Technical Manual

For

SRU Course Scheduling

-Python Update-

Version 1.0

Prepared by William Davis, Samantha Ricketts

Slippery Rock University

March 27 2020

# Overview

This document provides a detailed description of the worked done within the Python Update addition to the SRU Course Scheduling Project.

# System Requirements

* Operating System: Microsoft Windows 7, 8, or 10 (64 bit)
* Working Installation of Python:
  + Python Editor of choice (this will cover the Anaconda Distribution including Spyder)

# Preparing your system

The following steps are needed in order to run convert the generalized Excel document into readable input for the MATLAB algorithm.

3.1.1 Download the GitHub Project

Download the project from the GitHub repository at <https://github.com/williamjwdavis/SRUCourseScheduling> and unzip it to the desired location.

3.1.2 Download & Installation of Anaconda

Download the Anaconda distribution from <https://www.anaconda.com/distribution/>. Note in the “Advanced Installation Options” menu during the installation process, both “Add Anaconda to my PATH…” and “Register Anaconda as my default…” should be checked.

# 3.2 Setup (Only necessary to view and edit the Python scripts)

Upon completion of both downloads, open Spyder (Windows Key + type “Spyder”). Now click “File-->Open-->”and use the browser to navigate to the location of the unzipped GitHub Project. The Python scripts lie within the pythonUpdate folder, and at this point, any files can be viewed or run in isolation using the Spyder environment.

# Running the Script

To run the script from Spyder, simply navigate to the run.py script and press F5, or click run.

Alternatively, the script can be executed by navigating to the pythonUpdate folder through command prompt, GitBash, or any comparable software. To do so, open the terminal and navigate to the file location, followed by using the command “python run.py”.

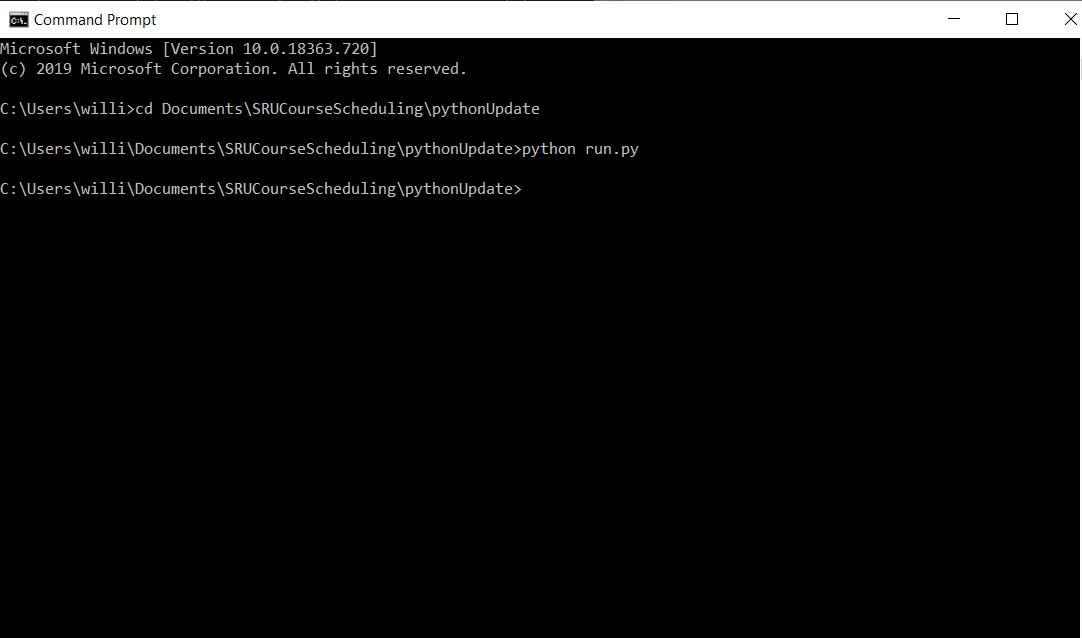


Figure 1- Command Prompt with the command to run the python script

# Explanation of Files and Folders

5.1 dictionaries

The dictionaries folder contains all of the dictionaries that are used in the preprocessing step. While these dictionaries come premade from GitHub, if these dictionaries aren’t present, they will be created and added to this folder.

5.1.1 classDict.pk1

The classDict dictionary contains a dictionary of class objects that each describe 1 class as directed by the input file in \_\_\_\_\_\_\_\_\_. These objects are directly dependent on the room objects.

5.1.2 profDict.pk1

The profDict dictionary contains a dictionary of professor objects that each describe 1 professor as directed by the input file in \_\_\_\_\_\_\_\_\_. These objects are directly dependent on both the class objects and room objects.

5.1.3 roomDict.pk1

The roomDict dictionary contains a dictionary of room objects that each describe 1 room as directed by the input file in \_\_\_\_\_\_\_\_\_. These objects are not independent of the other objects.

5.1.4 timeEncodingDict.pk1

The timeEncodingDict dictionary contains the first level of encodings of our implementation of the time slots available (this is viewable in \_\_\_\_\_\_\_). This level of encoding still contains meaning and is a product of various levels of modular logic.

5.1.5 timeEncodingDictFinal.pk1

The timeEncodingDictFinal dictionary contains the second level of encodings of the implementation of the time slots available (this is viewable in \_\_\_\_\_). This level of encoding loses the meaning that was present in 5.1.3. This encoding takes the previous encodings and pushes them together, so all of the values are numerically adjacent. This encoding is the final encoding used and passed to MATLAB.

5.2 input

The input folder, containing only Input.xlsx serves as the container for the input file that is used to read in all the information and transform it into a readable MATLAB format. The manual for using Input.xlsx are housed in \_\_\_\_\_\_\_\_\_\_.

5.3 output

The output folder, containing only matlabInputFinal.m serves as the container for the output file that is read into MATLAB, and used as the basis for the algorithm in MATLAB.

5.4 scripts

The scripts folder contains all of the general scripts used in the python preprocessing steps, apart from run.py.

5.4.1 buildTimeConstraints.py

The buildTimeConstraints script takes the timeEncodingDictFinal.pk1 from 5.1.5 and builds an array of forbidden pairs, or time slots that are not permitted to be used for the same class. Time slots are forbidden if they occur on the same day, or on different days but different time windows (i.e. Monday 8AM, Tuesday 9AM). This script is only run if the forbidden pairs array doesn’t exist.

5.4.2 buildTimeDict.py

The buildTimeDict script builds the first encoded dictionary for the time slots. Since the time window in consideration is arbitrary and subjective in the first place, it follows that this level of encoding is rather manual. A visual explanation of the encoding can be found in \_\_\_\_\_\_\_.

5.4.3 buildTimeDict2.py

The buildTimeDict script builds the second encoded dictionary for the time slots. This takes the previous encoding and puts all of the numerical values adjacent to each other according to a regular mod 10 system. This is then exported into timeEncodingDictFinal dictionary.

5.4.4 classInfo.py

The classInfo script contains the Class class, which attributes that come directly from

5.4.5 createInputVars.py

asdf

5.4.6 exportToTxt.py

asdf

5.4.7 handleTime.py

asdf

5.4.8 profInfo.py

asdf

5.4.9 roomInfo.py

asdf