# 2. Written Responses

Submit one PDF file in which you respond directly to each prompt. **Clearly label your responses 2a–2d in order. Your response to all prompts combined must not exceed 750 words, exclusive of the Program Code.**

## Program Purpose and Development

### 2a. Provide a written response or audio narration in your video that:

- identifies the programming language;
- identifies the purpose of your program; and
- explains what the video illustrates.

*(Approximately 150 words)*

The tool is python3.6. My program is made for saving each of our student's time for calculating the non-cumulative GPA for each semester. For users, which I assume do not understand the skills of computer science, the algorithm automatically installs all path and modules through cmd, if path and modules are not installed, before any further process is made. After all modules are loaded, the test module opens the login page of powerschool to login with the information that is inputted by the user. After driver gets to the page with the grade, html tables from this page is converted to the params that only contain array of each semester's score. Score of each subject are converted into GPA and saved in new params. Through special equation each semester's GPA is calculated and printed. Program requires enter to close the program. From the second run, no more user information is required. (149 words)

### 2b. Describe the incremental and iterative development process of your program, focusing on two distinct points in that process. Describe the difficulties and/or opportunities you encountered and how they were resolved or incorporated. In your description clearly indicate whether the development described was collaborative or independent. At least one of these points must refer to independent program development.

*(Approximately 200 words)*

There were two major difficulties during the development of program. First difficulty was getting the html information of the grade page, which includes the table of information for scores. The request module's login function did not work for the powerschool login page, therefore, I removed all the code made with request module, and used selenium module and chrome driver to open the actual website, to login and enter the grade page, to get the html information from the browser rather from the website server. Another difficulty was converting html table to table in python so that I can read the data collected. During the research, I found a code of Scott Rome for converting

html table to a readable python table, however, the structure of html table I collected didn't match his code perfectly. Therefore, I made the collected values stored into an array rather than a chart, and convert the array into two type of array inside one params to organize the data to the form that is easy to calculate. (172 words)

**2c. Capture and paste the program code segment that implements an algorithm (marked with an oval in section 3 below) that is fundamental for your program to achieve its intended purpose. Your code segment must include an algorithm that integrates other algorithms and integrates mathematical and/or logical concepts. Describe how each algorithm within your selected algorithm functions independently, as well as in combination with others, to form a new algorithm that helps to achieve the intended purpose of the program.**

*(Approximately 200 words)*

```
def calGPA(x,array):
    convertGPA(x,array)
    if(GPA[array][x]!='--'):
        numberofsub[array]=numberofsub[array]+1
        if(numberofsub[array]!=1):
            ss[array]=str(GPA[array][x])+'+'+str(ss[array])
        else:
            ss[array]=str(GPA[array][x])+str(ss[array])
    printGPA(x,array)
```

I chose this algorithm because it handles all the process of data conversion and calculation for GPA of imputed semester. When all the fundamental data is collected, the algorithm first converts $x^{th}$ score data into a GPA value, and add that into the array in the params. Then stored value is called to make the equation for the calculation. Finally condition for starting the calculation is checked and the result is printed when condition meets.

The algorithm convertGPA(x,array) shown below is integrated into the above algorithm. The algorithm convertGPA(x,array) is called to convert the called score value into a GPA value. The algorithm check if the value is above certain value continuously. When the condition meets, it puts the assigned GPA value to the array in the new params. If the value is not a score it puts '--' for the place.

```
def convertGPA(x,array):
    g=GPA[array]
    if(eval(array)[x]!='--'):
        if(int(eval(array)[x])>91):
            g[x]=4
        elif(int(eval(array)[x])>89):
            g[x]=3.667
        elif(int(eval(array)[x])>87):
            g[x]=3.333
        elif(int(eval(array)[x])>81):
            g[x]=3.000
```

```
        elif(int(eval(array)[x])>79):
            g[x]=2.667
        elif(int(eval(array)[x])>77):
            g[x]=2.333
        elif(int(eval(array)[x])>71):
            g[x]=2.000
        elif(int(eval(array)[x])>69):
            g[x]=1.667
        elif(int(eval(array)[x])>67):
            g[x]=1.333
        elif(int(eval(array)[x])>61):
            g[x]=1.000
        elif(int(eval(array)[x])>59):
            g[x]=0.667
        else:
            g[x]=0.000
    else:
        g[x]='--'
```

The algorithm printGPA(x,array) shown below is another integrated algorithm that calculates the equation and prints the result when the x is equal to number of GPA minus one. Both of these algorithm (convertGPA(x,array) and printGPA(x,array)) are noteworthy algorithms that are integrated in the main algorithm in order to calculate each semester's GPA value. (196 words)

```
def printGPA(x,array):
    if(x==int(z/20)-1):
        if(ss[array]!=''):
            print(str(array)+' GPA: '+str((eval(ss[array])+0.666*AP+0.333*HONOR)/numberofsub[array]))
```

**2d. Capture and paste the program code segment that contains an abstraction you developed (marked with a rectangle in section 3 below). Your abstraction should integrate mathematical and logical concepts. Explain how your abstraction helped manage the complexity of your program.**

*(Approximately 200 words)*

```
def editCon(con):
    condition=open("condition.txt","w")
    condition.write(str(int(con)+1))
    condition.close()

def inputUser():
    userinfo=open("user.txt","w")
    userinfo.write("userName='"+str(input('username: '))+"'"+"\n"+"passWord='"+str(input('password: '))+"'"+"\n"+"AP="+str(input('number of ap: '))+"\n"+"HONOR="+str(input('number of honor: ')))
    userinfo.close()

def install(module):
    subprocess.call(str('pip.exe install '+module),shell=True)

def checkCon():
    if(con=="0"):
```

```
    editCon(con)
elif(con=="1"):
    install('pandas')
    install('selenium')
    install('bs4')
    install('lxml')
    editCon(con)
    inputUser()
else:
    if(input('fix the user info [t/f] ')=='t'):
        inputUser()
```

this code implements a large amount of abstraction because it condenses code pieces that are essential to the program into single functions and then calls those in checkCon function. All the editing and installing for fundamental sources for the code are set by functions called editCon, inputUser, install. The editCon function contains 3 lines of code that changes the condition for next run, depending on its current condition, that will be checked in the next run for deciding which codes to run. The inputUser function contains 3 lines of code that edits user information through 4 types of input that will be used for login to the powerschool. The install function contains one line of code that will install different types of module depending on its input. The checkOn function is also a good abstraction because it determines which other functions need to be run by checking the value of con, which is edited by editCon. (156 words)

## 3. Program Code

Capture and paste your entire program code in this section.

- Mark with an **oval** the segment of program code that implements the algorithm you created for your program that integrates other algorithms and integrates mathematical and /or logical concepts.
- Mark with a **rectangle** the segment of program code that represents an abstraction you developed.
- Include comments or citations for program code that has been written by someone else.

```
#--------------------------------------------------------------#
# Citation for Beautiful Soup Module
#Copyright (c) 2004-2017 Leonard Richardson
#
#Permission is hereby granted, free of charge, to any person obtaining a
copy
#of this software and associated documentation files (the "Software"), to
deal
#in the Software without restriction, including without limitation the
rights
#to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
#copies of the Software, and to permit persons to whom the Software is
#furnished to do so, subject to the following conditions:
#
#The above copyright notice and this permission notice shall be included in
all
#copies or substantial portions of the Software.
#
#THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
#IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
#FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE
#AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
#LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
#OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE
#SOFTWARE.
#--------------------------------------------------------------#
# Citation for Selenium Module
#Copyright [2008-2012] [Software Freedom Conservancy]
#
#Licensed under the Apache License, Version 2.0 (the "License");
#you may not use this file except in compliance with the License.
#You may obtain a copy of the License at
#
#    http://www.apache.org/licenses/LICENSE-2.0
#
#Unless required by applicable law or agreed to in writing, software
#distributed under the License is distributed on an "AS IS" BASIS,
#WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
#See the License for the specific language governing permissions and
#limitations under the License.
#--------------------------------------------------------------#
# Citation for Pandas Module
#Copyright (c) 2011-2012, Lambda Foundry, Inc. and PyData Development Team
#All rights reserved.
#
#Copyright (c) 2008-2011 AQR Capital Management, LLC
#All rights reserved.
#
#Redistribution and use in source and binary forms, with or without
#modification, are permitted provided that the following conditions are
#met:
#
#    * Redistributions of source code must retain the above copyright
```

```
#       notice, this list of conditions and the following disclaimer.
#
#    * Redistributions in binary form must reproduce the above
#       copyright notice, this list of conditions and the following
#       disclaimer in the documentation and/or other materials provided
#       with the distribution.
#
#    * Neither the name of the copyright holder nor the names of any
#       contributors may be used to endorse or promote products derived
#       from this software without specific prior written permission.
#
#THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDER AND CONTRIBUTORS
#"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
#LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
#A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
#OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
#SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
#LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
#DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
#THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
#(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
#OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
#----------------------------------------------------------------#
# Citation for lxml library
#Copyright (c) 2004 Infrae. All rights reserved.
#
#Redistribution and use in source and binary forms, with or without
#modification, are permitted provided that the following conditions are
#met:
#
#  1. Redistributions of source code must retain the above copyright
#     notice, this list of conditions and the following disclaimer.
#
#  2. Redistributions in binary form must reproduce the above copyright
#     notice, this list of conditions and the following disclaimer in
#     the documentation and/or other materials provided with the
#     distribution.
#
#  3. Neither the name of Infrae nor the names of its contributors may
#     be used to endorse or promote products derived from this software
#     without specific prior written permission.
#
#THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
#"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
#LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
#A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INFRAE OR
#CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
#EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
#PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
#PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
#LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
#NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
#SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
#----------------------------------------------------------------#
# Citation for subprocess (public domain)
#Copyright (c) 2003-2005 by Peter Astrand
```

```
#-------------------------------------------------------------#
import subprocess
import getpass
condition=open("condition.txt","r")
con=condition.read()
condition.close()


if(con=="0"):
    subprocess.call(str('setx path
"%path%;C:\\Users\\'+getpass.getuser()+'\\AppData\\Local\\Programs\\Python\
\Python36\\Scripts\\;C:\GPAcal\"'),shell=True) #set path to the new file
generated to run the chrome driver

def editCon(con):
    condition=open("condition.txt","w")
    condition.write(str(int(con)+1))
    condition.close()

def inputUser():
    userinfo=open("user.txt","w")
    userinfo.write("userName='"+str(input('username:
'))+"'"+"\n"+"passWord='"+str(input('password:
'))+"'"+"\n"+"AP="+str(input('number of ap:
'))+"\n"+"HONOR="+str(input('number of honor: ')))
    userinfo.close()

def install(module):
    subprocess.call(str('pip.exe install '+module),shell=True)

def checkCon():
    if(con=="0"):
        editCon(con)
    elif(con=="1"):
        install('pandas')
        install('selenium')
        install('bs4')
        install('lxml')
        editCon(con)
        inputUser()
    else:
        if(input('fix the user info [t/f] ')=='t'):
            inputUser()

checkCon()

userinfo=open("user.txt","r")
for line in userinfo:
    exec(line)
userinfo.close()

from selenium import webdriver
import pandas as pd
from bs4 import BeautifulSoup

browser= webdriver.Chrome()
```

```
browser.get('https://powerschool.isqchina.com/public/')
idElem=browser.find_element_by_id('fieldAccount')
idElem.send_keys(userName)
passElem=browser.find_element_by_name('pw')
passElem.send_keys(passWord)
passElem.submit()
browser.get('https://powerschool.isqchina.com/guardian/grades.html')
html_string=browser.page_source
browser.quit()


soup = BeautifulSoup(str(html_string), 'lxml')
table=soup.find_all('table')[0]


new_table = pd.DataFrame(columns=range(0,20), index = [0,1]) # I know the
size

y={} # array of all info
z=0  # number of all info
s1={}# array of all value for semseter 1
s2={}# array of all value for semseter 2
GPA={'s1':{},'s2':{}}      #array of all GPA value for semester 1 and 2
ss={'s1':'','s2':''}       #addition formula of all GPA for semesterl and 2
numberofsub={'s1':0,'s2':0} #number of subjects in semester 1 and 2
#---below part is made by Scott Rome(public domain asked through email)---#
row_marker=0
for row in table.find_all('tr'):
    column_marker = 0
    columns = row.find_all('td')
    for column in columns:
        new_table.iat[row_marker,column_marker] = column.get_text()
    #-------------------------below is made by me----------------#
        y[z]=column.get_text()
        z=z+1
    #-------------------until this part is made by me------------#
        column_marker += 1
#---------------until this part is made by Scott Rome---------------------#

def convertGPA(x,array):
    g=GPA[array]
    if(eval(array)[x]!='--'):
        if(int(eval(array)[x])>91):
            g[x]=4
        elif(int(eval(array)[x])>89):
            g[x]=3.667
        elif(int(eval(array)[x])>87):
            g[x]=3.333
        elif(int(eval(array)[x])>81):
            g[x]=3.000
        elif(int(eval(array)[x])>79):
            g[x]=2.667
        elif(int(eval(array)[x])>77):
            g[x]=2.333
        elif(int(eval(array)[x])>71):
            g[x]=2.000
```

```
            elif(int(eval(array)[x])>69):
                g[x]=1.667
            elif(int(eval(array)[x])>67):
                g[x]=1.333
            elif(int(eval(array)[x])>61):
                g[x]=1.000
            elif(int(eval(array)[x])>59):
                g[x]=0.667
            else:
                g[x]=0.000
        else:
            g[x]='--'

    def printGPA(x,array):
        if(x==int(z/20)-1):
            if(ss[array]!=''):
                print(str(array)+' GPA:
    '+str((eval(ss[array])+0.666*AP+0.333*HONOR)/numberofsub[array]))

    def calGPA(x,array):
        convertGPA(x,array)
        if(GPA[array][x]!='--'):
            numberofsub[array]=numberofsub[array]+1
            if(numberofsub[array]!=1):
                ss[array]=str(GPA[array][x])+'+'+str(ss[array])
            else:
                ss[array]=str(GPA[array][x])+str(ss[array])
        printGPA(x,array)


    for x in range(0,int(z/20)):
        s1[x]=y[14+20*x]
        s2[x]=y[17+20*x]
        calGPA(x,'s1')
        calGPA(x,'s2')

    input('press enter to quit')
```