```
#Evan DePosit
#New Beginnings
#capstone
#this file contains the class definitions for Class List, Reading Groups, and Student

#
#-------------------------------------------------------------------------------------------------------------
#                                                       $student
#
#-------------------------------------------------------------------------------------------------------------

class Class_List():
    def __init__(self, studentFile):
        #add number of days in week
        #start and stop times
        self.studentList=[]
        #self.readingGroupList=[]
        #delete this
        self.numOfGroups=0
        self.groupNumberList=[]
        self.readingGroups= {}

        studentData= self.read_stu_file(studentFile)

        #make list of of all students
        for kid in studentData:
            self.studentList.append(Student(kid))


        #count how many reading groups/levels and make list of group numbers
        #or make list of readingGroups an make dictionary for readingGroups
        maxGroupNum=0
        for student in self.studentList:
            if student.groupNumber > maxGroupNum:
                maxGroupNum= student.groupNumber
        self.numOfGroups=maxGroupNum
        #delete above
-------------------------------------------------------------------------

        #make list of student groups to allow iterating through groups in dictionary
        groupNumberList=[]
        for student in self.studentList:
            if student.groupNumber not in groupNumberList:
                groupNumberList.append(student.groupNumber)
        self.groupNumberList=groupNumberList

        #make dictionary of student groups
        for student in self.studentList:
            if student.groupNumber in self.readingGroups:
                self.readingGroups[student.groupNumber].add_student_to_group(student)
            else:

self.readingGroups[student.groupNumber]=Reading_Group(student.groupNumber)
                self.readingGroups[student.groupNumber].add_student_to_group(student)

        #test print students in each reading group
        #for group in self.groupNumberList:
            #group.print_reading_group()

    def read_stu_file(self, filePath):
    #input: filepath to student csv file
    #output: list of students, each student is list of frist, last and reading group/
```

```
level
    #reading group number/level must start at 0 and be contiguous to highest group
number
        studentCount=0
        columnCount=0
        fin = open(filePath, 'rt')

        #count how many fields in csv
        line=fin.readline()
        line=line[:-1:]
        #columnHeaders=[]
        #columnHeaders=line.split(',')
        #columnCount=len(columnHeaders)

        classData=[]

        while True:
            #read in each student by line and count total
            line= fin.readline()
            if not line:
                break
            line=line[:-1:]
            studentCount+=1
            #print each line of student data
            #print(line)

            #add to classData list
            studentData=line.split(',')
            classData.append(studentData)

        fin.close()
        return classData

    def sched_readingGroups(self, weekLen):
        for student in self.studentList:
            for day in range(0, weekLen):
                if day in student.actSched:
                    for act in student.actSched[day]:
                        #act.print_act()
                        #act.mate.print_event()
                        if day in student.eventSched:
                            student.eventSched[day].append(act.mate)
                        else:
                            student.eventSched[day]=[]
                            student.eventSched[day].append(act.mate)

    def make_free_list(self, eventTimes, weekLen):
        for student in self.studentList:
            #print(student.fullName)
            student.make_free_list(eventTimes, weekLen)

    def print_freeList_sched(self, weekLen):
        for student in self.studentList:
            print()
            print('---------------------------------------')
            print(student.fullName)
            print('---------------------------------------')
            student.print_freeList(weekLen)
            student.print_sched(weekLen)

    def print_freeList(self, weekLen):
        for student in self.studentList:
```

```
            print()
            print('--------------------------------------')
            print(student.fullName)
            print('--------------------------------------')
            student.print_freeList(weekLen)

    def print_sched(self, weekLen):
        for student in self.studentList:
            print()
            print('--------------------------------------')
            print(student.fullName)
            print('--------------------------------------')
            student.print_sched(weekLen)

    def sched_to_file(self, weekLen):


        fout=open('student_sched.csv', 'wt')
        fout.close()
        for student in self.studentList:
            studentLines=student.sched_to_file(weekLen)

            temp=dict(studentLines[0])
            headers=list(temp.keys())
            #print(headers)

            fout=open('student_sched.csv', 'at')
            fout.write(student.fullName + '\n')
            cout = csv.DictWriter(fout, headers)
            cout.writeheader()
            cout.writerows(studentLines)
            fout.close()

        #with open('student_sched.csv', 'wt') as fout:
            #cout = csv.DictWriter(fout, headers)
            #cout.writeheader()
            #cout.writerows(studentLines)

class Reading_Group():
    def __init__(self, groupNumber):
        self.groupNumber= groupNumber
        self.studentList=[]
        self.activityList=[]

    def add_student_to_group(self, student):
        self.studentList.append(student)

    def print_reading_group(self):
        print('Reading Group ', self.groupNumber)
        print('Student List:')
        if self.studentList:
            for student in self.studentList:
                student.print_student()
        else:
            print('no students in group')

    def add_actList(self, groupActList):
        #input list of readingGroup activities for group
        #output add list to groups act list and to each students sched
        self.activityList=groupActList

        for student in self.studentList:
```

```
            student.add_group_act(groupActList)


class Student():
    def __init__(self, studentData):
        self.first= studentData[0]
        self.last= studentData[1]
        self.readingLevel= int(studentData[2])
        self.groupNumber= int(studentData[2])
        self.actSched={}
        self.eventSched={}
        self.freeList={}
        self.fullName= studentData[0] + ' ' + studentData[1]

    def add_group_act(self, groupActList):
        for activity in groupActList:
            if activity.day in self.actSched:
                self.actSched[activity.day].append(activity)
            else:
                self.actSched[activity.day]=[]
                self.actSched[activity.day].append(activity)

    def print_student(self):
        print('{} {} {} {} {}'.format('student: ', self.first, self.last, 'reading
level: ', self.readingLevel))

    def make_free_list(self, eventTimes, weekLen):
        #print('eventTimes')
        #print(eventTimes)
        for day in range(0, weekLen):
            dayFreeList=[]
            #needs to have conditional if no events scheduled
            #print(self.eventSched[day])

            #if no events planned on day
            if day not in self.eventSched:
                for startEnd in eventTimes[day]:

                    #did I add 'free list event' for some reason. messed up the
constructor.
                    newFreeEvent= rg.Event(day, startEnd[0], startEnd[1], 'Free List
Event')
                    newFreeEvents.stuents=self
                    #newFreeEvent.print_event()

                    #newFreeEvent.students=self
                    dayFreeList.append(newFreeEvent)
                #add freelist to hash table for specified day
                self.freeList[day]=dayFreeList

            else:
                for startEnd in eventTimes[day]:
                    noEvent=True
                    for event in self.eventSched[day]:
                        if event.start == startEnd[0]:
                            noEvent=False
                    if noEvent:
                        #add student to event constructor?
                        newFreeEvent= rg.Event(day, startEnd[0], startEnd[1], 'Free
List Event')
                        newFreeEvent.students=self
```

```
                        dayFreeList.append(newFreeEvent)

                #add to freelist for to day to hash table
                self.freeList[day]=dayFreeList

            #self.print_freeList(day, weekLen)
        #for event in dayFreeList:
            #event.print_event()

    def print_freeList(self, weekLen):
        print('-------------------------------------')
        print('print_freeList')
        print('-------------------------------------')
        for day in range(0, weekLen):
            if day in self.freeList:
                for event in self.freeList[day]:
                    event.print_event()
            else:
                print('No Free List Events')

    def print_sched(self, weekLen):
        print('-------------------------------------')
        print('print_sched')
        print('-------------------------------------')
        for day in range(0, weekLen):
            #print('day', day)
            if day in self.eventSched:
                for event in self.eventSched[day]:
                    event.print_event()
            else:
                print('No Events on day', day)

    def sched_to_file(self, weekLen):
        fileLines=[]
        startTimeList=[]

        #go through all schedules to get lst of all the times
        for day in range(0, weekLen):
            for event in self.eventSched[day]:
                if event.start not in startTimeList:
                    startTimeList.append(event.start)
        startTimeList.sort()

        #print(self.fullName)
        #print(startTimeList)

        #initalize dictionary just to avoide using if else statements later
        schedTime={}
        for time in startTimeList:
            schedTime[time]={}
            schedTime[time]['Time']=tm.min_to_time(time)

        #go through student sched dictionary again
        for day in range(0, weekLen):
            for event in self.eventSched[day]:
                if event.teacher:
                    schedTime[event.start]['Day '+ str(event.day+1)]=event.type + '-'
+ event.teacher.name
                else:
                    schedTime[event.start]['Day '+
str(event.day+1)]=event.type
        #fileLines.append(self.fullName)
```

```
        for time in startTimeList:
            #print(schedTime[time])
            fileLines.append(schedTime[time])
        #print(fileLines)
        return fileLines

import readingGroups as rg
import timeConvert as tm
import csv
```