

Chocolate Machine

Contents:

Introduction:	1
Analysis:	1
Design:	4
Developing Coded Solution:.....	9

Introduction:

The purpose of the program being created is to simulate a vending machine that will be used to teach python and tkinter. It will be a chocolate bar vending machine written in python 3.3.2 that allows a user to choose a chocolate bar, input money and then receive an appropriate amount of change. The change will be given in as few coins as possible meaning coins of the highest value will be given first. This vending machine will be displayed on a GUI using tkinter on windows. A chocolate vending machine is a suitable program for teaching people to use python and tkinter because it is simple and uses lots of the main process and components that are commonly used.

Analysis:

Stakeholders

Suitable stakeholders for this project will be people learning to code, teachers and python and tkinter learning resources such as websites because the program can be used as an example code or step by step tutorial. This would require the program to be easy to understand which could be done by giving things suitable variable names, commenting on the code to explain individual parts and by putting the code in chronological order (linear) where possible. This program will be appropriate for their needs because it involves quite simple processes and everyone is familiar with how a vending machine works. Other stakeholders for this project would be people selling vending machines as this project could be used to program them however this would require the code to consider the amount of stock left and the amount of change the machine holds.

Research

Essential Features

Based on my research I have found that the essential features are:

1. A list of chocolate bars with their prices displayed to be chosen from
2. A way to enter an amount of money
3. A way to check the amount of money entered is valid/enough
4. The correct amount of change dispensed
5. Change given in order of highest valued coins to lowest
6. Ability to exit the program

Description and Justification of these essential features:

1. This will be a series of radio buttons (from tkinter) with the name of a chocolate bar and its price on each one. This is used to ensure a valid option is chosen and that a price can be outputted to work out the change. Radio buttons are used so that only one option can be chosen at a time to avoid errors.
2. This will include an entry widget so that an amount can be entered by the user and a button. This is an essential feature because it allows the amount paid to be outputted so that the process of working out the change can begin.
3. This will use iteration to make sure the amount paid is higher than the price of the chocolate bar and this is essential because without it there would be errors as the amount of change needed would be negative.
4. Change will be dispensed using a while loop that outputs coins until change needed equals 0. The correct amount of change must be dispensed because if it didn't the user or the vending machine would lose money which is not what a correctly functioning vending machine does.
5. The change given will be in order of highest valued coins to lowest using a list of coins in that order and a while loop that moves along the list if the change is lower than the coin it is currently on. This is essential so that minimum amount of coins is dispensed because this is the most efficient way and it is part of the requirements.
6. This will be done using a button on tkinter that triggers a command that quits the program. This is essential so that the program can be closed easily.

Limitations

The final code will have some limitations such as not being able to store the amount of stock available of products or the amount of change left. This will mean there is an unlimited amount of change and products which is unrealistic for a real vending machine as in reality it would run out of these items as it dispensed them to customers. If this program were to be used in a real vending machine with these limitations it would eventually take in money without giving out any change or products however these limitations do not matter because the main purpose of this program is to be used as a learning tool for tkinter and python or as an example of a GUI (graphical user interface) which it does effectively despite these limitations.

Another limitation is that an error will occur if the user enters the price as anything other than an integer. This will cause red text to appear in the python shell and the program won't be able to work out the amount of change due however this limitation is not a major problem as if the user enters a new amount of money the program will run as normal. Additionally, if the program were used for a real vending machine it would not be possible for this error to occur because only coins could be inserted.

An additional limitation is that the money must be entered in pence instead of pounds which could confuse some users because if they wanted to enter £1 they would have to write it as 100. Fortunately, this is not an issue because the chocolate bars are all less than a pound and it helps users to learn python by making them look at the code to see how it works and therefore how to enter the data correctly which is the main purpose of this program.

Requirements

Because of the purpose of the project the program will require:

1. A program written using the software python 3.3.2
2. A way for the user to choose a chocolate bar with a set price

3. A way for the user to enter an amount of money
4. The program to calculate and display an appropriate amount of change
5. The change to be given in as few coins as possible
6. The essential features to be displayed on a GUI using the software tkinter

Justification of these requirements:

1. Because the final programs purpose is to be used as a learning tool for python which it could not be used for unless written in python and certain features might not work unless using the same version.
2. Because the program will be a simulation of a chocolate vending machine so the user should only be able to pick one option out of a set variety of chocolate bars. The chosen bar will need to have a set price because one of the other requirements it's to calculate change which cannot be done without a price.
3. Because the program will be a simulation of a chocolate vending machine so it must also simulate a user entering money because money is required to buy the chocolate and the amount paid is then used to calculate the change due.
4. Because to be a simulation of a real vending machine it must be able to give change so that people don't lose money from paying extra. It will also teach people how to use dynamic text boxes on tkinter because when the change is displayed the text in the text widget must be updated.
5. Because this is what happens in a real vending machine so that dispensing the coins is more efficient as it takes less time and people don't have to carry as much change with them.
6. Because the final programs purpose is to be used as a learning tool for tkinter which it could not be used for unless written in tkinter. It must be a GUI to make it more of a simulation because this is more like a real vending machine as both have buttons.

Success Criteria

1. The program must run successfully using python 3.3.2 without any errors occurring that prevent it from working out the correct change due.
2. The program must create a GUI using tkinter
3. Tkinter will create a window titled "Chocolate Vending Machine:" that is big enough to fit all the other components within it without making them overlap.
4. A series of radio buttons that are all the same size will be displayed on the top left of the screen with the title "Choose your favourite chocolate bar:" above them. The first three buttons will have the names of a chocolate bar on it with their price to the left of the text. When these buttons are clicked their price will be outputted.
5. The last button in the series of radio buttons will include the text "EXIT" and will end the program when clicked.
6. The GUI must only allow one of the radio buttons to be selected at a time.
7. On the top right of the tkinter window must be an entry widget that's the same height as the current text size and the default width. To the left of the widget will be the text "Enter Money:"
8. On the top right of the tkinter window to the right of the entry widget will be a button labelled "Pay". When clicked this button will output the amount the user entered so that the program works out the change.
9. In the centre of the window is a display widget that has a height of 8 and a width of 50. It will display the text "Change:" at the top and will show the correct amount of change in as few coins as possible.

10. On the right of the display widget will be a scroll bar that enables the user to scroll the text up and down so that it can all be seen.

Justification for Success Criteria:

1. This is because if there are errors in the code it will not be able to work out the correct amount of change which would mean the program could not fulfil its purpose of simulating a vending machine and the people using it as a learning tool may learn incorrect information. It must use python 3.3.2 so that the program works correctly on all computers because they can all use the same version.
2. This is so that it simulates a vending machine more realistically than it would without a GUI and it must be in tkinter so that the program can be used to teach tkinter or as an example.
3. It will be labelled as this so that users know it is a vending machine and the components can't overlap because it could cause errors in the program from the wrong things being clicked and something not being visible.
4. They will be laid out this way because it is clear and simple. The price will be outputted so that it can later be used to calculate the change required.
5. This is so that the program can be closed in an easy way and to teach people how to make exit buttons.
6. This is to avoid errors and to make the GUI more realistic because in a real vending machine only one item can be purchased at a time.
7. It must be this height so that the amount of money the user can enter is limited and it should only take up one-line maximum. The text is there to show the user where to enter the money.
8. This is so that the amount paid can be outputted and the process of working out the change can be triggered.
9. It must be this size so that what is meant to be one line of text is not shifted onto the next line and so that it is the same height as the rest of the components making the layout neater. As few coins as possible will be used to make the program more efficient.
10. This is because as more chocolate bars are purchased the change displayed in the widget is not removed so more text keeps getting added and a way to view all this text is necessary.

Design:

Structure and Steps of Solution

The program being created can be broken down into a series of smaller problems suitable for computational solutions. These steps include:

- Importing everything from tkinter then create frames
- Creating a series of radio buttons with different chocolate bar options and prices on them
- Allowing the user to enter a valid amount of money
- Working out and displaying the correct amount of change in as few coins as possible
- Allowing the user to choose another item and repeat the process
- Allowing the user to exit the program and therefore not choose another item
- Displaying the simulation using a GUI on tkinter

Justification of steps:

- The program will be created using these steps because it fulfils everything on the success criteria and is the simplest yet most effective way of creating a vending machine simulation.

- These steps make the code easier to understand which helps to achieve the programs purpose of being a learning tool.
- Breaking the problem down into these steps makes it suitable for computational solutions because programs like python work by doing things step by step in a certain order.

Structure of steps:

- The program is using tkinter so the first step will be to import everything from tkinter.
- The layout of the GUI will be developed using the frame method so the frames will be defined next.
- Below this is the function for working out the correct amount of change and inserting it to the text widget.
- Next the list of radio buttons is created and when they are clicked a command is triggered. For the exit buttons the quit command is executed but for the other buttons when clicked the items set price is outputted.
- The entry widget is created next and afterwards the “Pay” button which, when clicked, outputs the amount paid and triggers the function for working out the change which was defined previously.
- After the entry widget is generated the text widget and scroll bar are created so that the amount of change can be displayed.

Algorithms

The previously mentioned “series of smaller problems” will be used to create a series of algorithms for each step that will be put together for the final code to create the vending machine. This is so that the program is easier to make because there is a step by step guide that explains each part of the process.

Importing everything from tkinter then create frames:

```
from tkinter import all

root = tk

root title = “Chocolate Machine”

frame = Frame(root)

toleftframe = Frame(root)

toleftframe = left side

toprightframe = Frame(root)

toprightframe = right side

bottomrightframe = Frame(root)

bottomrightframe = right side
```

Working out and displaying the correct amount of change in as few coins as possible:

Function ShowPaid

```
    If amount paid > price:

        Change = paid – price

        Print (‘you need’ change ‘p change’)
```

```

Coins = [200, 100, 50, 20, 10, 5, 2, 1]

X = 0

While change > 0:

    While change >= coins[x]

        Change = change – coins[x]

        Quote = coins + 'p dispensed'

    x = x + 1

Text box. Insert quote

```

Creating a series of radio buttons with different chocolate bar options and prices on them:

```

Chocolates = [['Mars = 78p', 78], [Galaxy=99p',99], ['Dairy Milk=59p',59], ['EXIT',0]]

Label in topleftframe, text = "Choose your favourite chocolate bar"

For i in range (0 to length(chocolates) – 1)

    Create radiobutton in top left frame (, text = chocolates[i][0],

                                                price = chocolates[i][1])

```

Allowing the user to exit the program and therefore not choose another item:

```

Create radiobutton in top left frame (, text = chocolates[i+1] [0],

                                Command = quit )

```

Allowing the user to enter a valid amount of money:

```

Create label (toprightframe, text = "Enter Money:")

Paid = Entry(toprightframe)

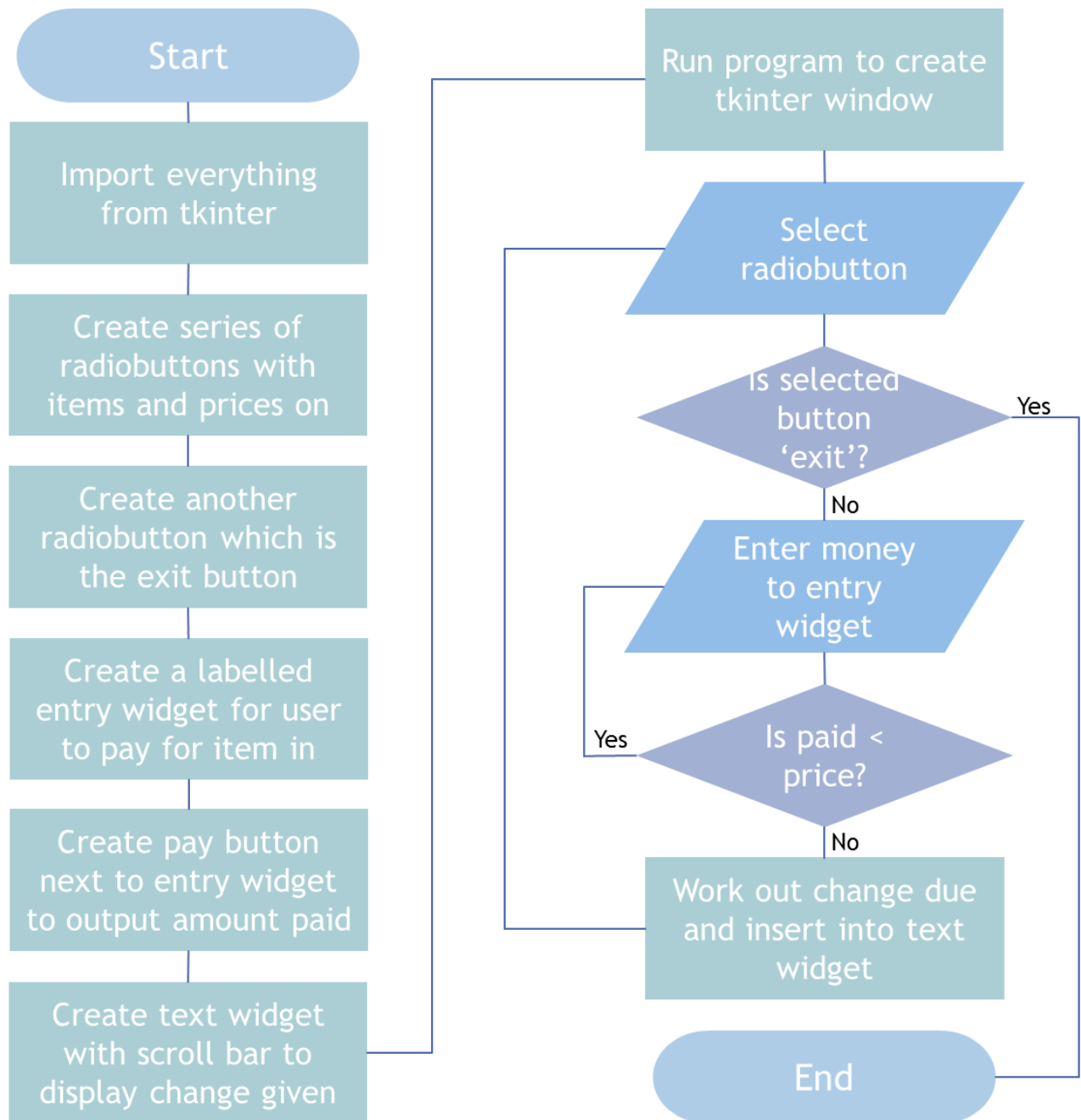
Create Button(toprightframe, text ='Pay', command= ShowPaid

```

Justification of Algorithms:

- These algorithms will form a complete solution to the problem by combining them together in the order previously mentioned in the 'structure of steps'. This will allow the program to receive information in the correct order to avoid parts of the code running before the information that they will process has been outputted. The functions are defined at the top of the page for the same reason.
- I wrote the separate algorithms using pseudo code because anyone can understand it since it is easy to follow and isn't written using a programming language so everyone can understand it.
- I wrote the algorithm for the full code using a flowchart because it clearly shows the main functions and processes of the program and highlights the main validation rules.

Flow Chart:



Usability Features

The GUI must include usability features to make the program easy to use so that it can achieve its purpose of being a python and tkinter tutorial/example. These features will be:

- Buttons – this is a box that can be clicked to trigger a command and it makes it easier for the user to select something because there is no way to input something invalid as there is only one thing to click and no other options.

- Entry widgets – this is where a user can input text which means the user will not accidentally type in the wrong area which could cause errors because there is a set place to type and a label can be used to tell the user what is meant to be entered.
- Text widgets – this is a method for displaying text and it makes it easier for the user to read outputted text because it is in a set area surrounded by a box instead of surrounded by more text.
- Radiobuttons – this is a series of buttons which causes no errors to occur from multiple buttons being pressed because only one of these buttons can be selected at a time.
- Exit buttons – this is a button that, when clicked, executes the quit command to close the program which makes it easy for the user to finish using the vending machine.

Key Variables

I will use these variable names for my code:

Variable Name:	Type of data:	Description:
quote	string	Stores the text that will be inserted into the text widget.
change	integer	Stores the amount of change that is still due.
coins	list	Stores an array of the coins that can be outputted as change in descending order so that fewest amount of coins possible dispensed.
chocolates	list	Stores an array of all the items that can be dispensed by the vending machine and their set price.
choice	string	Stores the item that the user chose to buy.
paid	integer	Stores the amount of money the user entered to pay for the item.
x	integer	Used to move along a list of coins by increasing it in a loop.
ShowPaid	functions	Used to work out and display the change using the 'paid' and 'price' variable.
price	integer	Stores the price of the item that is chosen by the user.

Justification:

These variable names were chosen because they are suitable for their function and sum up their purpose clearly for example the 'change' variable is the amount of 'change' due. This will make it easier to develop the code further and also for people viewing or editing my code which will be very useful for students using it as a tutorial or example which is the main purpose of the program.

Testing Plan

This is the data that will be used in the development of the solution to make sure that everything is working the way that it should be without errors and unexpected results:

Function being Tested:	Test:	Expected Result:
Whether the program creates a GUI using python.	Run the python 3.3.2 code by pressing f5.	A tkinter window titled "Chocolate Machine: " with all the required assets inside.

Whether only one of the radio buttons with items on can be selected at once.	Try to select multiple options by clicking multiple buttons e.g. click Mars button then Galaxy button.	Only one of the buttons is selected at one time and it changes depending on which one was last clicked.
Whether clicking on an items buttons outputs the correct price for that buttons.	Program the code to print the price in the python shell then click an items button e.g. click Diary Milk button	The price displayed on the selected button is the same as the price printed e.g. 59
Whether the 'EXIT' button exists the GUI and ends the program.	Click on the exit button.	A window opens asking you if you want to end the program, if yes is clicked the python shell and tkinter window close.
What happens when invalid data is entered to the entry widget.	Enter text to the entry widget e.g. 'fifty-nine'	Nothing happens so the change is not worked out allowing the user to enter valid data.
What happens when invalid data is entered to the entry widget.	Enter an amount of money that is less than the cost of the item select e.g. select Mars and enter 70p.	"Invalid amount" printed in text widget, the change is not worked out allowing the user to enter valid data.
Whether the correct amount of change in as few coins as possible is outputted.	Select a product and enter an amount of money greater than the price of the product e.g. select Mars and enter 99p.	A series of coins outputted in the text widget that add up to the correct amount of change due in as few coins as possible e.g. '20p dispensed 1p dispensed'
Whether the scroll bar on the text widget allows the user to scroll up and down the text.	Select and pay for multiple products so there is not enough room for all the text to be displayed at once.	The scroll bar will start working allowing the user to view all the text.

Justification:

I decided to test these functions because they are the most important steps in my code and success criteria; any errors in these areas could prevent the program from running correctly and/or efficiently. I chose this set of test data because each one is appropriate for testing a specific function for example to test what happens when text is entered to the entry widget text must be used as the test data. I chose the text 'fifty-nine' because the entry widget is for prices and 59 is the price of one of the items. If a user were to enter text to this entry widget 'fifty-nine' is likely to be the sort of text they would likely enter.

Developing Coded Solution:

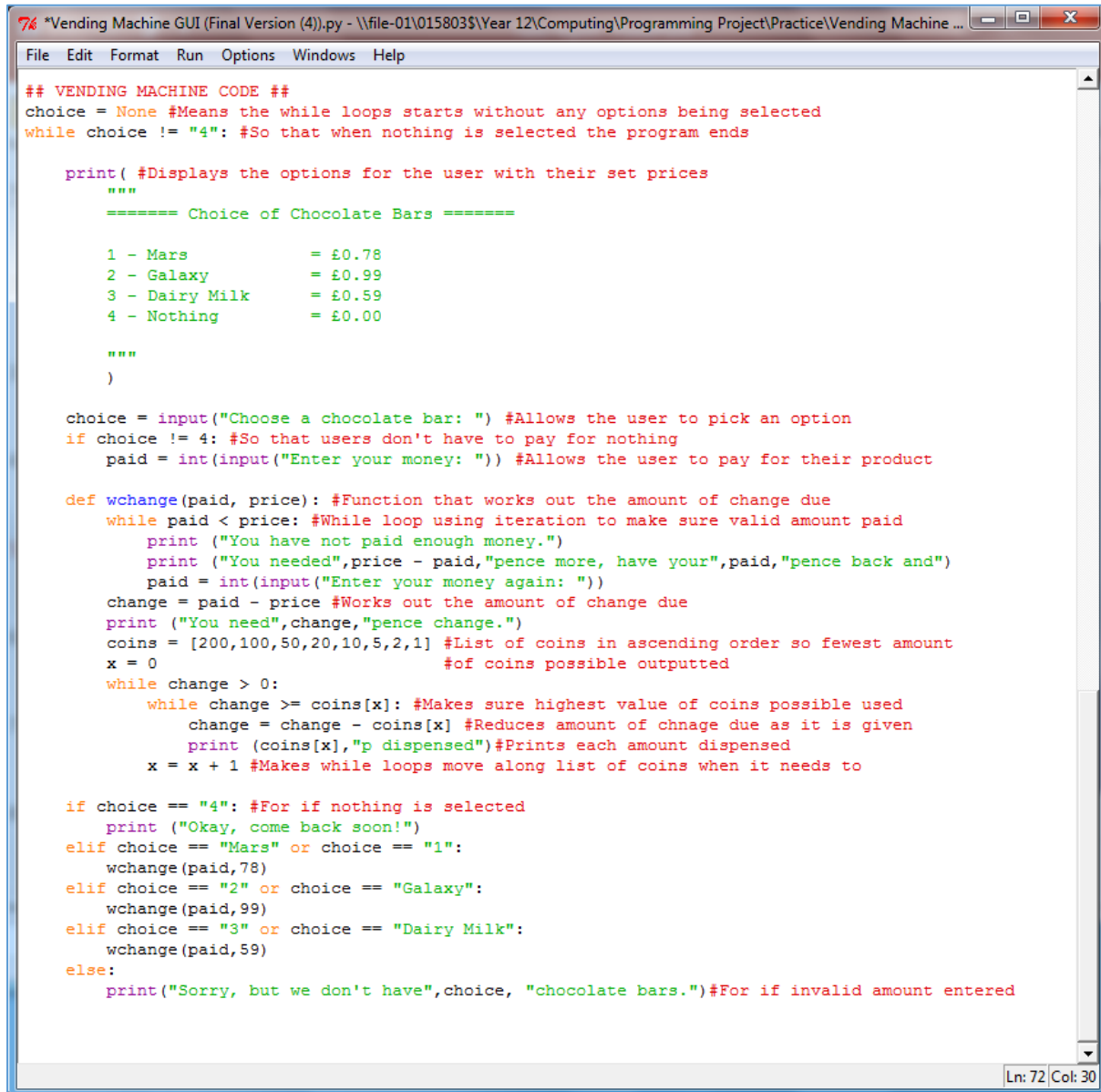
Development Plan

To program a chocolate vending machine that will be displayed on a GUI I will first program the code for just the vending machine without tkinter so that I can develop the main functions of the program correctly without having to worry about how it is displayed as this is the most important part of the program and I can't make a GUI without anything to display on it. After testing this prototype to

make sure it works correctly I will create a copy of the code and begin editing it to display the vending machine on a GUI using tkinter and its frame method because this is a requirement on the success criteria. To help me with this I will use my pseudo code algorithms and flow chart because they clearly show the main structure and steps of the program. I will also use the key variables chosen earlier because they have names that explain their process which will aid with future maintenance of the system because people will be able to understand it. I will test each version of the code using the testing plan to make sure everything works and once the code does everything required I will annotate it to help people understand it and aid future maintenance of the system.

First Prototype

This is the final code for the vending machine without a graphical user interface. The bits of red text with hashtags next to them are comments/annotations which explain the segments of code they are next to making it easier to understand which helps the program to attract suitable stakeholders like teachers because they can use it as a learning tool for their students.



```
74 "Vending Machine GUI (Final Version (4)).py - \\file-01\\015803$\\Year 12\\Computing\\Programming Project\\Practice\\Vending Machine ...
File Edit Format Run Options Windows Help

## VENDING MACHINE CODE ##
choice = None #Means the while loops starts without any options being selected
while choice != "4": #So that when nothing is selected the program ends

    print( #Displays the options for the user with their set prices
        """
        ===== Choice of Chocolate Bars =====

        1 - Mars           = £0.78
        2 - Galaxy          = £0.99
        3 - Dairy Milk      = £0.59
        4 - Nothing         = £0.00

        """
    )

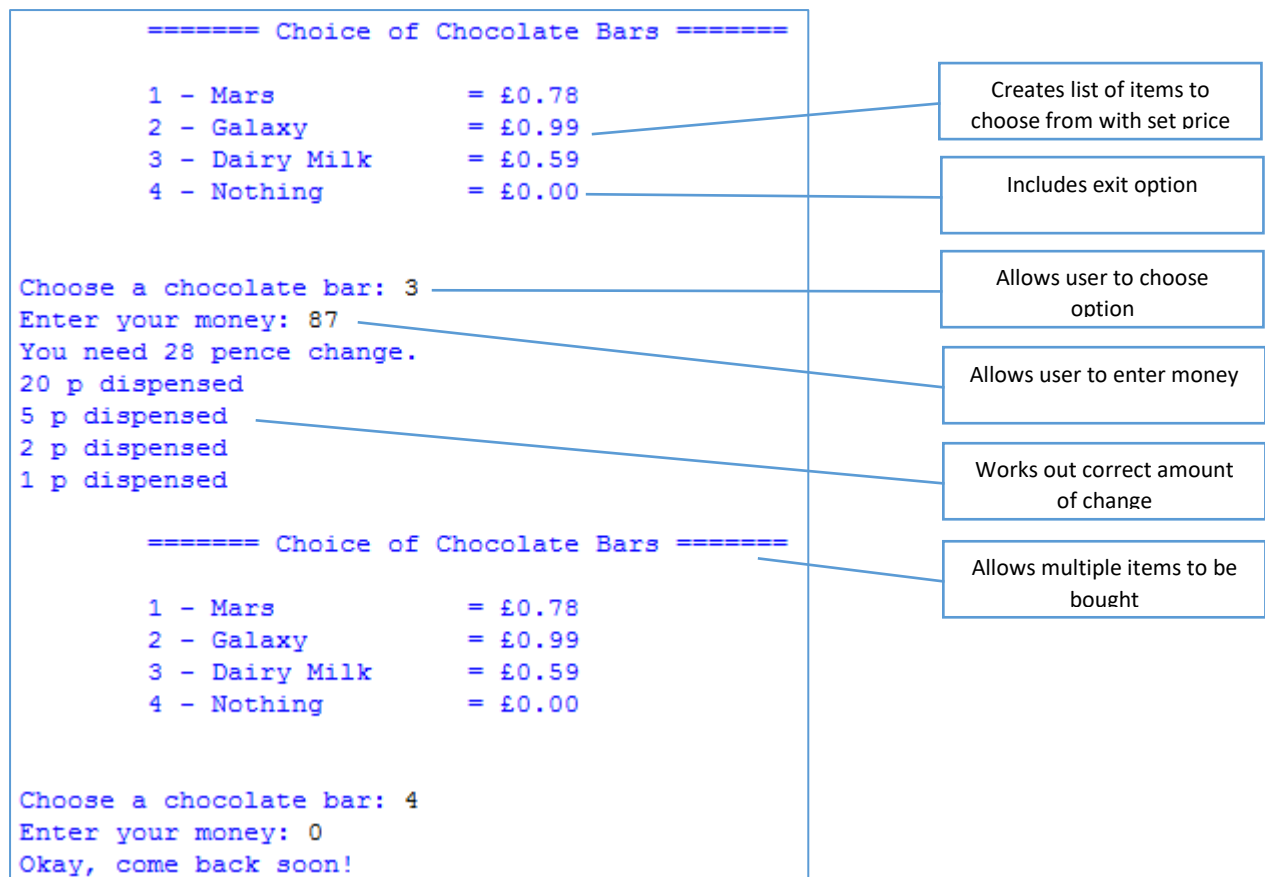
    choice = input("Choose a chocolate bar: ") #Allows the user to pick an option
    if choice != 4: #So that users don't have to pay for nothing
        paid = int(input("Enter your money: ")) #Allows the user to pay for their product

    def wchange(paid, price): #Function that works out the amount of change due
        while paid < price: #While loop using iteration to make sure valid amount paid
            print("You have not paid enough money.")
            print("You needed",price - paid,"pence more, have your",paid,"pence back and")
            paid = int(input("Enter your money again: "))
        change = paid - price #Works out the amount of change due
        print("You need",change,"pence change.")
        coins = [200,100,50,20,10,5,2,1] #List of coins in ascending order so fewest amount
        x = 0 #of coins possible outputted
        while change > 0:
            while change >= coins[x]: #Makes sure highest value of coins possible used
                change = change - coins[x] #Reduces amount of change due as it is given
                print(coins[x],"p dispensed") #Prints each amount dispensed
                x = x + 1 #Makes while loops move along list of coins when it needs to

    if choice == "4": #For if nothing is selected
        print("Okay, come back soon!")
    elif choice == "Mars" or choice == "1":
        wchange(paid,78)
    elif choice == "2" or choice == "Galaxy":
        wchange(paid,99)
    elif choice == "3" or choice == "Dairy Milk":
        wchange(paid,59)
    else:
        print("Sorry, but we don't have",choice, "chocolate bars.") #For if invalid amount entered

Ln: 72 Col: 30
```

Output of program in python shell when run:



This testing shows that the program effectively works as a chocolate vending machine and can now be used to develop a GUI using tkinter. I didn't use the testing plan I created earlier for this prototype since it does not yet create a GUI which is required to use the testing plan.

Development:

Next I started to develop my code in order to display the vending machine on a GUI using tkinter. To start with I imported everything from tkinter then tried to create an option box that allowed the user to pick one item. I used a label and a series of radio buttons created using a for loop and a list. I used radio buttons instead of normal buttons so that I could make it so that only one option could be selected.

Section of code for option box:

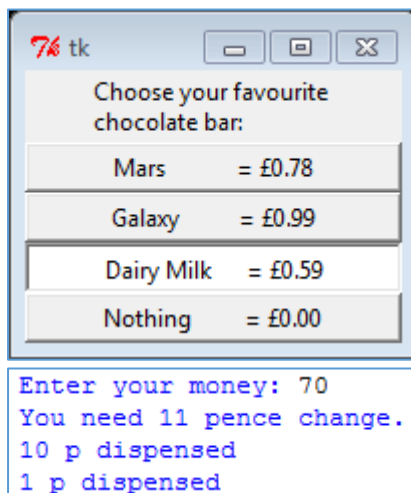
```

from tkinter import * #import everything from tkinter

#1 Options
root = Tk() #root used to put everything in the same area
price = IntVar() #means that price is the price of the option selected
price.set(1)
chocolates = [{"Mars" = £0.78",78}, {"Galaxy" = £0.99",99},
               {"Dairy Milk" = £0.59",59}, {"Nothing" = £0.00",0}]
               #what will be displayed on the button and prices
#2 Create option box
Label(root,
      text="Choose your favourite
chocolate bar:", #label to explain to user what to do with radiobuttons
      justify = LEFT, #layout of the text
      padx = 20).pack()
for i in range (0,len(chocolates)): #for loop to create buttons
    Radiobutton(root,
                text=chocolates[i][0], #text that will be displayed on button
                indicatoron = 0,
                width = 20,
                padx = 20, #size of each button
                variable=price, #used to set price depending on item chosen
                value=chocolates[i][1]).pack(anchor=W)

```

Output:



This code effectively lets you choose one item and outputs its price then works out the correct amount of change due in as few coins as possible however the window does not yet have a title and the code will have to be developed further by making it use the frame layout method in order to add other elements to the graphical user interface that still only display in the python shell.

Next I developed an entry widget for users to pay for their product with and a text widget that displays the change due with a scroll bar so that multiple product can be bought and still allow a way to read all the printed text.

Section of code for text and entry widgets:

```

#4 Paying
def ShowPaid(): #function that shows the change being dispensed
    root = Tk()
    S = Scrollbar(root) #creates a scroll bar in the text box
    T = Text(root, height=4, width=50) #creates the text box to display change
    S.pack(side=RIGHT, fill=Y) #positions the scroll bar
    T.pack(side=LEFT, fill=Y) #positions the text box
    S.config(command=T.yview)
    T.config(yscrollcommand=S.set)
    quote = "Paid: " + str(paid.get()) + "p\n"
    if int(paid.get()) > price.get(): #code from first prototype without GUI that works out change due
        change = int(paid.get()) - price.get()
        quote = quote + "You need " + str(change) + "p change.\n"
    coins = [200,100,50,20,10,5,2,1]
    x = 0
    while change > 0:
        while change >= coins[x]:
            change = change - coins[x]
            quote = quote + str(coins[x]) + "p dispensed\n"
            x = x + 1
    T.insert(END, quote) #inserts text to the text box
    mainloop()

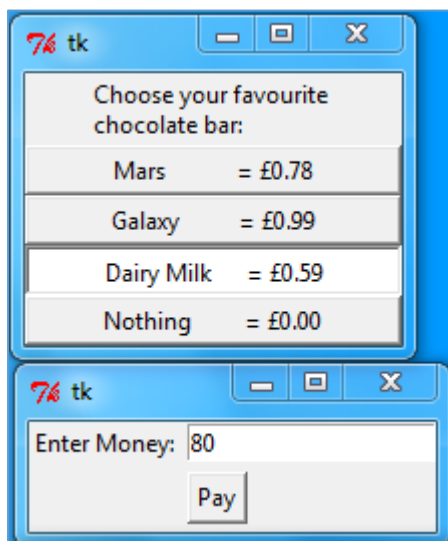
#3 Create paying box
master = Tk()
Label(master, text="Enter Money: ").grid(row=0) #label explaining to user what they put in entry widget

paid = Entry(master) #creates entry widget
paid.grid(row=0, column=1) #positions entry widget using grid method

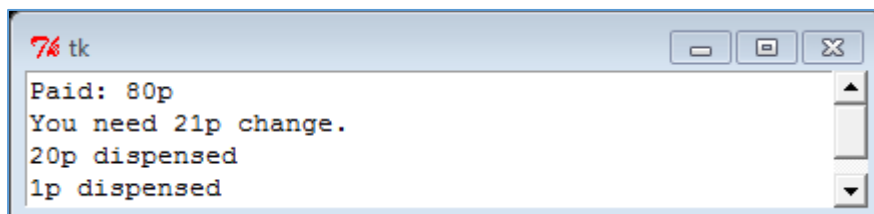
Button(master, text='Pay', command=ShowPaid).grid(row=3, column=1, sticky=W, pady=4) #creates button to output
#amount paid and trigger function
mainloop()

```

Output:



This code effectively allows users to choose a product and pay for it then display the correct amount of change due however the assets are currently in different text boxes and there still isn't a title for the window. You can currently not choose another item after buying your first so I will have to develop my code further to fix this so that it is more like a real vending machine since the purpose of the code is to be a chocolate vending machine simulation. To make it look more like a real vending machine I will use the frame layout method instead of the grid method because this means I won't have to change the grid coordinates every time I edit something or add something.



Next I further developed my code by defining a series of frames at the beginning of the code which go in different areas of one tkinter window then placing different widgets in each frame by changing 'root' to the name of the frame for example "topframe". I also created a title for the tkinter window which I learned how to do during my research.

Section of code that created frames and titled tkinter window:

```
#Frames
root = Tk() #creates a tkinter window
root.title("Chocolate Machine:") #gives the tkinter window a title
frame = Frame(root)
frame.pack()

topleftframe = Frame(root) #creates frame
topleftframe.pack(side=LEFT, anchor=N) #positions frame

toprightframe = Frame(root) #creates frame
toprightframe.pack(side=RIGHT, anchor=N) #positions frame

bottomrightframe = Frame(root) #creates frame
bottomrightframe.pack(side=RIGHT, anchor=N) #positions frame
```

Sections of code changed to use frames:

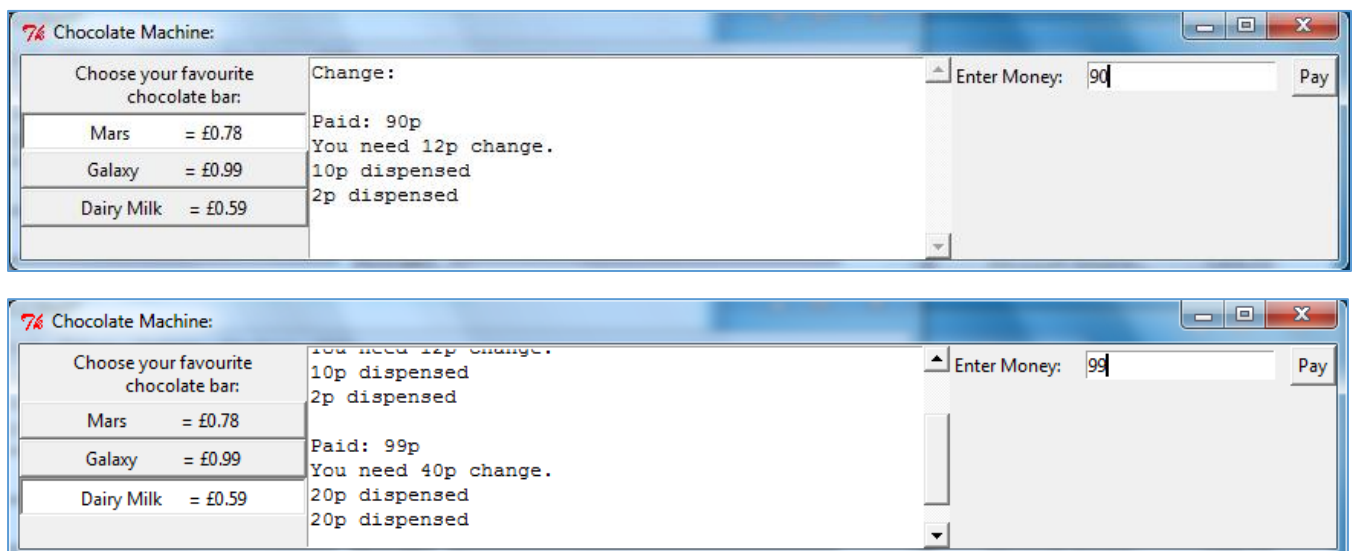
```
for i in range(0, len(chocolates)-1):
    choice = Radiobutton(topleftframe,
                        text=chocolates[i][0],
                        indicatoron=0,
                        width=20,
                        padx=20,
                        variable=price,
                        value=chocolates[i][1])
    choice.pack(side=TOP)
```

```
paying = Label(toprightframe, text="Enter Money: ")
paying.pack(side=LEFT)
paid = Entry(toprightframe)
paid.pack(padx=10, side=LEFT)

pay = Button(toprightframe, text='Pay', command=ShowPaid)
pay.pack(side=RIGHT)
```

```
S = Scrollbar(bottomrightframe)
T = Text(bottomrightframe, height=8, width=50)
```

Output:

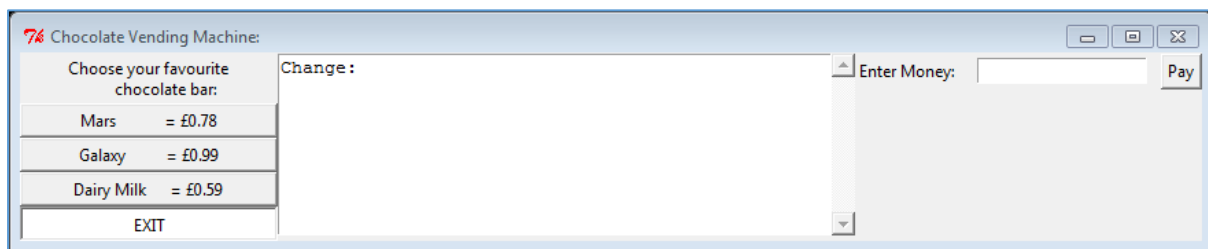


As you can see this code effectively created one tkinter window that contains a way to pick an item, a way to pay for it and a way to display the correct amount of change in a text widget with a working scroll bar without any errors occurring. This means it now completes almost every point on the success criteria with the exception of an exit button which I will now add because it allows the user to easily close the window and because the purpose of the code is to teach people tkinter and python so it will teach them how to create and exit button. I chose the name "Chocolate Machine:" because it clearly explains what the program creates.

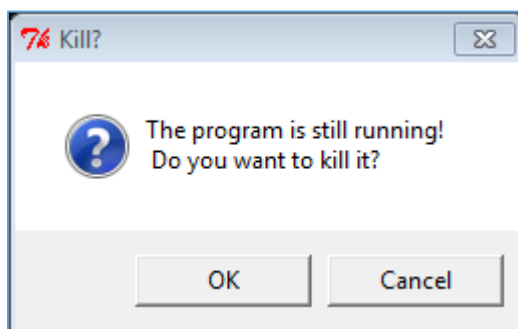
Code for exit button:

```
choice = Radiobutton(topleftframe,  
                    text=chocolates[i+1][0],  
                    indicatoron = 0,  
                    width = 20,  
                    padx = 20,  
                    variable=price,  
                    command=quit,  
                    value=chocolates[i+1][1])  
choice.pack(side=TOP)
```

Output:



When exit button clicked:



The exit button now successfully ends the program if you select the OK button on the “Kill?” window. If you select the cancel button the code continues as normal allowing you to choose an item. As you can see in the output I changed the windows title so that it states exactly what the program does. I choose to layout the widgets in the tkinter window this way because it is easy for users to understand what each part of the GUI is for and what they are meant to do with it as each frame is labelled. This means it makes a better learning tool which it is likely to be used for. The program now meets all the points on the success criteria so no further development is required.

Final Code

To aid future maintenance of the system I annotated my final code using comments (the red text) to explain each part and I spaced out each section of the code and gave it a title which stated the purpose of the section of code and which frame it would be displaying widgets in. This makes it much easier to understand for people that have not programmed things like this before which helps the program to fulfil one of its purposes which is to be able to be used as an educational resource.


```

## CHOCOLATE VENDING MACHINE V4 (frame layout)##
from tkinter import * #imports everything from tkinter
##Frames##
root = Tk() #creates a tkinter window
root.title("Chocolate Vending Machine:") #gives the tkinter window a title
frame = Frame(root)
frame.pack()

topleftframe = Frame(root) #creates frame
topleftframe.pack(side=LEFT,anchor=N)#positions frame in top left of window

toprightframe = Frame(root)#creates frame
toprightframe.pack(side=RIGHT,anchor=N)#positions frame in top right of window

centreframe = Frame(root)#creates frame
centreframe.pack(side=RIGHT,anchor=N)#positions frame in centre of window

##Work out and display change##
def ShowPaid(): #creates function to work out change due
    quote = "Paid: " + str(paid.get()) + "p\n" #adds amount paid to quote, \n move onto next line
    if int(paid.get()) > price.get(): #selection and iteration so that only works out change if change is needed
        change = int(paid.get()) - price.get()#works out amount of change needed
        quote = quote + "You need " + str(change) + "p change.\n" #adds amount of change needed to quote
        coins = [200,100,50,20,10,5,2,1] #array of the different coins that a vending machine can dispence
        x = 0 #variable used to move along the list
        while change > 0: #while loop and iteration so that only dispences coins if change is still needed
            while change >= coins[x]: #while loop and iteration so that coins of highest value possible dispensed
                change = change - coins[x] #amount of change due is decreased as change is given
                quote = quote + str(coins[x]) + "p dispensed\n" #add the coins dispensed to 'quote' variable,
                #used casting to make coins str becuse rest of quote is string
            x = x + 1 #to move along list when coin has too high a value
        T.insert(END, quote+"\n")#insert 'quote' to text widget

##Option Box (Top Left)##
price = IntVar()#makes price the variable
price.set(1)
chocolates = [{"Mars" = £0.78",78},{"Galaxy" = £0.99",99},{"Dairy Milk" = £0.59",59},{"EXIT",0}]
    #creates list of items available with their set prices
choose = Label(topleftframe, text="Choose your favourite
chocolate bar:") #radiobuttons labelled so user knows what they are for, label put in top left frame
choose.pack(side=TOP) #positions label at top of frame

for i in range (0,len(chocolates)-1):#for loop creates button for each item in list
    choice = Radiobutton(topleftframe, #puts radiobutton in top left frame
        text=chocolates[i][0], #puts name of chocolate bar and price on button
        indicatoron = 0, #make the radiobuttons design that they are
        width = 20, #defines width of each button
        padx = 20, #makes all the buttons the same dimensions
        variable=price, #makes the variable the price
        value=chocolates[i][1]) #makes the value the price
    choice.pack(side=TOP) #positions each radiobutton

choice = Radiobutton(topleftframe, #creates another radiobutton in top left frame
    text=chocolates[i+1][0], #puts "EXIT" on button
    indicatoron = 0, #makes the radiobutton the design that it is
    width = 20, #defines the width of the button
    padx = 20,#makes the button the same as the other radiobuttons created
    variable=price,#makes the variable the price
    command=quit, #so that when clicked the program will end
    value=chocolates[i+1][1]) #makes the value the price
choice.pack(side=TOP) #positions the radio button in line with the others

##Paying box (Top Right)##
paying = Label(toprightframe, text="Enter Money: ") #creates label for entry widget
paying.pack(side=LEFT) #positions the label
paid = Entry(toprightframe) #creates an entry widget in the top right frame
paid.pack(padx=10,side=LEFT) #positions the entry widget and defines its size

pay = Button(toprightframe, text='Pay', command=ShowPaid)#creates a button that starts the ShowPaid function when clicked
pay.pack(side=RIGHT)#positions the pay button next to the entry widget

##Process box (Centre)##
S = Scrollbar(centreframe)#creates a scroll bar in the centre frame so user can view all text
T = Text(centreframe, height=8, width=50)#creates text widget in centre frame an defines dimensions
S.pack(side=RIGHT, fill=Y)#positions scroll bar and makes it same height as text widget
T.pack(side=LEFT, fill=Y)#positons text widget and makes it same height as scroll bar
S.config(command=T.yview)#makes 'S' a working scroll bar
T.config(yscrollcommand=S.set)#makes 'S' a working scroll bar
T.insert(END, "Change:\n\n")#inserts title to text box so user knows what it is displaying

##Loop##
root.mainloop( )

```

Testing Final Code

To test the final code, I will use the testing plan I created during the design section as a template and add the columns 'Results' and 'Changes Needed' to keep track of the outcome. Some of the test data also needed to be changed because of changes in my code.

Function being Tested:	Test Data:	Expected Result:	Results:	Changes Needed:
Whether the program creates a GUI using python.	Run the python 3.3.2 code by pressing f5	A tkinter window titled "Chocolate Machine: " with all the required assets inside	A tkinter window titled "Chocolate Vending Machine: " with all the required assets inside.	None needed but could display images.
Whether only one of the radio buttons with items on can be selected at once.	Click Mars button then Galaxy button	Only Mars button is selected then only Galaxy button	Only Mars button is selected then only Galaxy button	None needed
Whether clicking on an items buttons outputs the correct price for that buttons.	Make code print price, click Diary Milk button and pay for it	59p printed in python shell	59 printed in python shell	None needed
Whether the 'EXIT' button exists the GUI and ends the program.	Click on the exit button	The program ends	A window opens asking you if you want to end the program, if yes is clicked the python shell and tkinter window close	None needed
What happens when invalid data is entered to the entry widget.	Enter 'fifty-nine' to text widget	Nothing happens so the change is not worked out allowing the user to enter valid data	Error text in python shell but code still works with no further issues	None needed but could prevent error text appearing in shell
What happens when invalid data is entered to the entry widget.	Select Mars and enter 70p. (70p is less than the price of Mars)	The change is not worked out allowing the user to enter valid data	"Invalid Amount" printed in text widget allowing user to enter valid data	None needed
Whether the correct amount of change in as few coins as possible is outputted.	Select Mars and enter 99p	'20p dispensed 1p dispensed'	"Paid: 99p You need 21p change. 20p dispensed 1p dispensed"	None needed

Whether the scroll bar on the text widget allows the user to scroll up and down the text.	Select Mars pay 99p, Select Dairy Milk pay 100p	The scroll bar will start working allowing the user to view all the text.	The scroll bar will start working allowing the user to view all the text.	None needed
---	--	---	---	-------------