Assignment 1 Solution

Joshua Guinness, guinnesj

January 18, 2019

Introductory blurb.

1 Testing of the Original Program

Description of approach to testing. Rationale for test case selection. Summary of results. Any problems uncovered through testing.

Under testing you should list any assumptions you needed to make about the program's inputs or expected behaviour.

2 Results of Testing Partner's Code

Summary of results.

3 Discussion of Test Results

3.1 Problems with Original Code

3.2 Problems with Partner's Code

4 Critique of Design Specification

5 Answers to Questions

(a) answer

(b) answer

(c) ...

F Code for ReadAllocationData.py

```
## @file ReadAllocationData.py
#  @author
#  @brief
#  @date

#How to open a file taken from:
     https://docs.python.org/3/tutorial/inputoutput.html#reading-and-writing-files


def readStdnts(s):

    # Creates a new dictionary
    student_info = []

    # Opens the file for reading
    f = open(s, 'r')

    # Reads in each line, converts to a dictionary, then adds the dictionay to the list
    for line in f:

        # Strips the newline character at the end of the string
        line = line.rstrip()
            #https://stackoverflow.com/questions/275018/how-can-i-remove-a-trailing-newline-in-python

        # Splits each line at each tab into a list
        student_string = line.split('\t')
            #https://www.pythoncentral.io/cutting-and-slicing-strings-in-python/

        # Splits the choices at commas and spaces
        choices = student_string[5].split(', ')

        # Transforms the current line student to a dictionary
        student_dictionary = {'macid': student_string[0], 'fname': student_string[1], 'lname':
            student_string[2],
                              'gender': student_string[3], 'gpa': float(student_string[4]), 'choices':
                                  choices}

        # Adds the current dictionary to a list of student info
        student_info.append(student_dictionary)

    # Closes the file
    f.close()

    #print(student_info)

    # Return statement
    return student_info

def readFreeChoice(s):

    # Creates an empty list
    students_free_choice = []

    # Opens the file
    f = open(s, 'r')

    # Iterates through the file, stripping the new line character, and appending each macid to the list
    for line in f:
        line = line.rstrip()
        students_free_choice.append(line)

    # Closes the file
    f.close()

    #print(students_free_choice)

    # Returns the list
    return students_free_choice

def readDeptCapacity(s):

    # Create an empty dictionary
    department_capacity = {}

    # Opens the file
    f = open(s, 'r')
```

```
    for line in f:
        line = line.rstrip()
        list = line.split(' ')
        department_capacity[list[0]] = int(list[1])
            #https://docs.python.org/3/tutorial/datastructures.html#dictionaries

    # Closes the file
    f.close()

    #print(department_capacity)

    # Returns a dictionary of the departments and their capacity
    return department_capacity


#readStdnts('rawStudentData')
#readFreeChoice('freeChoice')
#readDeptCapacity('rawDepartmentData')
```

G Code for CalcModule.py

```
## @file CalcModule.py
#  @Joshua Guinness
#  @brief
#  @date

from ReadAllocationData import *

def sort(S):

    # Sorts in students by GPA by highest to lowest
    # https://www.geeksforgeeks.org/python-program-for-bubble-sort/
    for i in range (len(S)):
        for j in range (0, len(S)-1-i):
            if (S[j].get('gpa') < S[j+1].get('gpa')):
                #https://www.pythonforbeginners.com/dictionary/how-to-use-dictionaries-in-python
                swap(S, j, j+1)

    return S

def average(L, g):

    total_sum = 0
    counter = 0

    for i in range(len(L)):
        if (L[i].get('gender') == g):
            total_sum += L[i].get('gpa')
            counter += 1

    if (counter == 0):
        return None
    else:
        average_gpa = total_sum / counter
        return average_gpa

def allocate(S, F, C):

    allocation_dictionary = {'civil': [], 'chemical': [], 'electrical': [], 'mechanical': [],
        'software': [], 'materials': [], 'engphys': []}

    # Students are now sorted from highest GPA to lowest
    sorted_student_dictionaries = sort(S)

    # Allocate students with free choice
    for i in F:
        student_choice = ""
        for j in sorted_student_dictionaries:
            if (i == j.get('macid')):
                student_choice = j.get('choices')[0]
                allocation_dictionary[student_choice] = [i]
                C[student_choice] = C[student_choice]-1
                sorted_student_dictionaries.remove(j)

    # Allocate all students with a gpa > 4
    for i in sorted_student_dictionaries:
        if (i.get('gpa') >= 4.0):
            first_choice = i.get('choices')[0]
            second_choice = i.get('choices')[1]
            third_choice = i.get('choices')[2]
            if (C.get(first_choice) >= 1):
                allocation_dictionary.get(first_choice).append(i.get('macid'))
                C[first_choice] = C[first_choice]-1
            elif (C.get(second_choice) >= 1):
                allocation_dictionary.get(second_choice).append(i.get('macid'))
                C[second_choice] = C[second_choice]-1
            elif (C.get(third_choice) >= 1):
                allocation_dictionary.get(third_choice).append(i.get('macid'))
                C[third_choice] = C[third_choice]-1

    return allocation_dictionary

# Function to swap two elements in a list
def swap(list, elem1, elem2):
    temp = list[elem1]
    list[elem1] = list[elem2]
    list[elem2] = temp
```

```
    return list


student_dictionaries = readStdnts('rawStudentData')
students_with_free_choice = readFreeChoice('freeChoice')
department_capacity = readDeptCapacity('rawDepartmentData')

sorted_student_dictionaries = sort(student_dictionaries)
average_gpa = average(student_dictionaries, 'male')
allocation_dictionary = allocate(student_dictionaries, students_with_free_choice, department_capacity)
print(allocation_dictionary)
print("\n")
print(department_capacity)
print("\n")
print(sorted_student_dictionaries)
```

H Code for testCalc.py

```
## @file testCalc.py
#  @author
#  @brief
#  @date
```

I Code for Partner's CalcModule.py

```
## @file CalcModule.py
#  @author Partner
```

J Makefile

```
PY = python
PYFLAGS =
DOC = doxygen
DOCFLAGS =
DOCCONFIG = docConfig

SRC = src/testCalc.py

.PHONY: all test doc clean

test:
	$(PY) $(PYFLAGS) $(SRC)

doc:
	$(DOC) $(DOCFLAGS) $(DOCCONFIG)
	cd latex && $(MAKE)

all: test doc

clean:
	rm -rf html
	rm -rf latex
```