CS-119 Lab #9

Expected Learning Objectives

- Introduction to Graphical User Interfaces (GUIs) in Python
- Forms
- Labels
- Buttons
- Text entry boxes
- Review of general Python coding (sequential code, selection structures, etc.)
- Text formatting

Overview

This lab activity gives you an opportunity to apply some of the programming principles presented in chapter 9 of the text book. You will be revisiting some previous labs and reworking them into GUI applications. Updating an existing application with new features and/or functional changes is very much a part of "real world" software development.

GUI programming in Python or any other language can get complicated – especially if you must do it all in code! There are numerous GUI controls, numerous ways to design/layout the GUI, set colors, align controls, etc. However, in this lab, a very basic approach will be used. Once you complete this first exercise, you will find that you can easily replicate the work you did here to the remaining exercises.

Most modern programming language libraries provide a set of commonly used GUI *controls* (often referred to as *components* or *widgets*). These include labels, buttons, text boxes, check boxes, etc. These are all pre-coded, tested and immediately available for any GUI application you wish to create.

When creating a GUI, you will start with a *form* (sometimes referred to as a *frame* or a *canvas*) and then add the various user interface controls you wish to use. Most GUI libraries have various layout managers to help you arrange your controls on the form. In this lab, you will use a grid layout for the simple reason it's the easiest to learn and use.

For getting user input you will use the Python Entry (single line text box). For labeling each Entry (text box), you will use Label controls. You will also use label controls for displaying output as these can't be modified by the user. For initiating actions, you will use buttons.

From chapter 9 in the text book, you learned the GUI environment is event driven. A GUI that responds to no events is completely useless! For this lab, you will write code often referred to as an *event handler* or *callback function* to respond to when the user presses (clicks) a button on the form.

Exercise 1 Guided – Gas Mileage Calculator

That Python console application you did earlier in the semester to compute gas mileage and cost per mile is a bit outdated! It's time to modernize it with a GUI. Your application must allow the user to enter the number of miles driven, number of gallons put into the gas tank and cost in dollars per gallon of gas. Your outputs must include MPG, total fuel cost and cost per mile.

Key formulas:
mpg = milesDriven / gallons
totalFuelCost = gallons * costPerGallon
costPerMile = totalFuelCost / milesDriven

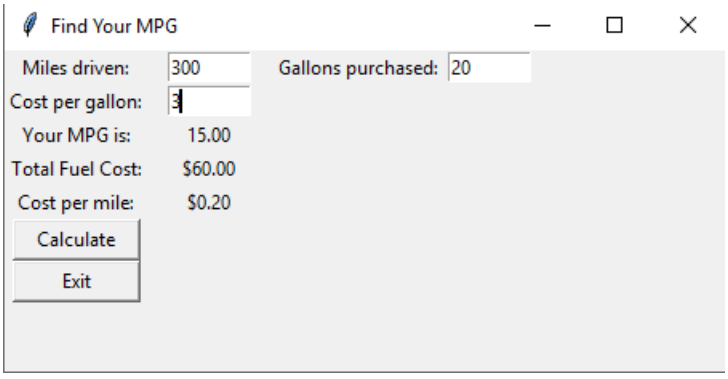
Before writing any code, revise your pseudocode from before and update it with the new steps for creating the GUI. The subtle lesson here: Your pseudocode/flowchart and other system design documents are *living documents* for the life of your project; meaning you should be revising them as you add features, fix bugs, make functional changes, etc.

Being a guided exercise, you are given the Python code:

```
#!/user/bin/env python3

# Cuyamaca College CS-119
# Your name here!
# Lab 9, exercise 1 MPG revisited

# import the Python tkinter GUI library
# Python 3 tkinter (all lower case), earlier Python version 2.x Tkinter (note upper case 'T')
from tkinter import *


# event handlers or callback functions
# click event handler for the calculate button
def calc_mileage():
    # input
    miles = float(txtMiles.get() )
    gallons = float(txtGallons.get() )
    cost_per_gal = float(txtCostPerGal.get() )

    # process
    mpg = miles / gallons
    total_fuel_cost = gallons * cost_per_gal
    cost_per_mile = total_fuel_cost / miles

    # output - note the string formatting
    # https://mkaz.blog/code/python-string-format-cookbook/
    str_mpg.set( "{:.2f}".format(mpg) )
    str_cpm.set( "${:.2f}".format(cost_per_mile) )
    str_tot_fuel_cost.set( "${:.2f}".format(total_fuel_cost) )


def exit_program():
    #exit(0)  # this prompts for confirmation
    frm.destroy()


# create the form
frm = Tk()
frm.title("Find Your MPG")
frm.geometry('450x200') # set the form size

# add controls, components or "widgets" We'll use a grid layout
lblMiles = Label(frm, text="Miles driven:", height=1, width=12)
lblMiles.grid(row=0, column=0)

# Entry is a single line text box
txtMiles = Entry(frm, width=8)
txtMiles.grid(row=0, column=1)

lblGallons = Label(frm, text="Gallons purchased:", height=1, width=15)
lblGallons.grid(row=0, column=2)

txtGallons = Entry(frm, width=8)
txtGallons.grid(row=0, column=3)

lblCostPerGal = Label(frm, text="Cost per gallon:", height=1, width=12)
lblCostPerGal.grid(row=1, column=0)

txtCostPerGal = Entry(frm, width=8)
txtCostPerGal.grid(row=1, column=1)
```

```
lblTextMPG = Label(frm, text="Your MPG is:",height=1, width=10)
lblTextMPG.grid(row=2, column=0, )

str_mpg = StringVar()
lblMPG = Label(frm, text="",height=1, width=10, textvariable=str_mpg)
lblMPG.grid(row=2, column=1)


str_tot_fuel_cost = StringVar()
lblTextTotFuel = Label(frm, text="Total Fuel Cost:",height=1, width=12)
lblTextTotFuel.grid(row=3, column=0, )

lblTotFuel = Label(frm, text="",height=1, width=10, textvariable=str_tot_fuel_cost)
lblTotFuel.grid(row=3, column=1)


str_cpm = StringVar()
lblTextCPM = Label(frm, text="Cost per mile:",height=1, width=12)
lblTextCPM.grid(row=4, column=0, )

lblCPM = Label(frm, text="",height=1, width=10, textvariable=str_cpm)
lblCPM.grid(row=4, column=1)



# in the command= we bind the button press event handler to the control
btnCalc = Button(frm, text="Calculate",command=calc_mileage, height=1, width=10)
btnCalc.grid(row=6, column=0)

btnExit = Button(frm, text="Exit",command=exit_program, height=1, width=10)
btnExit.grid(row=7, column=0)


# display main window
frm.mainloop()
```

There is room to be creative with placement of controls, wording prompts, etc. but your completed, running GUI should look (roughly) something like this:



Exercise 2 – BMI

For this exercise, you will update another previous lab exercise with a GUI. Design and develop an application that allows the user to enter their height in inches, weight in pounds and then calculate their body mass index (BMI). To calculate BMI, we will use the following formulas:

bmi = kilograms / meters2
meters = inches / 39.36
kilograms = pounds / 2.2

Once you calculate the BMI percentage, you will need to display one of the messages below based on the BMI percentage calculated.

BMI range	Message
Less than 18.5	Under weight
18.5 – 24.99	Normal weight
25 – 29.99	Over weight
30 – 39.99	Obese
40 or over	Morbid Obesity

Be sure to complete pseudocode or a flowchart for this exercise. You may reuse and update your previous work.

Exercise 3 – Coyote Inn

Coyote Inn has asked you to modernize the application you developed to estimate charges for staying at their new inn with a graphical user interface. From the original exercise, their nightly rates are based on months of the year and provided in the following table:

Months	Rate
1 – 3 (Jan – Mar)	$80/night
4 – 6 (Apr – Jun)	$90/night
7 – 9 (Jul – Sept)	$120/night
10 – 12 (Oct – Dec)	$100/night

To keep things simple, you can still allow the month to be entered by the month number. Normally, this would be done by a date period but do it by month number to keep the exercise simple.

The user must be able to enter a month number (1 – 12) and the number of nights. Use a decision structure to find the correct price per night for the month entered and then multiply it by the number of nights entered.

You may reuse and update the pseudocode or flowchart you completed for the original exercise.

Grading Criteria:

Deliverable	Points	Breakdown
Guided exercise pseudocode	5	Complete, logic is clearly outlined
Guided exercise code and run	5	GUI is reasonably clean. Code is clean, runs, produces correct output
Exercise 2 (BMI) pseudocode	5	Complete, logic is clearly outlined
Exercise 2 Python code	15	GUI is reasonably clean. Code is clean, runs, produces correct output
Exercise 3 (Coyote Inn) pseudocode	5	Complete, logic is clearly outlined
Exercise 2 Python code	15	GUI is reasonably clean. Code is clean, runs, produces correct output
Lab Total	**50**	