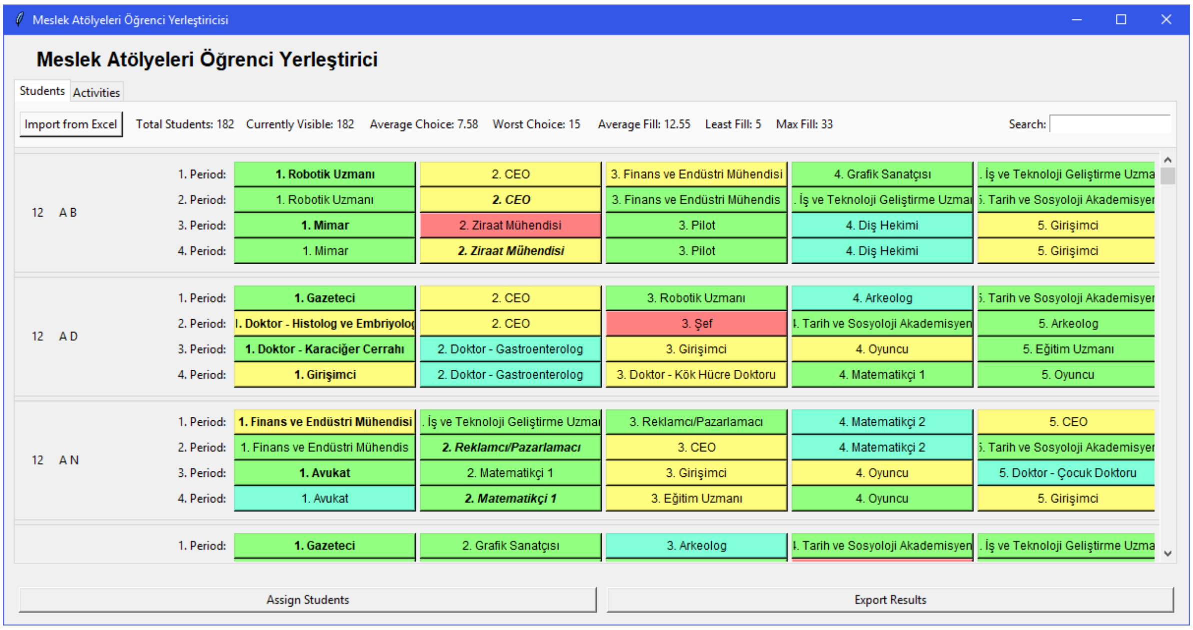
Criterion E: Product development

Complex Techniques Used

- Multiple different classes
- Arrays
- File Handling
- Functions with parameters and returns
- for, foreach, while loops, with and without exit conditions
- If-else
- UI

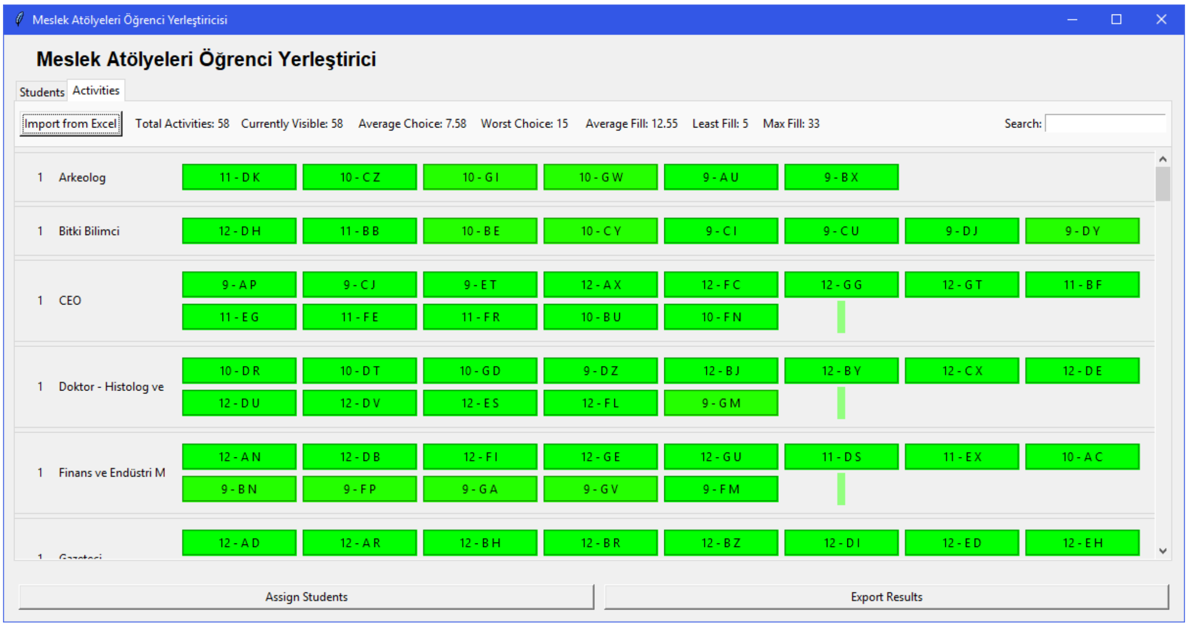
Final Interface


Meslek Atölyeleri Öğrenci Yerleştiricisi
Meslek Atölyeleri Öğrenci Yerleştirici
Students
Activities
Import from Excel
Search:
Assign Students
Export Results

Total Students: 182 Currently Visible: 182 Average Choice: 7.58 Worst Choice: 15 Average Fill: 12.55 Least Fill: 5 Max Fill: 33

12 A B	1. Period:	**1. Robotik Uzmanı**	2. CEO	3. Finans ve Endüstri Mühendisi	4. Grafik Sanatçısı	. İş ve Teknoloji Geliştirme Uzma
	2. Period:	1. Robotik Uzmanı	***2. CEO***	3. Finans ve Endüstri Mühendis	. İş ve Teknoloji Geliştirme Uzmaı	ı. Tarih ve Sosyoloji Akademisyeı
	3. Period:	**1. Mimar**	2. Ziraat Mühendisi	3. Pilot	4. Diş Hekimi	5. Girişimci
	4. Period:	1. Mimar	***2. Ziraat Mühendisi***	3. Pilot	4. Diş Hekimi	5. Girişimci
12 A D	1. Period:	**1. Gazeteci**	2. CEO	3. Robotik Uzmanı	4. Arkeolog	ı. Tarih ve Sosyoloji Akademisyeı
	2. Period:	**I. Doktor - Histolog ve Embriyolo**	2. CEO	3. Şef	I. Tarih ve Sosyoloji Akademisyen	5. Arkeolog
	3. Period:	**1. Doktor - Karaciğer Cerrahı**	2. Doktor - Gastroenterolog	3. Girişimci	4. Oyuncu	5. Eğitim Uzmanı
	4. Period:	**1. Girişimci**	2. Doktor - Gastroenterolog	3. Doktor - Kök Hücre Doktoru	4. Matematikçi 1	5. Oyuncu
12 A N	1. Period:	**1. Finans ve Endüstri Mühendisi**	. İş ve Teknoloji Geliştirme Uzmaı	3. Reklamcı/Pazarlamacı	4. Matematikçi 2	5. CEO
	2. Period:	1. Finans ve Endüstri Mühendis	***2. Reklamcı/Pazarlamacı***	3. CEO	4. Matematikçi 2	ı. Tarih ve Sosyoloji Akademisyeı
	3. Period:	**1. Avukat**	2. Matematikçi 1	3. Girişimci	4. Oyuncu	5. Doktor - Çocuk Doktoru
	4. Period:	1. Avukat	***2. Matematikçi 1***	3. Eğitim Uzmanı	4. Oyuncu	5. Girişimci
	1. Period:	**1. Gazeteci**	2. Grafik Sanatçısı	3. Arkeolog	I. Tarih ve Sosyoloji Akademisyen	. İş ve Teknoloji Geliştirme Uzma

Students screen

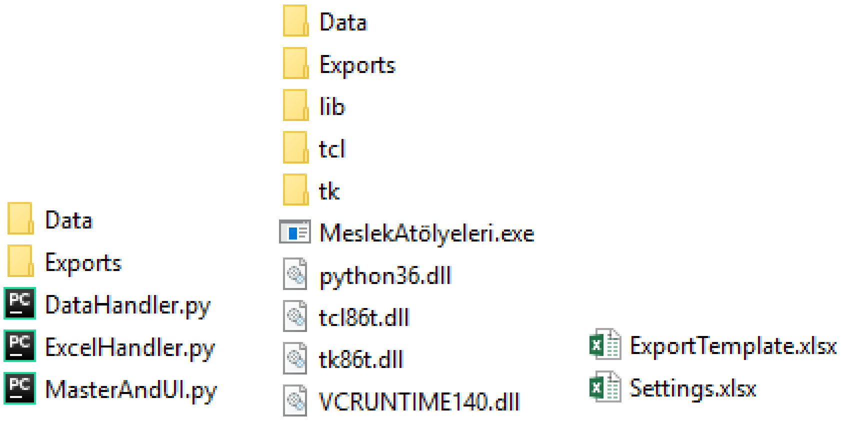

Meslek Atölyeleri Öğrenci Yerleştiricisi
Meslek Atölyeleri Öğrenci Yerleştirici
Students
Activities
Import from Excel
Search:
Assign Students
Export Results

Total Activities: 58 Currently Visible: 58 Average Choice: 7.58 Worst Choice: 15 Average Fill: 12.55 Least Fill: 5 Max Fill: 33

1	Arkeolog	11 - D K	10 - C Z	10 - G I	10 - G W	9 - A U	9 - B X		
1	Bitki Bilimci	12 - D H	11 - B B	10 - B E	10 - C Y	9 - C I	9 - C U	9 - D J	9 - D Y
1	CEO	9 - A P	9 - C J	9 - E T	12 - A X	12 - F C	12 - G G	12 - G T	11 - B F
		11 - E G	11 - F E	11 - F R	10 - B U	10 - F N			
1	Doktor - Histolog ve	10 - D R	10 - D T	10 - G D	9 - D Z	12 - B J	12 - B Y	12 - C X	12 - D E
		12 - D U	12 - D V	12 - E S	12 - F L	9 - G M			
1	Finans ve Endüstri M	12 - A N	12 - D B	12 - F I	12 - G E	12 - G U	11 - D S	11 - E X	10 - A C
		9 - B N	9 - F P	9 - G A	9 - G V	9 - F M			
1	Gazeteci	12 - A D	12 - A R	12 - B H	12 - B R	12 - B Z	12 - D I	12 - E D	12 - E H

Activities screen

Folder Structures



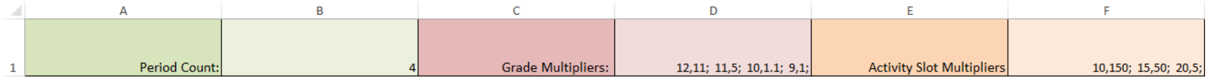
- Data
- Exports
- DataHandler.py
- ExcelHandler.py
- MasterAndUI.py

- Data
- Exports
- lib
- tcl
- tk
- MeslekAtölyeleri.exe
- python36.dll
- tcl86t.dll
- tk86t.dll
- VCRUNTIME140.dll

- ExportTemplate.xlsx
- Settings.xlsx

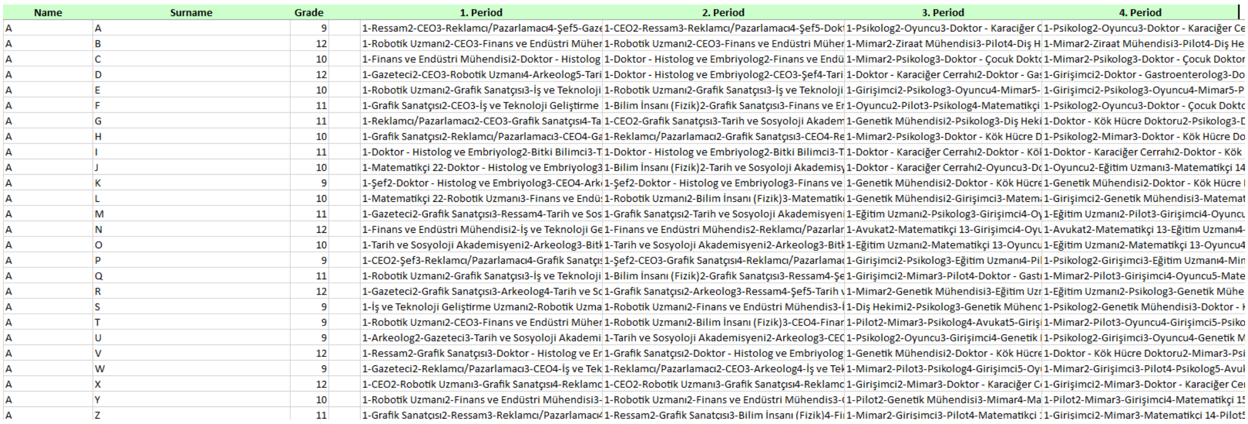
The folder structures for the source files (left), the final exe program (middle), the data folder (right). The settings file is loaded at startup and needs to be edited by hand to change algorithm settings.

Settings File



	A	B	C	D	E	F
1	Period Count:	4	Grade Multipliers:	12,11; 11,5; 10,1.1; 9,1;	Activity Slot Multipliers	10,150; 15,50; 20,5;

Import Excel Structure (Generated by Jotfrom.com)

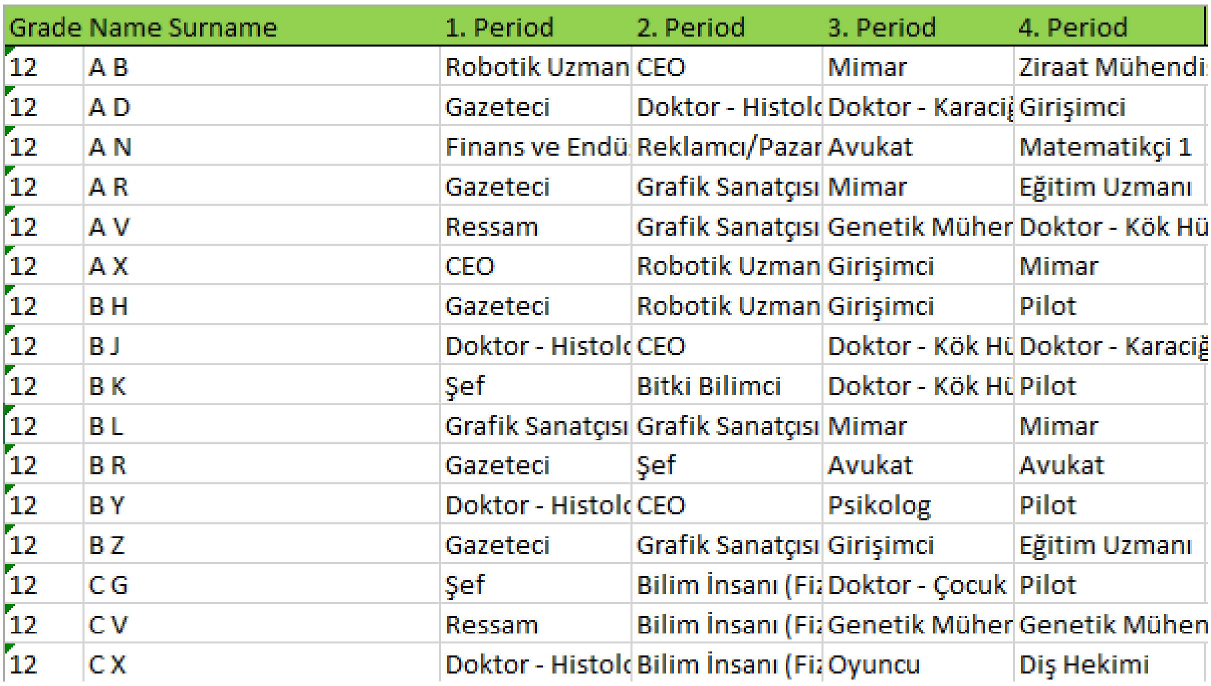


Name	Surname	Grade	1. Period	2. Period	3. Period	4. Period
A	A	9	1-Ressam2-CEO3-Reklamcı/Pazarlamacı4-Şef5-Gaze	1-CEO2-Ressam3-Reklamcı/Pazarlamacı4-Şef5-Dokt	1-Psikolog2-Oyuncu3-Doktor - Karaciğer C	1-Psikolog2-Oyuncu3-Doktor - Karaciğer Ce
A	B	12	1-Robotik Uzmanı2-CEO3-Finans ve Endüstri Müher	1-Robotik Uzmanı2-CEO3-Finans ve Endüstri Müher	1-Mimar2-Ziraat Mühendisi3-Pilot4-Diş H	1-Mimar2-Ziraat Mühendisi3-Pilot4-Diş He
A	C	10	1-Finans ve Endüstri Mühendisi2-Doktor - Histolog	1-Doktor - Histolog ve Embriyolog2-Finans ve Endü	1-Mimar2-Psikolog3-Doktor - Çocuk Dokto	1-Mimar2-Psikolog3-Doktor - Çocuk Doktor
A	D	12	1-Gazeteci2-CEO3-Robotik Uzmanı4-Arkeolog5-Tari	1-Doktor - Histolog ve Embriyolog2-CEO3-Şef4-Tari	1-Doktor - Karaciğer Cerrahı2-Doktor - Ga	1-Girişimci2-Doktor - Gastroenterolog3-Do
A	E	10	1-Robotik Uzmanı2-Grafik Sanatçısı3-İş ve Teknoloji	1-Robotik Uzmanı2-Grafik Sanatçısı3-İş ve Teknoloji	1-Girişimci2-Psikolog3-Oyuncu4-Mimar5-	1-Girişimci2-Psikolog3-Oyuncu4-Mimar5-P
A	F	11	1-Grafik Sanatçısı2-CEO3-İş ve Teknoloji Geliştirme	1-Bilim İnsanı (Fizik)2-Grafik Sanatçısı3-Finans ve Er	1-Oyuncu2-Pilot3-Psikolog4-Matematikçi	1-Psikolog2-Oyuncu3-Doktor - Çocuk Dokto
A	G	11	1-Reklamcı/Pazarlamacı2-CEO3-Grafik Sanatçısı4-Ta	1-CEO2-Grafik Sanatçısı3-Tarih ve Sosyoloji Akadem	1-Genetik Mühendisi2-Psikolog3-Diş Heki	1-Doktor - Kök Hücre Doktoru2-Psikolog3-D
A	H	10	1-Grafik Sanatçısı2-Reklamcı/Pazarlamacı3-CEO4-Ga	1-Reklamcı/Pazarlamacı2-Grafik Sanatçısı3-CEO4-Re	1-Mimar2-Psikolog3-Doktor - Kök Hücre D	1-Psikolog2-Mimar3-Doktor - Kök Hücre Do
A	I	11	1-Doktor - Histolog ve Embriyolog2-Bitki Bilimci3-T	1-Doktor - Histolog ve Embriyolog2-Bitki Bilimci3-T	1-Doktor - Karaciğer Cerrahı2-Doktor - Kö	1-Doktor - Karaciğer Cerrahı2-Doktor - Kök
A	J	10	1-Matematikçi 22-Doktor - Histolog ve Embriyolog3	1-Bilim İnsanı (Fizik)2-Tarih ve Sosyoloji Akademisy	1-Doktor - Karaciğer Cerrahı2-Oyuncu3-Do	1-Oyuncu2-Eğitim Uzmanı3-Matematikçi 14
A	K	9	1-Şef2-Doktor - Histolog ve Embriyolog3-CEO4-Ark	1-Şef2-Doktor - Histolog ve Embriyolog3-Finans ve	1-Genetik Mühendisi2-Doktor - Kök Hücre	1-Genetik Mühendisi2-Doktor - Kök Hücre
A	L	10	1-Matematikçi 22-Robotik Uzmanı3-Finans ve Endü	1-Robotik Uzmanı2-Bilim İnsanı (Fizik)3-Matematik	1-Genetik Mühendisi2-Girişimci3-Matem	1-Girişimci2-Genetik Mühendisi3-Matemat
A	M	11	1-Gazeteci2-Grafik Sanatçısı3-Ressam4-Tarih ve Sos	1-Grafik Sanatçısı2-Tarih ve Sosyoloji Akademisyeni	1-Eğitim Uzmanı2-Psikolog3-Girişimci4-Oy	1-Eğitim Uzmanı2-Pilot3-Girişimci4-Oyuncu
A	N	12	1-Finans ve Endüstri Mühendisi2-İş ve Teknoloji Ge	1-Finans ve Endüstri Mühendis2-Reklamcı/Pazarlar	1-Avukat2-Matematikçi 13-Girişimci4-Oyu	1-Avukat2-Matematikçi 13-Eğitim Uzmanı4-
A	O	10	1-Tarih ve Sosyoloji Akademisyeni2-Arkeolog3-Bitk	1-Tarih ve Sosyoloji Akademisyeni2-Arkeolog3-Bitk	1-Eğitim Uzmanı2-Matematikçi 13-Oyuncu	1-Eğitim Uzmanı2-Matematikçi 13-Oyuncu4
A	P	9	1-CEO2-Şef3-Reklamcı/Pazarlamacı4-Grafik Sanatçı	1-Şef2-CEO3-Grafik Sanatçısı4-Reklamcı/Pazarlama	1-Girişimci2-Psikolog3-Eğitim Uzmanı4-Pil	1-Psikolog2-Girişimci3-Eğitim Uzmanı4-Min
A	Q	11	1-Robotik Uzmanı2-Grafik Sanatçısı3-İş ve Teknoloji	1-Bilim İnsanı (Fizik)2-Grafik Sanatçısı3-Ressam4-Şe	1-Girişimci2-Mimar3-Pilot4-Doktor - Gast	1-Mimar2-Pilot3-Girişimci4-Oyuncu5-Mate
A	R	12	1-Gazeteci2-Grafik Sanatçısı3-Arkeolog4-Tarih ve Sc	1-Grafik Sanatçısı2-Arkeolog3-Ressam4-Şef5-Tarih v	1-Mimar2-Genetik Mühendisi3-Eğitim Uzr	1-Eğitim Uzmanı2-Psikolog3-Genetik Mühe
A	S	9	1-İş ve Teknoloji Geliştirme Uzmanı2-Robotik Uzma	1-Robotik Uzmanı2-Finans ve Endüstri Mühendis3-İ	1-Diş Hekimi2-Psikolog3-Genetik Mühenc	1-Psikolog2-Genetik Mühendisi3-Doktor - K
A	T	9	1-Robotik Uzmanı2-CEO3-Finans ve Endüstri Müher	1-Robotik Uzmanı2-Bilim İnsanı (Fizik)3-CEO4-Finar	1-Pilot2-Mimar3-Psikolog4-Avukat5-Girişi	1-Mimar2-Pilot3-Oyuncu4-Girişimci5-Psiko
A	U	9	1-Arkeolog2-Gazeteci3-Tarih ve Sosyoloji Akademi	1-Tarih ve Sosyoloji Akademisyeni2-Arkeolog3-CEC	1-Psikolog2-Oyuncu3-Girişimci4-Genetik	1-Psikolog2-Girişimci3-Oyuncu4-Genetik M
A	V	12	1-Ressam2-Grafik Sanatçısı3-Doktor - Histolog ve Er	1-Grafik Sanatçısı2-Doktor - Histolog ve Embriyolog	1-Genetik Mühendisi2-Doktor - Kök Hücre	1-Doktor - Kök Hücre Doktoru2-Mimar3-Psi
A	W	9	1-Gazeteci2-Reklamcı/Pazarlamacı3-CEO4-İş ve Tek	1-Reklamcı/Pazarlamacı2-CEO3-Arkeolog4-İş ve Tek	1-Mimar2-Pilot3-Psikolog4-Girişimci5-Oy	1-Mimar2-Girişimci3-Pilot4-Psikolog5-Avuk
A	X	12	1-CEO2-Robotik Uzmanı3-Grafik Sanatçısı4-Reklamc	1-CEO2-Robotik Uzmanı3-Grafik Sanatçısı4-Reklamc	1-Girişimci2-Mimar3-Doktor - Karaciğer C	1-Girişimci2-Mimar3-Doktor - Karaciğer Cer
A	Y	10	1-Robotik Uzmanı2-Finans ve Endüstri Mühendisi3-	1-Robotik Uzmanı2-Finans ve Endüstri Mühendis3-(	1-Pilot2-Genetik Mühendisi3-Mimar4-Ma	1-Pilot2-Mimar3-Girişimci4-Matematikçi 15
A	Z	11	1-Grafik Sanatçısı2-Ressam3-Reklamcı/Pazarlamacı4	1-Ressam2-Grafik Sanatçısı3-Bilim İnsanı (Fizik)4-Fi	1-Mimar2-Girişimci3-Pilot4-Matematikçi 1	1-Girişimci2-Mimar3-Matematikçi 14-Pilot5

This is an example input file. There are student names and students’ activity choices for each period. The picture is the Example Students.xlsx file which includes the student choices collected by my client during the event. The student names are changed for privacy, and the seminar names are in their original Turkish form.

Period data is in the following format:
[choice index]-[seminar name][choice index]-[seminar name]…

Export Excel Structure

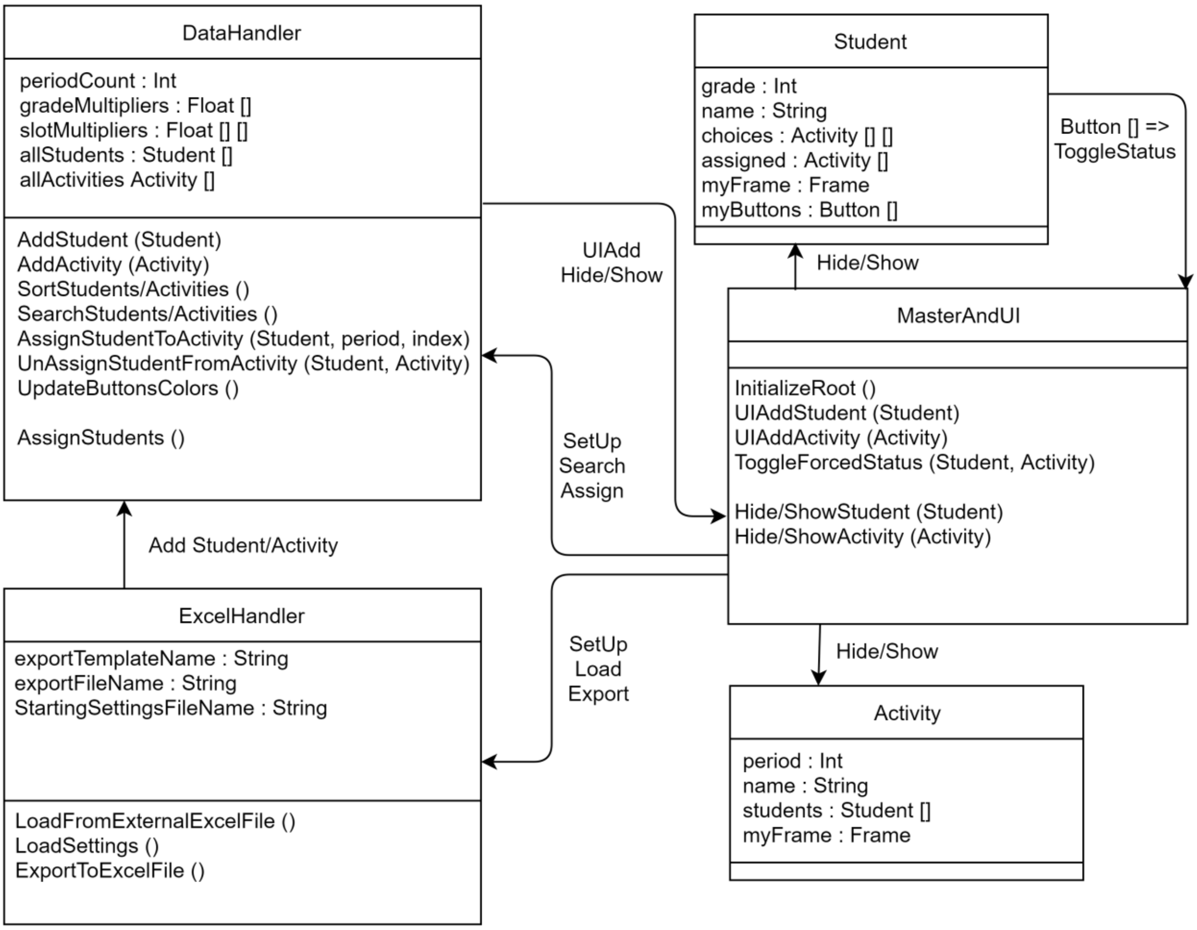


Grade	Name Surname	1. Period	2. Period	3. Period	4. Period
12	A B	Robotik Uzman	CEO	Mimar	Ziraat Mühendi
12	A D	Gazeteci	Doktor - Histolo	Doktor - Karaciğ	Girişimci
12	A N	Finans ve Endü	Reklamcı/Pazar	Avukat	Matematikçi 1
12	A R	Gazeteci	Grafik Sanatçısı	Mimar	Eğitim Uzmanı
12	A V	Ressam	Grafik Sanatçısı	Genetik Müher	Doktor - Kök Hü
12	A X	CEO	Robotik Uzman	Girişimci	Mimar
12	B H	Gazeteci	Robotik Uzman	Girişimci	Pilot
12	B J	Doktor - Histolo	CEO	Doktor - Kök Hü	Doktor - Karaciğ
12	B K	Şef	Bitki Bilimci	Doktor - Kök Hü	Pilot
12	B L	Grafik Sanatçısı	Grafik Sanatçısı	Mimar	Mimar
12	B R	Gazeteci	Şef	Avukat	Avukat
12	B Y	Doktor - Histolo	CEO	Psikolog	Pilot
12	B Z	Gazeteci	Grafik Sanatçısı	Girişimci	Eğitim Uzmanı
12	C G	Şef	Bilim İnsanı (Fiz	Doktor - Çocuk	Pilot
12	C V	Ressam	Bilim İnsanı (Fiz	Genetik Müher	Genetik Mühen
12	C X	Doktor - Histolo	Bilim İnsanı (Fiz	Oyuncu	Diş Hekimi

The file exported by the program.
Please note that originals of both of these files are in Turkish, and so the files exported by the program will have Turkish headers.

Multiple classes/Objects

UML Diagram – Overview of the classes

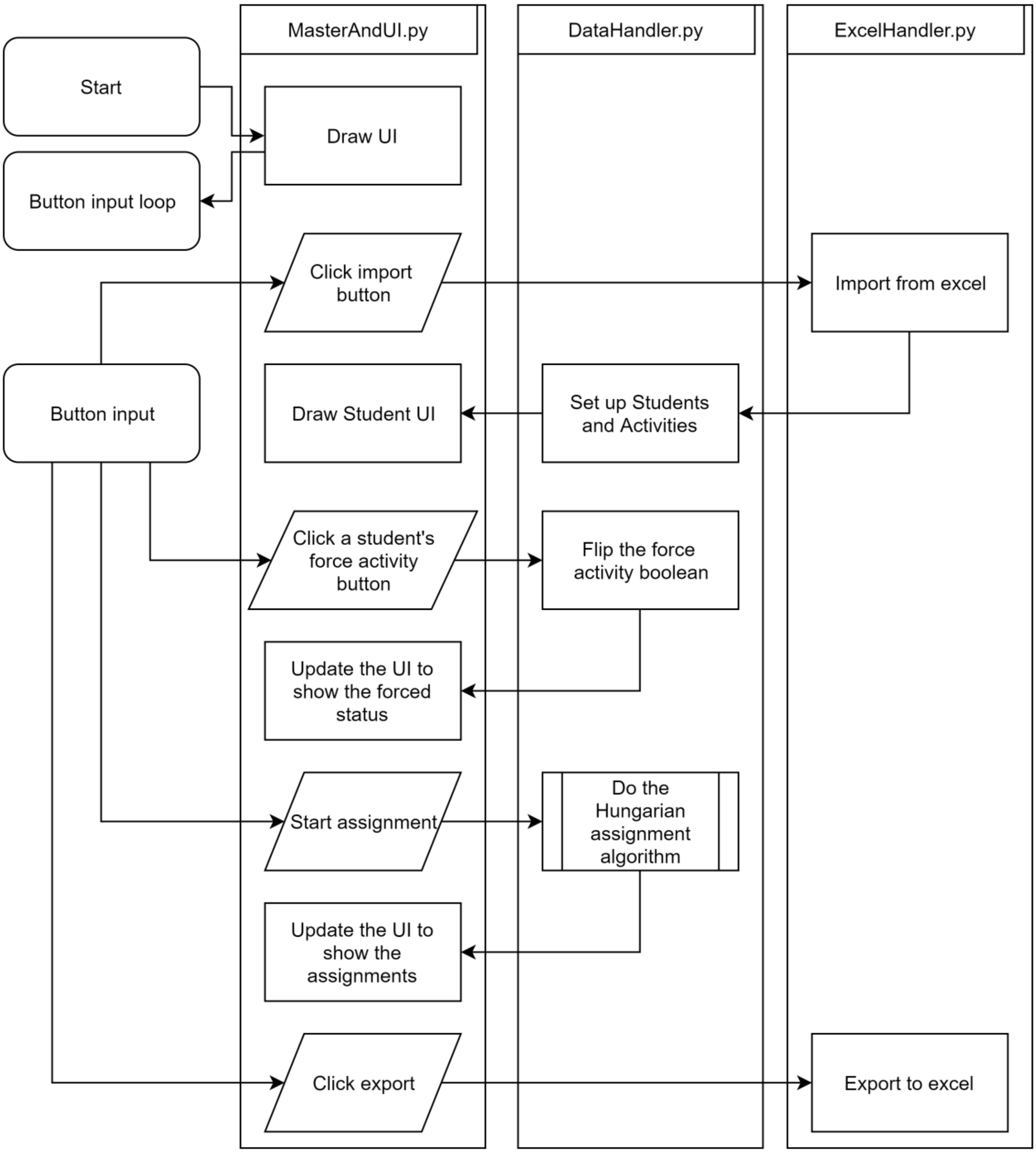


DataHandler.py: Stores the student and activity information and deals with operations for them. The main assignment algorithm is also in this file. The storage classes Student and Activity is also in this file.

MasterAndUI.py: Control and draws the UI. Initializes the other classes & passes the required references to those classes.

ExcelHandler.py: Deals with excel reading & writing

Flowchart


MasterAndUI.py
DataHandler.py
ExcelHandler.py
Start
Draw UI
Button input loop
Click import button
Import from excel
Button input
Draw Student UI
Set up Students and Activities
Click a student's force activity button
Flip the force activity boolean
Update the UI to show the forced status
Start assignment
Do the Hungarian assignment algorithm
Update the UI to show the assignments
Click export
Export to excel

Explanation of the code

DrawUI:

A section from the MasterAndUI.py:

Function with return

UI

```
def InitializeRoot():
    root = Tk ()
    root.geometry ("800x600")
    root.minsize = (800, 600);
    root.title ("Meslek Atölyeleri Öğrenci Yerleştiricisi")

    # Add a grid
    mainframe = Frame (root)
    mainframe.config (width = 800, height = 800);
    mainframe.pack (side = "top", fill = "both", expand = True, padx = 10, pady = 10);


    Label (mainframe, text = "Meslek Atölyeleri Öğrenci Yerleştirici", font =
("TkDefaultFont", 15, "bold")).pack (side = "top", anchor = "w", padx = 20);


    # ------------------------------General Tab Setup
    nb = ttk.Notebook (mainframe)
    tabStudents = Frame (nb)  # first page, which would get widgets gridded into it
    tabActivities = Frame (nb)  # second page
    nb.add (tabStudents, text = 'Students')
    nb.add (tabActivities, text = 'Activities')
    nb.pack (side = "top", fill = "both", expand = True);

    statsVar = StringVar ()
    statsVar.set ("Average Choice: -     Worst Choice: -" + "      " + "Average Fill: -
Least Fill: -     Max Fill: -")

    # ------------------------------Tab Student stuff
    stuTopStuff = Frame (tabStudents, bg = "#f9f9f9")
    stuTopStuff.pack (fill = "x", ipadx = 5, ipady = 10)

    addStuButton = Button (stuTopStuff, text = "Import from Excel", command =
ExcelHandler.LoadFromExternalExcelFile, bg = "#f9f9f9");
    addStuButton.pack (side = "left", padx = 5)

    stuSearchBar = Frame (stuTopStuff, bg = "#f9f9f9")
    stuSearchBar.pack (side = "right", padx = 5)

    stuSearchVar = StringVar()
    stuSearchVar.trace_add ("write", DataHandler.StuSearchBarUpdate)
    stuSearchEntry = Entry (stuSearchBar, textvariable=stuSearchVar);
    Label (stuSearchBar, text = "Search:", bg = "#f9f9f9").pack (side = "left")
    stuSearchEntry.pack (side = "left");

    tabStuScrollable = VerticalScrolledFrame (tabStudents, "Students", bg = "purple");
    tabStuScrollable.pack (fill = "both", expand = True)

    # ------------------------------End of Tab stuff
    Label (mainframe, height = 1).pack (fill = "x");  # spacer
    bottomButtonFrame = Frame(mainframe)
    bottomButtonFrame.pack(fill = "x", ipady=2)
    Button (bottomButtonFrame, text = "Assign Students", command = lambda :
DataHandler.AssignStudents()).pack (side="left", fill = "x", expand=1, padx=5);
    Button (bottomButtonFrame, text = "Export Results", command = lambda:
ExcelHandler.ExporttoExcelFile()).pack (side = "left", fill = "x", expand=1, padx=5);

    return root;
```

I used Tkinter to draw my UI. Tkinter “packs” elements into “frames” to organize the UI. I also use some of the premade UI elements like a “Notebook” for the tabs in my application. There are also some StringVars used here, which are updateable texts in the UI. I also used “VerticalScrolledFrame” which is a collection of other Tkinter elements made into a nice package that I took from http://tkinter.unpythonic.net/wiki/VerticalScrolledFrame.

Import Students

After drawing the UI the program waits for the user to import students.

```
def LoadFromExternalExcelFile ():
    inputPath = askopenfilename (initialdir = "myPath", title = "Please Select a Jotfrom
Generated Excel File");
    LoadFromExternalExcelFileWithPath (inputPath);


#load from an excel file generated with a jotform
def LoadFromExternalExcelFileWithPath (filePath):
    print("-*-*-");
    print("Loading from External Excel File")

    try:
        wb = load_workbook (filename = filePath)
        ws = wb.active
    except Exception as e:
        print(e)
        showerror("I/O Error", "Can't read the file. " + str(e));
        return


    print("Adding Students")
    x = 2
    while ws.cell(x,4).value != None:
        myName = ws.cell(x,4).value; #name
        myName += " " + ws.cell(x,5).value #surname
        stu = DataHandler.Student(grade = int(ws.cell(x,6).value), name = myName);

        stu.choices = []
        for n in range(DataHandler.periodCount):
            stu.choices.append([]);

        for period in range(DataHandler.periodCount):
            for act in ws.cell(x, 8 + period).value.split("\n"):
                if not (act == ""):
                    act = act.split("-");
                    act[0] = ""
                    actName = ""
                    for part in act:
                        if not part == "":
                            actName += part.replace("-","") + "-"

                    actName = actName[:-1]
                    #print(act[0] + " - " + act[1] + " - " + act[2])

                    stu.choices[period].append(
                       DataHandler.Student.Act(name=actName, isForced=False))

        stu.assigned = [None] * DataHandler.periodCount
        DataHandler.AddStudent(stu)

        x+=1
    print("-*-*-");

    print("Loaded from External Excel File Successfuly")
    print("-*-*-");
```

File I/O

Arrays

While loop

For loop

This section from ExcelHandler.py imports students from the excel file. It creates Student objects in the DataHandler file and adds the student object to the list with the AddStudent method.

```
def AddStudent (stu : Student):
    allStudents.append(stu)
    UIAddStudent (stu)
    SortStudents();
    StuSearchBarUpdate();

    p = 0
    for period in stu.choices:
        for act in period:
            isThereDuplicate = False
            for existingAct in allActivities:
                if(existingAct.period == p and existingAct.name == act.name):
                    isThereDuplicate = True;
                    break;
            if(not isThereDuplicate):
                AddActivity (Activity (name = act.name, period = p))
        p += 1;
```

Foreach loop

Break condition

This bit of code in DataHandler.py adds the students, updates the UI and if the student have a never seen before activity, adds that to the activity list. AddActivity method does the same adding and updating the UI.

UIAddStudent in MasterAndUI.py just adds the specific UI elements and assigns the various variables.

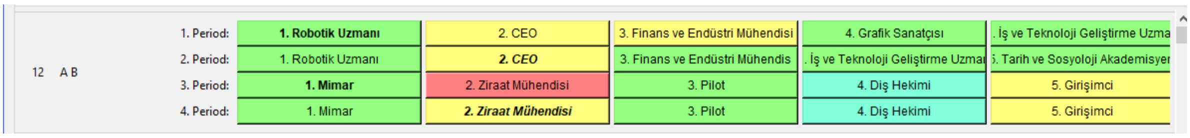
```
class Student:
    name = "Stu"
    grade = 9
    choices = []
    assigned = []
    myFrame : Frame = None
    myButtons : Button = []

    class Act:
        name = "Act"
        isForced = False;

        def __init__ (self, name, isForced):
            self.name = name
            self.isForced = isForced

    def __init__ (self, name, grade):
        self.name = name
        self.grade = grade

class Activity:
    name = "Act"
    period = 0
    assigned = []
    assignedCount = 0;
    myFrame : Frame = None
    stuNameParentFrame : Frame = None;
    stuNameFrame : Frame = None;

    def __init__ (self, name, period):
        self.name = name
        self.period = period;
        self.assigned = [];
```

Multiple classes/Objects

Arrays

Student and Activity classes in the DataHandler.py. These classes mostly hold data.

Force an Activity:



12 A B	1. Period:	**1. Robotik Uzmanı**	2. CEO	3. Finans ve Endüstri Mühendisi	4. Grafik Sanatçısı	. İş ve Teknoloji Geliştirme Uzma
	2. Period:	1. Robotik Uzmanı	***2. CEO***	3. Finans ve Endüstri Mühendis	. İş ve Teknoloji Geliştirme Uzmaı	i. Tarih ve Sosyoloji Akademisyer
	3. Period:	**1. Mimar**	2. Ziraat Mühendisi	3. Pilot	4. Diş Hekimi	5. Girişimci
	4. Period:	1. Mimar	***2. Ziraat Mühendisi***	3. Pilot	4. Diş Hekimi	5. Girişimci

Clicking any of the activities selected by the student will force assign that activity to that student. Assigned activities are in bold and force assigned activities are in italic.

From the MasterAndUI.py:

```python
def ToggleActivityForcedStatus (stu : DataHandler.Student, period, index):
    if stu.choices[period][index].isForced == True:
        stu.choices[period][index].isForced = False
        stu.myButtons[period][index].configure(font = "arial 9 normal")
        DataHandler.UpdateButtonColors()
    else:
        stu.choices[period][index].isForced = True
        stu.myButtons[period][index].configure(font = "arial 9 bold")

        DataHandler.AssignStudentToActivity(stu, period, index, True, "");
        DataHandler.UpdateActivityAssignmentsUI();
        DataHandler.CalculateAverages();

        n = 0
        for act in stu.choices[period]:
            if(act.isForced) and (n != index):
                ToggleActivityForcedStatus(stu,period,n);
            n += 1;
```

Function with parameters

Assigning students to the activities is also simple:

```
def AssignStudentToActivity (stu : Student, period, index, boolShouldUpdateButtonColors, actName):
    if(stu == None): #the hungarian algorithm have to assign dummy students
        print("Assigning dummy student")
        return;

    isForced = False;
    try:
        actName = stu.choices[period][index].name;
        isForced = stu.choices[period][index].isForced;
    except:
        #do nothing
        kek = 5;

    #if this student is already assigned to an activity for this period
    if (stu.assigned[period] != None):
        if (stu.assigned[period].name == actName): #if the same activity
            print(stu.name + " is already assigned to " + actName)
            if (boolShouldUpdateButtonColors):
                UpdateButtonColors();
            return;
        else: #if a different one, unassign first
            UnAssignStudentFromActivity (stu, period, GetActivityWithName (period,
stu.assigned[period].name))

    myAct: Activity = GetActivityWithName (period, actName);

    stu.assigned[period] = Student.Act(name = actName, isForced = isForced)

    isAssigned = False;
    for n in range(len(myAct.assigned)):
        if (myAct.assigned[n] == None): #assign the student to the first empty slot
            myAct.assigned[n] = stu;
            isAssigned = True
            break;
    if not isAssigned: #if no slot is available add new slot
        myAct.assigned.append(stu);

    myAct.assignedCount += 1;

    if(boolShouldUpdateButtonColors):
        UpdateButtonColors();

def UnAssignStudentFromActivity (stu : Student, period, act : Activity):
    act.assigned.remove(stu);
    act.assignedCount -= 1;
    print (stu.name + " is unassigned from " + str (period) + " - " + act.name)
```

If-else

Assign Students

Student assignment algorithm is the most complicated part of this application. I followed the matrix interpretation of the Hungarian algorithm from this Wikipedia article: https://www.wikiwand.com/en/Hungarian_algorithm. Unfortunately the article's explanation had a few errors and I had to fix it but the algorithm worked in the end.

The Hungarian algorithm is an algorithm for a situation like this: if you have set of "jobs" and a set of "workers" that each will do each of the "jobs" for a different "cost", the algorithm will match the jobs with the workers so that the cost is as small as possible. For my case, I need to match students with activities, trying to make sure everyone gets their best possible choice. So for that to work with the algorithm I give better choices better "value". Also 12th graders choices get multiplied. To fill a minimum amount of slots while having flexibility to have more students in an activity I make the first slots in a activity more valuable, and the rest less valuable, so the minimum number of slots should be matched by the algorithm. All these "values" get flipped in the end because the algorithm works to find the least, not the maximum.

The algorithm generates a 2D array as the "cost matrix"

```
costMatrix = [];

#populate the cost matrix
n = 0
for stu in allStudents:
    costMatrix.append([])
    for act in acts:
        for actSlot in range (MaxSlotCount):
            costMatrix[n].append(CostSlot(stu, act, CalculateCost(stu, act, actSlot, period)))
    n += 1;


length = totalActSlotCount;
#the hungarian algorithm works to minimize the cost, but we need to maximize it, so subtract
everything from the max cost
theMaxCost = 0

for r in range(length):
    for c in range(length):
        theMaxCost = max(theMaxCost, costMatrix[r][c].cost);

for rows in costMatrix:
    for slots in rows:
        slots.cost = theMaxCost - slots.cost;

class CostSlot:
    stu : Student;
    act : Activity;
    cost = 0;
    markCount = 0;
    assigned = False;
```

The cost function:

```
def CalculateCost (stu : Student, act : Activity, actSlot, period):
    cost = 0;

    if stu == None:
        return 0;

    choiceIndex = 0;
    forcedMult = 1;
    for choice in stu.choices[period]:
        if choice.name == act.name:
            if(choice.isForced):
                forcedMult = 1000;
            break;
        choiceIndex += 1;

    choiceMult = pow(2,-choiceIndex)
    if(choiceIndex > len(stu.choices[period])):
        choiceMult = 0;

    slotMult = slotMultipliers[3][1];

    if (actSlot < slotMultipliers[0][0]): slotMult = slotMultipliers[0][1];
    elif (actSlot < slotMultipliers[1][0]): slotMult = slotMultipliers[1][1];
    elif (actSlot < slotMultipliers[2][0]): slotMult = slotMultipliers[2][1];


    cost = gradeMultipliers[stu.grade] * slotMult * choiceMult * forcedMult;

    return cost;
```

After this the more complicated Hungarian algorithm parts come in. The full explanation the Hungarian algorithm is not in the scope of this IA, but I will put some code snippets and will explain some of the code sections. Please check the source code while looking the Wikipedia article if you want a deeper understanding.

Some of the preparation steps:

```
#find the least value in each row and subtract
for r in range(length):
    minVal = theMaxCost;
    for c in range(length):
        minVal = min(minVal, costMatrix[r][c].cost);
    for c in range(length):
        costMatrix[r][c].cost -= minVal;



#find the least value in each row and subtract
for r in range(length):
    minVal = theMaxCost;
    for c in range(length):
        minVal = min(minVal, costMatrix[r][c].cost);
    for c in range(length):
        costMatrix[r][c].cost -= minVal;
```

The most complicated part:

```
        ----Main algorithm comes from here----
        #Mark all rows having no assignments
        for unAssRow in nonAssignedRows:
            rMarks[unAssRow] = True;
            #Mark all columns having zeros in newly marked rows
            MarkColumnWithZeroes(costMatrix, cMarks, rMarks, unAssRow, length) #RECURSIVE MARKING METHOD
        ----Main algorithm continues after here----
```

```
def MarkColumnWithZeroes (costMatrix, cMarks, rMarks, row, length):
    # Mark all columns having zeros in newly marked rows
    for c in range (length):
        if costMatrix[row][c].cost == 0 and not cMarks[c]:
            cMarks[c] = True;
            DebugDrawMatrix (costMatrix, length, "Marking column with zero " + str(c), rMarks, cMarks)
            MarkRowWithAssignment(costMatrix, cMarks, rMarks, c, length)


def MarkRowWithAssignment (costMatrix, cMarks, rMarks, column, length):
    # Mark all rows having assignments in newly marked columns
    for r in range (length):
        if costMatrix[r][column].assigned and rMarks[r] == False:
            rMarks[r] = True;
            DebugDrawMatrix (costMatrix, length, "Marking row with assignment " + str(r), rMarks, cMarks)
            # mark all columns having zeros in newly marked rows
            MarkColumnWithZeroes (costMatrix, cMarks, rMarks, r, length)
```

This is a recursive method set that does one part of the 3 step process in the Wikipedia article. Each method calls the other one until there are no possible step points left.

After this process there are numerous more steps similar to these ones after. In the end all of the students are assigned to activities and the UI is updated:

```
UpdateButtonColors();

CalculateAverages();

#update activity student placements
UpdateActivityAssignmentsUI();

print("ASSIGNMENTS DONE");
```

I also wrote a debug console matrix drawer to fix the errors within the algorithm. You can only see this if you use the python version:

```python
def DebugDrawMatrix (_costMatrix, _length, message, _rMarks = [], _cMarks = []):
    print("-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-")
    print("DebugDrawing >> " + message)
    return

    columnNames = ""

    for c in range(_length):
        columnNames += _costMatrix[0][c].act.name + " - ";
    print(columnNames)

    topMarks = "   "
    if (len (_cMarks) >= _length):
        for b in _cMarks:
            if b:
                topMarks += "    *     -"
            else:
                topMarks += "          -"
        print(topMarks)

    for r in range(_length):
        row = ""
        if(_costMatrix[r][0].stu != None):
            row += _costMatrix[r][0].stu.name[0]
        else:
            row += "D"
        row += " > "
        for c in range(_length):
            middleThing = " - "
            marks = ""
            marks += "*" * _costMatrix[r][c].markCount;
            if(_costMatrix[r][c].assigned):
                marks += "~"
            else:
                marks += " "
            marks += " " * (2 - _costMatrix[r][c].markCount);
            row += "{: >5d}".format(int(_costMatrix[r][c].cost)) + marks + middleThing
        if (len(_rMarks) > r):
            if(_rMarks[r]):
                print(row + "*")
            else:
                print(row)
        else:
            print(row)


    print("-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-")
```

This bit of code draws the cost matrix into console. (it is disabled with a return statement at top as default to make the algorithm run faster)

Exporting to Excel

Export part is also simply writing the list of students & activities into an excel file (handled by the ExcelHandler.py):

```
def ExporttoExcelFile ():
    print ("-*-*-");
    print ("Exporting Excel File")

    date = datetime.datetime.now ();
    dateStr = str (date.day) + "-" + str (date.month) + "-" + str (date.year) + " - " + str
(date.hour) + "-" + str (date.minute);

    try:
        wb = load_workbook (filename = exportTemplate)
        ws_students = wb["Öğrenciler"]
        ws_activities = wb["Aktiviteler"]

        wb.save(exportFileName + dateStr + '.xlsx')
    except Exception as e:
        print(e)
        if (askretrycancel ("I/O Error", "There was an error while Exporting. " +
                              "You may continue your work, but your results aren't saved. " +
                              "To fix this issue please try closing any open files, checking
your antivirus, or running with administrator rights fix this issue. "
                                        + str(e))):
            return ExporttoExcelFile ();
        else:
            return


    print ("-*-*-");
    print("Students")
    x = 2
    for stu in DataHandler.allStudents:
        ws_students.cell (x, 1).value = str(stu.grade);
        ws_students.cell (x, 2).value = stu.name;

        p = 0
        assignedVals = ""
        for act in stu.assigned:
            if act != None:
                ws_students.cell (x, 3 + p).value = act.name;
                p+=1
        x+=1;

    print ("-*-*-");
    print("Activities")
    y = 1
    for act in DataHandler.allActivities:
        ws_activities.cell(1,y).value = str(act.period+1)+ " - " + str(act.name);
        x = 2
        for assStu in act.assigned:
            ws_activities.cell(x,y).value = str(assStu.name);
            x+=1
        y+=1

    wb.save (exportFileName + dateStr + '.xlsx')
    print ("Exported Successfuly: " + exportFileName + dateStr + '.xlsx')
    print ("-*-*-");
```

A full source code is in the product folder.

Word count: 906