Name – **Onkar Kajrolkar**

CampbuzzId – **2020Sep@03308**

Email – **onkarkajrolkar7798@gmail.com**

**Case Study on Java/C#/Python LBJ 2020**

**LBJ Cycle-2**

**1. Problem Statement - 1**

1. **Synopsis –**

Python is very easy interpreted language with a number of libraries which makes problem solving and the code very easy and efficient. Python has in-built “CSV” module and GUI application module which allowed me to read and edit through the csv file very easily and creating the GUI was fast. As compared to other programming languages, Python takes significantly less lines of code and is easy to understand

1. **Source Code –**

import tkinter as tk

from tkinter import \*

import csv

root = tk.Tk()

root.title("Colleges")

root.geometry("300x300")

global top

top = tk.Frame(root)

top.grid()

global bottom

bottom = tk.Frame(root)

bottom.grid()

def hide\_bottom():

root.geometry("300x300")

for widget in bottom.winfo\_children():

widget.destroy()

class College:

def \_\_init\_\_(self, cId, cName, course, city, fees, pin):

self.collegeId = cId

self.collegeName = cName

self.courseType = course

self.city = city

self.fees = fees

self.pinCode = pin

def register(self):

with open('colleges.csv', 'a') as csv\_file:

csv\_writer = csv.writer(csv\_file)

csv\_writer.writerow([self.collegeId, self.collegeName, self.courseType, self.city, self.fees, self.pinCode])

l1 = Label(bottom, text="Successfully registered college!")

l1.grid(columnspan=3, row=8)

def newCollege(cId, cName, course, city, fees, pin):

obj = College(cId, cName, course, city, fees, pin)

obj.register()

def register():

hide\_bottom()

l1 = Label(bottom, text="College Id: ")

l1.grid(column=0, row=1)

cId = Entry(bottom, width=30)

cId.grid(column=1, row=1, columnspan=2)

l2 = Label(bottom, text="College Name: ")

l2.grid(column=0, row=2)

cName = Entry(bottom, width=30)

cName.grid(column=1, row=2, columnspan=2)

l3 = Label(bottom, text="Course: ")

l3.grid(column=0, row=3)

course = Entry(bottom, width=30)

course.grid(column=1, row=3, columnspan=2)

l4 = Label(bottom, text="City: ")

l4.grid(column=0, row=4)

city = Entry(bottom, width=30)

city.grid(column=1, row=4, columnspan=2)

l5 = Label(bottom, text="Fees: ")

l5.grid(column=0, row=5)

fees = Entry(bottom, width=30)

fees.grid(column=1, row=5, columnspan=2)

l6 = Label(bottom, text="Pincode: ")

l6.grid(column=0, row=6)

pin = Entry(bottom, width=30)

pin.grid(column=1, row=6, columnspan=2)

btn = Button(bottom, text="Submit", command=lambda: newCollege(cId.get(), cName.get(), course.get(), city.get(), fees.get(), pin.get()))

btn.grid(columnspan=3, row=7)

def searchCollege(cName, course):

l1 = Label(bottom)

l1.grid(columnspan=3, row=4)

success = True

with open('colleges.csv', 'r') as csv\_file:

csv\_reader = csv.reader(csv\_file)

for i in csv\_reader:

if cName in i and course in i:

if success:

l1.configure(text="Colleges Found!")

success = False

root.geometry("525x300")

str1 = "College Id= "+i[0]+", College Name= "+i[1]+", Course= "+i[2]+", City= "+i[3]+", Fees= "+i[4]+", Pincode= "+i[5]

l2 = Label(bottom, text=str1).grid(columnspan=3)

if success:

l1.configure(text="No Colleges found for the given name and course")

def search():

hide\_bottom()

l1 = Label(bottom, text="College Name: ")

l1.grid(column=0, row=1, sticky="e")

cName = Entry(bottom, width=25)

cName.grid(column=1, row=1, columnspan=2, sticky="w")

l2 = Label(bottom, text="Course: ")

l2.grid(column=0, row=2, sticky="e")

course = Entry(bottom, width=25)

course.grid(column=1, row=2, columnspan=2, sticky="w")

btn = Button(bottom, text="Submit", command=lambda: searchCollege(cName.get(), course.get()))

btn.grid(columnspan=3, row=3)

def removeCollege(cId):

li = list()

with open('colleges.csv', 'r') as csv\_file:

csv\_reader = csv.reader(csv\_file)

for i in csv\_reader:

li.append(i)

with open('colleges.csv', 'w', newline='') as csv\_file:

csv\_writer = csv.writer(csv\_file)

for lines in li:

if lines[0] == str(cId):

continue

else:

csv\_writer.writerow(lines)

str1 = "College with Id: " + str(cId) + " removed successfully!"

l1 = Label(bottom, text=str1)

l1.grid(columnspan=3, row=3)

def remove():

hide\_bottom()

l1 = Label(bottom, text="College Id: ")

l1.grid(column=0, row=1)

cId = Entry(bottom, width=30)

cId.grid(column=1, row=1, columnspan=2)

btn = Button(bottom, text="Submit", command=lambda: removeCollege(cId.get()))

btn.grid(columnspan=3, row=2)

bt1 = Button(top, text="Register College", command=register)

bt1.grid(column=0, row=0)

bt2 = Button(top, text="Search Colleges", command=search)

bt2.grid(column=1, row=0)

bt3 = Button(top, text="Remove College", command=remove)

bt3.grid(column=2, row=0)

root.mainloop()

1. **Approach –**

The approach towards this problem statement was fairly simple. The CSV module is used to read and write the ‘colleges.csv’ file. The user was given three options which were Registering a new college, Searching a college and Deleting a college from the csv file. So first the user’s input was considered for the above options and depending on the input, the program flow was redirected to the specified functions which performed the desired task. A new college is stored in an object of the College class with it’s different attributes and method. This was done solely on the Python console to run and compile the commands for the csv file editing. When everything was done, I started building the GUI using the “Tkinter” module. The tkinter module is very useful and efficient for designing python applications. The user is given three options with the three buttons and depending on which button is pressed, the respective data entry options appear. For example if user chooses to search a college, after the button click, the window will display entry boxes so that user can provide the parameters such as College Name and Course type.

1. **Coding Practices –**

The use of CSV and Tkinter modules makes the code more efficient than other programming languages. Lambda function was also used. Lambda is a small anonymous function which contains only single expression. It is mainly used to write functions which are going to be used only once. In this code, it was used to pass the user entries as formal parameters to the called function. This made the code more concise and short. The use of different methods made the code very modular. So even if one part of the code malfunctions, it won’t affect the whole program.

1. **Screenshots –**

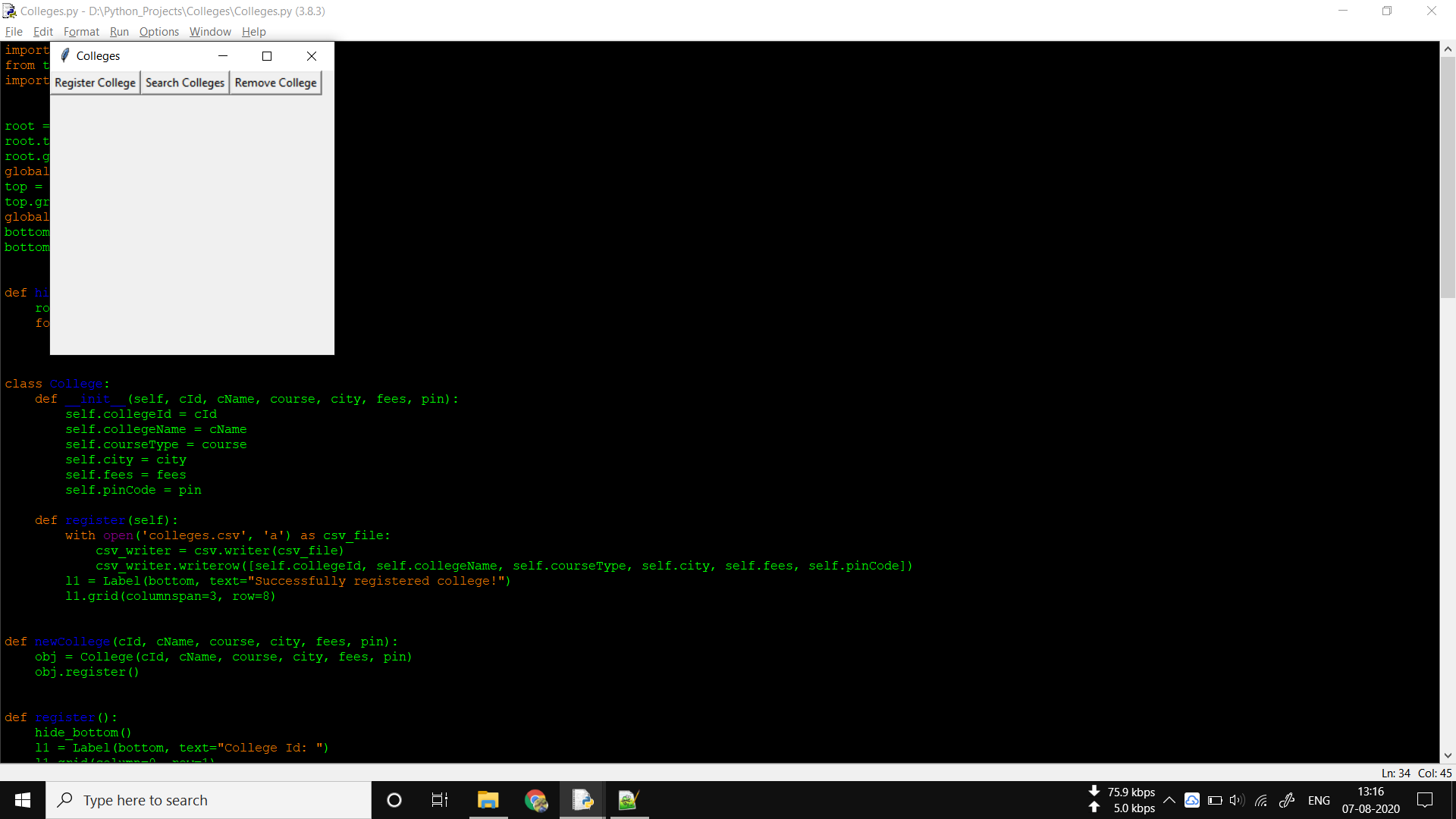


Fig1 – Options given to the user in the application

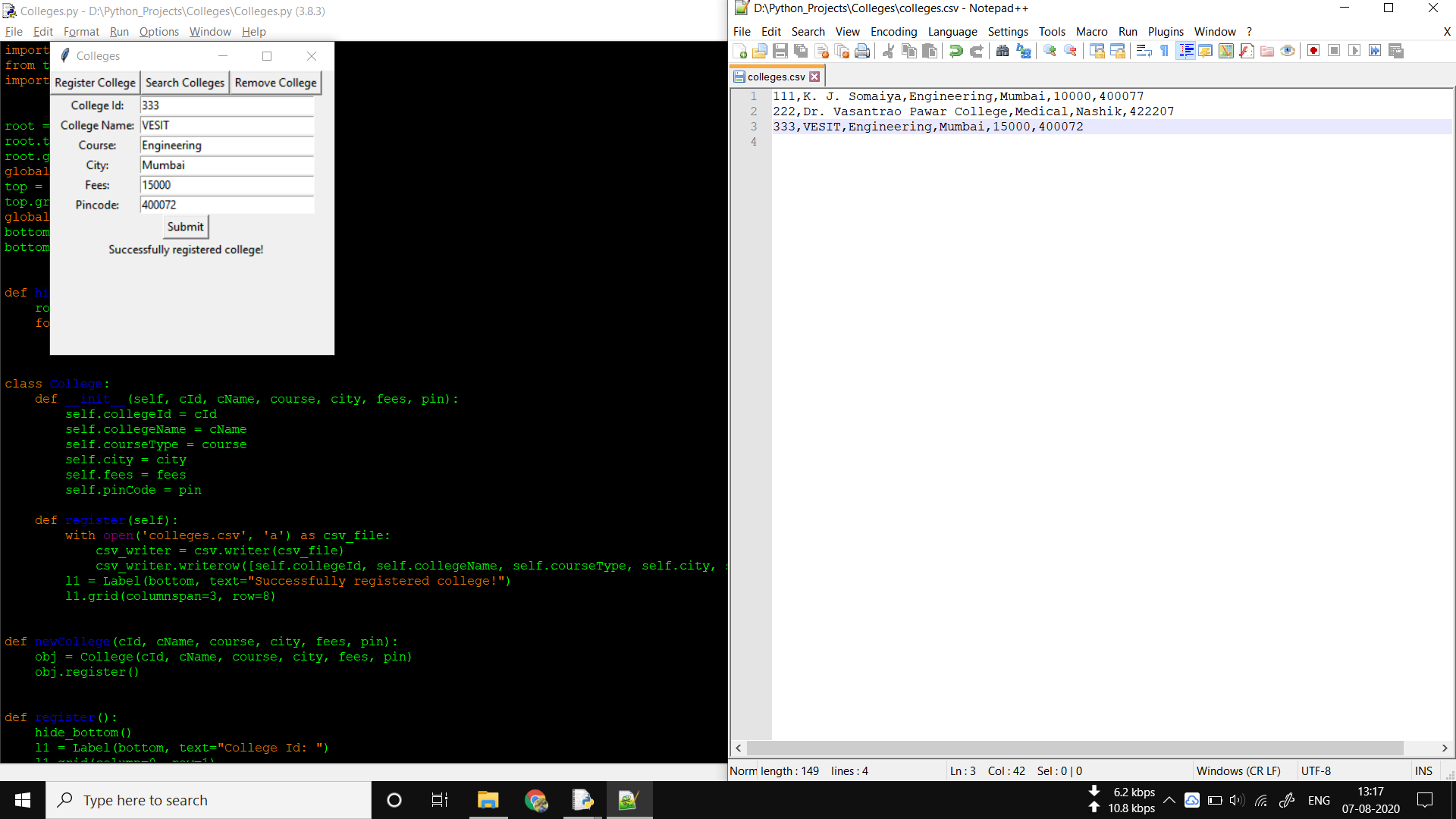


Fig2 – Registering a new College

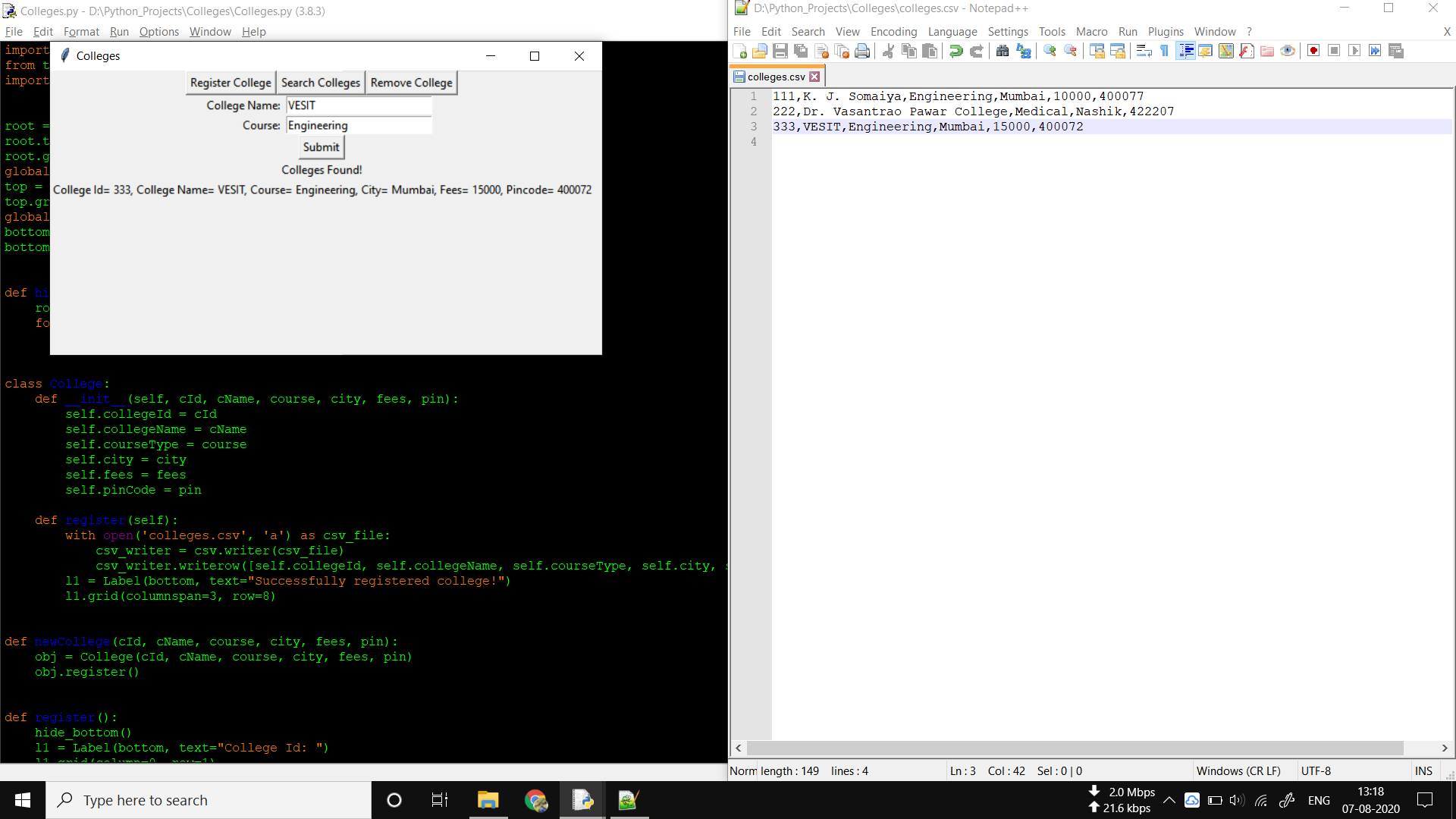


Fig3 – Searching a College

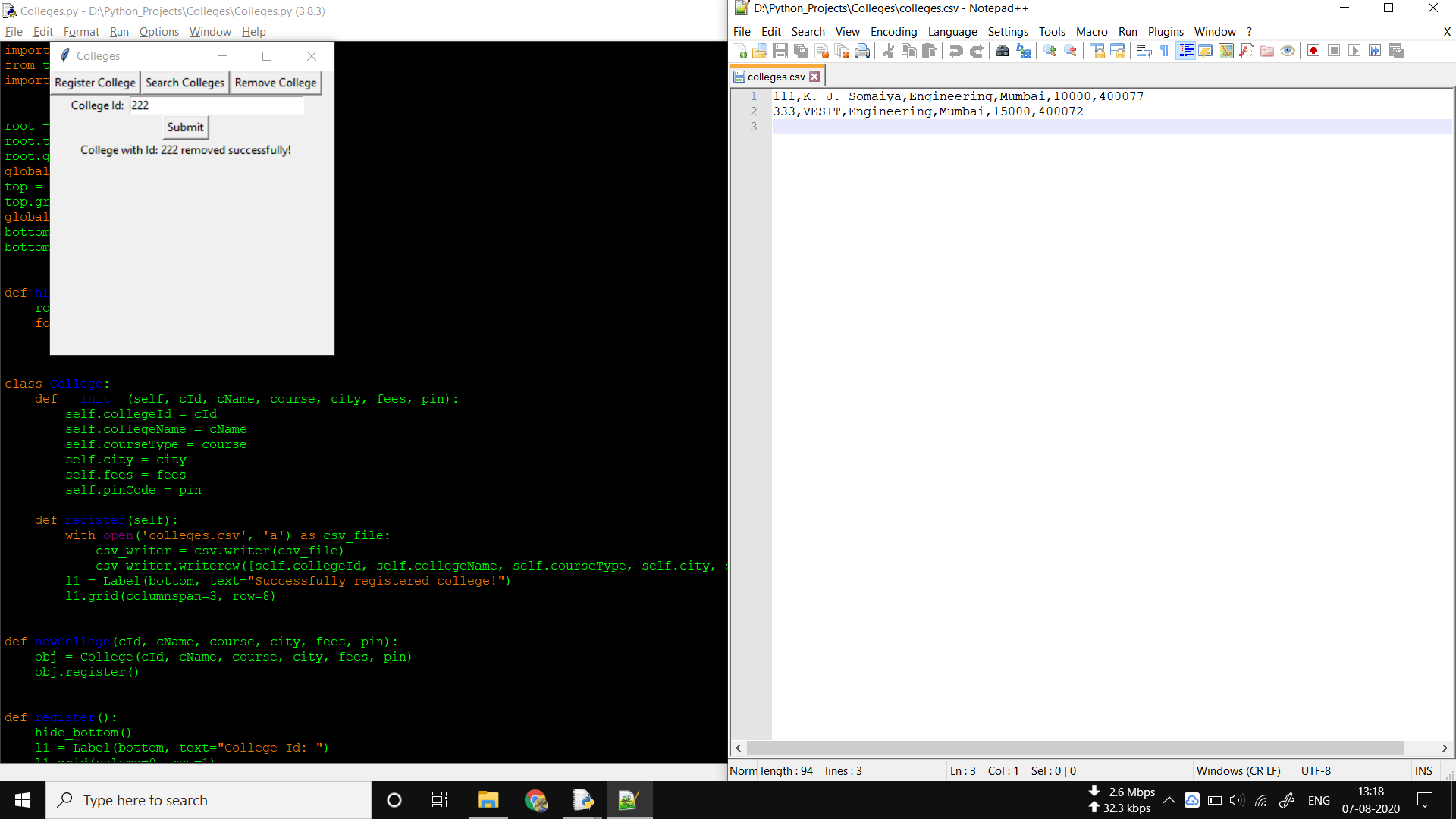


Fig4 – Removing a College