By Michel Salazar-Garcia

Monroe community college – cpt 100-101 spring 2016

Prof. Les dudek

tic tac toe – a simple implementation in python 3.5 with tkinter

Table of Contents

[name\_entry class 2](#_Toc450649898)

[name\_entry class: UML diagram 2](#_Toc450649899)

[name\_entry class: IPO charts 3](#_Toc450649900)

[name\_entry class: Flowchart 4](#_Toc450649901)

[name\_entry class: Pseudocode 5](#_Toc450649902)

[\_\_init\_\_ function: 5](#_Toc450649903)

[\_\_get\_name function: 6](#_Toc450649904)

[get\_name function: 6](#_Toc450649905)

[window function: 6](#_Toc450649906)

[square class 7](#_Toc450649907)

[square class: UML diagram 7](#_Toc450649908)

[square class: IPO charts 8](#_Toc450649909)

[square class: Flowchart 10](#_Toc450649910)

[square class: Pseudocode 11](#_Toc450649911)

[\_\_init\_\_ function: 11](#_Toc450649912)

[right\_click function: 11](#_Toc450649913)

[get\_text function: 11](#_Toc450649914)

[mark\_o function: 11](#_Toc450649915)

[id function: 11](#_Toc450649916)

[get\_mark\_count function: 11](#_Toc450649917)

[clear function: 11](#_Toc450649918)

[boaster class 12](#_Toc450649919)

[boaster class: UML diagram 12](#_Toc450649920)

[boaster class: IPO charts 13](#_Toc450649921)

[boaster class: Flowchart 14](#_Toc450649922)

[boaster class: Pseudocode 15](#_Toc450649923)

[\_\_init\_\_ function: 15](#_Toc450649924)

[Change\_color function: 15](#_Toc450649925)

[board class 16](#_Toc450649926)

[board class: UML diagram 16](#_Toc450649927)

[board class IPO charts 17](#_Toc450649928)

[board class: Flowchart 21](#_Toc450649929)

[board class: Pseudocode 24](#_Toc450649930)

[\_\_init\_\_ function: 24](#_Toc450649931)

[show\_history function: 25](#_Toc450649932)

[square\_clicked function: 25](#_Toc450649933)

[out\_of\_board function: 26](#_Toc450649934)

[already\_marked function: 26](#_Toc450649935)

[check\_win function: 26](#_Toc450649936)

[display\_result function: 26](#_Toc450649937)

[check\_marks function: 27](#_Toc450649938)

[search\_completing function: 27](#_Toc450649939)

[search\_corners function: 27](#_Toc450649940)

[search\_side function: 28](#_Toc450649941)

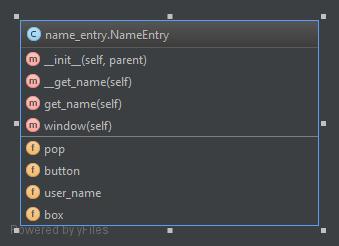
[board\_full function: 28](#_Toc450649942)

[new\_game function: 28](#_Toc450649943)

[confirm\_quit function: 28](#_Toc450649944)

# name\_entry class

## name\_entry class: UML diagram



## name\_entry class: IPO charts

|  |  |  |
| --- | --- | --- |
| \_\_init\_\_ | | |
| Input | Processing | Output |
| Self  Parent window | Initializes the main characteristics of the name\_entry object: mandatory focus and relative location (foreground), capability to deactivate parent window, binding to the function that processes the entered input (user name), etc. | None |

|  |  |  |
| --- | --- | --- |
| \_\_get\_name | | |
| Input | Processing | Output |
| Self  Caller event | Checks if anything was entered and if so, rejects it if it’s longer than 10 characters. | The entered name or “USER” if nothing was entered. |

|  |  |  |
| --- | --- | --- |
| get\_name | | |
| Input | Processing | Output |
| Self | Gets the entered user name. | The processes user name. |

|  |  |  |
| --- | --- | --- |
| window | | |
| Input | Processing | Output |
| Self | Gets the name\_entry object (tkinter pop object). | The name\_object. (This is needed for the wait\_window method in the parent window (board object). |

## name\_entry class: Flowchart

Start

NO

Set the user name to “USER”

YES

Is there something in the entry box?

Initialize main characteristic

User clicks Submit or closes the window

NO

Is the entered string > 10 characters?

Destroy the window

Stop

Set the user name to the entered string

YES

Tell user to enter something < 10 characters and erase the entry box

## name\_entry class: Pseudocode

### \_\_init\_\_ function:

Initialize the toplevel object (pop window) with the parent (board) as the master argument.

Set the size of the windows to 220 x 400 and located it at 400 and 500 from the screen edges.

Make it non resizable in width or height.

Set the title to “Name”.

Set it to be displayed on top of all other windows.

Lock the focus to the window.

Give it a “transient” style over the parent window. (Ie., do not show it in the taskbar).

Set the window to capture that Enter key pressed and then trigger the \_\_get\_name to handle what happens next.

Capture when the windows is closed and let the function \_\_get\_name handle what happens next.

Initialize a top frame on the pop window.

Initialize a text entry box on the top frame with border width to 2.

Initialize a label on the top frame showing “Enter your name: “

Set the focus on the text entry box.

Initialize a button on the pop window with text “Submit” and \_\_get\_name as the function to handle the click event.

Set the user.name attribute of the class to an empty string.

Pack to top frame on the top of the pop window.

Pack the label on the left of the top frame.

Pack the entry box on the left of the top frame.

Pack the button on the bottom of the pop window.

### \_\_get\_name function:

Check if entry box is empty. If it is set the user\_name attribute of the class instance to “USER” and destroy the pop window.

If the entry box content is longer than 10 characters, tell the user and erase the entry box.

If the entry box content is no longer than 10 characters, set the user\_name attribute of the class instance to the entered text and destroy the pop window.

### get\_name function:

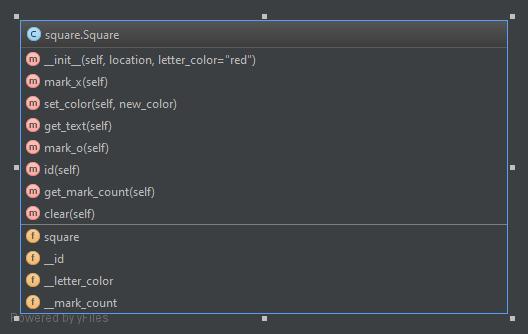
Return the user\_name attribute of the class instance.

### window function:

Return the instance of the tkinter.Toplevel object.

# square class

## square class: UML diagram



## square class: IPO charts

|  |  |  |
| --- | --- | --- |
| \_\_init\_\_ | | |
| Input | Processing | Output |
| Self  The name of the frame the square will be positioned on. | Initializes the main characteristics of the square object: location, binding to right click event, times that has been clicked, id, etc. | None |

|  |  |  |
| --- | --- | --- |
| mark\_x | | |
| Input | Processing | Output |
| Self | Set a blue “X” and increments the mark count. | None |

|  |  |  |
| --- | --- | --- |
| set\_color | | |
| Input | Processing | Output |
| Self  New color for the machine’s marks. | Changes the color of the machine marks (“O”). | None |

|  |  |  |
| --- | --- | --- |
| get\_text | | |
| Input | Processing | Output |
| Self  None | Gets the current text of the square, if any. | The current text of the square. |

|  |  |  |
| --- | --- | --- |
| mark\_o | | |
| Input | Processing | Output |
| Self  None | Marks the square with a red “O” and increments the click count. | The current text of the square. |

|  |  |  |
| --- | --- | --- |
| id | | |
| Input | Processing | Output |
| Self  None | Gets the ID of the square. | The ID of the square object. |

|  |  |  |
| --- | --- | --- |
| get\_mark\_count | | |
| Input | Processing | Output |
| Self  None | Gets how many times the square has been clicked. | How many time the square has been clicked. |

|  |  |  |
| --- | --- | --- |
| clear | | |
| Input | Processing | Output |
| Self  None | Erases the square’s text and resets the click count to 0. | None |

## square class: Flowchart

Start

Initialize main characteristic of the instance and wait for functions to be accessed from outside.

Stop

## square class: Pseudocode

### \_\_init\_\_ function:

Initialize the tkinter.Label object located in the frame specified by the passed location argument, with border width 2, relief solid, no text, height of 2, width of 4 and font Tahoma 62.

Set the \_\_location attribute of the class instance to the specified frame in the location argument.

Bind the right click event to the righ\_click method of the instance

pack the instance in the left of the frame passed by location.

Set the \_\_mark\_count attribute of the class instance to 0.

Set the \_\_id attribute of the class instance to a formatted string (format < .54557904.54558128>).

Set the letter.color attribute of the class instance to the passed argument.

### mark\_x function:

Set the text of the square to a blue “X”.

Increase the \_\_mark\_count attribute by 1.

### set\_color function:

Sets the color of the text to the argument passed in new\_color.

### get\_text function:

Return the current text of the square.

### mark\_o function:

Set the text of the square to an “O” the color of the \_\_letter\_color attribute of the instance.

Increase the \_\_mark\_count attribute by 1.

### id function:

Return the \_\_id attribute of the square.

### get\_mark\_count function:

Return the \_\_mark\_count attribute of the square.

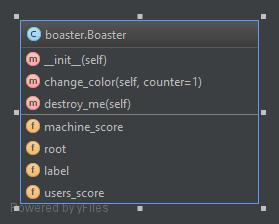
### clear function:

Set the text of the square to nothing.

Reset the \_\_mark\_count attribute to 0.

# boaster class

## boaster class: UML diagram



## boaster class: IPO charts

|  |  |  |
| --- | --- | --- |
| \_\_init\_\_ | | |
| Input | Processing | Output |
| Self  The name of the frame the square will be positioned on. | Initializes the main characteristics of the boaster object: size, tkinter subjects, font, not resizable, etc. Sets the object to autodestroy after 15 seconds of appearing. | None |

|  |  |  |
| --- | --- | --- |
| change\_color | | |
| Input | Processing | Output |
| Self  Counter | Changes the color of the main label randomly every 0.3 seconds for 50 times (15 seconds total). | None |

|  |  |  |
| --- | --- | --- |
| Destroy\_me | | |
| Input | Processing | Output |
| Self | Destroy the object instance. (Closes the windows by itself) | None |

## boaster class: Flowchart

Start

Is there an IOError?

Change the color of the main label randomly

Wait 0.3 seconds

NO

Stop

Close the window

Has the function iterated < 50 times?

NO

YES

Open the stored historic score and extract it

Initialize main characteristics of the instance. (Autodestroy itself in 15 seconds)

## boaster class: Pseudocode

### \_\_init\_\_ function:

Initialize the tkinter.Tk object.

Set the object to not resizable.

Set the windows title to “Historic Score”.

Set the dimensions and position of the window.

Initialize the “Historic Score” label, located in the window, set the text and set the font properties.

Initialize the users\_score label, located in the window, set the text and set the font properties.

Initialize the machine\_score label, located in the window, set the text and set the font properties.

Pack the “Historic Score” label on the top of the main window.

Pack the user\_score label.

Pack the machine\_score label.

Set the main window to autodestroy after 15 seconds.

Call the function to change the color of the “Historic Score” label.

### Change\_color function:

Mark the beginning of the error handled block.

Open the file containing the historic score for reading.

Unpickle the file.

Close the file.

If there was an IO error, return.

Set the font color of the “Historic Score” label to a random choice out or 6 colors.

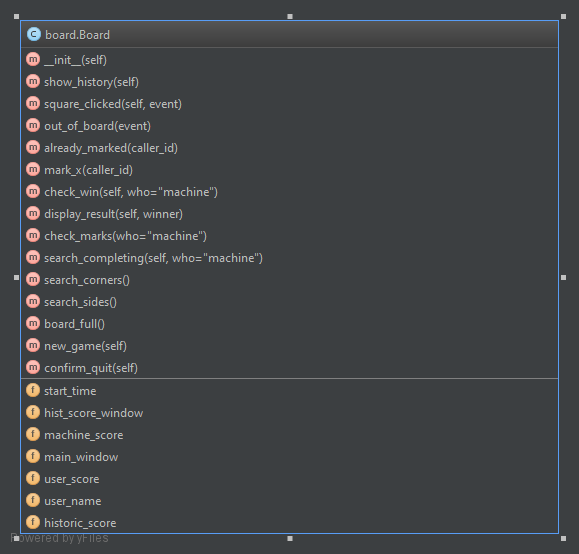
Add the users’ score to the users’ score label.’

Add the machine’s score the machine score label.

After 3 seconds call the same function recursively.

# board class

## board class: UML diagram



## board class IPO charts

|  |  |  |
| --- | --- | --- |
| \_\_init\_\_ | | |
| Input | Processing | Output |
| Self | Initializes the main characteristics of the board instance and its objects (frames, labels, squares instances, etc). Set the function that will handle the window closure.  Binds the right click to the square\_clicked function.  Initializes all 9 squares as instances of the square class.  Instantiates once the user\_name class and get the user name, if any is given.  This is the function that contains the only tkinter.mainloop() in the program. | None |

|  |  |  |
| --- | --- | --- |
| show\_history | | |
| Input | Processing | Output |
| Self | Instantiates the boaster object and assigns to the a handler variable. | None. |

|  |  |  |
| --- | --- | --- |
| square\_clicked | | |
| Input | Processing | Output |
| Self  Caller event | Checks if the click was outside any square and returns immediately if so.  Drives the program execution and main game algorithm after each user click. | None. |

|  |  |  |
| --- | --- | --- |
| out\_of\_board | | |
| Input | Processing | Output |
| Caller event | Checks if the click was outside any square. | True if the click was capture by a non-square object. False if the click was capture by a square. |

|  |  |  |
| --- | --- | --- |
| already\_marked | | |
| Input | Processing | Output |
| Self  ID of the widget that captured the event (mouse left click). | Checks if the clicked square is already marked by the user or machine. | True if the clicked square is already marked.  False if the clicked square is not marked. |

|  |  |  |
| --- | --- | --- |
| mark\_x | | |
| Input | Processing | Output |
| Self  ID of the widget that captured the event (mouse left click). | Marks a blue “X” in the clicked square. | Nothing. |

|  |  |  |
| --- | --- | --- |
| check\_win | | |
| Input | Processing | Output |
| Self  Party to check if won (“user” or “machine”). | Checks if the passed party has just won. | True if the specified party just won.  False if the specified party just won. |

|  |  |  |
| --- | --- | --- |
| display\_result | | |
| Input | Processing | Output |
| self  The party that just won (“user or “machine”). | Displays a system information message with the result of the game. | None. |

|  |  |  |
| --- | --- | --- |
| check\_marks | | |
| Input | Processing | Output |
| Party to check which squares are marked for (“user” or “machine”). | Gets a list with the number of the squares that the party specified by “who” has already marked. | The gathered list of square(s). |

|  |  |  |
| --- | --- | --- |
| search\_completing | | |
| Input | Processing | Output |
| Self  Party to check if it’s about to complete a line. | Searches for lines that only need one square marked to complete for the specified party. | The number of the first found square that can complete a line for the specified party, or “nothing found” if none found. |

|  |  |  |
| --- | --- | --- |
| search\_corners | | |
| Input | Processing | Output |
| None. | Searches for free corners. If more than 1, picks one randomly. | The selected free corner, or “nothing found” if none found. |

|  |  |  |
| --- | --- | --- |
| search\_sides | | |
| Input | Processing | Output |
| None. | Searches for free sides. If more than one, picks one randomly. | The selected free side, or “nothing found” if none found |

|  |  |  |
| --- | --- | --- |
| board\_full | | |
| Input | Processing | Output |
| None. | Checks if the board is full. | True if the board is full. False if it’s not. |

|  |  |  |
| --- | --- | --- |
| new\_game | | |
| Input | Processing | Output |
| Self | Decides who starts the next game. Clears the board and resets the clicked counter for each. | None. |

|  |  |  |
| --- | --- | --- |
| confirm\_quit | | |
| Input | Processing | Output |
| Self | Displays the exit confirmation box. If exit is confirmed, saves the result to a file and destroy the game. | None. |

## board class: Flowchart

Initialize main characteristics of the instance. Draw the board, instantiate other objects. Make the board react to a right click.

Start

Show the historic score and close the windows after 15 seconds

Ask to enter a name < 10 characters

Ask for the user’s name

YES

ee

Is the entered string > 10 characters?

Did the user enter anything?

YES

ee

NO

NO

Set the user’s name to the entered string

Set the user’s name to “USER”

YES

Is the square already marked?

Wait for a right click and mark it with a blue X

Was the right click on top of a square?

NO

Is the board full?

NO

YES

YES

Announce that it’s a match

Play a sound

Clear the board

NO

Did the user win?

YES

Increase the user’s score count

Announce that user won

Increase the machine’s score count

Change the border color of the window to red

Mark it with is red O

NO

YES

Clear the board

Change the border color of the window to blue

Is there a square that completes a line for the machine?

Did the machine start this game?

YES

Machine starts next

User starts next

Did the machine win?

NO

Is there a square that completes a line for the user?

NO

YES

YES

Announce that machine won

NO

Is there a free corner?

YES

Change the border color of the window to red

Clear the board

After the user acknowledges the winner message, change the border color of the window back to the system default

Did the machine start this game?

NO

Is the center free?

YES

NO

YES

NO

Is there a free side?

User starts next

YES

Machine starts next

NO

YES

After the user acknowledges the winner message, change the border color of the window back to the system default

Clear the board

## board class: Pseudocode

### \_\_init\_\_ function:

Store the application start time.

Initialize the main window tkinter.Tk object.

Set the main windows object to not resizable.

Set the main windows title.

Set the main windows size and position.

Bind the main window to execute the confirm\_quit function on mouse right click event.

Declare variable for the upcoming boaster object.

Set the main windows to instantiate call the function that instantiates the historic score announcement 3 seconds after being opened.

Initialize the “instructions” frame on the main window.

Initialize the user’s name label and set its main characteristics.

Create the StringVar object that will handle the user score.

Set the StringVar object to 0.

Initialize the user’s score label, set its main characteristics and assign the StringVar object to its “textvariable” property.

Initialize the historic score list.

Initialize the buffer variable (keeps the “Welcome to Tic Tac Toe” label somewhat centered).

Initialize the left buffer label inside the “instructions” frame and assign the variable buffer as its text.

Initialize the right buffer label inside the “instructions” frame and assign the variable buffer as its text.

Initialize the “machine” label and set its main characteristics.

Create the StringVar object that will handle the user score.

Set the StringVar object to 0.

Initialize the machine’s score label, set its main characteristics and assign the StringVar object to its “textvariable” property.

Initialize the top frame in inside the main window.

Initialize the middle frame in inside the main window.

Initialize the bottom frame in inside the main window.

Pick a random color out of the colors list for the machine marks.

Instantiate square number 4 located in the top frame, pass it the chosen color and store it in the squares dictionary.

Instantiate square number 9 located in the top frame, pass it the chosen color and store it in the squares dictionary.

Instantiate square number 2 located in the top frame, pass it the chosen color and store it in the squares dictionary.

Instantiate square number 3 located in the middle frame, pass it the chosen color and store it in the squares dictionary.

Instantiate square number 5 located in the middle frame, pass it the chosen color and store it in the squares dictionary.

Instantiate square number 7 located in the middle frame, pass it the chosen color and store it in the squares dictionary.

Instantiate square number 8 located in the bottom frame, pass it the chosen color and store it in the squares dictionary.

Instantiate square number 1 located in the bottom frame, pass it the chosen color and store it in the squares dictionary.

Instantiate square number 6 located in the bottom frame, pass it the chosen color and store it in the squares dictionary.

Pack the “instructions” frame (on the top of the main window).

Pack the user’s name label (on the left of the “instructions” frame).

Pack the user’s score label (to the left of the “instructions” frame).

Pack the left buffer label (to the left of the “instructions” frame).

Pack the “welcome” label (to the left of the “instructions” frame).

Pack the right buffer label (to the left of the “instructions” frame).

Pack the “machine” label (to the left of the “instructions” frame).

Pack the machine’s score label (to the left of the “instructions” frame).

Pack the top frame (to the top of the main window).

Pack the middle frame (to the top of the main window).

Pack the bottom frame (to the top of the main window).

Instantiate a NameEntry class object.

Set the main window to wait for the NameEntry object to be closed to activate.

Get the user’s name captured by the NameEntry object.

Set the user’s name label to the retrieved user’s name.

Resize the left buffer depending on the user’s name length.

Resize the right buffer depending on the user’s name length.

Enter the tkinter mainloop.

### show\_history function:

Instantiate the Boaster class object and assign it to a handler variable.

### square\_clicked function:

Call the out\_of\_board function to check if the righ click was out of a square. If yes, return.

Call the already\_marked function passing the event object as an argument to check if the clicked square is already marked. If so, return.

Call the mark\_x function to mark a blue “X” in the clicked square.

Call the check\_win function passing “user” as the argument to check if the user just won. If so, return.

Call the search\_completing function passing “machine” as the argument to search for line-completing squares for the machine. If one is found, call the mark\_o method for that square number to mark it with a red “O” (the machine mark).

Call the search\_completing function passing “user” as the argument to search for line-completing squares for the user. If one is found, call the mark\_o method for that square number to mark it with a red “O” (the machine mark).

Call the search\_corners function to search for free corners. If one is found, call the mark\_o method for that square number to mark it with a red “O” (the machine mark).

Call the get\_text method for the center square to check if it’s free. If it is, call the mark\_o method for that square number to mark it with a red “O” (the machine mark).

Call the search\_sides function to search for free sides. If one is found, call the mark\_o method for that square number to mark it with a red “O” (the machine mark).

Call the check\_win function passing “machine” as the argument to check if the machine just won. If not, call the board\_full function to check if the board is full. If so, call the display\_result function to announce that there is a match.

### out\_of\_board function:

Start a loop of 9 iterations.

Extract each square object from the containing dictionary by its key (1-9) and check if its id method returns the same id than the widget that detected the right click. If it is return False.

If nothing matches return True.

### already\_marked function:

Convert the given argument caller\_id (from float) to string.

Iterate over the dictionary containing the square objects extracting the keys and their value.

Check if the given caller id matches the id of the square object.

If so, check if that square object has been clicked more than once. If so play the system’s stop sound. Return True.

If not, return False after the loop.

### mark\_x function:

Convert to id of the clicked square to string.

Extract all key and values of the squares dictionary one by one.

If the current id of the clicked square matched the id of the currently extracted square object:

call the mark\_x method for that square.

### check\_win function:

Iterate over a list containing the numbers of each 3 squares that form a line.

Declare a list to contain matching squares below.

Iterate over each string of 3 squares that form a line.

If the current square number matches any of the already marked squares for the party given in the argument (“machine” or “user”), append that square to the list declared above.

If the resulting list has a length of 3, increase the score for the party given in the argument.

Announce the winner.

Return True.

Return False if all 3 groups of squares that complete a line have been checked and nobody won.

### display\_result function:

Ring the system’s bell.

If the passed argument is “user”, color the main window in blue.

Show a tkinter.messagebox announcing that the user won.

If the passed argument is “machine”, color the main window in red.

Show a tkinter.messagebox announcing that the machine won.

If the passed argument is not “user” or “machine”, show a tkinter.messagebox announcing that it’s a match.

Call the new\_game functions.

If the machine starts next, mark a random corner with a red “O”.

### check\_marks function:

If the passed argument is “user”, will search for an “X”.

If the passed argument is “machine”, will search for an “O”.

Declare a list to contain matching squares below.

Iterate over the dictionary containing the square objects extracting the keys and their value.

If the current square’s mark matches the searched letter, append that square number to the list declared above.

Return the list.

### search\_completing function:

If the passed argument is “machine”, the other party is the user.

If the passed argument is “user”, the other party is the machine.

Iterate over a list containing the numbers of each 3 squares that form a line.

Declare a list to contain matching squares below.

Iterate over each string of 3 squares that form a line.

If the current square number matches any of the already marked squares for the party given in the argument (“machine” or “user”), append that square to the list declared above.

If the resulting list has a length of 2, determine the square that will complete that line by subtracting the sum of both square numbers in the list from 15.

If the resulting square number is already marked by the determined other party, return that square number.

If none of the above conditions are ever met, return “nothing found”.

### search\_corners function:

Declare a list to contain matching squares below.

Iterate from 2 to 10 in steps of 2.

If the number of square that corresponds to the current number in the loop is empty, append it to the list declared above.

If the list has 1 element return it.

If the list has more than 1 element, return one randomly.

If the list is empty return “nothing found”.

### search\_side function:

Declare a list to contain matching squares below.

If square 1 is empty, append tis number to the list declared above.

If square 3 is empty, append tis number to the list declared above.

If square 7 is empty, append tis number to the list declared above.

If square 9 is empty, append tis number to the list declared above.

If the list has 1 element return it.

If the list has more than 1 element, return one randomly.

If the list is empty return “nothing found”.

### board\_full function:

Set a counter to 0.

Iterate from 1 to 9 (inclusive).

If the square number that matches the current iteration is empty, increase the counter.

If the counter is 9 return True, if not return False.

### new\_game function:

Declare global variable that controls who starts.

If the machine starts now, set the machine to not start next game.

If not, set the machine to start the next game.

Iterate from 1 to 9 (inclusive).

Clear each square whose number matches the current iteration number.

Color the main window to the system’s default.

### confirm\_quit function:

Play the system’s question sound.

Display a message confirming quit. If the user confirms, check if the difference between the time when the game started and the current time. It it’s greater than 3 seconds and smaller than 19 seconds, call the destroy\_me method of the boaster object (closes that window).

If the score of the user or the machine is greater than 1,

Declare a False variable.

Open the historic score file for reading.

Start and error-handled block:

Unplickle the opened file.

If there is an error, set the False variable to True.

Close the historic score file.

Open the historic score file for writing.

If the variable is False,

Add the current score to the saved historic score in binary form.

If the variable is True,

Save the current score the historic score in binary form.

Close the historic score file.

Form the current score file name by obtain the Desktop path for the current user + “Tic Tac Toe Score” + current data and time formatted + “.txt”.

Open that file for writing.

Write to the file the user’s name, their score, and machine’s score under “MACHINE”.

Close the file.

Show a message box informing that the current score was saved to the Desktop.

Destroy the main window (close the game).

**END**