EUROPEAN UNIVERSITY OF LEFKE
Faculty of Engineering
Department of Software Engineering

COMP 337

Prepared by Anthony Olori 194234
Prepared by Praise Adeoti 194484

Project Description

Language of program : python

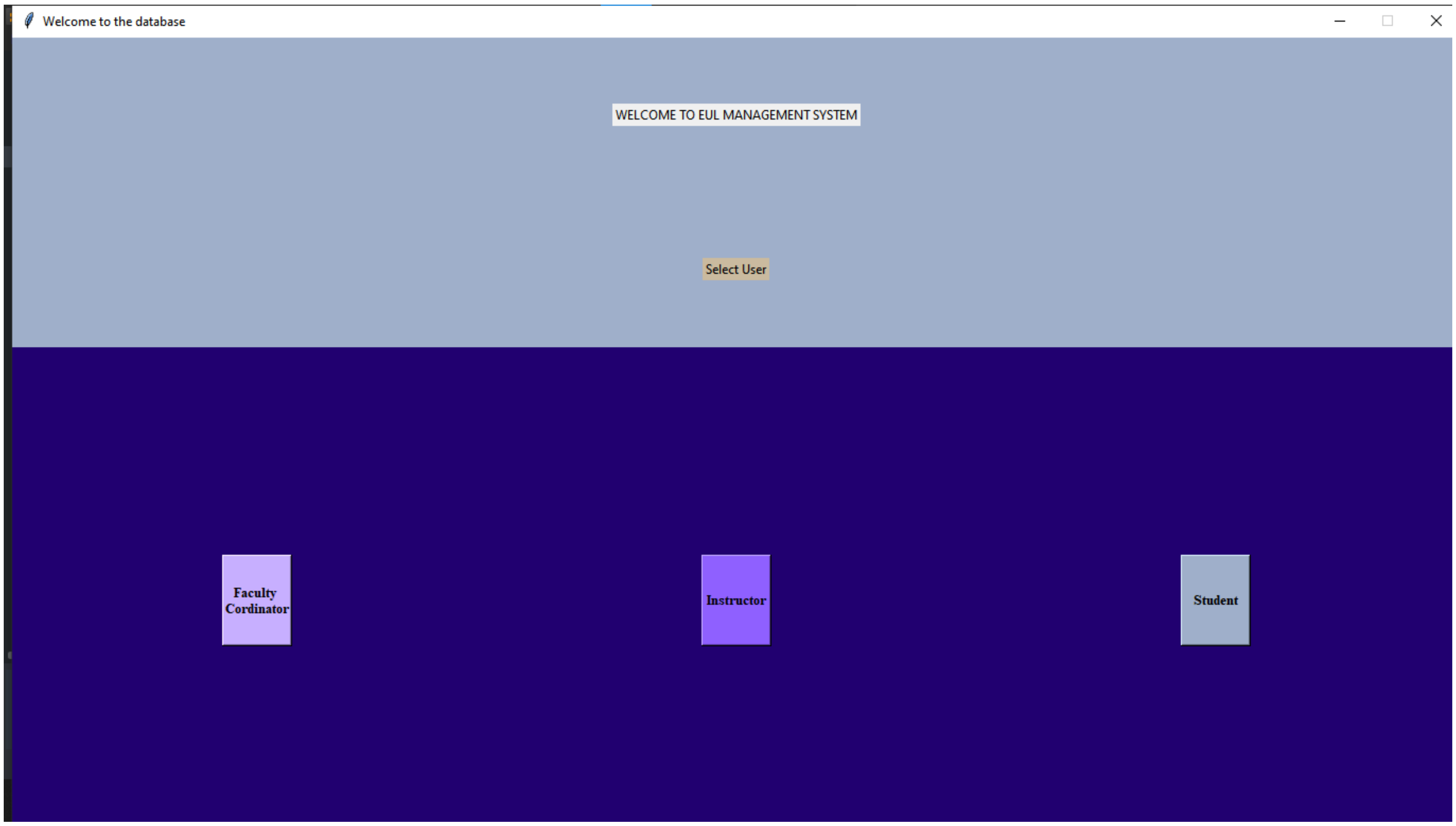
Module used: tkinter

Head of logic design and implementation: Anthony Olori

UI/UX design head: Praise Adeoti

This program comprises of 4 scenes namely,

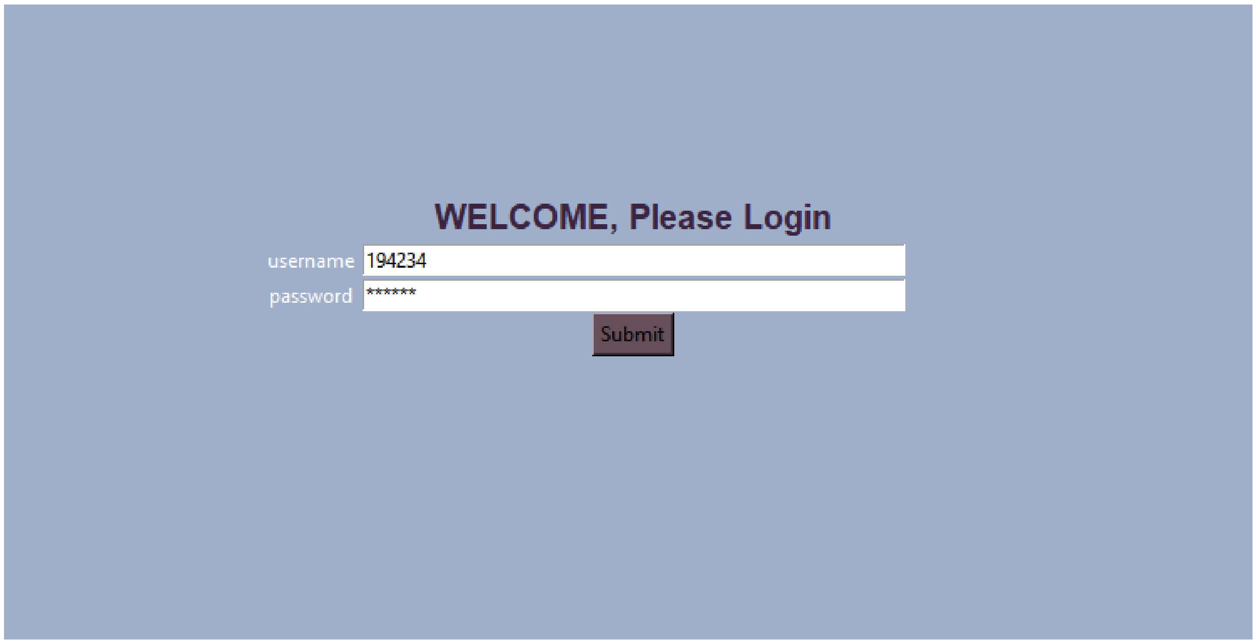
The selection scene:



the user is asked to select whether they're a faculty member, an instructor or a student, clicking on a button sets a variable to either 1, 2, or 3 respectively.

The next scene consists of a a login form
Where the user is asked to input the correct username and password.

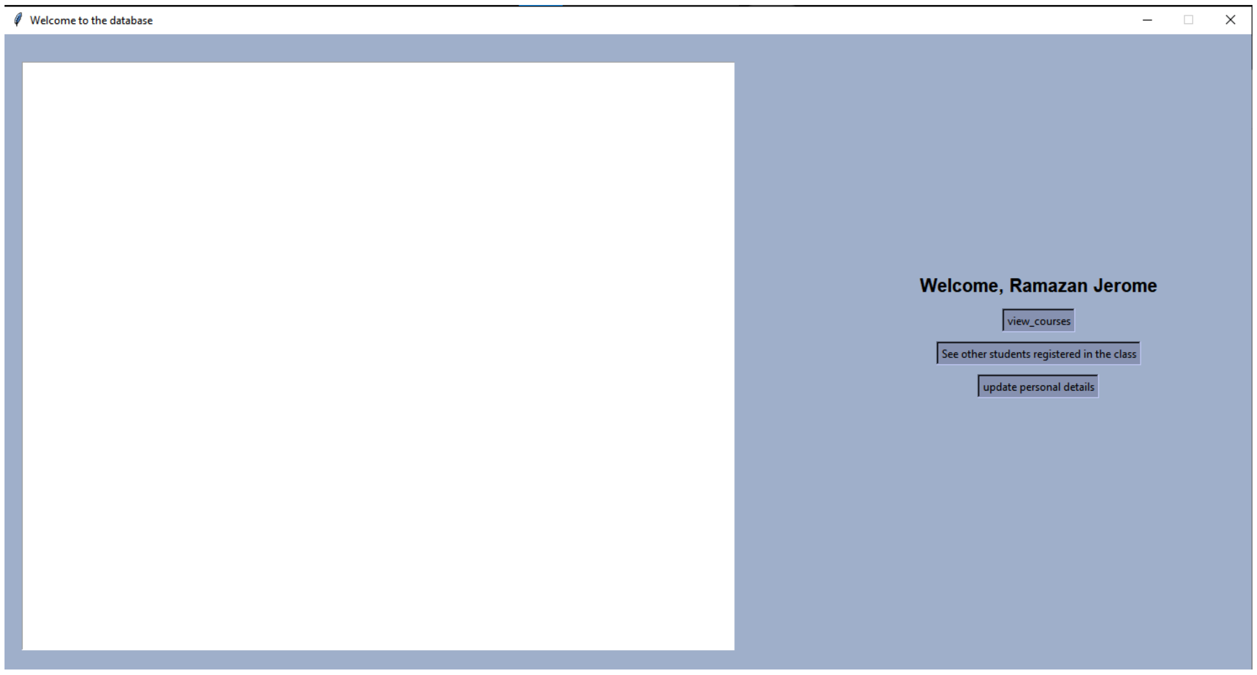
Login scene:

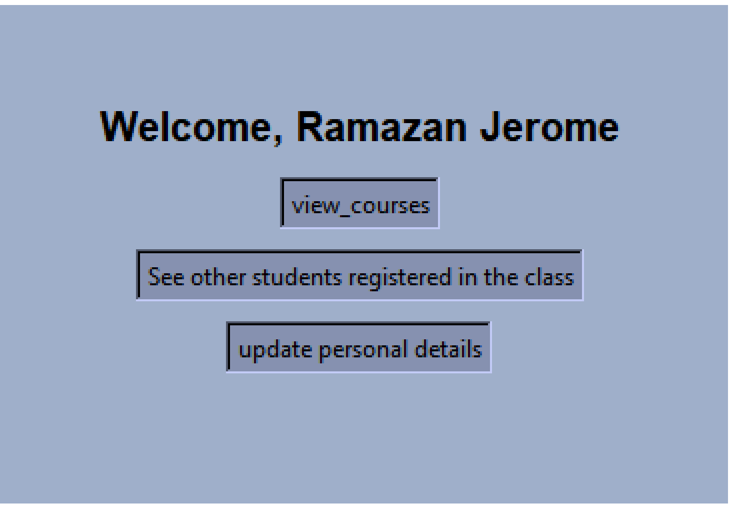


This scene is common to all users, meaning they see the same display at this step of the process. Depending on what was chosen in the selection scene the user input would be compared only with the correct credentials for the selection they made, this way the user cannot log in with the correct credentials on the wrong account.

The next scenes are:

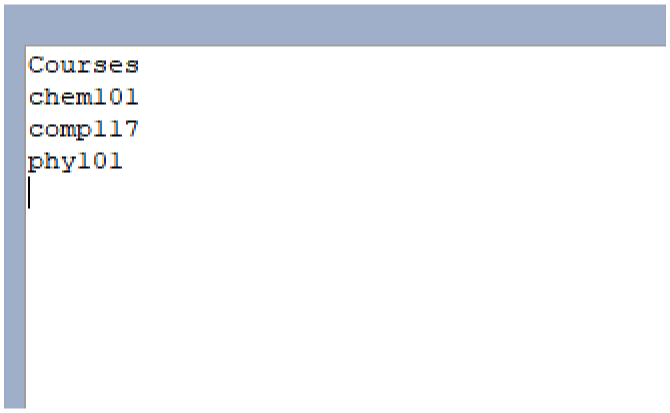
Student:





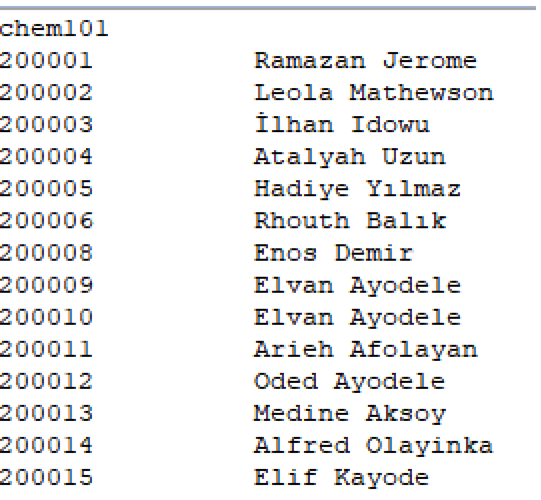
View courses:

This is what comes up when the user clicks on the view courses button, the application goes to the database and retrieves the data from the table and prints to the widget.

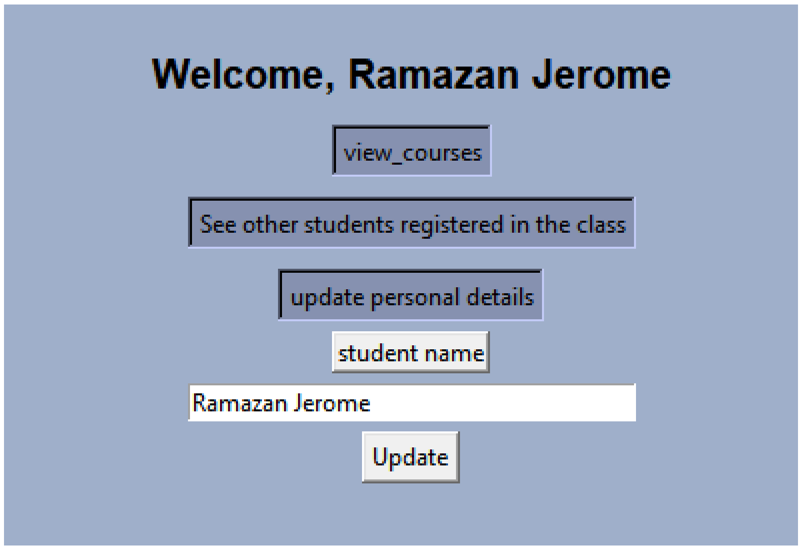


```
Courses
chem101
comp117
phy101
```

See other students registered in the class:
This prints out students' names and numbers grouped by their courses.

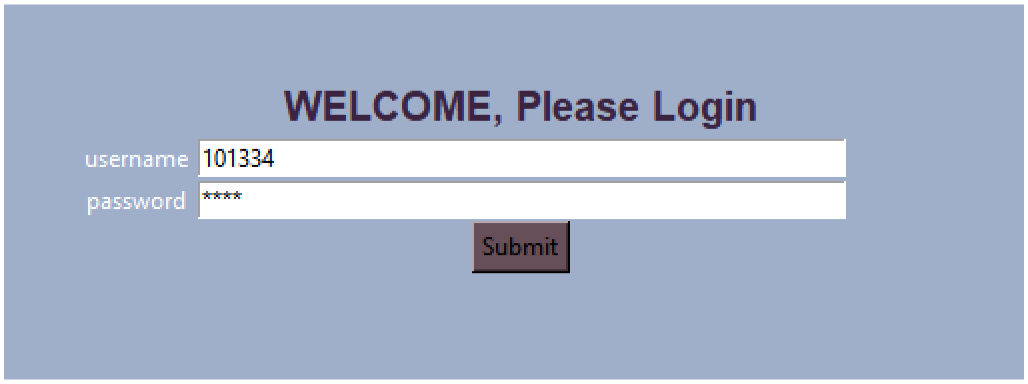


```
chem101
200001          Ramazan Jerome
200002          Leola Mathewson
200003          İlhan Idowu
200004          Atalyah Uzun
200005          Hadiye Yılmaz
200006          Rhouth Balık
200008          Enos Demir
200009          Elvan Ayodele
200010          Elvan Ayodele
200011          Arieh Afolayan
200012          Oded Ayodele
200013          Medine Aksoy
200014          Alfred Olayinka
200015          Elif Kayode
```

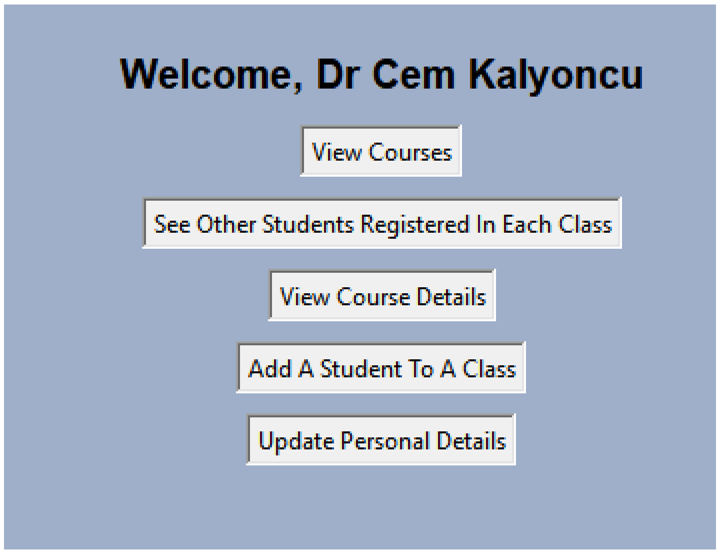


Update personal details: this prints the students name and allows the user to make changes to it

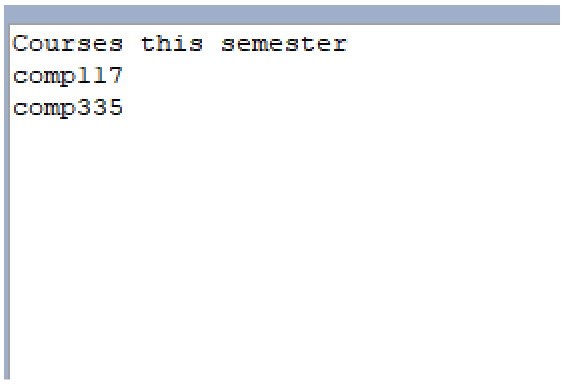
Instructor:



username 101334
password ****

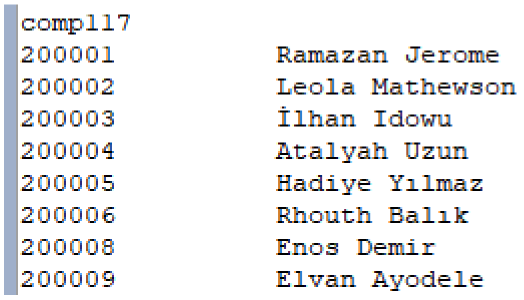


A view of all the viable button options, clickable by the instructor for the management system.



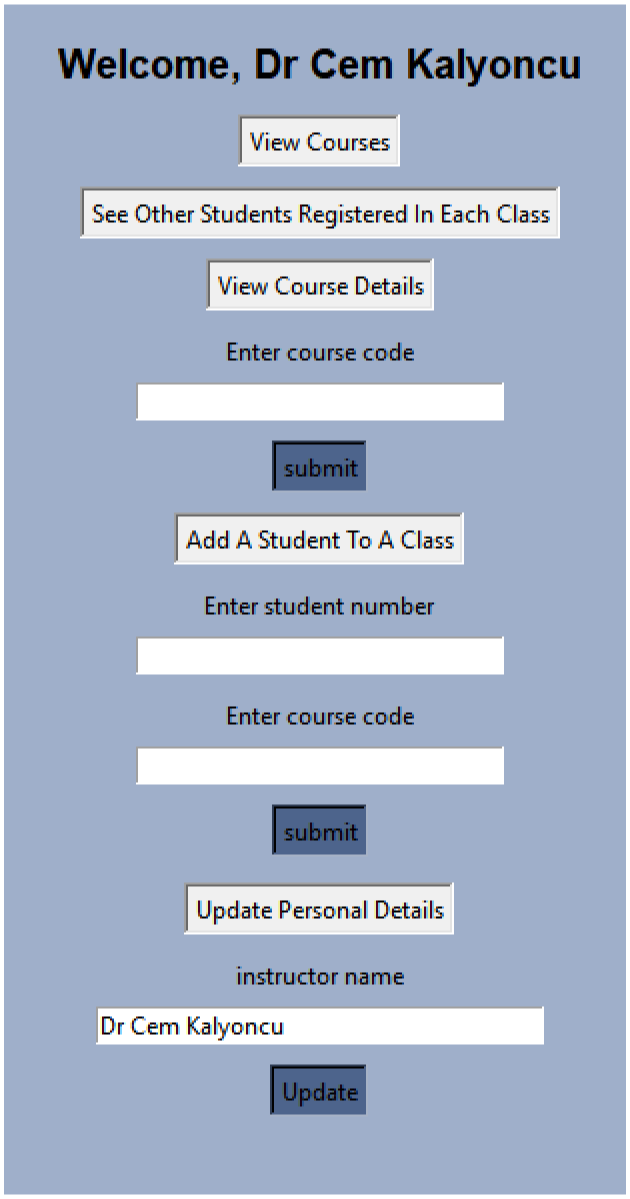
```
Courses this semester
comp117
comp335
```

View courses: This is the result that comes up when the user clicks on the view courses button, it sends a query to the database and retrieves the courses from “teaches” for the specific year and semester, namely 2021 and FALL respectively.

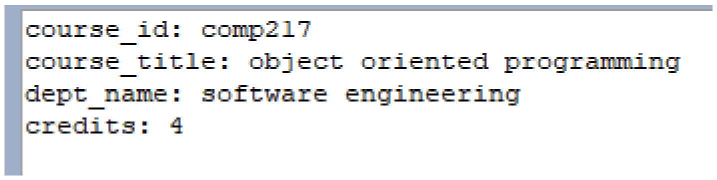


```
comp117
200001          Ramazan Jerome
200002          Leola Mathewson
200003          İlhan Idowu
200004          Atalyah Uzun
200005          Hadiye Yılmaz
200006          Rhouth Balık
200008          Enos Demir
200009          Elvan Ayodele
```

See other students registered in each class: this sends the same query as the student button but this time using the courses the instructor teaches as automatic input.



View course details: this takes the input from the teacher, compares with the database and prints the result, it works with any course, regardless whether the instructor is currently teaching it or not. If the details are invalid (i.e not matching with any course in the database) an error will be displayed.

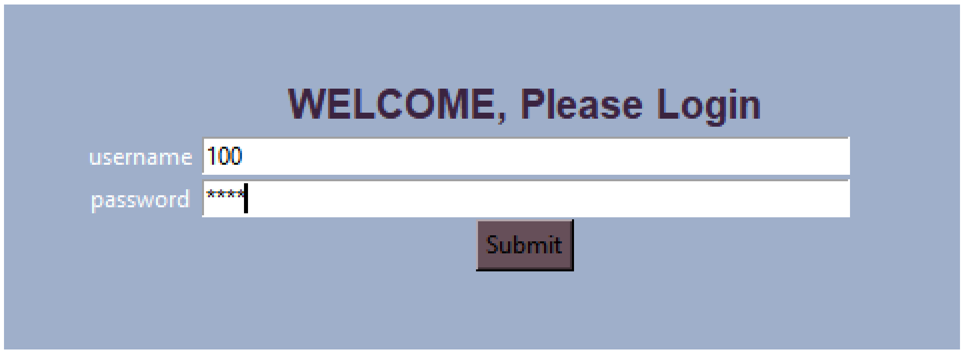


```
course_id: comp217
course_title: object oriented programming
dept_name: software engineering
credits: 4
```

Add a student to A class: this takes a student number and course code and sends it to the database. All errors are directed to the screen.

Update Personal Details: this first-off displays the users name then allow the users to make and submit changes.

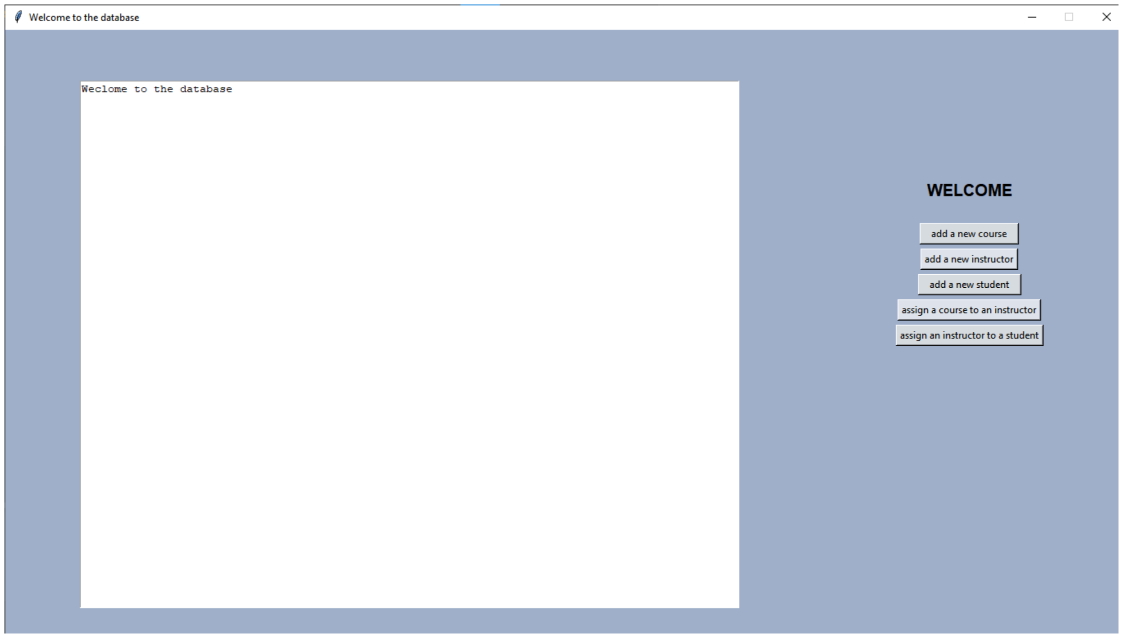
Faculty Coordinator



WELCOME, Please Login

username 100

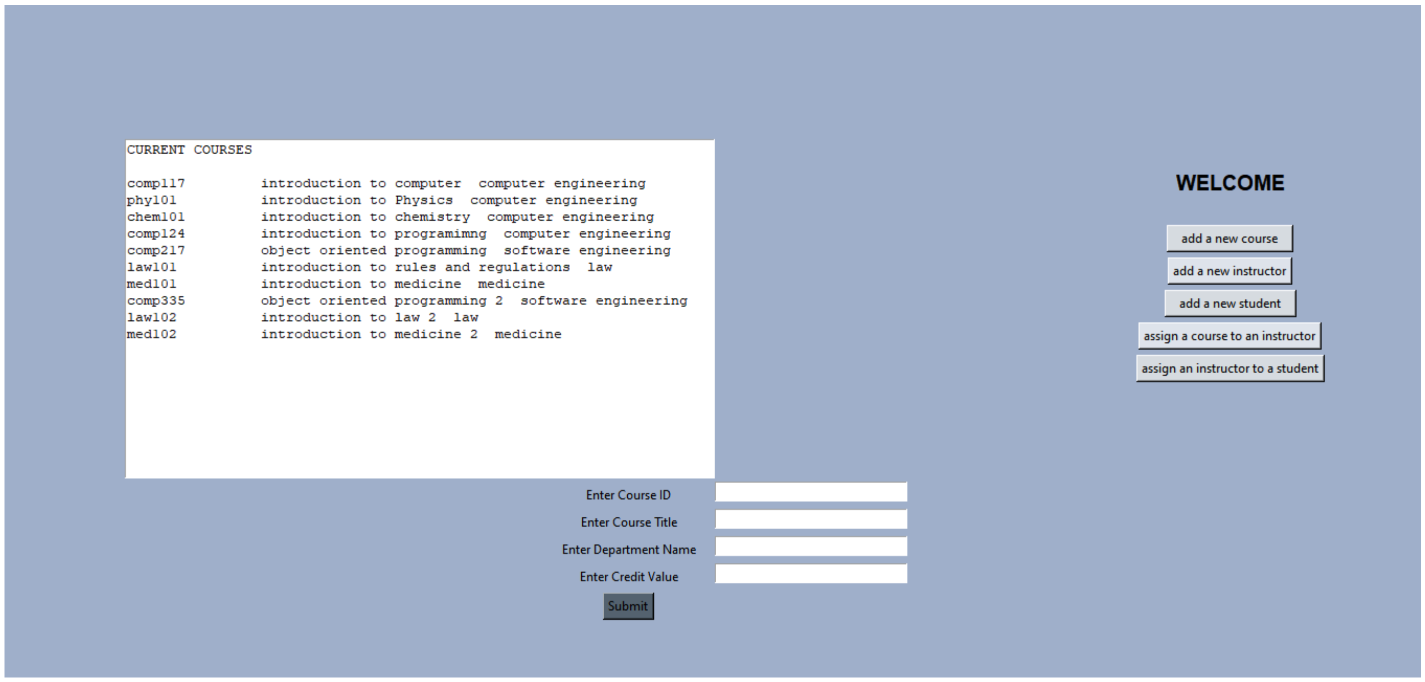
password ****



Welcome to the database

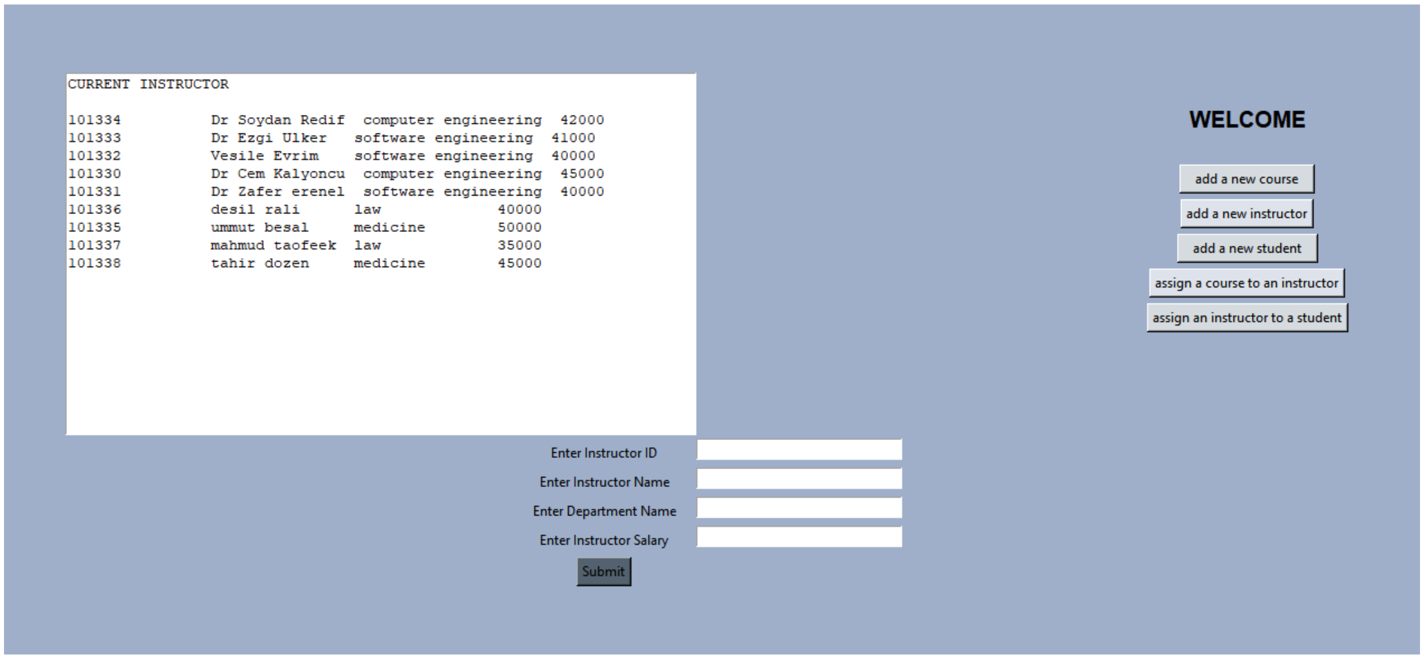
```
Weclome to the database
```

WELCOME
add a new course
add a new instructor
add a new student
assign a course to an instructor
assign an instructor to a student



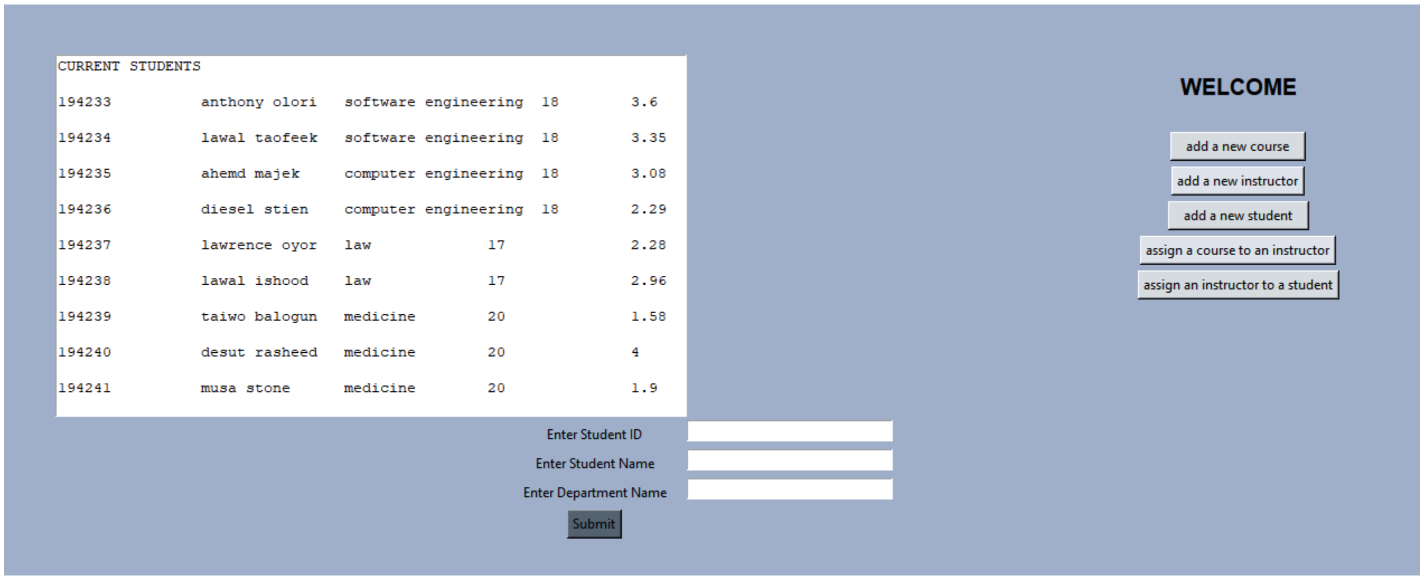
```
CURRENT COURSES

comp117        introduction to computer  computer engineering
phy101         introduction to Physics  computer engineering
chem101        introduction to chemistry  computer engineering
comp124        introduction to programimng  computer engineering
comp217        object oriented programming  software engineering
law101         introduction to rules and regulations  law
med101         introduction to medicine  medicine
comp335        object oriented programming 2  software engineering
law102         introduction to law 2  law
med102         introduction to medicine 2  medicine
```

The screenshot above is the scene displayed when the faculty coordinator clicks the “add a new course” button, with the entry box for input of details.



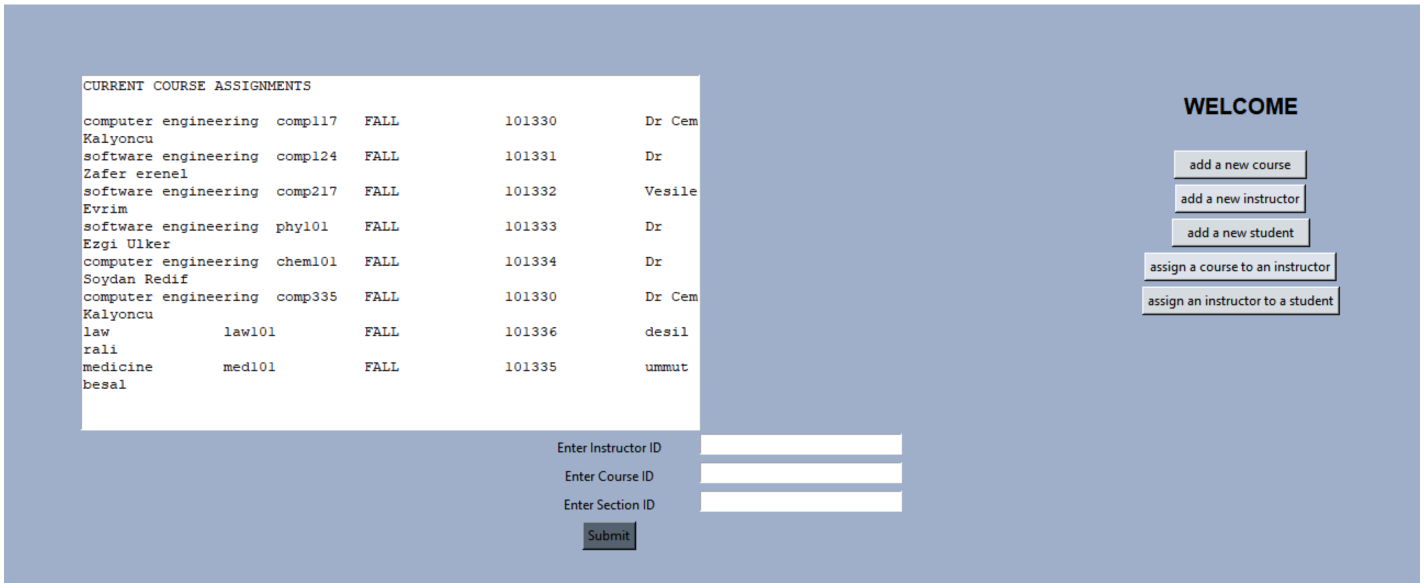
```
CURRENT  INSTRUCTOR

101334          Dr Soydan Redif  computer engineering  42000
101333          Dr Ezgi Ulker   software engineering  41000
101332          Vesile Evrim    software engineering  40000
101330          Dr Cem Kalyoncu  computer engineering  45000
101331          Dr Zafer erenel  software engineering  40000
101336          desil rali      law             40000
101335          ummut besal     medicine        50000
101337          mahmud taofeek  law             35000
101338          tahir dozen     medicine        45000
```

The screenshot above is the scene displayed when the faculty coordinator clicks the "add a new instructor" button, with the entry box for input of details.



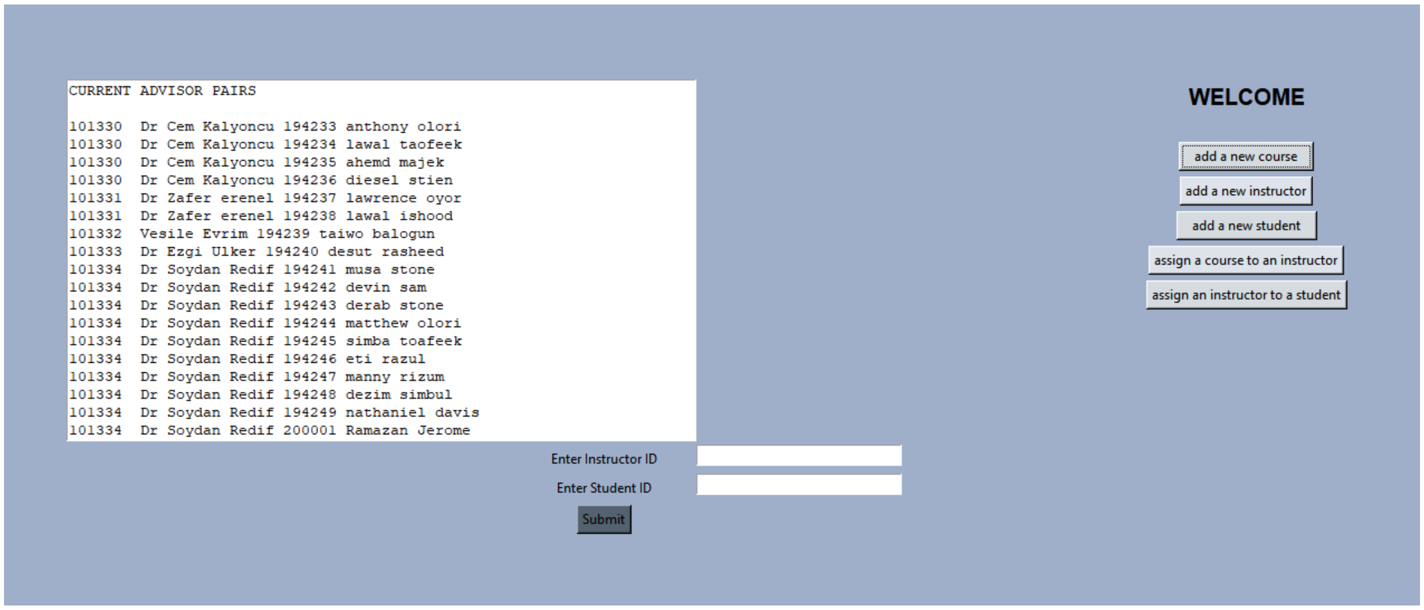
```
CURRENT  STUDENTS

194233          anthony olori   software engineering  18        3.6

194234          lawal taofeek   software engineering  18        3.35

194235          ahemd majek     computer engineering  18        3.08

194236          diesel stien    computer engineering  18        2.29

194237          lawrence oyor   law             17              2.28

194238          lawal ishood    law             17              2.96

194239          taiwo balogun   medicine        20              1.58

194240          desut rasheed   medicine        20              4

194241          musa stone      medicine        20              1.9
```

The screenshot above is the scene displayed when the faculty coordinator clicks the "add a new student" button, with the entry box for input of details.



```
CURRENT COURSE ASSIGNMENTS

computer engineering  comp117   FALL           101330         Dr Cem
Kalyoncu
software engineering  comp124   FALL           101331         Dr
Zafer erenel
software engineering  comp217   FALL           101332         Vesile
Evrim
software engineering  phy101    FALL           101333         Dr
Ezgi Ulker
computer engineering  chem101   FALL           101334         Dr
Soydan Redif
computer engineering  comp335   FALL           101330         Dr Cem
Kalyoncu
law             law101          FALL           101336         desil
rali
medicine        med101          FALL           101335         ummut
besal
```

The screenshot above is the scene displayed when the faculty coordinator clicks the “assign a course to an instructor” button, with the entry box for input of details.



```
CURRENT ADVISOR PAIRS

101330  Dr Cem Kalyoncu 194233 anthony olori
101330  Dr Cem Kalyoncu 194234 lawal taofeek
101330  Dr Cem Kalyoncu 194235 ahemd majek
101330  Dr Cem Kalyoncu 194236 diesel stien
101331  Dr Zafer erenel 194237 lawrence oyor
101331  Dr Zafer erenel 194238 lawal ishood
101332  Vesile Evrim 194239 taiwo balogun
101333  Dr Ezgi Ulker 194240 desut rasheed
101334  Dr Soydan Redif 194241 musa stone
101334  Dr Soydan Redif 194242 devin sam
101334  Dr Soydan Redif 194243 derab stone
101334  Dr Soydan Redif 194244 matthew olori
101334  Dr Soydan Redif 194245 simba toafeek
101334  Dr Soydan Redif 194246 eti razul
101334  Dr Soydan Redif 194247 manny rizum
101334  Dr Soydan Redif 194248 dezim simbul
101334  Dr Soydan Redif 194249 nathaniel davis
101334  Dr Soydan Redif 200001 Ramazan Jerome
```

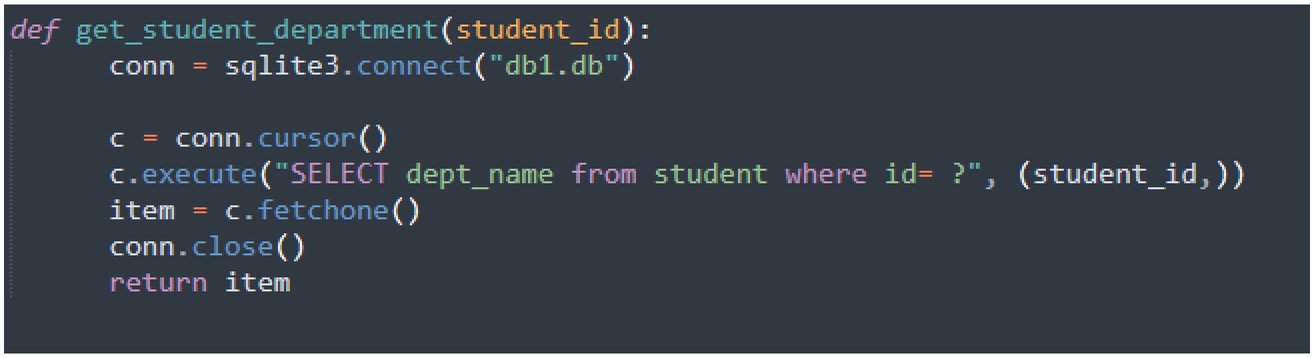
The screenshot above is the scene displayed when the faculty coordinator clicks the “assign an instructor to a student” button, with the entry box for input of details.

Functions

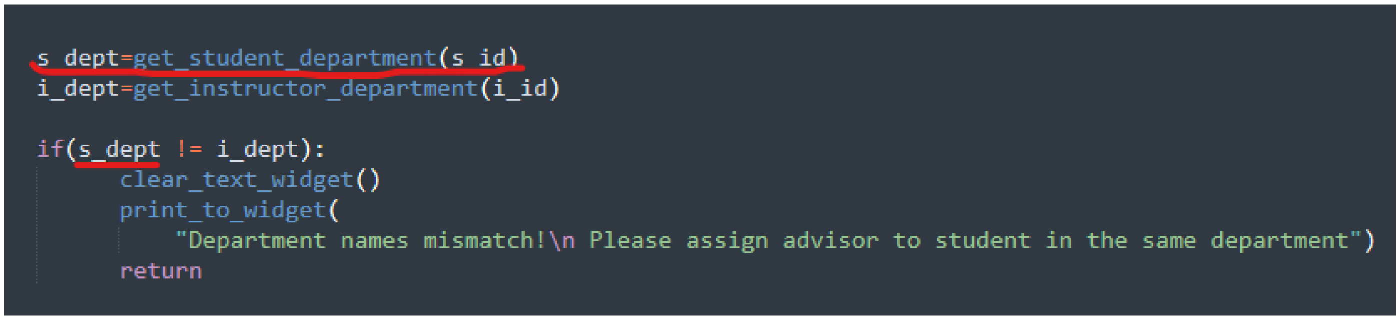
Get functions:

These are a class of functions whose purpose is to execute an sql select statement and return an object that can then be referenced/compared with,

Usage:



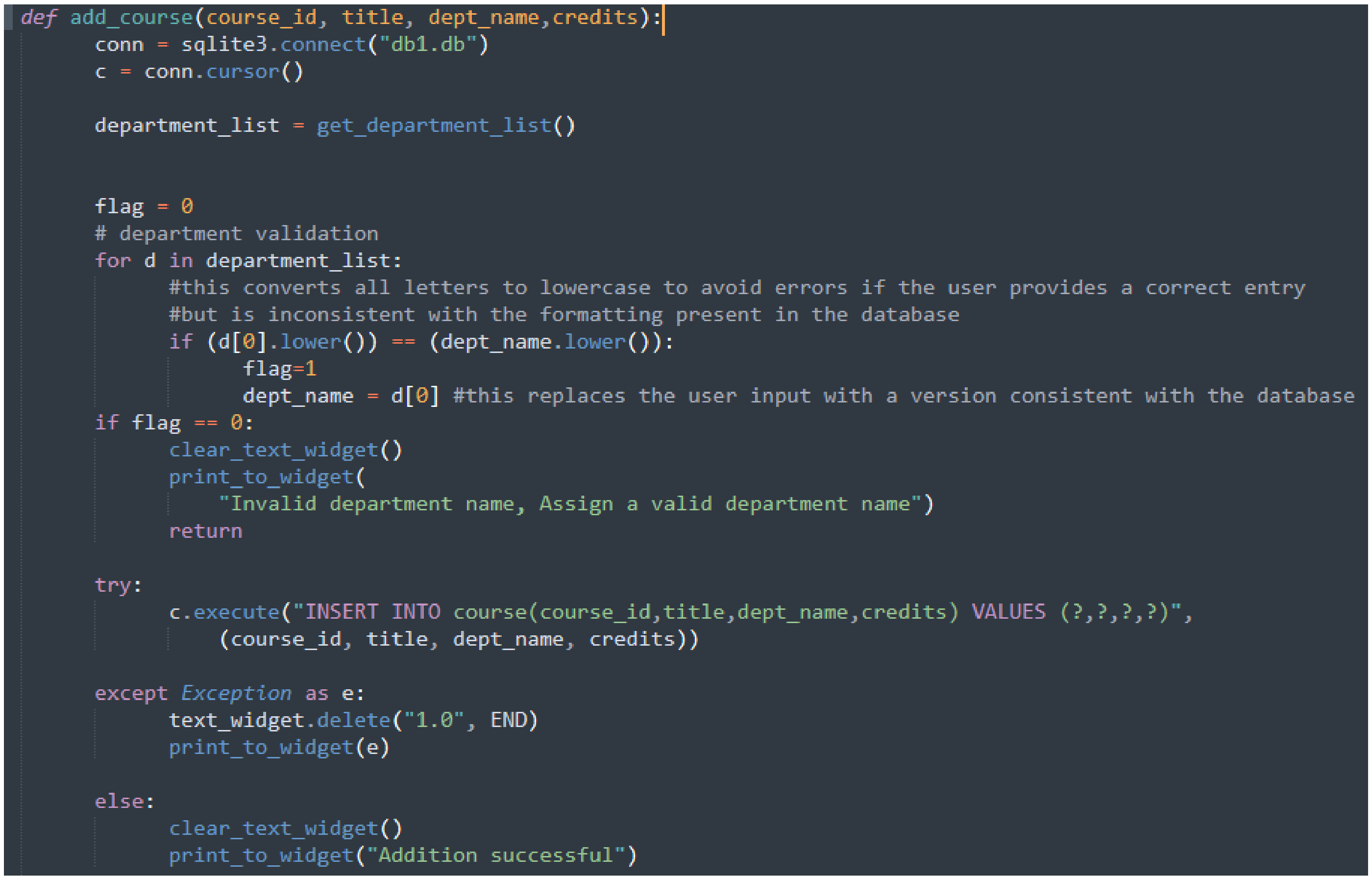
```python
def get_student_department(student_id):
      conn = sqlite3.connect("db1.db")

      c = conn.cursor()
      c.execute("SELECT dept_name from student where id= ?", (student_id,))
      item = c.fetchone()
      conn.close()
      return item
```



```python
s_dept=get_student_department(s_id)
i_dept=get_instructor_department(i_id)

if(s_dept != i_dept):
      clear_text_widget()
      print_to_widget(
          "Department names mismatch!\n Please assign advisor to student in the same department")
      return
```

Add_named functions

These are functions that receive parameters from the user input, where validation is necessary like in the last example, it is also carried out here.



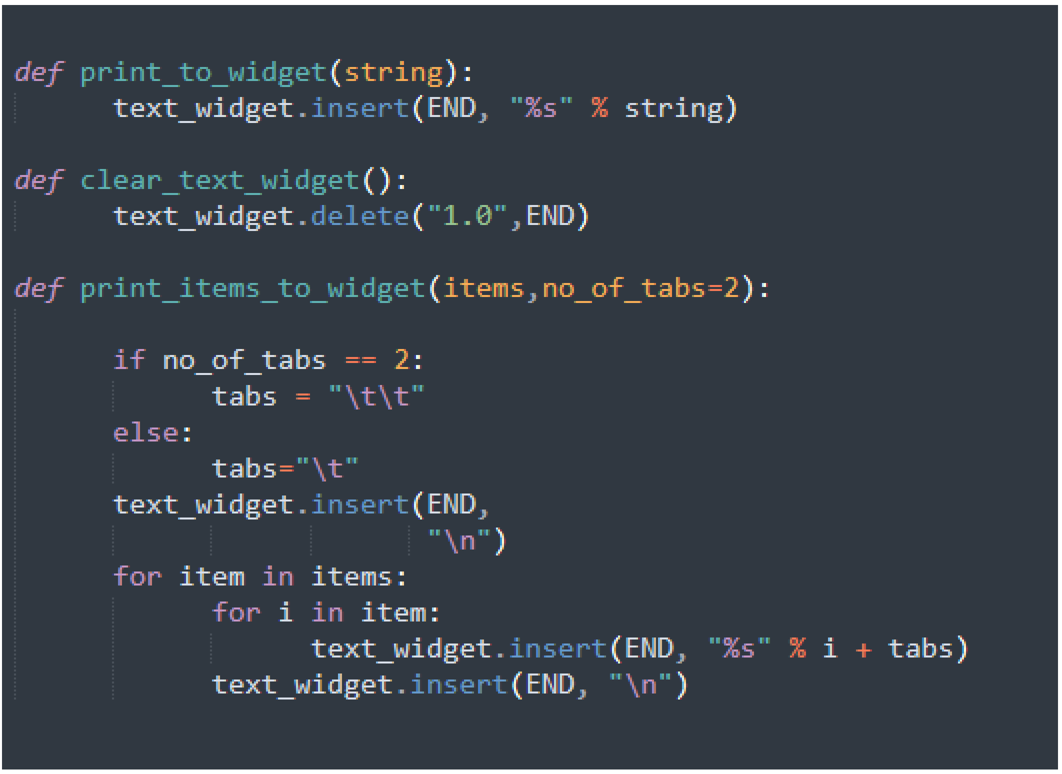
```
def add_course(course_id, title, dept_name,credits):
    conn = sqlite3.connect("db1.db")
    c = conn.cursor()

    department_list = get_department_list()


    flag = 0
    # department validation
    for d in department_list:
        #this converts all letters to lowercase to avoid errors if the user provides a correct entry
        #but is inconsistent with the formatting present in the database
        if (d[0].lower()) == (dept_name.lower()):
            flag=1
            dept_name = d[0] #this replaces the user input with a version consistent with the database
    if flag == 0:
        clear_text_widget()
        print_to_widget(
            "Invalid department name, Assign a valid department name")
        return

    try:
        c.execute("INSERT INTO course(course_id,title,dept_name,credits) VALUES (?,?,?,?)",
            (course_id, title, dept_name, credits))

    except Exception as e:
        text_widget.delete("1.0", END)
        print_to_widget(e)

    else:
        clear_text_widget()
        print_to_widget("Addition successful")
```

- Here in this function we first connect to the database then we retrieve a database_list object
- We set a flag object to 0
- We loop through the department object.
    - If there is a match between the inputted department name and the department names we compare with, set flag to 1
- If flag is equal to 0 print an error message
- Next we enter a try block
- Execute the statement using the user inputted values
- All sqlite database errors are caught and displayed.
- If no errors are displayed then print an “addition successful” message
- done

This is the general flow for all functions that are prefixed with “add”

Functions of Note

These are functions that are called frequently in the application code, they are regarded as important tools that facilitate the smooth working and readability of the application.



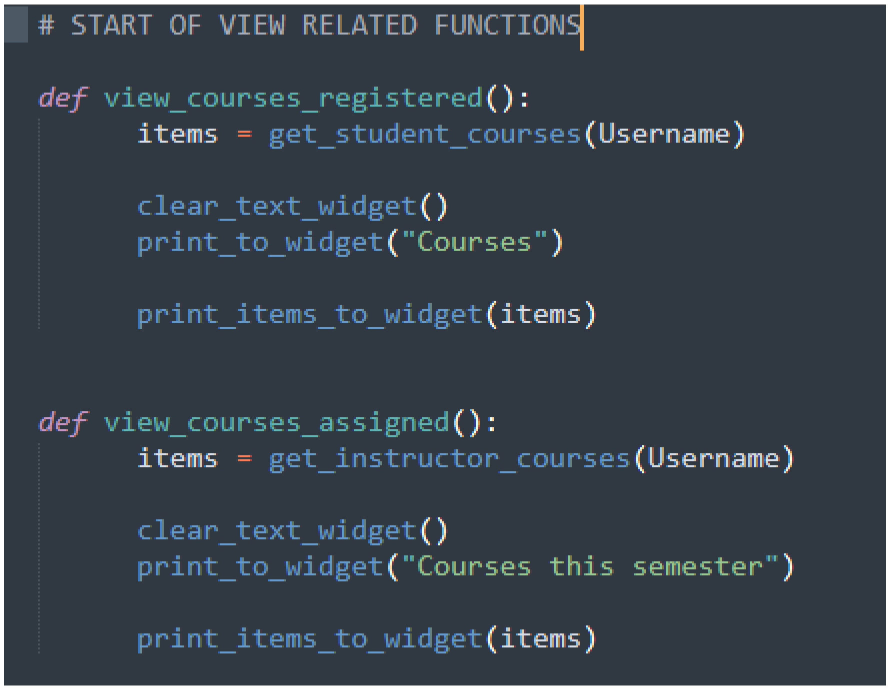
```python
def print_to_widget(string):
    text_widget.insert(END, "%s" % string)

def clear_text_widget():
    text_widget.delete("1.0",END)

def print_items_to_widget(items,no_of_tabs=2):

    if no_of_tabs == 2:
        tabs = "\t\t"
    else:
        tabs="\t"
    text_widget.insert(END,
                        "\n")
    for item in items:
        for i in item:
            text_widget.insert(END, "%s" % i + tabs)
        text_widget.insert(END, "\n")
```

Print_to_widget takes a string and prints it in the display box.

The *clear_text_widgets* is used to call the method that clears the display box without inputting parameters

Print_items_to_widget is a function that works with many types of objects,it is most commonly used to print objects that are a list of tuples(i.e " [(194234,stone),...] " It is designed to unpack these values and display them using 2 tabs as the default spacing, only between 1 and 2 tab spaces are used so the flexibility for the tab selection was kept to the minimum.

View_named Functions

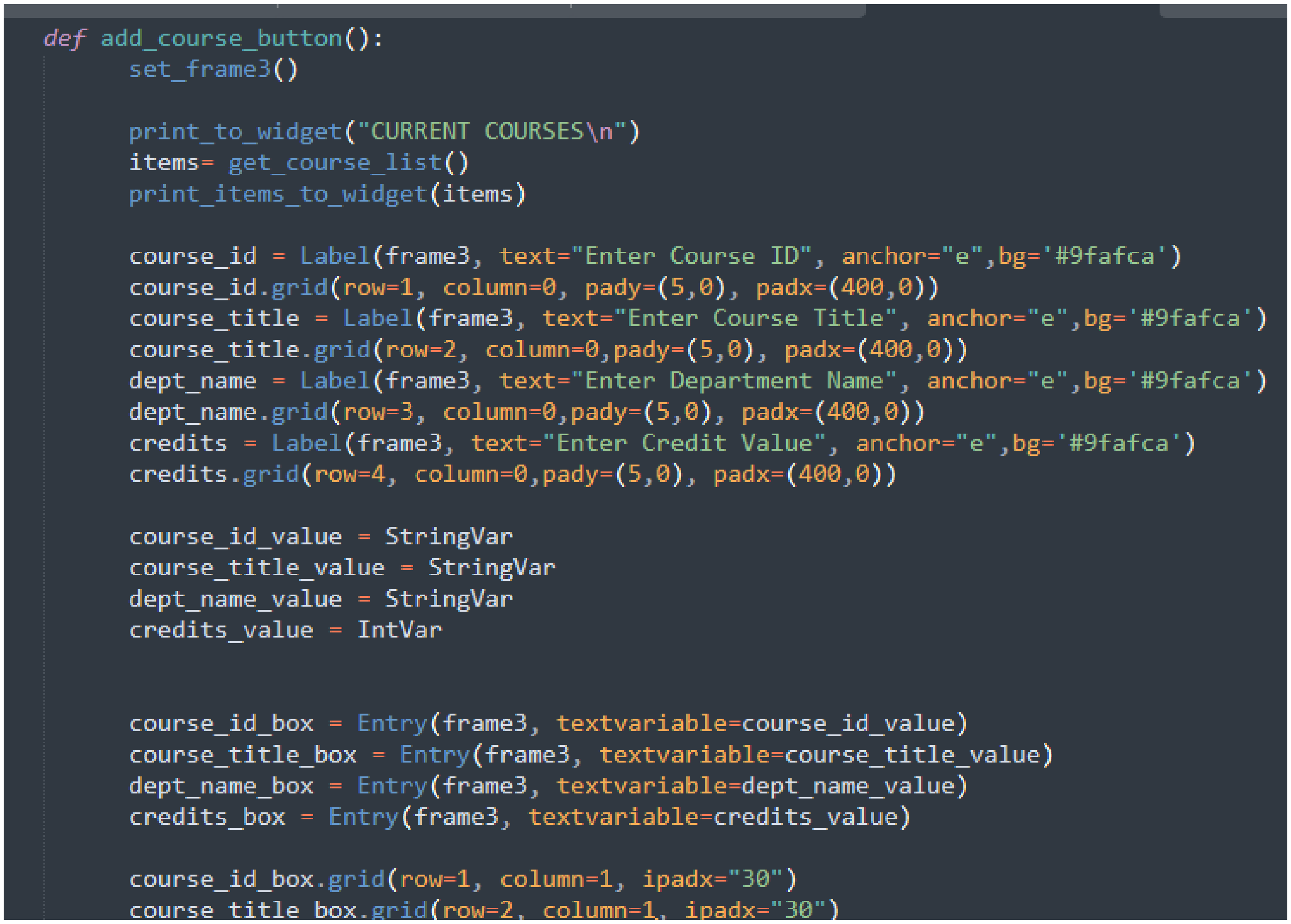


```python
# START OF VIEW RELATED FUNCTIONS

def view_courses_registered():
    items = get_student_courses(Username)

    clear_text_widget()
    print_to_widget("Courses")

    print_items_to_widget(items)


def view_courses_assigned():
    items = get_instructor_courses(Username)

    clear_text_widget()
    print_to_widget("Courses this semester")

    print_items_to_widget(items)
```

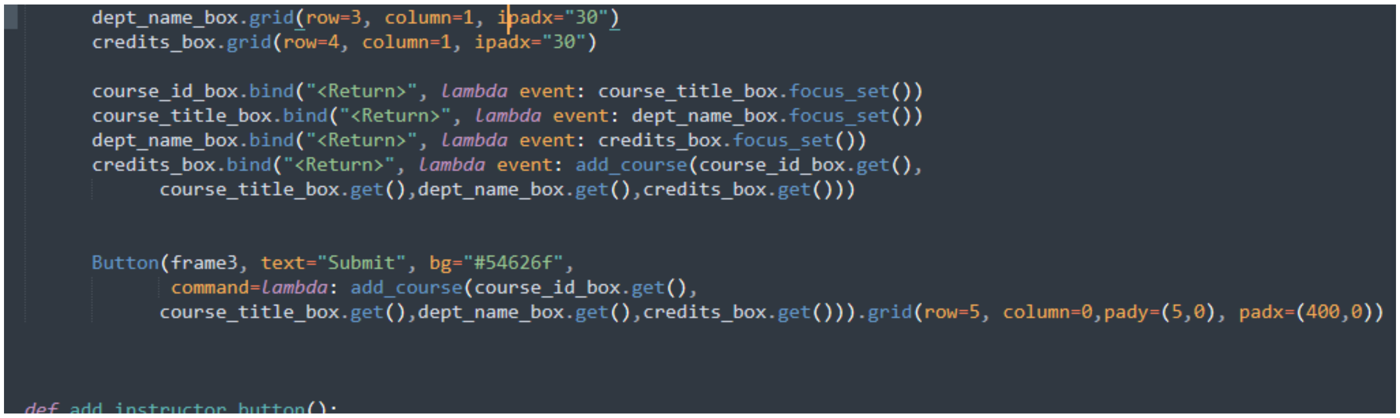
These functions have the task of calling the get function related to it and calling the *print_to_widget* function with the output from the get function as a parameter.

Add_named Functions(faculty)

These functions are used in the faculty section of the code to manage actions that will occur when the user clicks on the related button, for this example, when the faculty user clicks on the button, 4 labels,4 entry boxes and a submit button have to be created and printed to the screen, it also runs a special function called “bind” on the entry boxes which allows the user to fill in a detail and press enter to go to and prints out errors). This is the general structure for Add_named functions in this faculty section of the code.

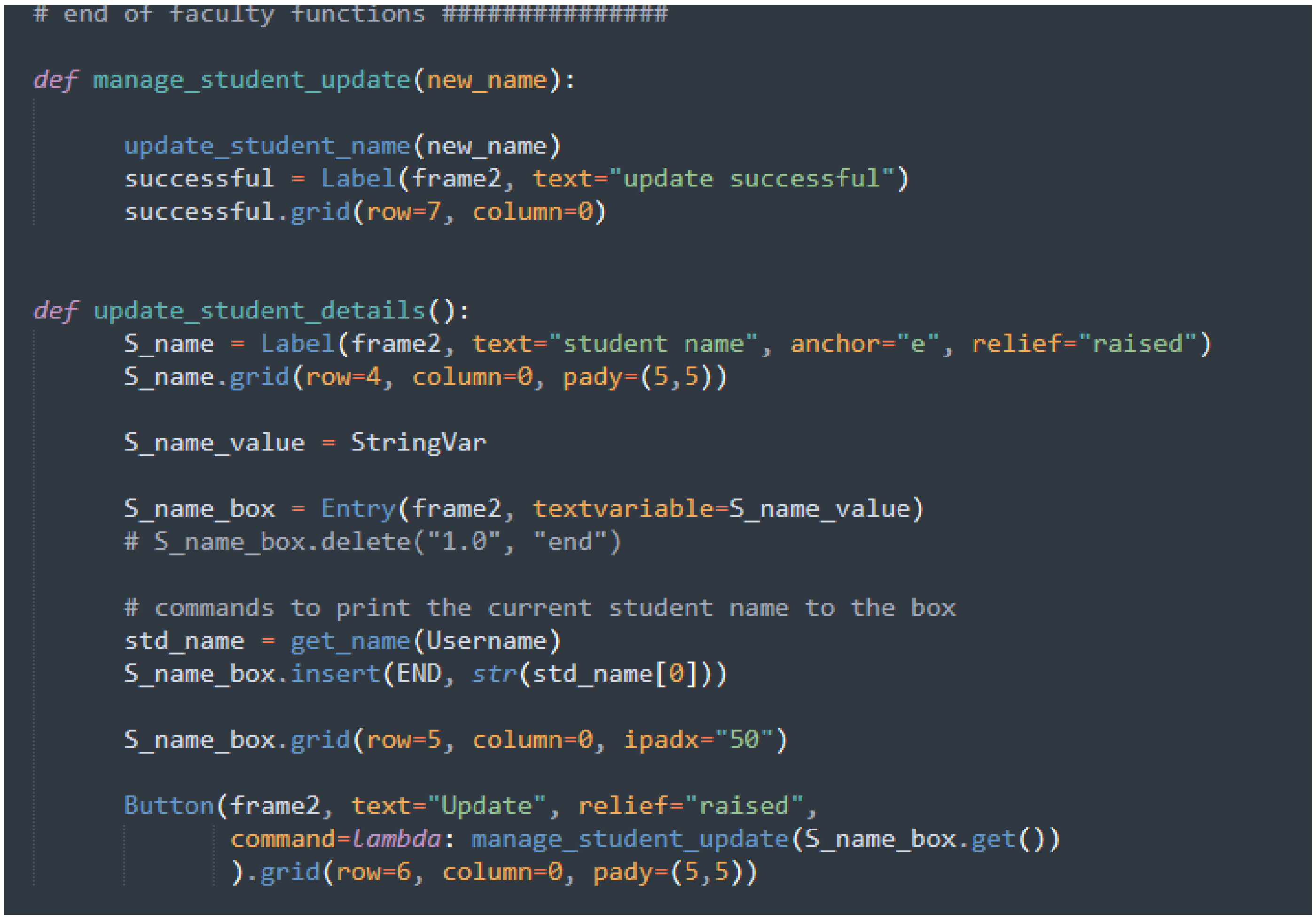


```
def add_course_button():
        set_frame3()

        print_to_widget("CURRENT COURSES\n")
        items= get_course_list()
        print_items_to_widget(items)

        course_id = Label(frame3, text="Enter Course ID", anchor="e",bg='#9fafca')
        course_id.grid(row=1, column=0, pady=(5,0), padx=(400,0))
        course_title = Label(frame3, text="Enter Course Title", anchor="e",bg='#9fafca')
        course_title.grid(row=2, column=0,pady=(5,0), padx=(400,0))
        dept_name = Label(frame3, text="Enter Department Name", anchor="e",bg='#9fafca')
        dept_name.grid(row=3, column=0,pady=(5,0), padx=(400,0))
        credits = Label(frame3, text="Enter Credit Value", anchor="e",bg='#9fafca')
        credits.grid(row=4, column=0,pady=(5,0), padx=(400,0))

        course_id_value = StringVar
        course_title_value = StringVar
        dept_name_value = StringVar
        credits_value = IntVar


        course_id_box = Entry(frame3, textvariable=course_id_value)
        course_title_box = Entry(frame3, textvariable=course_title_value)
        dept_name_box = Entry(frame3, textvariable=dept_name_value)
        credits_box = Entry(frame3, textvariable=credits_value)

        course_id_box.grid(row=1, column=1, ipadx="30")
        course_title_box.grid(row=2, column=1, ipadx="30")
```



```
        dept_name_box.grid(row=3, column=1, ipadx="30")
        credits_box.grid(row=4, column=1, ipadx="30")

        course_id_box.bind("<Return>", lambda event: course_title_box.focus_set())
        course_title_box.bind("<Return>", lambda event: dept_name_box.focus_set())
        dept_name_box.bind("<Return>", lambda event: credits_box.focus_set())
        credits_box.bind("<Return>", lambda event: add_course(course_id_box.get(),
                course_title_box.get(),dept_name_box.get(),credits_box.get()))


        Button(frame3, text="Submit", bg="#54626f",
                command=lambda: add_course(course_id_box.get(),
                course_title_box.get(),dept_name_box.get(),credits_box.get())).grid(row=5, column=0,pady=(5,0), padx=(400,0))


def add_instructor_button():
```

Update_named Details

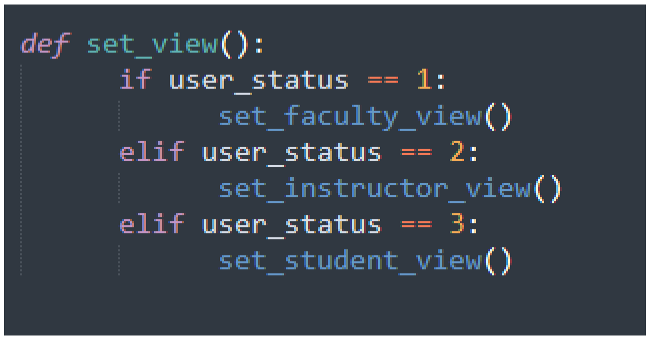


```
# end of faculty functions ###############

def manage_student_update(new_name):

    update_student_name(new_name)
    successful = Label(frame2, text="update successful")
    successful.grid(row=7, column=0)


def update_student_details():
    S_name = Label(frame2, text="student name", anchor="e", relief="raised")
    S_name.grid(row=4, column=0, pady=(5,5))

    S_name_value = StringVar

    S_name_box = Entry(frame2, textvariable=S_name_value)
    # S_name_box.delete("1.0", "end")

    # commands to print the current student name to the box
    std_name = get_name(Username)
    S_name_box.insert(END, str(std_name[0]))

    S_name_box.grid(row=5, column=0, ipadx="50")

    Button(frame2, text="Update", relief="raised",
           command=lambda: manage_student_update(S_name_box.get())
           ).grid(row=6, column=0, pady=(5,5))
```

There are two functions responsible for managing the update_details part of the student and instructor scenes. They display a label and an entry box supplied with the users current name, taken from a get function, then it displays a submit button which takes the data in the entry box and passes it as a parameter to a function which calls an update function and displays “successful”

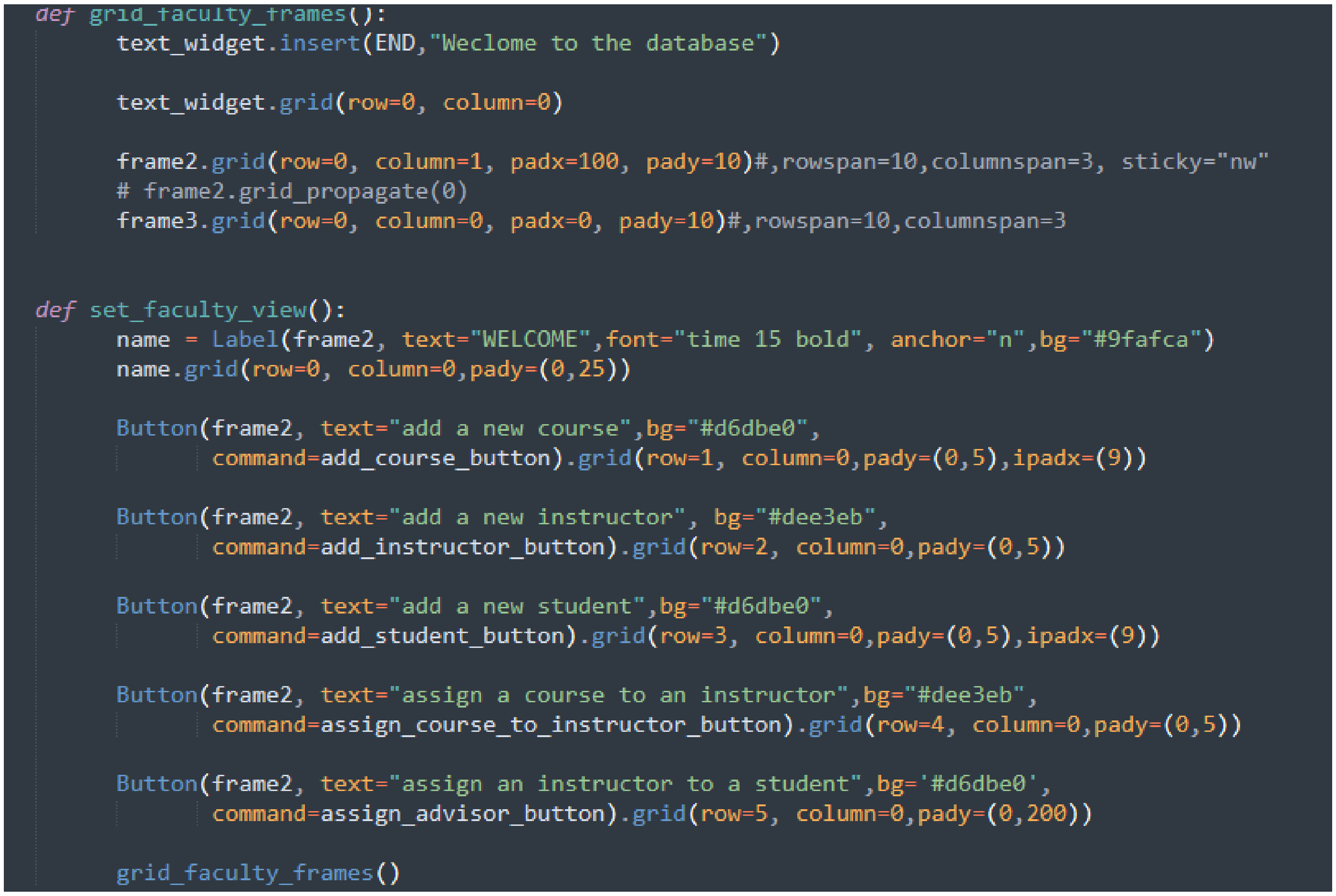
Set_view named Functions

These are functions responsible for placing the elements that the user will see and interact with on the screen. They are split into 3, 1 for each type of user and are customized to display the specific buttons that the user needs.



```
def set_view():
    if user_status == 1:
        set_faculty_view()
    elif user_status == 2:
        set_instructor_view()
    elif user_status == 3:
        set_student_view()
```

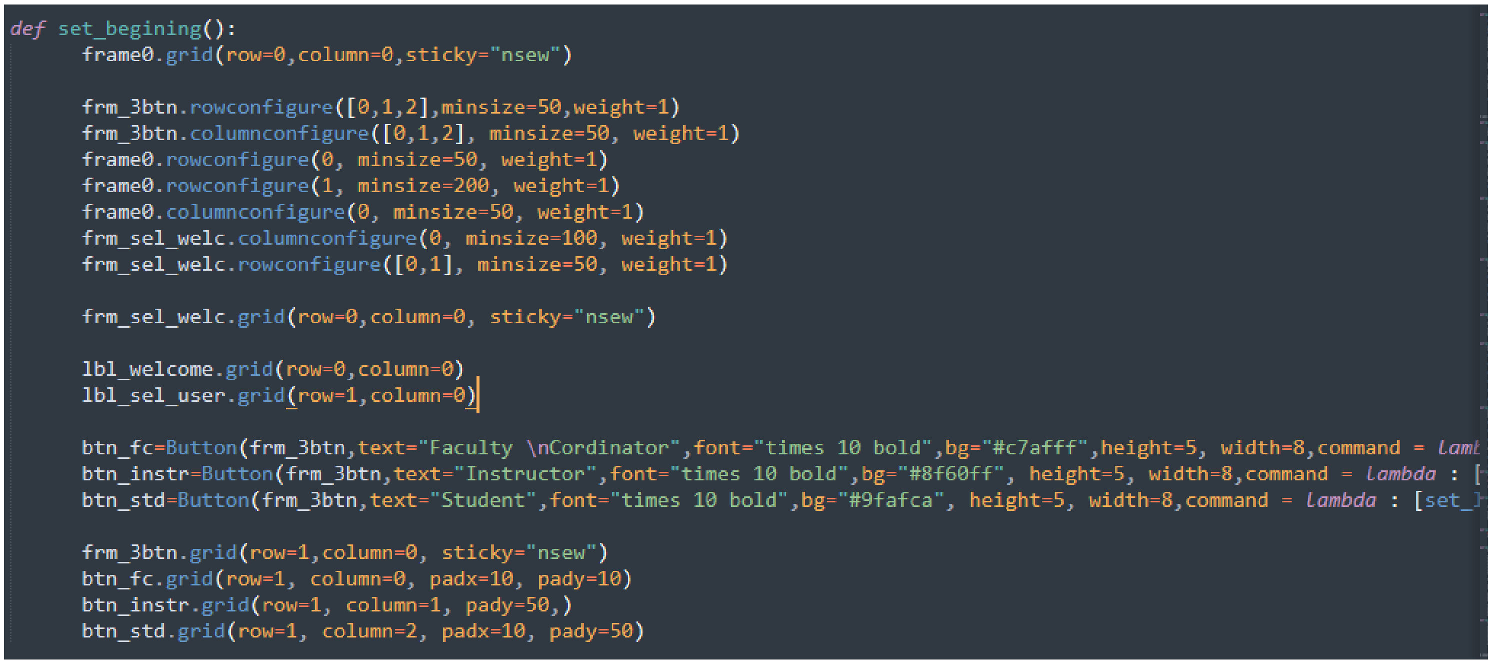
This is function of note whose job is to change the scene according to the *user_status* variable stated earlier. This allows the scenes to begin to diverge to satisfy the different needs of the applications users.



```
def grid_faculty_frames():
        text_widget.insert(END,"Weclome to the database")

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

        frame2.grid(row=0, column=1, padx=100, pady=10)#,rowspan=10,columnspan=3, sticky="nw"
        # frame2.grid_propagate(0)
        frame3.grid(row=0, column=0, padx=0, pady=10)#,rowspan=10,columnspan=3


def set_faculty_view():
        name = Label(frame2, text="WELCOME",font="time 15 bold", anchor="n",bg="#9fafca")
        name.grid(row=0, column=0,pady=(0,25))

        Button(frame2, text="add a new course",bg="#d6dbe0",
                command=add_course_button).grid(row=1, column=0,pady=(0,5),ipadx=(9))

        Button(frame2, text="add a new instructor", bg="#dee3eb",
                command=add_instructor_button).grid(row=2, column=0,pady=(0,5))

        Button(frame2, text="add a new student",bg="#d6dbe0",
                command=add_student_button).grid(row=3, column=0,pady=(0,5),ipadx=(9))

        Button(frame2, text="assign a course to an instructor",bg="#dee3eb",
                command=assign_course_to_instructor_button).grid(row=4, column=0,pady=(0,5))

        Button(frame2, text="assign an instructor to a student",bg='#d6dbe0',
                command=assign_advisor_button).grid(row=5, column=0,pady=(0,200))

        grid_faculty_frames()
```

This view function creates a label and displays it, then creates and displays the buttons for the faculty user, which are linked to different *add_named* functions in the database, lastly it calls a function which is responsible for printing the necessary things to the screen.

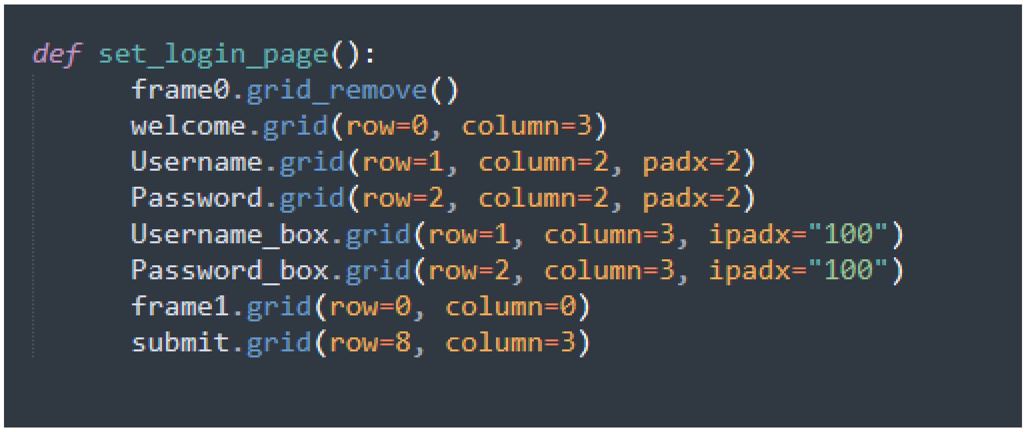
The Beginning



```
def set_begining():
      frame0.grid(row=0,column=0,sticky="nsew")

      frm_3btn.rowconfigure([0,1,2],minsize=50,weight=1)
      frm_3btn.columnconfigure([0,1,2], minsize=50, weight=1)
      frame0.rowconfigure(0, minsize=50, weight=1)
      frame0.rowconfigure(1, minsize=200, weight=1)
      frame0.columnconfigure(0, minsize=50, weight=1)
      frm_sel_welc.columnconfigure(0, minsize=100, weight=1)
      frm_sel_welc.rowconfigure([0,1], minsize=50, weight=1)

      frm_sel_welc.grid(row=0,column=0, sticky="nsew")

      lbl_welcome.grid(row=0,column=0)
      lbl_sel_user.grid(row=1,column=0)

      btn_fc=Button(frm_3btn,text="Faculty \nCordinator",font="times 10 bold",bg="#c7afff",height=5, width=8,command = lamb
      btn_instr=Button(frm_3btn,text="Instructor",font="times 10 bold",bg="#8f60ff", height=5, width=8,command = lambda : [
      btn_std=Button(frm_3btn,text="Student",font="times 10 bold",bg="#9fafca", height=5, width=8,command = lambda : [set_

      frm_3btn.grid(row=1,column=0, sticky="nsew")
      btn_fc.grid(row=1, column=0, padx=10, pady=10)
      btn_instr.grid(row=1, column=1, pady=50,)
      btn_std.grid(row=1, column=2, padx=10, pady=50)
```

This is where the first thing any user sees is printed from, and where we start using frames, these are containers for buttons and labels that we print to before displaying them on the screen, this scene and the login scene use only one frame but the final scene uses two frames to contain the labels and widget for application.

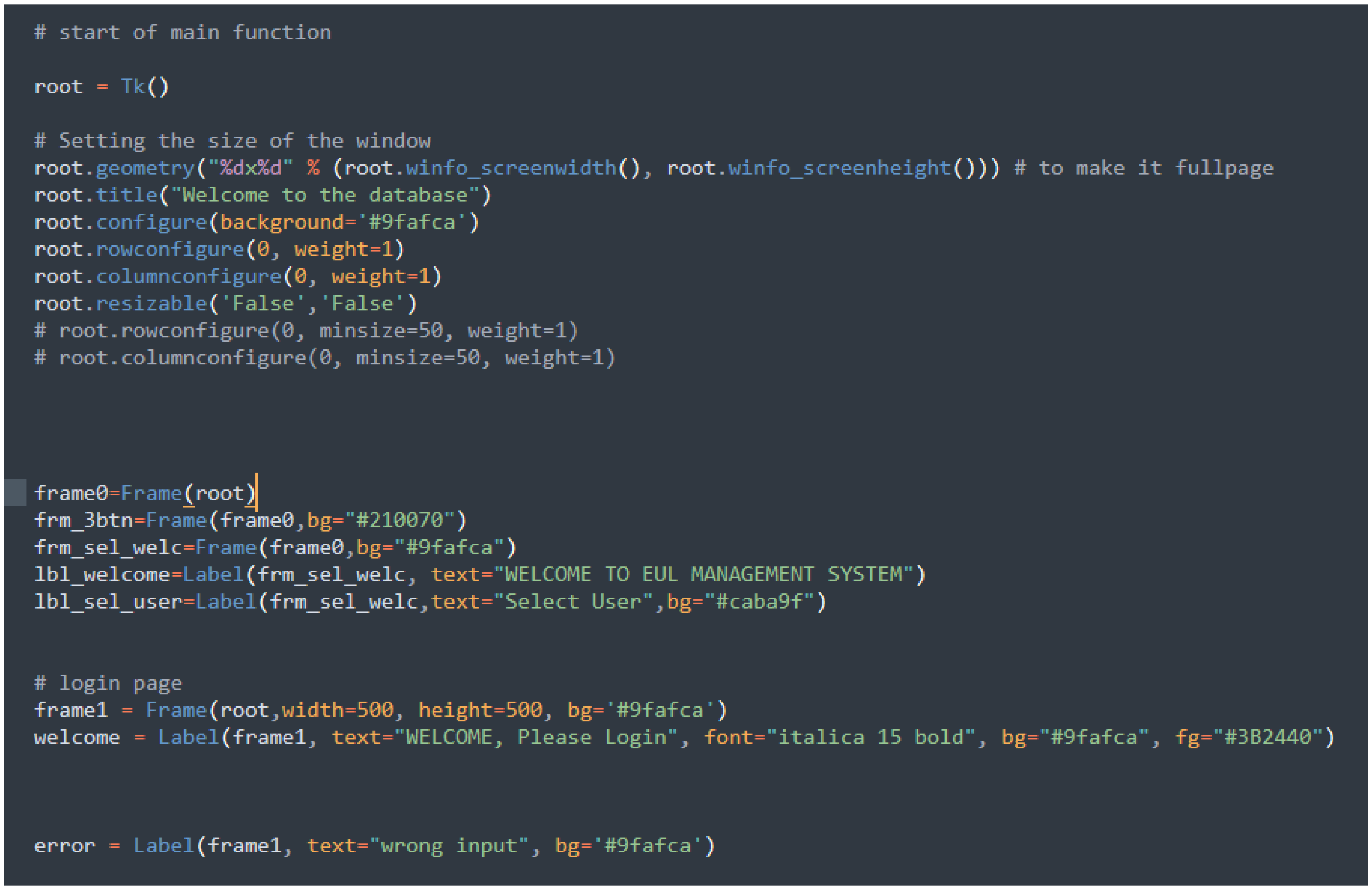
The login page



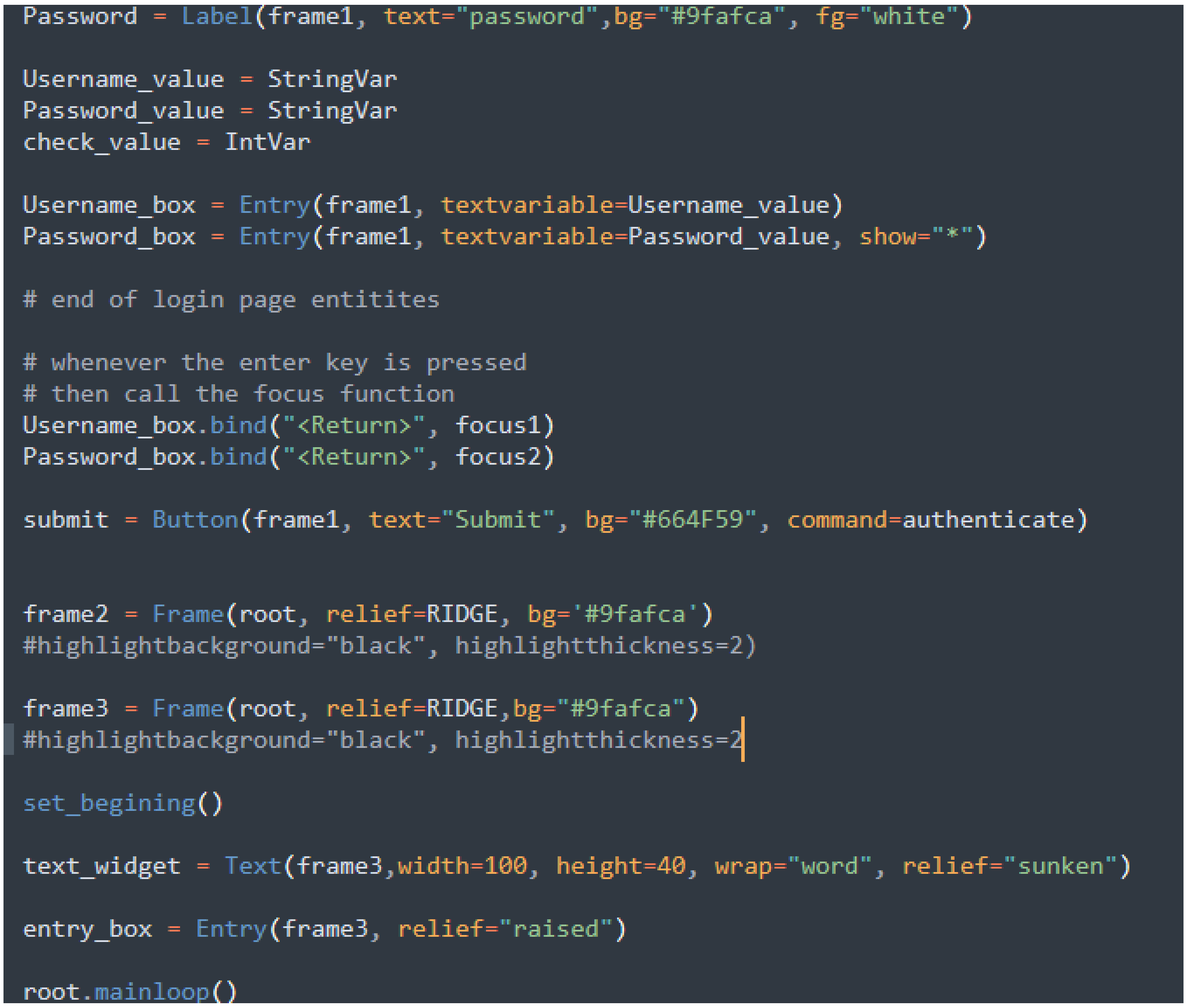
```
def set_login_page():
      frame0.grid_remove()
      welcome.grid(row=0, column=3)
      Username.grid(row=1, column=2, padx=2)
      Password.grid(row=2, column=2, padx=2)
      Username_box.grid(row=1, column=3, ipadx="100")
      Password_box.grid(row=2, column=3, ipadx="100")
      frame1.grid(row=0, column=0)
      submit.grid(row=8, column=3)
```

This is the function that displays the entities for the second scene, the login scene.

Main Function



```
# start of main function

root = Tk()

# Setting the size of the window
root.geometry("%dx%d" % (root.winfo_screenwidth(), root.winfo_screenheight())) # to make it fullpage
root.title("Welcome to the database")
root.configure(background='#9fafca')
root.rowconfigure(0, weight=1)
root.columnconfigure(0, weight=1)
root.resizable('False','False')
# root.rowconfigure(0, minsize=50, weight=1)
# root.columnconfigure(0, minsize=50, weight=1)




frame0=Frame(root)
frm_3btn=Frame(frame0,bg="#210070")
frm_sel_welc=Frame(frame0,bg="#9fafca")
lbl_welcome=Label(frm_sel_welc, text="WELCOME TO EUL MANAGEMENT SYSTEM")
lbl_sel_user=Label(frm_sel_welc,text="Select User",bg="#caba9f")


# login page
frame1 = Frame(root,width=500, height=500, bg='#9fafca')
welcome = Label(frame1, text="WELCOME, Please Login", font="italica 15 bold", bg="#9fafca", fg="#3B2440")



error = Label(frame1, text="wrong input", bg='#9fafca')
```



```
Password = Label(frame1, text="password",bg="#9fafca", fg="white")

Username_value = StringVar
Password_value = StringVar
check_value = IntVar

Username_box = Entry(frame1, textvariable=Username_value)
Password_box = Entry(frame1, textvariable=Password_value, show="*")

# end of login page entitites

# whenever the enter key is pressed
# then call the focus function
Username_box.bind("<Return>", focus1)
Password_box.bind("<Return>", focus2)

submit = Button(frame1, text="Submit", bg="#664F59", command=authenticate)


frame2 = Frame(root, relief=RIDGE, bg='#9fafca')
#highlightbackground="black", highlightthickness=2)

frame3 = Frame(root, relief=RIDGE,bg="#9fafca")
#highlightbackground="black", highlightthickness=2

set_begining()

text_widget = Text(frame3,width=100, height=40, wrap="word", relief="sunken")

entry_box = Entry(frame3, relief="raised")

root.mainloop()
```

This is the main function, in here all the general/shared entities that will be used by the functions are declared, in here is a root entity where all frames for each scene are placed onto, the main program will run with the call to the mainloop method.