Regex Games

Generated by Doxygen 1.8.12

Contents

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 checkForClick Namespace Reference

We check for a click on the gameboard Each time we get a click we update the screen We make sure that the mouse is visible as well.

Variables

- resp = None
- dispEnd = time.time()

  we display the time
- ms = pygame.mouse.get_pressed()

  This is the variable that holds the mouse click itself.
- rt = dispEnd-dispSt

4.1.1 Detailed Description

We check for a click on the gameboard Each time we get a click we update the screen We make sure that the mouse is visible as well.

4.1.2 Variable Documentation

4.1.2.1 checkForClick.dispEnd = time.time()

we display the time

4.1.2.2 checkForClick.ms = pygame.mouse.get_pressed()

This is the variable that holds the mouse click itself.

4.1.2.3 string checkForClick.resp = None

4.1.2.4 checkForClick.rt = dispEnd-dispSt

4.2 FinalImplementation Namespace Reference

This is the final implementation for the terminal version of our game.

Functions

- def fillQuestionListBasic ()

  These are the functions for our "easy" mode, using basic questions.
- def fillAnswerListBasic ()

  These are the functions for our "easy" mode, using basic questions.
- def fillQuestionListDifficulty (difficultyString)

  Here is the function for our difficulty selector.
- def fillAnswerListDifficulty (difficultyString)

  Here is the function for our difficulty selector.
- def gameDeclaration ()

  We declare an instance of the "player" class that we have declared.

Variables

- db = MySQLdb.connect("127.0.0.1", "root", "root", "regexQuestions")

  This is the DB that we connect to, it holds all of our questions.
- cur = db.cursor()
- list questionList = [ ]

  This is the list that we append all of our questions to.
- list answerList = [ ]

  This is the list we append all of our answers to.

4.2.1 Detailed Description

This is the final implementation for the terminal version of our game.

We have implemented our 'hardness-selection' functionality, we pull in from our database of questions and we are able to create a player and track their score.

4.2.2 Function Documentation

4.2.2.1 def FinalImplementation.fillAnswerListBasic ( )

These are the functions for our "easy" mode, using basic questions.

We pull a single question/answer from the database at a time using a continuous for loop. We then append the lists with questions/answers

4.2.2.2 def FinalImplementation.fillAnswerListDifficulty (*difficultyString*)

Here is the function for our difficulty selector.

It follows the same principle as above, except it has a input string check for Hard/Medium settings. This fills the Answer List that we have

4.2.2.3 def FinalImplementation.fillQuestionListBasic ( )

These are the functions for our "easy" mode, using basic questions.

We pull a single question/answer from the database at a time using a continuous for loop. We then append the lists with questions/answers"""

4.2.2.4 def FinalImplementation.fillQuestionListDifficulty (*difficultyString*)

Here is the function for our difficulty selector.

It follows the same principle as above, except it has a input string check for Hard/Medium settings

4.2.2.5 def FinalImplementation.gameDeclaration ( )

We declare an instance of the "player" class that we have declared.

4.2.3 Variable Documentation

4.2.3.1 FinalImplementation.answerList = [ ]

This is the list we append all of our answers to.

It's also a simple python list.

4.2.3.2 FinalImplementation.cur = db.cursor()

4.2.3.3 FinalImplementation.db = MySQLdb.connect("127.0.0.1", "root", "root", "regexQuestions")

This is the DB that we connect to, it holds all of our questions.

4.2.3.4 FinalImplementation.questionList = [ ]

This is the list that we append all of our questions to.

It's a simple python list

4.3 mouseClick Namespace Reference

Here is our functionality for mouse clicking within the graphical version.

Functions

- def draw (canvas)

  We create the screen necessary for the click here.
- def do_on_click (clicked_position)

Variables

- list click_pos = [ ]

  This is to keep track of the position of the mouse click.
- list click = [ ]

  We store our 'click' in the list here.
- frame = simplegui.create_frame("Mouse", 400,400)

4.3.1 Detailed Description

Here is our functionality for mouse clicking within the graphical version.

4.3.2 Function Documentation

4.3.2.1 def mouseClick.do_on_click (*clicked_position*)

4.3.2.2 def mouseClick.draw (*canvas*)

We create the screen necessary for the click here.

4.3.3 Variable Documentation

4.3.3.1 mouseClick.click = [ ]

We store our 'click' in the list here.

4.3.3.2 mouseClick.click_pos = [ ]

This is to keep track of the position of the mouse click.

4.3.3.3 mouseClick.frame = simplegui.create_frame("Mouse", 400,400)

4.4 playerClass Namespace Reference

This is the definition for our player class.

Classes

- class player

4.4.1 Detailed Description

This is the definition for our player class.

Each player object is initialized with a score of 0 and can add to that score and print from that score.

Chapter 5

Class Documentation

5.1 playerClass.player Class Reference

Public Member Functions

- def __init__ (self)

  This is the constructor for the player class.
- def addScore (self)

  We define the addScore function here, to add to the score of the player.
- def printScore (self)

  We define the printScore function here, so as to print the current score of the player.

Public Attributes

- score

Static Public Attributes

- int score = 0

5.1.1 Constructor & Destructor Documentation

5.1.1.1 def playerClass.player.__init__ (*self*)

This is the constructor for the player class.

We initialize the score to 0.

5.1.2 Member Function Documentation

5.1.2.1 def playerClass.player.addScore (*self*)

We define the addScore function here, to add to the score of the player.

5.1.2.2 def playerClass.player.printScore (*self*)

We define the printScore function here, so as to print the current score of the player.

5.1.3 Member Data Documentation

5.1.3.1 int playerClass.player.score = 0 `[static]`

5.1.3.2 playerClass.player.score

The documentation for this class was generated from the following file:

- playerClass.py

Chapter 6

File Documentation

6.1 checkForClick.py File Reference

Namespaces

- checkForClick

  We check for a click on the gameboard Each time we get a click we update the screen We make sure that the mouse is visible as well.

Variables

- checkForClick.resp = None
- checkForClick.dispEnd = time.time()

  we display the time
- checkForClick.ms = pygame.mouse.get_pressed()

  This is the variable that holds the mouse click itself.
- checkForClick.rt = dispEnd-dispSt

6.2 FinalImplementation.py File Reference

Namespaces

- FinalImplementation

  This is the final implementation for the terminal version of our game.

Functions

- def FinalImplementation.fillQuestionListBasic ()

  These are the functions for our "easy" mode, using basic questions.
- def FinalImplementation.fillAnswerListBasic ()

  These are the functions for our "easy" mode, using basic questions.
- def FinalImplementation.fillQuestionListDifficulty (difficultyString)

  Here is the function for our difficulty selector.
- def FinalImplementation.fillAnswerListDifficulty (difficultyString)

  Here is the function for our difficulty selector.
- def FinalImplementation.gameDeclaration ()

  We declare an instance of the "player" class that we have declared.

Variables

- FinalImplementation.db = MySQLdb.connect("127.0.0.1", "root", "root", "regexQuestions")

    This is the DB that we connect to, it holds all of our questions.
- FinalImplementation.cur = db.cursor()
- list FinalImplementation.questionList = [ ]

    This is the list that we append all of our questions to.
- list FinalImplementation.answerList = [ ]

    This is the list we append all of our answers to.

6.3 mouseClick.py File Reference

Namespaces

- mouseClick

    Here is our functionality for mouse clicking within the graphical version.

Functions

- def mouseClick.draw (canvas)

    We create the screen necessary for the click here.
- def mouseClick.do_on_click (clicked_position)

Variables

- list mouseClick.click_pos = [ ]

    This is to keep track of the position of the mouse click.
- list mouseClick.click = [ ]

    We store our 'click' in the list here.
- mouseClick.frame = simplegui.create_frame("Mouse", 400,400)

6.4 playerClass.py File Reference

Classes

- class playerClass.player

Namespaces

- playerClass

    This is the definition for our player class.

Index