRect10.center = ((windowWidth/2), ((windowHeight/4)+60))
        displaySurf.fill(bgColour)
        displaySurf.blit(textSurf3, textRect3)
        displaySurf.blit(textSurf10, textRect10)
        pygame.display.update()
        while 1:
```

```
                    event = pygame.event.poll()
                    if event.type == KEYDOWN:
                        if event.unicode == '\r': break
                        usersolution3 += event.unicode
                        if event.key == K_BACKSPACE:
                            usersolution3 = usersolution2[0:-2]
                        textSurf8, textRect8 = text_objects(usersolution3,smallText)
                        textRect8.center = ((windowWidth/2), (windowHeight/2))
                        displaySurf.fill(bgColour)
                        displaySurf.blit(textSurf3, textRect3)
                        displaySurf.blit(textSurf10, textRect10)
                        displaySurf.blit(textSurf8, textRect8)
                        pygame.display.update()
                textSurf11, textRect11 = text_objects("Solution four: ",smallText)
                textRect11.center = ((windowWidth/2), ((windowHeight/4)+60))
                displaySurf.fill(bgColour)
                displaySurf.blit(textSurf3, textRect3)
                displaySurf.blit(textSurf11, textRect11)
                pygame.display.update()
                while 1:
                    event = pygame.event.poll()
                    if event.type == KEYDOWN:
                        if event.unicode == '\r': break
                        usersolution4 += event.unicode
                        if event.key == K_BACKSPACE:
                            usersolution4 = usersolution2[0:-2]
                        textSurf8, textRect8 = text_objects(usersolution2,smallText)
                        textRect8.center = ((windowWidth/2), (windowHeight/2))
                        displaySurf.fill(bgColour)
                        displaySurf.blit(textSurf3, textRect3)
                        displaySurf.blit(textSurf11, textRect11)
                        displaySurf.blit(textSurf8, textRect8)
                        pygame.display.update()
    return UserAnswer


def Question2():
    global UserAnswer
    #gets numbers for equation
    A = random.randint(-20,20)
    B = random.randint(-20,20)
    C = random.randint(-20,20)
    D = random.randint(-20,20)
    E = random.randint(-20,20)
    T = "t"

    while D == 0 or C == 0 or (((int(A)-int(B))/int(C))-1) < 0: # some numbers cannot equal 0, or the
program will crash.
```

```
    A = random.randint(-20,20)
    B = random.randint(-20,20)
    C = random.randint(-20,20)
    D = random.randint(-20,20)
    E = random.randint(-20,20)

if C >= 0:
    C = "+",str(C)
    C = str(C)[2] + str(C)[7]
if E <= 0:
    E = "-"
else:
    E = "+"

if E == "-": # formats the equation so it displays on the screen correctly.
    equation = str(B)+str(C)+"e"+"^("+str(E)+str(T)+"/"+str(D)+")"+" = "+str(A)
    if B == 0:
        equation = str(C)+"e"+"^("+str(E)+str(T)+"/"+str(D)+")"+" = "+str(A)
    if C == "+1":
        equation = str(B)+"+"+"e"+"^("+str(E)+str(T)+"/"+str(D)+")"+" = "+str(A)
    if C == -1:
        equation = str(B)+"-"+"e"+"^("+str(E)+str(T)+"/"+str(D)+")"+" = "+str(A)
    if C == "+1" and B == 0:
        equation = "e"+"^("+str(E)+str(T)+"/"+str(D)+")"+" = "+str(A)
    if C == -1 and B == 0:
        equation = "-"+"e"+"^("+str(E)+str(T)+"/"+str(D)+")"+" = "+str(A)

if E == "+":# formats the equation so it displays on the screen correctly.
    equation = str(B)+str(C)+"e"+"^("+str(T)+"/"+str(D)+")"+" = "+str(A)
    if B == 0:
        equation = str(C)+"e"+"^("+str(T)+"/"+str(D)+")"+" = "+str(A)
    if C == "+1":
        equation = str(B)+"+"+"e"+"^("+str(T)+"/"+str(D)+")"+" = "+str(A)
    if C == -1:
        equation = str(B)+"-"+"e"+"^("+str(T)+"/"+str(D)+")"+" = "+str(A)
    if C == "+1" and B == 0:
        equation = "e"+"^("+str(T)+"/"+str(D)+")"+" = "+str(A)
    if C == -1 and B == 0:
        equation = "-"+"e"+"^("+str(T)+"/"+str(D)+")"+" = "+str(A)

question = str("Find the value of 't' for the equation: "+equation)
textSurf3, textRect3 = text_objects(question,smallText)
textRect3.center = ((windowWidth/2),(windowHeight/4))
displaySurf.fill(bgColour)
displaySurf.blit(textSurf3, textRect3)
pygame.display.update()
if E == "-":#finds answers to the question asked
    x1 = -math.log1p(((int(A)-int(B))/int(C))-1)*int(D)
    elif E == "+":#finds answers to the question asked
        x1 = math.log1p(((int(A)-int(B))/int(C))-1)*int(D)

    x1 = format(float(x1),'.2f') # formats answers to two decimal places
    if str(x1[0]) == "-" and str(x1[1]) == "0":
        x1 = 0
    attempts = 5
    usersolution = "" # resets user solutions from previous asked questions
    textSurf7, textRect7 = text_objects("Solution:",smallText)
    textRect7.center = ((windowWidth/2), ((windowHeight/4)+60))
    displaySurf.fill(bgColour)
    displaySurf.blit(textSurf3, textRect3)
    displaySurf.blit(textSurf7, textRect7)
    pygame.display.update()
    while 1: # gets user answer
        event = pygame.event.poll()
        if event.type == KEYDOWN:
            if event.unicode == '\r': break
            usersolution += event.unicode
            if event.key == K_BACKSPACE:
                usersolution = usersolution[0:-2]
            textSurf8, textRect8 = text_objects(usersolution,smallText)
            textRect8.center = ((windowWidth/2), (windowHeight/2))
            displaySurf.fill(bgColour)
            displaySurf.blit(textSurf3, textRect3)
            displaySurf.blit(textSurf7, textRect7)
            displaySurf.blit(textSurf8, textRect8)
            pygame.display.update()
    while usersolution != str(x1) and attempts > 1: # gets another answer if the user was wrong
        attempts = attempts - 1
        usersolution = ""
        textSurf7, textRect7 = text_objects("Incorrect solution, "+str(attempts)+" attempts remaining.
Solution: ",smallText)
        textRect7.center = ((windowWidth/2), ((windowHeight/4)+60))
        displaySurf.fill(bgColour)
        displaySurf.blit(textSurf3, textRect3)
        displaySurf.blit(textSurf7, textRect7)
        pygame.display.update()
        while 1:
            event = pygame.event.poll()
            if event.type == KEYDOWN:
                if event.unicode == '\r': break
                usersolution += event.unicode
                if event.key == K_BACKSPACE:
                    usersolution = usersolution[0:-2]
                textSurf8, textRect8 = text_objects(usersolution,smallText)
                textRect8.center = ((windowWidth/2), (windowHeight/2))
                displaySurf.fill(bgColour)
        displaySurf.blit(textSurf3, textRect3)
        displaySurf.blit(textSurf7, textRect7)
        displaySurf.blit(textSurf8, textRect8)
        pygame.display.update()
  if usersolution != str(x1):
    UserAnswer = "Incorrect"
    textSurf7, textRect7 = text_objects("Incorrect solution, 0 attempts remaining. Solution: 
",smallText)
    textRect7.center = ((windowWidth/2), ((windowHeight/4)+60))
    textSurf4, textRect4 = text_objects(UserAnswer,smallText)
    textRect4.center = ((windowWidth/2), (windowHeight/2))
    displaySurf.fill(bgColour)
    displaySurf.blit(textSurf3, textRect3)
    displaySurf.blit(textSurf4, textRect4)
    displaySurf.blit(textSurf7, textRect7)
    pygame.display.update()
    pygame.time.wait(2000)
  else:
    UserAnswer = "Correct"
  return UserAnswer

def Question3():
  global UserAnswer
  p=True
  while p == True:#gets numbers to form an equation.
    A = random.randint(-10,10)
    while A == 0:
      A = random.randint(-10,10)
    B = random.randint(1,10)
    while B == 0:
      B = random.randint(1,10)
    C = random.randint(-10,10)
    while C == 0:
      C = random.randint(-10,10)
    D = random.randint(1,10)
    while D == 0:
      D = random.randint(-10,10)
    E = random.randint(1,10)
    while E == 0 or E == 1:
      E = random.randint(-10,10)
    X = random.randint(-20,20)
    while X == 0 or X == -1 or X == 1:
      X = random.randint(-15,15)
    while (D == 1 and B == 1) or D == B:
      D = random.randint(-10,10)
      B = random.randint(-10,10)
    sol1 = 0.00001
    F = str((X)**(B-1))
    if F == "0":
      F = 0
    else:
      F = F[-1]
    G = str((C*D)*(X)**(D-1))
    if G == "0":
      G == 0
    else:
      G = G[-1]
    H = str(A*(X)**(B))
    if H == "0":
      H = 0
    else:
      H = H[-1]
    I = str(C*(X)**(D))
    if I == "0":
      I == 0
    else:
      I = I[-1]
    while A == 0 or B == 0 or C == 0 or D == 0 or E == 0 or E == 1 or X == 0 or X == 1 or X == -1 or (B == 1 and D == 1) or B == D or (A*X**(B)+C*X**(D)) == 0 or E*((A*B)*(X)**(B-1)+(C*D)*(X)**(D-1)) == 0 or F[-1] == "0" or G[-1] == "0" or H[-1] == "0" or I[-1] == "0": # this can crash the program if some numbers and in 0.
      A = random.randint(-10,10)
      while A == 0:
        A = random.randint(-10,10)
      B = random.randint(1,10)
      while B == 0:
        B = random.randint(1,10)
      C = random.randint(-10,10)
      while C == 0:
        C = random.randint(-10,10)
      D = random.randint(1,10)
      while D == 0:
        D = random.randint(-10,10)
      E = random.randint(1,10)
      while E == 0 or E == 1:
        E = random.randint(-10,10)
      X = random.randint(-15,15)
      while X == 0 or X == 1 or X == -1:
        X = random.randint(-15,15)
      while (B == 1 and D == 1) or B == D:
        B = random.randint(-10,10)
        D = random.randint(-10,10)
      F = str((X)**(B-1))
      if F == "0":
        F = 0
      else:
```

```
                F = F[-1]
            G = str((C*D)*(X)**(D-1))
            if G == "0":
                G == 0
            else:
                G = G[-1]
            H = str(A*(X)**(B))
            if H == "0":
                H = 0
            else:
                H = H[-1]
            I = str(C*(X)**(D))
            if I == "0":
                I == 0
            else:
                I = I[-1]

        sol1 = E*((A*B)*(X)**(B-1)+(C*D)*(X)**(D-1))*((A*(X)**(B))+(C*(X)**(D)))**(E-1) # uses chain rule
to find answer in terms of "x"
        if sol1 < -500 or sol1 > 500 or type(sol1) != int:
            p = True
        else: # formats equation so it displays on screen correctly.
            equation = "y="+"("+str(A)+"x^("+str(B)+")+"+str(C)+"x^("+str(D)+"))"+"^"+str(E)
            if C < 0:
                equation = "y = ("+str(A)+"x^("+str(B)+")"+str(C)+"x^("+str(D)+"))"+"^"+str(E)
            if C > 0:
                equation = "y = ("+str(A)+"x^("+str(B)+")+"+str(C)+"x^("+str(D)+"))"+"^"+str(E)
            if C == 1:
                equation = "y = ("+str(A)+"x^("+str(B)+")+"+"x^("+str(D)+"))"+"^"+str(E)
            if C == -1:
                equation = "y = ("+str(A)+"x^("+str(B)+")-"+"x^("+str(D)+"))"+"^"+str(E)
            if D == 1:
                equation = "y =("+str(A)+"x^("+str(B)+")+("+str(C)+"x)"+"^"+str(E)
            if A == 1 and C < 0:
                equation = "y = ("+"x^("+str(B)+")"+str(C)+"x^("+str(D)+"))"+"^"+str(E)
            if A == 1 and C > 0:
                equation = "y = ("+"x^("+str(B)+")+"+str(C)+"x^("+str(D)+"))"+"^"+str(E)
            if A == 1 and C == 1:
                equation = "y = ("+"x^("+str(B)+")+"+"x^("+str(D)+"))"+"^"+str(E)
            if A == 1 and C == -1:
                equation = "y = "+"("+"x^("+str(B)+")-"+"x^("+str(D)+"))"+"^"+str(E)
            if A == -1 and C < 0:
                equation = "y = (-x^("+str(B)+")"+str(C)+"x^("+str(D)+"))"+"^"+str(E)
            if A == -1 and C > 0:
                equation = "y = (-x^("+str(B)+")+"+str(C)+"x^("+str(D)+"))"+"^"+str(E)
            if A == -1 and C == 1:
                equation = "y = (-x^("+str(B)+")+"+"x^("+str(D)+"))"+"^"+str(E)
            if A == -1 and C == -1:
```

```
            equation = "y = (-x^("+str(B)+")-"+"x^("+str(D)+"))"+"^"+str(E)
        if B == 1 and C > 0:
            equation = "y = ("+str(A)+"x+"+str(C)+"x^("+str(D)+"))"+"^"+str(E)
        if B == 1 and C < 0:
            equation = "y = ("+str(A)+"x"+str(C)+"x^("+str(D)+"))"+"^"+str(E)
        if B == 1 and C == 1:
            equation = "y = ("+str(A)+"x+"+"x^("+str(D)+"))"+"^"+str(E)
        if B == 1 and C == -1:
            equation = "y = ("+str(A)+"x-"+"x^("+str(D)+"))"+"^"+str(E)
        if D == 1 and C > 0:
            equation = "y = ("+str(A)+"x^("+str(B)+")+"+str(C)+"x)"+"^"+str(E)
        if D == 1 and C < 0:
            equation = "y = ("+str(A)+"x^("+str(B)+")"+str(C)+"x)"+"^"+str(E)
        if D == 1 and C == 1:
            equation = "y = ("+str(A)+"x^("+str(B)+")+"+"x)"+"^"+str(E)
        if D == 1 and C == -1:
            equation = "y = ("+str(A)+"x^("+str(B)+")-"+"x)"+"^"+str(E)
        if D == 1 and A == 1:
            equation = "y = ("+"x^("+str(B)+")+("+str(C)+"x)"+"^"+str(E)
        if D == 1 and A == 1 and C > 0:
            equation = "y = ("+"x^("+str(B)+")+"+str(C)+"x)"+"^"+str(E)
        if D == 1 and A == 1 and C < 0:
            equation = "y = ("+"x^("+str(B)+")"+str(C)+"x)"+"^"+str(E)
        if D == 1 and A == 1 and C == -1:
            equation = "y = ("+"x^("+str(B)+")-x)"+"^"+str(E)
        if D == 1 and A == -1 and C > 0:
            equation = "y = (-x^("+str(B)+")+"+str(C)+"x)"+"^"+str(E)
        if D == 1 and A == -1 and C < 0:
            equation = "y = (-x^("+str(B)+")"+str(C)+"x)"+"^"+str(E)
        if D == 1 and A == -1 and C == -1:
            equation = "y = (-x^("+str(B)+")-x)"+"^"+str(E)
        if D == 1 and A == 1 and C == 1:
            equation = "y = (x^("+str(B)+")"+"+x)"+"^"+str(E)
        if D == 1 and C == 1 and A == -1:
            equation = "y = (-x^("+str(B)+")+x)"+"^"+str(E)
        if B == 1 and C > 0 and A == -1:
            equation = "y = (-x+"+str(C)+"x^("+str(D)+"))"+"^"+str(E)
        if B == 1 and C < 0 and A == -1:
            equation = "y = (-x"+str(C)+"x^("+str(D)+"))"+"^"+str(E)
        if B == 1 and C == 1 and A == -1:
            equation = "y = "+"(-x+x^("+str(D)+"))"+"^"+str(E)
        if B == 1 and C == -1 and A == -1:
            equation = "y = (-x-x^("+str(D)+"))"+"^"+str(E)
        if A == 1 and B == 1 and C > 0:
            equation = "y = (x+"+str(C)+"x^("+str(D)+"))"+"^"+str(E)
        question = str("Find the gradient on the curve: "+equation+", when x = "+str(X)) # displays
question on screen
        textSurf3, textRect3 = text_objects(question,smallText)
```

```
      textRect3.center = ((windowWidth/2),(windowHeight/4))
      displaySurf.fill(bgColour)
      displaySurf.blit(textSurf3, textRect3)
      pygame.display.update()
      p = False
  attempts = 5
  usersolution = ""
  textSurf7, textRect7 = text_objects("Solution:",smallText)
  textRect7.center = ((windowWidth/2), ((windowHeight/4)+60))
  displaySurf.fill(bgColour)
  displaySurf.blit(textSurf3, textRect3)
  displaySurf.blit(textSurf7, textRect7)
  pygame.display.update()
  while 1:
    event = pygame.event.poll()
    if event.type == KEYDOWN:
      if event.unicode == '\r': break
      usersolution += event.unicode
      if event.key == K_BACKSPACE:
        usersolution = usersolution[0:-2]
      textSurf8, textRect8 = text_objects(usersolution,smallText)
      textRect8.center = ((windowWidth/2), (windowHeight/2))
      displaySurf.fill(bgColour)
      displaySurf.blit(textSurf3, textRect3)
      displaySurf.blit(textSurf7, textRect7)
      displaySurf.blit(textSurf8, textRect8)
      pygame.display.update()
  while usersolution != str(sol1) and attempts > 1: # asks again if the user was incorrect
    attempts = attempts - 1
    usersolution = ""
    textSurf7, textRect7 = text_objects("Incorrect solution, "+str(attempts)+" attempts remaining.
Solution: ",smallText)
    textRect7.center = ((windowWidth/2), ((windowHeight/4)+60))
    displaySurf.fill(bgColour)
    displaySurf.blit(textSurf3, textRect3)
    displaySurf.blit(textSurf7, textRect7)
    pygame.display.update()
    while 1:
      event = pygame.event.poll()
      if event.type == KEYDOWN:
        if event.unicode == '\r': break
        usersolution += event.unicode
        if event.key == K_BACKSPACE:
          usersolution = usersolution[0:-2]
        textSurf8, textRect8 = text_objects(usersolution,smallText)
        textRect8.center = ((windowWidth/2), (windowHeight/2))
        displaySurf.fill(bgColour)
        displaySurf.blit(textSurf3, textRect3)
```

```
                displaySurf.blit(textSurf7, textRect7)
                displaySurf.blit(textSurf8, textRect8)
                pygame.display.update()

    if usersolution == str(sol1):
        UserAnswer = "Correct"
    else:
        UserAnswer = "Incorrect"
        textSurf7, textRect7 = text_objects("Incorrect solution, 0 attempts remaining. Solution:
",smallText)
        textRect7.center = ((windowWidth/2), ((windowHeight/4)+60))
        textSurf4, textRect4 = text_objects(UserAnswer,smallText)
        textRect4.center = ((windowWidth/2), (windowHeight/2))
        displaySurf.fill(bgColour)
        displaySurf.blit(textSurf3, textRect3)
        displaySurf.blit(textSurf4, textRect4)
        displaySurf.blit(textSurf7, textRect7)
        pygame.display.update()
        pygame.time.wait(2000)
    return UserAnswer

def mainmenu():
    menu = True
    #y coordinates for moving tokens
    redw1 = -200
    redw2 = -200
    redw3 = -200
    blackw1 = 776
    blackw2 = 776
    blackw3 = 776
    redw1, redw2, redw3, blackw1, blackw2, blackw3= menuCounters(redw1, redw2, redw3, blackw1,
blackw2, blackw3)
    while menu:
        for event in pygame.event.get(): # event handler
            if event.type == pygame.QUIT:
                pygame.quit()
                quit()
            if event.type == MOUSEBUTTONDOWN:
                if 150 + 100 > mouse[0] > 150 and 450 + 50 > mouse[1] >450:
                    main()
                if 724 + 100 > mouse[0] > 724 and 450 + 50 > mouse[1] > 450:
                    pygame.quit()
                    quit()

        displaySurf.fill(bgColour)

        if redw1 > 700 and redw2 > 700 and redw3 > 700: # resets pixel locations
            redw1 = -200
            redw2 = -200
            redw3 = -200
        else:
            displaySurf.blit(redTokenIMG, (120, redw1))
            if redw1 < 710:
                redw1 += 2
            if redw1 > 700:
                displaySurf.blit(redTokenIMG, (450, redw2))
                redw2 += 2
            if redw1 > 700 and redw2 > 700:
                displaySurf.blit(redTokenIMG, (900, redw3))
                redw3 += 2
        if blackw1 < -200 and blackw2 < -200 and blackw3 < -200:
            blackw1 = 776
            blackw2 = 776
            blackw3 = 776
            blackw4 = 776
        else:
            displaySurf.blit(blackTokenIMG, (220, blackw1))
            if blackw1 > -210:
                blackw1 += -2
            if blackw1 < -200:
                displaySurf.blit(blackTokenIMG, (530, blackw2))
                blackw2 += -2
            if blackw1 < -200 and blackw2 < -200:
                displaySurf.blit(blackTokenIMG, (800, blackw3))
                blackw3 += -2

        TextSurf, TextRect = text_objects("Connect 4",largeText)
        TextRect.center = ((windowWidth/2),(windowHeight/4))
        displaySurf.blit(TextSurf, TextRect)
        mouse = pygame.mouse.get_pos()
        if 150 + 100 > mouse[0] > 150 and 450 + 50 > mouse[1] >450:
            pygame.draw.rect(displaySurf, bright_green, (150,450,100,50))
        else:
            pygame.draw.rect(displaySurf, green, (150,450,100,50))
        if 724 + 100 > mouse[0] > 724 and 450 + 50 > mouse[1] > 450:
            pygame.draw.rect(displaySurf, bright_red, (724,450,100,50))
        else:
            pygame.draw.rect(displaySurf, red1, (724,450,100,50))

        textSurf, textRect = text_objects("Start",smallText)
        textRect.center = ((150 + (100/2)), (450 + (50/2)))
        textSurf1, textRect1 = text_objects("Quit",smallText)
        textRect1.center = ((724 + (100/2)), (450 + (50/2)))
        displaySurf.blit(textSurf, textRect)
        displaySurf.blit(textSurf1, textRect1)
        pygame.display.update()
```

```
def getQuestionType():
    number = random.randint(1,3) # gets a random number to decide the type of question
    if number == 1:
        Question1()
    elif number == 2:
        Question2()
    else:
        Question3()

if __name__ == '__main__':# starts the program.
    mainmenu()
```

Evaluation

Were the objectives met?

General objectives

Objective	Met	Comment
The system must improve the logical and quick thinking skills of students.	Yes	This objective is achieved, as the questions require the student to answer then multiple times, and the more practice at a topic the student has, they are likely to improve. Also, the student will have to work out how to win the game.
To offer students a way to improve their maths skills as well as playing a problem solving game.	Yes	The maths skills will be improves by answering questions. Problem solving skills will be improved by working out how to win the game.
The target user of the system will be someone studying A2 maths.	Yes	The questions in the game are based on topics from A2 maths text book, and exam paper questions.
The system should be easily understood by the target audience. The terminology used must be understood by the target audience. The language and grammar must be of a high standard.	Yes	The terminology used should be understood by someone studying A2 maths, as they are frequently used in lessons and other questions.
The system should be user friendly as otherwise the terminology used could make the questions difficult to understand. User friendly systems are also more likely to not frustrate the user.	Yes	The system only uses two buttons, and an area for the user to type their answer. This will make the program user friendly and easy to use.
The system must be able to run quickly and efficiently.	Yes	The system does not require a powerful system to use.
The navigation of the system must be clear. The user must be	Yes	The system navigates between windows by itself, after a question is answered, and when

able to navigate around the system with no problems.		the computer has had a turn on the game.

Specific objectives

Objective	Met	Comment
The system will be able to run in school, or on the student's personal computer.	Yes	The system can run on school computers, as long as the necessary components are installed, python and pygame.
The student will be able to complete a different set of questions each time, to stop them from being able to remember answers, and instead improve their maths skills.	Yes	The questions are randomly generated each time.
The system will have at least three topics from the A2 maths syllabus.	Yes	The system used three topics for the questions.
The system should have different options to change the size of the game.	No	This is possible to do easily in the source code of the program, however this is not currently available in the game window.
The student will be able to play the game against the computer, so it is not required to have two players.	Yes	The game is single player against the computer, as most students revise alone.
The computers first move will be different each time to stop it from generating the same group of moves each game.	Yes	The computers move is random each time the game starts.
The user will only get a turn on the game if they get a question correct.	Yes	The user must get their question correct, otherwise the computer will continue to have another turn.
The type of question that is asked to the user is different each time.	Yes	The questions are randomly generated each time.
The number of moves in which the student wins the game will be shown after completion of the game.	No	This is not shown, as the main point of the game is to be a revision tool and not about how quickly the user can win the game.

The number of questions that the user answers correctly in a row will be shown after completion of the game.	No	This could be implemented, but it is likely that the user will get all questions right once they learn the method.
The game instructions will be shown as the user is playing the game, therefore will not require a separate instructions page.	Yes	The help arrow is shown when the user has their first move on the game.
The user will be rewarded in game when they get a set number of questions correct, in a row.	No	Rewarding the user with extra turns made the game too easy to win.
The user will have the ability to change the difficulty of the computer (how many moves are calculated in advanced).	No	The difficulty is fixes, as the game will become too hard or too easy to win, should the difficulty be changed.

Feedback

All questions were answered by students studying A level maths at A High School.

Do you think the objectives will create a suitable program?

"Yes, the objectives will allow the program to have a range of topics and will help when revising."

Are there any features which would help improve the program?

"Having extra topics would help, as well as having the choice to choose the topics which you want to revise"

Do you think the program is easy to use and user friendly?

"Yes the program is very easy to use."

Did you run into any problems when using the program?

"No, I did not have any problems when using the game."

Possible extensions

The system could have all the above objectives that were not met. With the exception of the objectives which were purposely not met. In addition to this, there could be a log in system, where the user can track their progress, or save a game that did not finish.

Other subjects could also be implemented into the game, as the input can be altered to accept different data types. For subjects where questions cannot be randomly generated, it would also be possible so use a database to store a range of questions, and then randomly choose one from there.

A timer could also be implemented, to see how quickly a user could answer a question. This could also encourage a user to try and answer the questions quickly, to beat their fastest time.