Last Update: December 15, 2018

**Author**

Onur Y. Özcan <<https://ozcan.io>>

**Description**

ADist stands for "Assistant Distribution" and has been created to help managing personnel to distribute teaching assistants to supervisory roles in exams within the department. This program has originally been used for the stated purpose at the Environmental Engineering Department at Istanbul Technical University, Turkey.

**Aim**

Create a program that:

* receives as input each assistant's official schedule
* receives as input every available exam date and timeframe
* asigns assistants to exams based on a fair point-based system and displays the results

**Modules**

-1- Assistant Schedule Input Interface

a module that lets assistants or the user enter their schedule to the input files to be processed by the program

---- this can be an online form with fields: assistant's name, name of course, course day, start time, end time

---- or assistans can directly edit a csv template file and send that, but i think the online form is a more elegant solution

---- the resulting csv file will have the data in tabular form to be processed by the program

---- a download button will be available for each file created

---- the online components will be coded after the program has been finalized

-2- Exam Schedule Input Interface

a module that lets the user enter the exam schedule to the input files to be processed by the program

---- also an online form with fields: course code, course registration number (crn), exam date, day of the week, start time, end time

---- or the user can directly edit a csv template file and use that as the input exam file for the program

---- the resulting csv file will have the data in tabular form to be processed by the program

---- a download button will be available for the file that has been created

---- the online components will be coded after the program has been finalized

-3- Input Processor

a module that reads both the assistant and exam schedule input files created in modules -1- and -2-, and stores them in memory as a data structure that the program can use

---- use strings to store 2d data. example:

dict = {"John Doe" :

"CEV201,Monday,0930,1130;CEV301,Wednesday,1330,1530;CEV401,Friday,1030,1230"}

---- convert the inputs to an integer form usable by the program (i.e. use a cypher)

-------- for example, instead of using "Monday\_0830" as an input datetime, this value will be converted simply to 1, "Monday\_0900" will be converted to 2, and so on

---- read files into a 2d tuple/list/dictionary (?)

---- later on, the input processor will be expanded to include specific dates that the assistants are unavailable, in addition to regular weekly duties

-4- Comparison Module

a module that compares the assistant schedules and exam dates and lists available names for each exam

---- the way to do this would be to create a loop that for each exam date and timeframe goes through each assistant's schedule and returns True for that specific assistant if there is no conflict

---- the output of this module would be a 2d tuple/list/dictionary containing the exams as keys and the list of available assistants for each exam as values

---- create an output file containing all names for all

-5- Distribution Module

a module that handles fair distribution of assistants

---- get the total number of exam supervisions required throughout the semester

---- divide that number by the total number of assistants available to get an average number of exams per assistant

---- if each assistant's current points total has been given as an input, take that into account. otherwise just distribute the exams as equally as possible

-6- Output Interface

a module that displays the results of the program

---- An online interface that displays the results in two different ways:

-------- the first is a list ordered by assistant name, with exams associated with each assistant listed under or next to their name

-------- the second is a list ordered by exam, with assistants assigned to each exam appearing next to the exam info

---- In addition to the online interface, two csv files that list the assistants and the exams in a similar manner will be created and a download link will be created for these files

---- the online components will be coded after the program has been finalized

Module Component Breakdown

-1- Assistant Schedule Input Interface

not thinking about this component yet

-2- Exam Schedule Input Interface

not thinking about this component yet

-3- Input Processor

this is where the design starts

-3.1- Read the Input Files into Memory

code examples:

// using 'with ... as' closes any open files automatically

// when the code block is done

filepath = 'input/assistants.csv'

with open(filepath) as fp:

for cnt, line in enumerate(fp):

print("Line {} : {}".format(cnt, line))

// using the split(',') method on a string breaks the lines down

// into lists containing the components split by the comma as strings

filepath = 'input/assistants.csv'

with open(filepath) as fp:

for line in readline(fp):

current\_line = line.split(',')

// learn the difference between read, readline, and readlines methods

// for reading lines from file objects created by using the open() method

-4- Comparison Module

-5- Distribution Module

-6- Output Interface

ADist stands for "Assistant Distribution" and has been created to help managing personnel to distribute teaching assistants to supervisory roles in exams within the department. This program has originally been used for the stated purpose at the Environmental Engineering Department at Istanbul Technical University, Turkey.