Strathmore
UNIVERSITY

STRATHMORE INSTITUTE OF MATHEMATICAL SCIENCES

MASTER OF SCIENCE IN DATA SCIENCE AND ANALYTICS

END OF SEMESTER EXAMINATION

DSA 8103: FUNDAMENTAL OF COMPUTING CONCEPTS

DATE: 7th October 2022 **Time:** 3 Hours

Instructions

1. This examination consists of **FOUR** questions.
2. Answer Question **ONE AND TWO (COMPULSORY)** and any other **ONE** question.

Question 1 (5 Marks): *Attempt this question without using python.*

In this question you are required to construct a function to make the computer's move in a game of Pig. Follow the instructions, and write one single statement for each instruction. Do not provide more than one statement. Show indentation.

a) Write the header for a method named computers_move that takes no parameters.	
b) Write a statement to print "COMPUTER'S TURN", preceded by a blank line	

c) Set an integer variable named sumx to zero.	
d) Start a loop that will run until sumx is 17 or larger.	
e) Set a variable dice_roll to a random integer between 1 and 6, inclusive. (Assume random has been imported.)	
f) Print out a message saying what the computer has just rolled.	
g) Test if the number rolled is a 1 …	
h) And if it is, return a zero.	
i) . . . but if it isn't a 1 . . .	
j) Add the number rolled to sumx.	
k) After the loop is finished, print out a message saying how much the computer has gained this turn.	
l) Return from the function with the amount gained this turn.	

Question 2 (30 marks): (*Attempt this question without using python.)*

a) Suppose abc =

i. What is abc[2:4] ?
ii. What is abc[5:] ?
iii. What is abc[:5] ?
iv. What is abc[-3 : -1] ?
v. What is abc * 3 ?

(5 Marks)

b) Suppose cards = {"ace" : 1, "deuce" : 2, "trey" : 3}
What does the following print? **(3 Marks)**

```
for i in cards:
  print(i)
```

c) Write an anonymous ("lambda") function that takes one integer argument and returns twice that number. **(2 Marks)**

d) Write an anonymous ("lambda") function that takes one integer argument and returns half that number if it is even, but three times that number if it is odd. **(2Marks)**

e) Write a statement that uses the format method to print out a floating point number x with two digits after the decimal point. **(2 Marks)**

f) Suppose:
a = [1, 2, [3, 4]]
b = a[:]
b[2] = [5, 6]

What is the value of b? **(2 Marks)**

g) Write a for loop statement to set a variable squares to be a list of the squares of the numbers 1 through 1000. **(1 Marks)**

h) Suppose small is defined as:

```
def small(a, b=5):
    return a < b
```

What is the result of: **(3 Marks)**

i. list(map(small, [1, 2, 3, 4, 5]))
ii. list(filter(small, [1, 2, 3, 4, 5]))
iii. reduce(small, [1, 2, 3, 4, 5])

i) Write a function highest_rank that takes in a Dictionay of {Name: Integers} and returns the Name with the highest Integer. **(3 Marks)**

j) Using indexing write an expression that takes reverses the order of a string. **(1 Marks)**

k) Using numpy:

i. Write a code that creates a matrix A that has 100 rows and 100 columns and whose entries are randomly selected positive integers from 1 to 100. **(3Marks)**

ii. Write a code that calculates the sum of the numbers on the diagonal of the matrix A. **(2 Marks)**

iii. Write a code that finds the largest element of each of the rows of the matrix A. **(1Marks)**

Question 3 (20 Marks)

This is a case that involves implementation of a managing expenses system. You are required to implement the functions as per the given instruction. In the system, users should be able to add and deduct expenses, update expenses, sort expenses, and export filtered expenses to a file. The program should initially load a collection of expenses from 2 different .txt files (in the same format) and store them in a dictionary.

Required Functions

Below are explanations of the functions that need to be written in the program (expenses.py). Do not change the names of the functions.

Add comments to your code.

def import_expenses(expenses, file):

- Reads data from the given file and stores the expenses in the given expenses dictionary,
- where the expense type is the key and the total expense amount for that expense is the value.
- The same expense type may appear multiple times in the given file.
- Ignores expenses with missing or invalid amounts.
- This function doesn't return anything. Rather, it updates the given expenses dictionary
- based on the expenses in the given file.
- Note: This function will be called twice in main with the same dictionary but different files.

def get_expense(expenses, expense_type):

- Returns the value for the given expense type in the given expenses dictionary.
- Prints a friendly message and returns None if the expense type doesn't exist. (Note: None is a specific keyword in Python of NoneType. You should not return a string "None" from this function.)
- Note: Printing a friendly message means that the program should not raise an error or otherwise terminate. Simply tell the user that the requested expense type does not exist and continue the program.

def add_expense(expenses, expense_type, value):

- Adds the given expense type and value to the given expenses dictionary.
- If the expense type already exists, add the value to the total amount.
- Otherwise, creates a new expense type with the value.
- Prints the expense.
- This function doesn't return anything.

def deduct_expense(expenses, expense_type, value):

- Deducts the given value from the given expense type in the given expenses dictionary.
- Prints a friendly message if the expense type doesn't exist. Note: Printing a friendly message means that the program should not raise an error or otherwise terminate. Simply tell the user that the requested expense type does not exist and continue the program.
- Raises a RuntimeError if the value is greater than the existing total of the expense type.
- Prints the expense.
- This function doesn't return anything.

def update_expense(expenses, expense_type, value):

- Updates the given expense type with the given value in the given expenses dictionary.
- Prints a friendly message if the expense type doesn't exist.
- Note: Printing a friendly message means that the program should not raise an error or otherwise terminate. Simply tell the user that the requested expense type does not exist and continue the program.
- Prints the expense.
- This function doesn't return anything.

def sort_expenses(expenses, sorting):

- Converts the key:value pairs in the given expenses dictionary to a list of tuples and sorts based on the given sorting argument.
- Returns the list of sorted items.
- If the sorting argument is the string 'expense_type', sorts the list of tuples based on the expense type (e.g. 'rent') in ascending alphabetical order, e.g. sorted results: ("coffee", 5), ("food", 5000), ("rent", 1000) Otherwise, if the sorting argument is 'amount', sorts the list of tuples based on the total expense amount (e.g. 825) in descending order, e.g. sorted results: ("food", 5000), ("rent", 1000), ("coffee", 5)

def export_expenses(expenses, expense_types, file):

- Exports the given expense types from the given expenses dictionary to the given file.
- Iterates over the given expenses dictionary, filters based on the given expense types (a list of strings), and exports to a file. Skips any expense type in the given list of expense types that doesn't exist.
- If the expenses argument is the dictionary {"food": 5000, "rent": 1000, "coffee": 5, "clothes": 58.92} and the expense_types argument is the list of strings 'coffee, clothes, rent', exports a file containing:
    - coffee: 5
    - clothes: 58.92
    - rent: 1000

- If the expenses argument is the dictionary {"food": 5000, "rent": 1000, "coffee": 5, "clothes": 58.92} and the expense_types argument is the list of strings 'coffee, clothes, sports', exports a file containing:
    - coffee: 5
    - clothes: 58.92
- Note, the specified expense type 'sports' does not exist in the expenses dictionary, so it is ignored.
- If an item is duplicated in the given expense types, don't worry about it, just export the data as is.
- This function doesn't return anything.

```
def main():
    #import expense file and store in dictionary expenses = import_expenses('expenses.txt')
    #for testing purposes
    #print(expenses)
    while True:
        #print welcome and options
        print('\nWelcome to the expense management system! What would you like to do?')
        print('1: Get expense info')
        print('2: Add an expense')
        print('3: Deduct an expense')
        print('4: Sort expenses')
        print('5: Export expenses')
        print('0: Exit the system')

        #get user input
        option_input = input('\n')

        #try and cast to int
        try:
            option = int(option_input)
        #catch ValueError
        except ValueError:
            print("Invalid option.")

    else:
        #check options
        if (option == 1):
            #get expense type and print expense info
            expense_type = input('Expense type? ')
```

```
            print(get_expense(expenses, expense_type))
        elif (option == 2):
            #get expense type
            expense_type = input('Expense type? ')
            #get amount to add and cast to float
            amount = float(input('Amount to add? '))
            #add expense
            add_expense(expenses, expense_type, amount)
        elif (option == 3):
            #get expense type
            expense_type = input('Expense type? ')
            #get amount to deduct and cast to float
            amount = float(input('Amount to deduct? '))
            #deduct expense
            deduct_expense(expenses, expense_type, amount)
        elif (option == 4):
            #get sort type
            sort_type = input('What type of sort? (\'expense_type\' or \'amount\')')
            #sort expenses
            print(sort_expenses(expenses, sort_type))
        elif (option == 5):
            # get filename to export to
            file_name = input('Name of file to export to?')
            # get expense types to export
            expense_types = []
            while True:
                expense_type = input("What expense type you want to export? Input
                N to quit:")
                if expense_type == "N":
                    break
                expense_types.append(expense_type)
                # export expenses
                export_expenses(expenses, expense_types, file_name)
        elif (option == 0):
            #exit expense system
            print('Good bye!')
                break

if __name__ == '__main__':
    main()
```

Expected Output

For example, in the scenario below, entering "1" will allow the user to get the information for a particular expense. Entering "coffee" will show 12.4, the total for that expense.

```
Welcome to the expense management system!  What would you like to do?
1: Get expense info
2: Add an expense
3: Deduct an expense
4: Update an expense
5: Sort expenses
6: Export expenses
0: Exit the system

1
Expense type? coffee
12.4
```

For another example, in the scenario below, entering "2" will allow the user to add an amount to an existing expense. Entering "coffee" and 1.32 will add to that expense, and show 13.72, the new total for that expense.

```
Welcome to the expense management system!  What would you like to do?
1: Get expense info
2: Add an expense
3: Deduct an expense
4: Update an expense
5: Sort expenses
6: Export expenses
0: Exit the system

2
Expense type? coffee
Amount to add? 1.32
Expense: 13.72
```

Required:

Implement the individual functions correctly in expenses.py **(20 Marks)**

Note that marks will be awarded for the Coding Style;

i. Appropriate naming of variables
ii. Naming of helper functions (with docstrings)
iii. Clear comments in your code

Submit

1. expenses.py: your program
2. expenses.txt and expenses_2.txt: the .txt files to be read by your program
   a. It is important that you DO NOT edit these files.
   b. DO NOT change the spacing or remove any blank lines.
   c. DO NOT copy/paste the text from these files into other files

Question 4 (20 Marks):

Use the following steps/questions to investigate the mammals sleep data set msleep.csv.

a. Write a code that creates a new data frame by removing the rows of the original frame where the value of vore is missing and use the new data frame without the missing values of vore in the subsequent analysis. Take note that some values of the variable vore are missing. **(2 Marks)**
b. Write a function that calculates the time on average mammals in each of the 4 different classes of vore spend sleeping. As a short answer, include a comment in the code that shows/indicate the average time different classes of mammals sleep. **(4 Marks)**
c. Write a code that creates a scatter plot using matlplotlib between the body weight of a mammal (bodywt) and the amount of time spent awake (awake). Remember to apply the principles of visualization. **(4 Marks)**
d. Create the same scatter plot as in Part c. using seaborn and modify it so that the points in the scatter plot are coloured according to the value of the variable vore . **(4 Marks)**
e. Create the same scatter plot as in Part c. using plotly and modify it so that the scatter plot only contains the data about mammals that are herbivores (herbi) and weigh 10 kilograms or less. **(6 Marks)**

--------THE END ---------
GOOD LUCK