""" Group: Group 1

    Names: Kyle Ingersoll, Dylan Howard, Vanya Bays

    Date: 4/17/2024

    Summary: This program is a partial remake of the CSCI101 Final Project

    by Kyle Ingersoll, in Python using Tkinter.

"""

import tkinter as tk # for the gui mainloop and to be easier to write and read

from tkinter import ttk # for the gui widgets

from PIL import ImageTk, Image # with the default tkinter Photolabel having errors with .jpg images,

                               # this import is needed to fix them

from dataclasses import dataclass # the dataclasses are used to replace the structs in the original

                                  # C++ code

@dataclass

class reservationType:

    """ This is our reservation dataclass, containing the customer's name,

        the time they will arrive at the restaurant, the number of people in the reservation, and whether

        the reservation is checked in.

    """

    customerName: str

    time: str

    numberOfPeople: int = 0

    checkInReservation: bool = False

@dataclass

class tableType:

    """ This is our table dataclass, containing the reservation the table is checked-in to, the maximum amount of

        people who can sit in the table, the number of people who are sitting at the table, the table number,

        and whether the table is being used or not.

    """

    reservation: reservationType

    maximumNumberOfPeopleWhoCanSitInThem: int = 10 # altered from the original C++ coce for simplicity's sake

    numberOfPeopleCurrentlySittingAtTheTable: int = 0

    tableNumber: int = 0

    tableUsage: bool = False

def welcome\_screen(reservations: list[reservationType], tables: list[tableType]) -> None:

    """ This is the beginning of our application, a menu

        where the user can select from two options, Make Reservation and

        Check-In Reservation.

    """

    # note: use Toplevel windows when you need to call the window function more than once

    # initalize Welcome Screen

    welcome\_screen\_gui = tk.Toplevel()

    # create title for window

    welcome\_screen\_gui.title("Welcome Screen")

    # set window size in pixels

    welcome\_screen\_gui.geometry("720x720")

    # prevents you from resizing the welcome screen window

    welcome\_screen\_gui.resizable(0,0)

    # sets the title for the welcome screen

    welcome\_screen\_title\_label = ttk.Label(welcome\_screen\_gui, text='Welcome to Messijoes', font=("Times New Roman", 32))

    welcome\_screen\_title\_label.grid(row = 0, column = 3)

    # sets up the stock image of an open sign in the welcome screen

    """ Note: tkinter doesn't support .jpg images! Use .png images instead or

        convert .jpg images to .png images.

    """

    photo1 = ImageTk.PhotoImage(Image.open('./Open\_Sign\_Photo\_For\_Welcome\_Screen.png')) # using ImageTK just in case

    photo1\_label = ttk.Label(welcome\_screen\_gui, image=photo1, padding=5)

    photo1\_label.grid(row = 5, column = 3)

    # set up both make\_reservation and check\_in\_reservation menu buttons in the Welcome Screen

    """ Note: Do not use grid and pack methods with objects that have the same parent in tkinter!

        It will create an error! Use either all grid or all pack.

    """

    make\_reservation\_button = ttk.Button(welcome\_screen\_gui, text="Make Reservation", command=lambda: make\_reservation(reservations, tables))

    make\_reservation\_button.grid(row=1, column=0)

    check\_in\_reservation\_button = ttk.Button(welcome\_screen\_gui, text="Check-In Reservation", command=lambda: check\_in\_reservation(reservations, tables))

    check\_in\_reservation\_button.grid(row=1, column=5)

    # todo: make exit button, since the program won't shut down on it's own even after you close all the visible windows due to the withdrawn root window

    # begin welcome screen

    welcome\_screen\_gui.mainloop()

    return

"""

    # These are the old functions to verify the number of people in the party input field in make\_reservation function.

    # I would note that trying to implement these caused various widgets to disappear and we lacked the time to solve that issue.

    # Thus the code is commented out to avoid accidentally calling it or running it.

    def verify\_number\_of\_people\_in\_the\_party\_input\_field\_in\_make\_reservation(number\_of\_people\_in\_the\_party\_integer: tk.IntVar) -> bool:

        """

"""

            This function verifies that the number of people in the party input field in make reseration

            has a number between 1 and 10 in it. It returns False if text or a float number

            was entered into the field, and returns False if a number was entered into the field

            but it was not between 1 and 10. If neither of those cases are true, then it returns True.

        """

"""

        if number\_of\_people\_in\_the\_party\_integer.get() == 0:

            return True

        elif type(number\_of\_people\_in\_the\_party\_integer.get()) == int:

            return True

        elif number\_of\_people\_in\_the\_party\_integer.get() < 1 or number\_of\_people\_in\_the\_party\_integer.get() > 10:

            return True

        else:

            return True

        """

"""

    def when\_invalid() -> None:

"""

"""     When the input is invalid, a message is displayed to the user stating that the input is wrong and

        that they should reenter it.

        """

"""

        when\_invalid\_gui = tk.Toplevel()

        when\_invalid\_gui.title("Error Message")

        when\_invalid\_gui.geometry("480x480")

        when\_invalid\_gui.resizable(0,0)

        when\_invalid\_error\_message\_text = ttk.Label(when\_invalid\_gui, text='The inputted value you entered was either not a number, or was not between 1 and 10. Put the number of people in the party in numerical form.', font=("Times New Roman", 12) )

        when\_invalid\_error\_message\_text.pack()

        when\_invalid\_gui.mainloop()

"""

def make\_reservation(reservations: list[reservationType], tables: list[tableType]) -> None:

    """ make\_reservation is the first in a series of functions for the Make Reservation choice

        on the Welcome Screen Menu. The user enters a name for the reservation, enters the number

        of people in the party (which is input verified) and enters the time for the reservation in

        HH::MM AM/PM. After that, the user clicks Next to go to the confirm\_reservation function/page.

    Args:

        reservations (list[reservationType]): A list of reservationType

        tables (list[tableType]): A list of tableType

    """

    # close welcome screen

    # make name input box variable

    name\_string = tk.StringVar()

    # make number of people input box variable

    number\_of\_people\_in\_the\_party\_integer = tk.IntVar()

    # make time input box variable

    time\_for\_the\_reservation\_string = tk.StringVar()

    # initialize window

    make\_reservation\_gui = tk.Toplevel()

    # Add title for window

    make\_reservation\_gui.title("Make Reservation Screen")

    # set window size in pixels

    make\_reservation\_gui.geometry("720x720")

    # prevents you from resizing the make reservation screen

    make\_reservation\_gui.resizable(0,0)

    # note: don't make frames or entry widgets ttk, make them tk, or they won't show up on the screen

    # sets the title for the make reservation screen

    make\_reservation\_title\_label = ttk.Label(make\_reservation\_gui, text='Make Reservation', font=("Times New Roman", 32))

    make\_reservation\_title\_label.grid(row = 0, column = 0)

    # make a label so that the user knows to enter their name in the associated input box

    make\_reservation\_name\_label = ttk.Label(make\_reservation\_gui, text='Enter a name for the reservation:', font=("Times New Roman", 12))

    make\_reservation\_name\_label.grid(row = 1, column = 0)

    # make the name input box

    make\_reservation\_name\_input\_window = tk.Frame(make\_reservation\_gui, width=100, height=50, bd=0, highlightbackground="black", highlightcolor="black", highlightthickness=2)

    make\_reservation\_name\_input\_window.grid(row = 1, column = 1)

    make\_reservation\_name\_input\_field = tk.Entry(make\_reservation\_name\_input\_window, font=('times new roman', 12, 'bold'), textvariable=name\_string, width=50, bg="#eee", bd=2, justify='left')

    make\_reservation\_name\_input\_field.pack() # You need to use the pack function here in order for the input field to work

    make\_reservation\_name\_input\_field.focus\_set()

    """

        # todo: fix other two labels and textboxes below not showing up when this is above them

        # note: it seems like the input validation and number of people in the party code block is the problem with the other labels and input boxes not showing up

        # make reservation number of people in the party input validation validate command and invalid command

        ver = make\_reservation\_gui.register(verify\_number\_of\_people\_in\_the\_party\_input\_field\_in\_make\_reservation(number\_of\_people\_in\_the\_party\_integer))

        inv = make\_reservation\_gui.register(when\_invalid())

        validate\_command = (ver, '%P')

        invalid\_command = (inv)

    """

    # make the number of people in the party label for the user

    make\_reservation\_number\_of\_people\_in\_the\_party\_label = ttk.Label(make\_reservation\_gui, text='Enter the number of people in the party (between 1 and 10 people) in numerical form:', font=("Times New Roman", 12))

    make\_reservation\_number\_of\_people\_in\_the\_party\_label.grid(row = 2, column = 0)

    # todo: add input verification here

    # make the number of people in the party input box

    make\_reservation\_number\_of\_people\_in\_the\_party\_input\_window = tk.Frame(make\_reservation\_gui, width=100, height=50, bd=0, highlightbackground="black", highlightcolor="black", highlightthickness=2)

    make\_reservation\_number\_of\_people\_in\_the\_party\_input\_window.grid(row = 2, column = 1)

    make\_reservation\_number\_of\_people\_in\_the\_party\_input\_field = tk.Entry(make\_reservation\_number\_of\_people\_in\_the\_party\_input\_window, font=('times new roman', 12, 'bold'), textvariable=number\_of\_people\_in\_the\_party\_integer, width=50, bg="#eee", bd=2, justify='left')

    make\_reservation\_number\_of\_people\_in\_the\_party\_input\_field.pack()

    """

        # Below is more validation code I didn't end up implementing:

        make\_reservation\_number\_of\_people\_in\_the\_party\_input\_field.config(validate="focus", validatecommand=validate\_command, invalidcommand=invalid\_command)

    """

    # make time label for the user's sake

    make\_reservation\_time\_label = ttk.Label(make\_reservation\_gui, text='Enter the time for the reservation in HH::MM AM/PM:', font=("Times New Roman", 12))

    make\_reservation\_time\_label.grid(row = 5, column = 0)

    # make time input box

    make\_reservation\_time\_input\_window = tk.Frame(make\_reservation\_gui, width=400, height=50, bd=0, highlightbackground="black", highlightcolor="black", highlightthickness=2)

    make\_reservation\_time\_input\_window.grid(row = 5, column = 1)

    make\_reservation\_time\_input\_field = tk.Entry(make\_reservation\_time\_input\_window, font=('times new roman', 12, 'bold'), textvariable=time\_for\_the\_reservation\_string, width=50, bg="#eee", bd=2, justify='left')

    make\_reservation\_time\_input\_field.pack()

    # make button to verify contents of make\_reservation\_number\_of\_people\_in\_the\_party\_input\_field and to go to Confirm Reservation Screen

    # note: using get() directly on StringVar() or IntVar() variables doesn't work, you have to use get() on the entry widget itself

    make\_reservation\_next\_button = ttk.Button(make\_reservation\_gui, text="Next", command=lambda: make\_reservation\_bind\_tkinter\_variables\_to\_newReservation(reservations, tables, make\_reservation\_name\_input\_field.get(),  make\_reservation\_number\_of\_people\_in\_the\_party\_input\_field.get(), make\_reservation\_time\_input\_field.get()))

    make\_reservation\_next\_button.grid(row = 7, column = 0)

    # begin make reservation screen

    make\_reservation\_gui.mainloop()

    return

def make\_reservation\_bind\_tkinter\_variables\_to\_newReservation(reservations: list[reservationType], tables: list[tableType], customer\_name: str, number\_of\_people\_in\_the\_party: int, time\_for\_the\_reservation: str) -> None:

    """ When the user decides to click the next button in Make Reservation Screen, it takes the reservations and tables lists, as well as the .get() for the 3 previous entry boxes

        And puts them into new\_reservation to be passed along to Confirm Reservation Screen.

    Args:

        reservations (list[reservationType]): the list of reservations that are not checked-in

        tables (list[tableType]): the list of tables that are avalible

        customer\_name (str): The customer's name as a string

        number\_of\_people\_in\_the\_party (int): The number of people in the party as an integer

        time\_for\_the\_reservation (str): The reservation time as a string

    """

    # add reservation dataclass to store data from tkinter variables

    new\_reservation = reservationType(customer\_name, time\_for\_the\_reservation, number\_of\_people\_in\_the\_party, False)

    # pass data to confirm reservation

    confirm\_reservation(new\_reservation, reservations, tables)

    return

def confirm\_reservation(new\_reservation: reservationType, reservations: list[reservationType], tables: list[tableType]) -> None:

    """ We allow the user to see what they put for their reservation, and decide whether they say yes, no, or cancel.

        If the user chooses to say yes, they return to welcome screen with the reservation added to the reservations list.

        If the user chooses to say no, they return to make reservation screen to fill in their reservation.

        If the user chooses to cancel the reservation, they return to welcome screen with no reservation added to reservation list.

    Args:

        new\_reservation (reservationType): The reservation made in the previous screen.

        reservations (list[reservationType]): the list of reservations that are not checked-in

        tables (list[tableType]): the list of tables that are avalible

    """

    # set up Confirm Reservation screen

    confirm\_reservation\_gui = tk.Toplevel()

    confirm\_reservation\_gui.geometry("720x720")

    confirm\_reservation\_gui.title("Confirm Reservation Screen")

    confirm\_reservation\_gui.resizable(0,0)

    # set up confirm reservation title label

    confirm\_reservation\_title\_label = ttk.Label(confirm\_reservation\_gui, text='Confirm Reservation', font=("Times New Roman", 32))

    confirm\_reservation\_title\_label.grid(row = 0, column = 0)

    # set up confirm reservation please confirm the reservation label

    confirm\_reservation\_please\_confirm\_the\_reservation\_label = ttk.Label(confirm\_reservation\_gui, text='Please confirm the reservation:', font=("Times New Roman", 12))

    confirm\_reservation\_please\_confirm\_the\_reservation\_label.grid(row = 1, column = 0)

    # todo: get new\_reservation variables to not be blank, completed

    # set up customer name label

    confirm\_reservation\_customer\_name\_label = ttk.Label(confirm\_reservation\_gui, text='Customer Name: ' + new\_reservation.customerName, font=("Times New Roman", 12))

    confirm\_reservation\_customer\_name\_label.grid(row = 2, column = 1)

    # set up reservation time label

    confirm\_reservation\_reservation\_time\_label = ttk.Label(confirm\_reservation\_gui, text='Reservation Time: ' + new\_reservation.time, font=("Times New Roman", 12))

    confirm\_reservation\_reservation\_time\_label.grid(row = 3, column = 1)

    # set up number of people in party label

    confirm\_reservation\_number\_of\_people\_in\_party\_label = ttk.Label(confirm\_reservation\_gui, text='Number of people in party: ' + new\_reservation.numberOfPeople, font=("Times New Roman", 12))

    confirm\_reservation\_number\_of\_people\_in\_party\_label.grid(row = 4, column = 1)

    # set up is this information correct label

    confirm\_reservation\_is\_this\_information\_correct\_label =  ttk.Label(confirm\_reservation\_gui, text='Is this information correct: ', font=("Times New Roman", 12))

    confirm\_reservation\_is\_this\_information\_correct\_label.grid(row = 5, column = 0)

    # set up yes button

    confirm\_reservation\_yes\_button = tk.Button(confirm\_reservation\_gui, text='Yes', command=lambda: confirm\_reservation\_yes\_option(new\_reservation, reservations, tables))

    confirm\_reservation\_yes\_button.grid(row = 6, column = 1)

    # set up cancel button

    confirm\_reservation\_cancel\_button = tk.Button(confirm\_reservation\_gui, text='Cancel', command=lambda: confirm\_reservation\_cancel\_option(reservations, tables))

    confirm\_reservation\_cancel\_button.grid(row = 7, column = 0)

    # set up no button

    confirm\_reservation\_no\_button = tk.Button(confirm\_reservation\_gui, text='No', command=lambda: confirm\_reservation\_no\_option(reservations, tables))

    confirm\_reservation\_no\_button.grid(row = 7, column = 1)

    # begin confirm reservation screen main loop

    confirm\_reservation\_gui.mainloop()

def confirm\_reservation\_yes\_option(new\_reservation: reservationType, reservations: list[reservationType], tables: list[tableType]) -> None:

    """ This function activates when the user presses the Yes button in Confirm Reservation Screen. It appends new\_reservation to the reservations list,

        and then passes the reservations and tables lists back to the Welcome Screen.

    Args:

        new\_reservation (reservationType): The reservation the customer is currently entering

        reservations (reservationType): The list of reservations that the customer entered

        tables (tableType): the list of tables avalible for the customer

    """

    # add new\_reseration at the end of the reservations list

    reservations.append(new\_reservation)

    # return to welcome screen with reservation added to reservations list

    welcome\_screen(reservations, tables)

    return

def confirm\_reservation\_cancel\_option(reservations: list[reservationType], tables: list[tableType]) -> None:

    """ If the user decides to cancel their reservation, it returns them to Welcome Screen without making any

        changes in the reservations or tables lists.

    Args:

        reservations (list[reservationType]): The list of reservations not checked-in

        tables (list[tableType]): The list of all tables

    """

    # return to welcome screen with no reservation added to reservations list

    welcome\_screen(reservations, tables)

    return

def confirm\_reservation\_no\_option(reservations: list[reservationType], tables: list[tableType]) -> None:

    """ If the user decides to say that they are not satisfied with the reservation, they are returned to Make Reservation

        Screen without saving anything to reservations or tables lists to redo their reservation.

    Args:

        reservations (list[reservationType]): The list of reservations not checked-in

        tables (list[tableType]): The list of all tables

    """

    # return to Make Reservation Screen to make new reservation

    make\_reservation(reservations, tables)

    return

def check\_in\_reservation(reservations: list[reservationType], tables: list[tableType]) -> None:

    """ This function is chosen directly from Welcome Screen, and is the second step the user will do after Make Reservation.

        Check-In Reservation Screen tells the user to check in a reservation from a drop down menu.

        After they are done, they will click the next button, which will bring them to check\_in\_reservation\_next\_button\_command function,

        which runs in the background and then brings them to check\_in\_reservation\_assign\_reservation\_to\_table\_screen.

    Args:

        reservations (list[reservationType]): The list of reservations pending

        tables (list[tableType]): The list of all tables

    """

    # initialize Check-In Reservation Screen

    check\_in\_reservation\_gui = tk.Toplevel()

    check\_in\_reservation\_gui.title("Check-In Reservation Screen")

    check\_in\_reservation\_gui.geometry("720x720")

    check\_in\_reservation\_gui.resizable(0,0)

    # Title label for Check-In Reservation Screen

    check\_in\_reservation\_title\_label = ttk.Label(check\_in\_reservation\_gui, text='Check-In Reservation', font=("Times New Roman", 32))

    check\_in\_reservation\_title\_label.grid(row = 0, column = 0)

    # Check-In Reservation Screen choose the reservation to check-in label

    check\_in\_reservation\_choose\_the\_reservation\_to\_check\_in\_label = ttk.Label(check\_in\_reservation\_gui, text='Choose the reservation to check-in:', font=("Times New Roman", 12))

    check\_in\_reservation\_choose\_the\_reservation\_to\_check\_in\_label.grid(row = 1, column = 0)

    # Initialize Check-In Reservation Screen reservation choice variable

    check\_in\_reservation\_reservation\_choice\_variable = tk.StringVar(check\_in\_reservation\_gui)

    # create check-in reservation reservation options list

    check\_in\_reservation\_reservation\_options = []

    # basically, if the reservation isn't checked-in, append the customer name to the reservation options list

    for i in reservations: # i is used to iterate across reservations in this function

        if i.checkInReservation == False:

            check\_in\_reservation\_reservation\_options.append(i.customerName)

    # initialize drop down menu

    check\_in\_reservation\_reservation\_choice\_drop\_down\_menu\_option\_menu = tk.OptionMenu(check\_in\_reservation\_gui, check\_in\_reservation\_reservation\_choice\_variable, "Options selectable are:", \*check\_in\_reservation\_reservation\_options)

    check\_in\_reservation\_reservation\_choice\_drop\_down\_menu\_option\_menu.grid(row = 2, column = 0)

    # make check-in reservation next button

    check\_in\_reservation\_next\_button = tk.Button(check\_in\_reservation\_gui, text = "Next", command=lambda: check\_in\_reservation\_next\_button\_command(reservations, tables, check\_in\_reservation\_reservation\_choice\_variable.get()))

    check\_in\_reservation\_next\_button.grid(row = 3, column = 1)

    # initialize infinite loop that keeps Check-In Reservation Screen on

    check\_in\_reservation\_gui.mainloop()

    return

def check\_in\_reservation\_next\_button\_command(reservations: list[reservationType], tables: list[tableType], selected\_option: str) -> None:

    """ When the next button in Check-In Reservation Screen is pressed, a empty reservation is created and the selected option

        is compared against each reservation in reservations. If the reservation in reservations isn't checked in and the customer name for that

        reservation equals the selected option, then the blank reservation is assigned to the reservation in reservations and we head to the Check-In Reservation

        Assign Reservation to Table Screen.

    """

    # initialize selected\_reservation variable

    selected\_reservation: reservationType = reservationType("", "", 0, False)

    # basically, if the reservation isn't checked-in, append the customer name to the reservation options list

    for j in reservations: # j is used to iterate across reservations in this function

        if j.checkInReservation == False and j.customerName == selected\_option:

            selected\_reservation = j

            check\_in\_reservation\_assign\_reservation\_to\_table\_screen(reservations, tables, selected\_reservation)

    return

def check\_in\_reservation\_assign\_reservation\_to\_table\_screen(reservations: list[reservationType], tables: list[tableType], selected\_reservation: reservationType) -> None:

    """ We assign a available table to the reservation in this screen through a drop down menu,

        and when the user is done they click the next button. This also shows an image of multiple tables in a row (2nd image in this program)

        to fulfill assignment requirements.

    """

    # initialize Check-In Reservation Assign Reservation to Table Screen

    check\_in\_reservation\_assign\_reservation\_to\_table\_screen\_gui = tk.Toplevel()

    check\_in\_reservation\_assign\_reservation\_to\_table\_screen\_gui.title("Check-In Reservation Assign Reservation to Table Screen")

    check\_in\_reservation\_assign\_reservation\_to\_table\_screen\_gui.geometry("720x720")

    check\_in\_reservation\_assign\_reservation\_to\_table\_screen\_gui.resizable(0,0)

    # Title label for Check-In Reservation Assign Reservation to Table Screen

    check\_in\_reservation\_assign\_reservation\_to\_table\_screen\_title\_label = ttk.Label(check\_in\_reservation\_assign\_reservation\_to\_table\_screen\_gui, text='Assign Reservation to Table Screen', font=("Times New Roman", 32))

    check\_in\_reservation\_assign\_reservation\_to\_table\_screen\_title\_label.grid(row = 0, column = 0)

    # Check-In Reservation Assign Reservation to Table Screen Please assign a table to the reservation label

    check\_in\_reservation\_assign\_reservation\_to\_table\_screen\_please\_assign\_a\_table\_to\_the\_reservation\_label = ttk.Label(check\_in\_reservation\_assign\_reservation\_to\_table\_screen\_gui, text='Please assign a table to the reservation:', font=("Times New Roman", 12))

    check\_in\_reservation\_assign\_reservation\_to\_table\_screen\_please\_assign\_a\_table\_to\_the\_reservation\_label.grid(row = 1, column = 0)

    # Initialize Check-In Reservation Screen Assign Reservation to Table Screen Table Choice variable

    check\_in\_reservation\_assign\_reservation\_to\_table\_screen\_table\_choice\_variable = tk.StringVar(check\_in\_reservation\_assign\_reservation\_to\_table\_screen\_gui)

    # create check-in reservation assign reservation to table screen table options list

    check\_in\_reservation\_assign\_reservation\_to\_table\_screen\_table\_options = []

    # basically, if there are no reservation checked-in to the table, append the table number to the table options list

    for k in tables: # k serves to iterate through tables in this function

        if k.reservation.checkInReservation == False:

            check\_in\_reservation\_assign\_reservation\_to\_table\_screen\_table\_options.append(str(k.tableNumber))

    # initialize drop down menu

    check\_in\_reservation\_table\_choice\_drop\_down\_menu\_option\_menu = tk.OptionMenu(check\_in\_reservation\_assign\_reservation\_to\_table\_screen\_gui, check\_in\_reservation\_assign\_reservation\_to\_table\_screen\_table\_choice\_variable, "Options selectable are:", \*check\_in\_reservation\_assign\_reservation\_to\_table\_screen\_table\_options)

    check\_in\_reservation\_table\_choice\_drop\_down\_menu\_option\_menu.grid(row = 2, column = 0)

    # make check-in reservation assign reservation to table screen next button

    check\_in\_reservation\_assign\_reservation\_to\_table\_screen\_next\_button = tk.Button(check\_in\_reservation\_assign\_reservation\_to\_table\_screen\_gui, text = "Next", command=lambda: check\_in\_reservation\_assign\_reservation\_to\_table\_screen\_next\_button\_command(reservations, tables, selected\_reservation, int(check\_in\_reservation\_assign\_reservation\_to\_table\_screen\_table\_choice\_variable.get())))

    check\_in\_reservation\_assign\_reservation\_to\_table\_screen\_next\_button.grid(row = 3, column = 1)

    # open and resize photo2 for Check-In Reservation Assign Reservation to Title screen

    photo2 = Image.open("./Restaurant\_Table\_For\_Check\_In\_Reservation\_Assign\_Reservation\_to\_Table\_Screen.png").resize((480,480)) # when resizing an image, pass the pixel dimensions of the resized image as a tuple

    # put the photo2 in PhotoImage so it can fit in a label

    resized\_photo2 = ImageTk.PhotoImage(image=photo2)

    # make photo2 label to store image in

    photo2\_label = ttk.Label(check\_in\_reservation\_assign\_reservation\_to\_table\_screen\_gui, image=resized\_photo2, padding=5)

    photo2\_label.grid(row = 4, column = 0)

    # initialize infinite loop that keeps Check-In Reservation Assign Reservation to Table Screen on

    check\_in\_reservation\_assign\_reservation\_to\_table\_screen\_gui.mainloop()

    return

def check\_in\_reservation\_assign\_reservation\_to\_table\_screen\_next\_button\_command(reservations: list[reservationType], tables: list[tableType], selected\_reservation: reservationType, table\_choice: int) -> None:

    """ Basically, when the next button in Check-In Reservation Assign Reservation to Table Screen is pressed,

        if table number equals table choice, copy over the information from selected reservation to that

        table's reservation except for ensuring that that table's checkInReservation variable is true.

        It also checks the reservations list afterwards to find where it matches with selected reservation and deletes that reservation from the reservations list.

        This is to avoid binding the same reservation to multiple tables.

    """

    n: int = 0 # serves as a counter variable to determine the index of m while in the loop

    for l in tables: # l serves to iterate through tables in this function

        if l.tableNumber == table\_choice:

            l.reservation.checkInReservation = True

            l.reservation.customerName = selected\_reservation.customerName

            l.reservation.time = selected\_reservation.time

            l.reservation.numberOfPeople = selected\_reservation.numberOfPeople

            for m in reservations: # m serves to iterate through reservations in this function

                if m.customerName == selected\_reservation.customerName and m.numberOfPeople == selected\_reservation.numberOfPeople and m.time == selected\_reservation.time and m.checkInReservation == selected\_reservation.checkInReservation:

                    reservations.pop(n)

                n += 1

            welcome\_screen(reservations, tables)

    return

# main part of code, initializes the memory and program

if \_\_name\_\_ == "\_\_main\_\_":

    """ We intitalize the two main lists of dataclasses here, reservations for the list of

    reservation dataclasses, and tables for the list of table dataclasses.

    We also initialize the root screen and make it visible to the user so that the user

    can completely close the application by exiting out of the root window.

    Then we begin the welcome\_screen function for the user.

    """

    # create 20 numbered tables from 1 to 20 that at most 10 people can sit at each table

    tables: list[tableType] = [tableType(reservationType("", "", 0, False), 10, 0, 1, False), tableType(reservationType("", "", 0, False), 10, 0, 2, False), tableType(reservationType("", "", 0, False), 10, 0, 3, False),

                         tableType(reservationType("", "", 0, False), 10, 0, 4, False), tableType(reservationType("", "", 0, False), 10, 0, 5, False), tableType(reservationType("", "", 0, False), 10, 0, 6, False),

                         tableType(reservationType("", "", 0, False), 10, 0, 7, False), tableType(reservationType("", "", 0, False), 10, 0, 8, False), tableType(reservationType("", "", 0, False), 10, 0, 9, False),

                         tableType(reservationType("", "", 0, False), 10, 0, 10, False), tableType(reservationType("", "", 0, False), 10, 0, 11, False), tableType(reservationType("", "", 0, False), 10, 0, 12, False),

                         tableType(reservationType("", "", 0, False), 10, 0, 13, False), tableType(reservationType("", "", 0, False), 10, 0, 14, False), tableType(reservationType("", "", 0, False), 10, 0, 15, False),

                         tableType(reservationType("", "", 0, False), 10, 0, 16, False), tableType(reservationType("", "", 0, False), 10, 0, 17, False), tableType(reservationType("", "", 0, False), 10, 0, 18, False),

                         tableType(reservationType("", "", 0, False), 10, 0, 19, False), tableType(reservationType("", "", 0, False), 10, 0, 20, False)

                         ]

    # create empty list for reservations

    reservations: reservationType = []

    # set up root window to be top level window and keep it visible so the user can close the application completely

    # without having to use task manager

    root = tk.Tk()

    root.title("Root Window")

    root.geometry("480x100")

    root\_label = ttk.Label(root, text='Do not close this window unless you want to close the application.', font=("Times New Roman", 12))

    root\_label.pack()

    # open Welcome Screen

    welcome\_screen(reservations, tables)