Programming Languages Final Report

Devam Patel
CS-352
11/29/21

GitHub Project: https://github.com/Devamp/CS352_Final_GUICalender

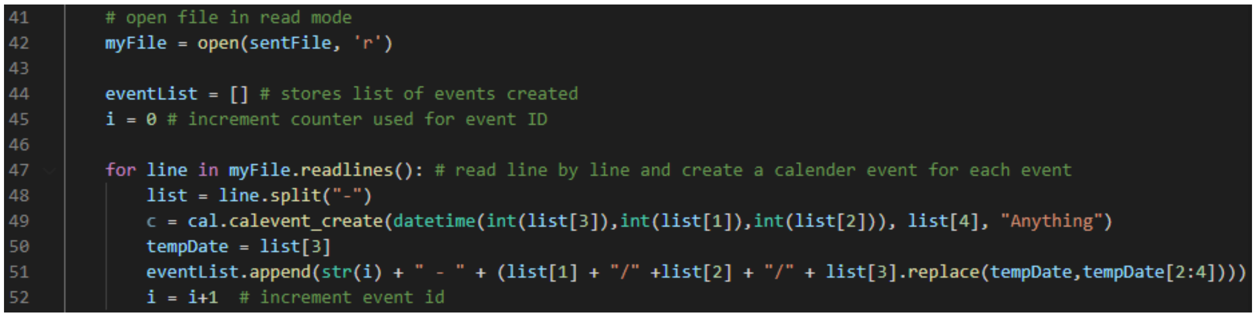
Overview

The primary purpose for this project was for me to start getting some experience with coding in Python. Python was one of the languages that I always wanted to work with because of how widely used and popular it is. Being a third year CS student without having once worked in the language, I felt like this assignment gave me that perfect opportunity to start getting some experience. I have programmed a simple assignment/event calendar which takes in a .txt file with the event id (any non-negative int), event date (M-D-YYY), and the assignment/event name (any string excluding whitespaces). I used the python tkinterface and the tkcalender libraries to complete this project. Here is the expected format for the input file and an example input:

Expected Format: *EventID-M-D-YYY-Assignment_Name*
Example: *(7-1-29-2001-Devam's_Birthday)*

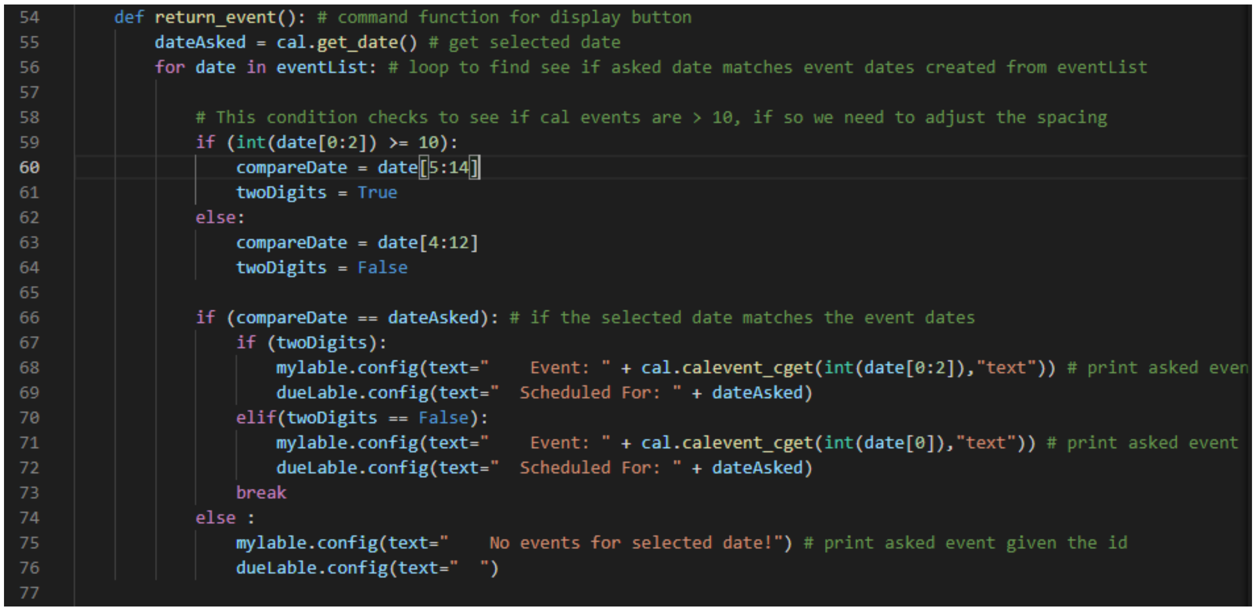
I wanted my program to act kind of like a planner or a syllabus reader where a person could simply give the program an input file and all of their homework/exam due dates would be added to a calendar for them to use. The .txt file I use is very simple and specifically modified so it is easily readable in the program. Once the program has read in the .txt file, the program will output a calendar GUI that is loaded with the user's events and due dates. From here, the user can click the *Display* button to see what events are associated with which dates. Users also have the option to delete an event by clicking the event and then clicking the *Delete Event*.

Code Highlights



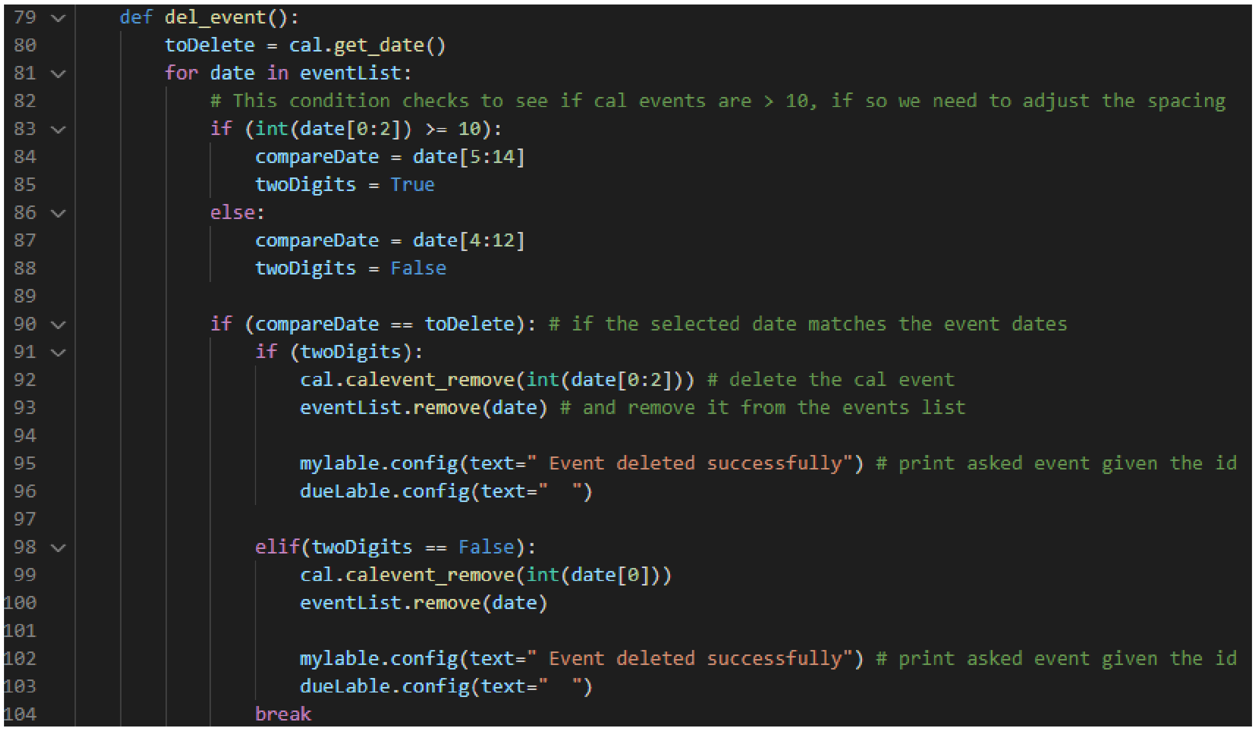
```
41      # open file in read mode
42      myFile = open(sentFile, 'r')
43
44      eventList = [] # stores list of events created
45      i = 0 # increment counter used for event ID
46
47      for line in myFile.readlines(): # read line by line and create a calender event for each event
48          list = line.split("-")
49          c = cal.calevent_create(datetime(int(list[3]),int(list[1]),int(list[2])), list[4], "Anything")
50          tempDate = list[3]
51          eventList.append(str(i) + " - " + (list[1] + "/" +list[2] + "/" + list[3].replace(tempDate,tempDate[2:4])))
52          i = i+1  # increment event id
```

The piece of code above is on lines 41-52 of the app.py file. This is one of the essential parts of the code as this is where I read in the events from the input file and create a calendar event out of them. First, I open up the input file in read mode as we will not be editing the file at all. Next, I create a list called *eventList* which will hold each of the events as I read them in. Within the for loop, we read in line by line from the input file. First, we split up whole line using the "-" as the delimiter. The split-up list now is in the format "*ID-M-D-YYYY-Some_Name*". Then using that new list, I create the calendar event using the *calevent_create()* which is part of the tkcalender library. Lastly, I add the new event to the *eventList* in the "MM/DD/YY" format, since that is the format we will use later. Then I update the counter variable which keeps track of the total number of events we have read in. This part of the code reads in the input file and creates all of the necessary events to be displayed on the calendar.



```
54      def return_event(): # command function for display button
55          dateAsked = cal.get_date() # get selected date
56          for date in eventList: # loop to find see if asked date matches event dates created from eventList
57
58              # This condition checks to see if cal events are > 10, if so we need to adjust the spacing
59              if (int(date[0:2]) >= 10):
60                  compareDate = date[5:14]
61                  twoDigits = True
62              else:
63                  compareDate = date[4:12]
64                  twoDigits = False
65
66              if (compareDate == dateAsked): # if the selected date matches the event dates
67                  if (twoDigits):
68                      mylable.config(text="    Event: " + cal.calevent_cget(int(date[0:2]),"text")) # print asked even
69                      dueLable.config(text="  Scheduled For: " + dateAsked)
70                  elif(twoDigits == False):
71                      mylable.config(text="    Event: " + cal.calevent_cget(int(date[0]),"text")) # print asked event
72                      dueLable.config(text="  Scheduled For: " + dateAsked)
73                  break
74              else :
75                  mylable.config(text="    No events for selected date!") # print asked event given the id
76                  dueLable.config(text="  ")
77
```

The piece of code above is on lines 54-76 of the app.py file. This part of the code is crucial to the program as this is the function that holds the functionality behind the *Display* button and returns the date corresponding event back to the user. The *returnEvent()* is called once the user selects a date on the calendar and then pressed the *Display* button. Once the function is called, it begins looping the *eventList* we had made earlier which holds all of the calendar events. From there, I first do some error checking to make sure the whitespaces are properly accounted for before creating a substring of a date from *eventList*. The main part of the function is the last conditional statement. Here we check to see if the trimmed date that is selected by the user matches any of the dates in our *eventList*. Once there is a match, function calls the *calevent_cget()* on the appropriate event, and then I update the label to display the asked event. Lastly, I also display the event date to let the user know what day the event was scheduled for. This part of the code fulfils the main purpose of the program as this allows the user to see their saved events and due dates.



```python
79      def del_event():
80          toDelete = cal.get_date()
81          for date in eventList:
82              # This condition checks to see if cal events are > 10, if so we need to adjust the spacing
83              if (int(date[0:2]) >= 10):
84                  compareDate = date[5:14]
85                  twoDigits = True
86              else:
87                  compareDate = date[4:12]
88                  twoDigits = False
89
90              if (compareDate == toDelete): # if the selected date matches the event dates
91                  if (twoDigits):
92                      cal.calevent_remove(int(date[0:2])) # delete the cal event
93                      eventList.remove(date) # and remove it from the events list
94
95                      mylable.config(text=" Event deleted successfully") # print asked event given the id
96                      dueLable.config(text="  ")
97
98                  elif(twoDigits == False):
99                      cal.calevent_remove(int(date[0]))
100                     eventList.remove(date)
101
102                     mylable.config(text=" Event deleted successfully") # print asked event given the id
103                     dueLable.config(text="  ")
104                 break
```

The piece of code above is on lines 79-104 of the app.py file. This snippet of code is essential due to the fact that this holds the functionality behind the *Delete Event* button. A user can load up events into the calendar by having them read in through an input file. However, once the user had completed the event or the date has passed, then the user is given the option to delete an event. The *del_event()* is called when the user selected a date on the calendar and then pressed the *Delete Event* button. This function has similar error checking as the *return_event()* but instead of calling *calevent_cget()*, the function calls *calevent_remove()* which is also part of the tkcalendar package. We again loop through the eventList, once the date that is asked to be deleted matches one from the *eventList*, we call the delete event function and remove the event from the calendar. The user is then prompted with a message saying their event was deleted successfully.

File Listing

- **app.py** – the program only has one file which has all of the code for the program. App.py also has the entry point to the program at the *main()* located within it.