ACIT 1515 Final Exam

This exam has three questions.

Part marks are awarded.

Marks will be deducted for:

* incorrect output and/or return values: function does not meet specification
* incorrect or unnecessarily complex function design strategies (your algorithm)
* inappropriate use of data structures and variables
* inappropriate length and/or use of functions
* poor commenting, variable names, and program layout
* code that is clearly cut/paste from internet, where the code does not reflect techniques learned in class or techniques that can’t be easily explained by the student

|  |  |
| --- | --- |
| Question 1: Code Reuse | / 16 |
| Question 2: Code Documentation and Annotation | / 16 |
| Question 3: Program Creation  Main – Arguments / Exceptions (10)  Student Averages (10)  Course Averages (10)  Report Generation (6) | / 36 |

Save and upload all required individual files for each of the questions using the names specified.

Save your programs as you finish them, as you will be required to upload your final copies to D2L at the end of the exam. It is your responsibility to keep track of your files, and to upload the correct versions. Only the last version of each copy will be checked.

# Rules:

1. Turn off your phone. Yes, off
2. Place your phone on your desk in front of you, where the instructor can see it.
3. No phones, no talking, no getting help from anyone else – which means no email, text, chat etc..
4. You CAN use your textbook, previous assignments, and code.
5. You CAN use any websites as long as you do not post a question anywhere or otherwise communicate anyone.
6. The test is timed: 2 hours max – at this point the D2L dropbox is closed

# Question 01: Code Reuse - deal.py

Write a program **deal.py** that imports the supplied module**: cards.py**, and performs the following tasks:

1. Creates a ‘deck’ of cards (i.e. a list of strings containing the card names)
2. Shuffles the deck of cards created above.
3. Deals hands of thirteen cards to four players (this will be stored as a list containing four lists of thirteen cards)
4. Prints out the cards that have been dealt to each player.
5. Prints out all of the cards in each player’s hand that are ‘Hearts’ (using a combination of statements of your choice).

Download **cards.py** from the Final Exam Assignment in D2L where you got this document. Make sure to include it in your solution submission.

When grading **cards.py** and **deal.py** are assumed to be in the same directory.

The tasks can be completed in sequence for part marks.

## Example Usage:

PS> python deal.py  
  
Player 0:  
  
 Hand:  
 9 of Diamonds  
 7 of Diamonds  
 8 of Spades  
 7 of Clubs  
 6 of Diamonds  
 10 of Diamonds  
 7 of Spades  
 2 of Clubs  
 Ace of Hearts  
 6 of Hearts  
 6 of Spades  
 5 of Hearts  
 5 of Clubs  
  
 Hearts:  
 Ace of Hearts  
 6 of Hearts  
 5 of Hearts  
  
Player 1:  
  
 Hand:  
 Ace of Diamonds  
 5 of Diamonds  
 5 of Spades  
 King of Hearts  
 9 of Hearts  
 7 of Hearts  
 2 of Diamonds  
 10 of Hearts  
 4 of Clubs  
 10 of Spades  
 3 of Spades  
 4 of Diamonds  
 6 of Clubs  
  
 Hearts:  
 King of Hearts  
 9 of Hearts  
 7 of Hearts  
 10 of Hearts  
  
Player 2:  
  
 Hand:  
 Queen of Spades  
 9 of Spades  
 4 of Hearts  
 8 of Hearts  
 Ace of Spades  
 Ace of Clubs  
 3 of Hearts  
 King of Spades  
 2 of Hearts  
 Queen of Clubs  
 Jack of Diamonds  
 Jack of Hearts  
 3 of Diamonds  
  
 Hearts:  
 4 of Hearts  
 8 of Hearts  
 3 of Hearts  
 2 of Hearts  
 Jack of Hearts  
  
Player 3:  
  
 Hand:  
 9 of Clubs  
 Queen of Hearts  
 Jack of Clubs  
 3 of Clubs  
 Queen of Diamonds  
 King of Diamonds  
 4 of Spades  
 2 of Spades  
 King of Clubs  
 8 of Diamonds  
 Jack of Spades  
 8 of Clubs  
 10 of Clubs  
  
 Hearts:  
 Queen of Hearts

## Question 02: Code Documentation and Annotation - word\_analysis.py

The supplied module **word\_analysis.py** is the solution to your [List Programming Assignment](https://1515.acit.site/collections/list_programming.html).

Modify **word\_analysis.py** so each of the functions in the file has Docstring (Google Style) and has annotated parameter and return types.

Download **word\_analysis.py** from the Final Exam Assignment in D2L where you got this document.

Sample data in the form of **story\_01.txt** and **story\_02.txt** is also available there if you would like to run **word\_analysis.py**

# Question 3: Programming - Loops, File I/O, Collections, and Exceptions - grades\_average.py

## Description

Write a program **grades\_average.py** that works as the follows.

1. It is invoked from the command line with a single argument. This argument is the name of a grade data file.
2. Using the data in the file, it will calculate the average grade for each student. It will store this data in a dictionary variable.
3. Using the data in the file it will calculate the average grade for each course, and store this data in dictionary variable.
4. After these dictionaries have been created, the program will save the data from each dictionary into two text files student\_avgs.txt and course\_avgs.txt The details of these are given below.

## Example Working Invocation:

python grade\_average.py grades.txt

## 

## grades.txt Data File

The file is specified as follows:

1. It is space delimited
2. The first row is a heading row, that is, course codes.
3. Each subsequent line starts with a student number followed by a grade for each course.

See the grades.txt from the D2L Final Exam Assignment for details; this file is used to evaluate your exam.

## Exception Handling: No File Argument Given

grade\_average.py will handle the event where no file argument is given on the command line.

In such a case the program will prompt the user to enter a valid file name.

It is assumed that the user enters a valid file name at the prompt.

### Example: No File Argument Given

PS> python grades\_avg.py  
Please enter a valid file name to process: grades.txt

## Exception Handling: Given File Unavailable

grade\_average.py will handle the case where the file argument given on the command line doesn’t exist.

In such a case, the program will prompt the user to enter a valid file name.

It is assumed that the user enters a valid file name at the prompt.

### Example: File Unavailable

PS> python grades\_avg.py non-existent-file.txt  
Please enter a valid file name to process: grades.txt

### Dictionary Creation: Student Averages

This will be accomplished by writing a function. The return value of this function is a dictionary with student number as key and the rounded grade average for the student as the value. The dictionary should match the following.

{'A00058160': 67,  
'A00063569': 77,  
'A00130020': 70,  
'A00150430': 69,  
'A00152653': 66,  
'A00160322': 75,  
'A00180087': 79,  
'A00185808': 65,  
'A00210137': 66,  
'A00221292': 67,  
'A00249805': 85,  
'A00267578': 73,  
'A00279480': 81,  
'A00286850': 85,  
'A00293220': 71,  
'A00299792': 66,  
'A00301541': 78,  
'A00308144': 71,  
'A00312395': 79,  
'A00321029': 67,  
'A00324108': 61,  
'A00358670': 69,  
'A00394127': 80,  
'A00406176': 82,  
'A00407552': 69,  
'A00409560': 74,  
'A00410027': 72,  
'A00423058': 87,  
'A00423734': 71,  
'A00428357': 67,  
'A00439211': 80,  
'A00445416': 73,  
'A00451251': 70,  
'A00451738': 63,  
'A00465186': 71,  
'A00470499': 71,  
'A00493950': 65,  
'A00509069': 71,  
'A00526549': 81,  
'A00527669': 79,  
'A00530273': 64,  
'A00546405': 76,  
'A00562796': 67,  
'A00567355': 80,  
'A00590036': 70,  
'A00601842': 88,  
'A00602876': 79,  
'A00609939': 67,  
'A00631188': 87,  
'A00644444': 68,  
'A00669478': 77,  
'A00686406': 69,  
'A00703154': 74,  
'A00704412': 72,  
'A00707106': 72,  
'A00721037': 83,  
'A00758971': 67,  
'A00766064': 71,  
'A00771007': 72,  
'A00791405': 75,  
'A00798984': 60,  
'A00803656': 73,  
'A00816385': 81,  
'A00834192': 69,  
'A00850399': 74,  
'A00880296': 68,  
'A00895591': 65,  
'A00925416': 68,  
'A00931398': 76,  
'A00943367': 69,  
'A00943935': 73,  
'A00945082': 77,  
'A00946319': 82,  
'A00960867': 62,  
'A00989640': 71}

### 

### Dictionary Creation: Course Averages

This will be accomplished by writing a function. The return value of this function is a dictionary with the course code as a key and the rounded average grade for the course as the value. The dictionary should match, the following.

{‘ACIT1420’: 74, ‘ACIT1515’: 71, ‘ACIT1620’: 71, ‘ACIT1630’: 74, ‘COMM1116’: 73, ‘MATH1310’: 72, ‘ORGB1100’: 75}

## Storing the Student Averages: Writing the Dictionary to a file

Using the above student dictionary the program will create a text file:

student\_avgs.txt

This is a space delimited file with one student’s data per line. It contains a student number followed by the student average for every student.

An example of the student\_avgs.txt file is available from the D2L in the Final Exam assignment.

## Storing the Course Averages: Writing the Dictionary to a file

Using the above course dictionary the program will create a text file:

course\_avgs.txt

This is a space delimited file with one course’s data per line. It contains the course code followed by the course average for all of the student scores in that course.

An example of the course\_avgs.txt file is available from the D2L in the Final Exam assignment.

## Development Breakdown

If you are looking for guidance or feel you won’t complete the activity as described above, you can progress through the following steps for part marks. Feel free to skip this if you would rather determine your own approach.

### Read Argument (File Name) From Command Line

Read argument from command line specifying grades file.

### Exception Handling: Lack of Arguments

Deal with invalid number of arguments from command line by issuing a prompt to user.

### Exception Handling: File Argument Exists and is Readable

Deal with invalid file by reissuing a prompt to user to enter a file name By issuing a prompt to user.

### Student Grade Averages: Part 1

Create function that accepts a file name as a parameter opens grades file and loops over all lines printing them

### Student Grade Averages: Part 2

Change above function so it only prints lines with student grades (i.e. it skips the first line or header)

### Student Grade Averages: Part 3

Add to function so that it prints a total of all the grades for each line in grades.txt file.

### Student Grade Averages: Part 3

Add to function so that it prints the average grade for each line in the grades.txt file

### Student Grade Averages: Part 4

Add to function so that it creates and returns a dictionary of grades with Student Number as the Key and the average grade as value

### Generate Average Report

Write function that accepts a dictionary parameter. This function will open a file and write one student number per line followed by a space and then the average grade by iterating over all of the items in the dictionary.

## Course Averages

Complete the student averages before completing this.

### Course Averages: Part 1

Create a function that reads the first line of the file (the header) and creates a list of courses.

### Course Averages: Part 2

Add to the above function so that it prints the total score for an individual course

### Course Averages: Part 3

Add to the above function so that it prints the average score for an individual course.

### Course Averages: Part 4

Add to the above function so that it prints the average score for an for all of the courses

### Course Averages: Part 5

Add to the above function so that it stores the course code and the average score for all of the courses

### Course Averages: Part 6

Modify the above function so that it returns a dictionary of the course code and course average.

### Generate Average Report

Write function that accepts a dictionary parameter. This function will open a file and write one course code per line followed by a space and then the average grade by iterating over all of the items in the dictionary.