# tkinter - Using Multiple Windows and Radio Buttons

This program will:

- Load a list of dictionaries storing the questions and the answers.
- Display a window with the instructions. When the user is ready to start, they will press a button
- A window with the question and the potential answers will appear. The user will choose their answer using radio buttons.
- Once the user makes their choice, another window will appear showing whether they are correct or not. If they are correct, the total number of correct answers will be outputted.
- Once all the questions have been asked, a window will appear stating this and the user will have the option of redoing the quiz or exiting the program.

The new tkinter items covered include multiple windows, window sizing and positioning and radio buttons. Here are some websites which describe these concepts:

- https://www.youtube.com/watch?v=XNF-y0QFNcM&list=PL8BEB66FBE6DB97AF&index=10
- http://www.tutorialspoint.com/python/tk_radiobutton.htm
- http://docstore.mik.ua/orelly/other/python2/1.7.htm

Note that it is poorly layed out using the pack() layout manager. You would probably wish to improve it - use the following web sites to help you with tkinter layout managers:

- https://www.youtube.com/watch?v=VtayMVa65cU&list=PL8BEB66FBE6DB97AF&index=3
- http://www.python-course.eu/tkinter_layout_management.php

It also requires the use of Python lambda functions which we have not covered to this point. Most tkinter objects have a function which is called when they are selected (ie. when you press a button, a function is called which will do something). Unfortunately, these “callback” functions cannot have parameters in tkinter. Using lambda functions allow parameters to be passed to the callback functions. Lambda functions are somewhat confusing but if you wish to learn more about them, look at the following website:

- https://pythonconquerstheuniverse.wordpress.com/2011/08/29/lambda_tutorial/

## Code

```
from tkinter import *

def loadList (questions):
    ''' Creates a list of four question - each question is organized as a dictionary
    questions is the name of the list of questions
    Obviously you can also load the questions from a file and the list can be as large as required. '''

    aSingleQuestion = {}
    aSingleQuestion["Question"]="What course is this?"
    aSingleQuestion["a"]="ICS 3U"
    aSingleQuestion["b"]="ICS 3C"
    aSingleQuestion["Answer"]="a"
    questions.append (aSingleQuestion)
```

```
    aSingleQuestion = {}
    aSingleQuestion["Question"]="What time is the exam?"
    aSingleQuestion["a"]="8:30 a.m."
    aSingleQuestion["b"]="12:00 p.m."
    aSingleQuestion["Answer"]="b"
    questions.append (aSingleQuestion)

    aSingleQuestion = {}
    aSingleQuestion["Question"]="What day is the exam?"
    aSingleQuestion["a"]="Friday"
    aSingleQuestion["b"]="Monday"
    aSingleQuestion["Answer"]="a"
    questions.append (aSingleQuestion)

    aSingleQuestion = {}
    aSingleQuestion["Question"]="What date is exam?"
    aSingleQuestion["a"]="February 1"
    aSingleQuestion["b"]="January 29"
    aSingleQuestion["Answer"]="b"
    questions.append (aSingleQuestion)

def start (questions):
    ''' Creates the introduction window.
    questions is the name of the list of questions which is not used in this specific function but passed to a
    function called in this function.'''

    # These integer variables must be global as they are used and changed in multiple functions. In Python, the
    #  programmer must use the global command to indicate the variable is global - if this is not used, the
    #  variable will automatically be local
    global numCorrect           # The number of correct answers the user has.
    global questionNum          # Which question is being displayed and answered

    questionNum = -1            # Must be initialized at -1 to insure the first element in the lis is used
    numCorrect=0

      instructionWindow = Toplevel (background = "red")   # Creates a new window to display the instructions
      instructionWindow.geometry ("400x400+200+200")      # Sizes the instruction window
      instructionWindow.wm_attributes('-topmost', 1)      # Insures that this window is on top of all others

      instructionArea = Canvas (instructionWindow)        # Creates a canvas to "draw" instruction text
      instructionArea.pack()
      instructionArea.create_text (200, 120, text="These are instructions", fill='blue', font=('verdana',20))

      # Creates a lambda function allowing a function with parameters to be passed to an "action" function
      # This function will ask a question.
      buttonFunction = lambda :newQuestion (questions, instructionWindow)
      startButton = Button (instructionWindow, text ="Start program", command= buttonFunction)
      startButton.pack()
```

```
def newQuestion (questions, previousWindow) :
    ''' Creates a window to display the question and the possible answers using radio buttons. When the user
    selects a radio button, a function to check their answer will be called.
    questions is the list of questions stored as dictionaries which are displayed using this function
    previousWindow is the window previously opened. It is closed in this function '''

    global numCorrect
    global questionNum
    questionNum += 1    # Every time this function is called, the question number increases to display the
                        next #question in the list

    previousWindow.destroy()    # Closes the Introduction window

    questionWindow = Toplevel (background="blue")  # Creates a new window to display the question
    questionWindow.geometry ("400x400+200+200")  # Sizes the instruction window
    questionWindow.wm_attributes ('-topmost', 1)       # Insures that this window is on top of all others

    if questionNum < len (questions) :    # Checks to ensure there are still questions to be displayed

        radioValue = IntVar()          # All radio buttons in a group must have the same variable

        # Uses a label to display the question from the list of dictionaries
        questionText = Label (questionWindow, text=questions[questionNum]["Question"])
        questionText.pack()

        # Creates a lambda function allowing a function with parameters to be passed to an "action"
        #function. This function will check to see if the user's answer is correct.
        radioButtonFunction = lambda :checkAnswer (questions[questionNum]["Answer"], radioValue, questionWindow)

            radButA= Radiobutton(questionWindow, text= questions[questionNum]["a"],
                    variable = radioValue, value=1, command= radioButtonFunction)
            radButA.pack()          # Radio button for option "a"

            radButB= Radiobutton(questionWindow, text=questions[questionNum]["b"],
                    variable = radioValue, value=2, command=radioButtonFunction)
            radButB.pack()

        else:    # If all the questions have been asked.

            questionNum=-1        # Resets the global variables
            numCorrect=0

            doneText = Label (questionWindow, text="All questions answered")
            doneText.pack()       # Displays message for the user

            # Creates a lambda function allowing a function with parameters to be passed to an "action"
            #function. This function will ask a question if the user wishes to re-start the quiz
            buttonFunction = lambda : newQuestion (questions, questionWindow)
```

```
        startButton = Button(questionWindow, text ="Re-start quiz", command = buttonFunction)
        startButton.pack()      # Button to restart the quiz

        # Creates a lambda function allowing a function with parameters to be passed to an "action"
        #function. This function will ask a question if the user wishes to exit the quiz
        exitButtonFunction = lambda : destroyStuff (questionWindow))

        exitButton = Button (questionWindow, text ="Exit program", command=exitButtonFunction)
        exitButton.pack()

def destroyStuff(win) :
    ''' Closes all windows and quits programs
    win is any open window '''
    win.destroy()
    root.destroy()

def checkAnswer(answer, radioValue, previousWindow):
    ''' Checks whether the user got the answer correct.
    answer is the answer to the question
    radioValue is which radio button has been selected
    previousWindow is the window previously opened. It is closed in this function '''

    previousWindow.destroy()          # Closes previous window

    global numCorrect     # numCorrect is updated in this function so it must be global

    answerWindow = Toplevel(background="yellow")  # Creates a new window to display the result
    answerWindow.geometry ("400x400+200+200")     # Sizes the answer results window
    answerWindow.wm_attributes('-topmost', 1)     # Insures that this window is on top of all others

    answerArea = Canvas (answerWindow)      # Creates a canvas to display the result
    answerArea.pack()

    if radioValue.get() == 1:       # Matches the buttons with the answers (ie 1st button is 'a', 2nd is 'b', etc.
        selection = "a"
    elif radioValue.get() == 2:
        selection = "b"

    if selection == answer:         # Outputs a "correct" message if the user's choice is correct

        answerArea.create_text (100,100, text="Correct", fill='blue', font=('verdana',20))
        answerArea.create_text (100,120, text="num right"+str(numCorrect), fill='blue', font=('verdana',20))
        numCorrect+=1       # Updates correct counter

    else :                  # Outputs a "wrong" message if the user's choice is incorrect
        answerArea.create_text (100,100, text="Wrong", fill='blue', font=('verdana',20))
```

```
    # Creates a button to display the next question
    # Note that the lambda function can be created as it is called instead of creating a function in a separate line.
    # This is faster but can be confusing
    continueButton = Button (answerWindow, text ="nextQuestion",
                    command = lambda :newQuestion(questions,answerWindow))
    continueButton.pack()
```

## #*Mainline*

```
root = Tk()                                 #Creates a tk object
root.geometry ("400x400+200+200")           # Sizes the root window so it appears under all the other windows

questions=[]            # Initializes the list that will store the list of dictionaries containing the question and answers
loadList(questions)     # Loads the list of dictionaries with the question and answers

start(questions)        # Starts the instruction window

root.mainloop()         # Event listener
```