```
#Evan DePosit
#New Beginnings
#capstone
#this file contains main program for the reading group scheduler

#
----------------------------------------------------------------------------------------------------
#                                                                    $main
#
----------------------------------------------------------------------------------------------------

import readingGroups as rg
import timeConvert as tm
import resource as rs
import student as st

runType= input('enter "run" and press enter to run program.  Enter "test" and press
enter to run test:')
if runType == 'run':
    test= False
elif runType == 'test':
   test= True
else:
    print('invalid selection, running program using test data')
    test= True
# .............................
# make class schedule parameteres
# .............................
if test:
    numberOfDays=4
    actPerDay=4
    actDuration=20
else:
    numberOfDays=int(input('Enter the number of days per week in the schedule:'))
    actPerDay=int(input('Enter the number of reading activities that will take place
each day:'))
    actDuration=int(input('Enter enter the number of minutes allocated for the
duration of each activity:'))

schedParams1= rg.Schedule_Parameters(numberOfDays, actPerDay, actDuration)

if test:
    weekTimes= rg.input_sched_testTimes(numberOfDays)
else:
    weekTimes= rg.input_sched_Times(numberOfDays)


schedParams1.set_weeks_eTimes(weekTimes)
#schedParams1.print_all_eTimes()


#...............................
# make classList of students
#...............................

#csv <name>,<last>,<number> then list[1] == <last>
filePath='students.csv'
myClassList= st.Class_List(filePath)

# .................................
# make teacher clas and schedule
# .................................
```

```
teacherSchedLines=[]
myStaff = rg.Staff_Schedule()
teacherSchedLines= myStaff.read_teachers('teachers.csv')
myStaff.teacher_sched(teacherSchedLines)

#myStaff.print_staff()


# .............................
# put it all together!!???
# .............................
readingGroupSched1= rg.Reading_Group_Sched(myStaff, myClassList, schedParams1)
readingGroupSched1.make_group_event()
readingGroupSched1.make_group_act()

if test:
    readingGroupSched1.add_teacher_pref_test()
else:
    readingGroupSched1.add_teacher_pref()

readingGroupSched1.set_edges()
readingGroupSched1.unionVU= readingGroupSched1.V + readingGroupSched1.U
readingGroupSched1.max_match()
#test print of matches
#readingGroupSched1.print_group_teacher()
#readingGroupSched1.print_matches()

# .............................
#          make free list
# .............................
#print(schedParams1.dailyEvents)

#adds reading group events to students schedule
myClassList.sched_readingGroups(numberOfDays)

# filles in empty slots with free events
myClassList.make_free_list(schedParams1.dailyEvents, numberOfDays)
#myClassList.print_freeList_sched(numberOfDays)

# .............................
#          $resource schedule
# .............................
resourceSched1= rs.Resource_Sched(myClassList)

if test:
    resourceSched1.make_resources_test()
else:
    resourceSched1.make_resources()

#resourceSched1.print_all_resources()

resourceSched1.make_resource_events(schedParams1.dailyEvents, numberOfDays)

resourceSched1.init_resource_use()
resourceSched1.set_edges()
resourceSched1.match_events()

# .............................
#          $print sched
# .............................
```

```
#myClassList.print_sched(numberOfDays)
myClassList.sched_to_file(numberOfDays)
myStaff.sched_to_file(numberOfDays)
```