MCAST

ASSESSMENT AND INTERNAL VERIFICATION FRONT SHEET (Individual Criteria)
(Note : This version is to be used for an assignment brief issued to students via Classter)

Course Title	MCAST ADVANCED DIPLOMA IN IT		Lecturer Name & Surname	Marco Farrugia, Chris Camilleri, Martin Rizzo, Laurent Azzopardi, Lucienne Micallef, Ian Attard	
Unit Number & Title	ITSFT-406-1501– Fundamentals of Scripting				
Assignment Number, Title / Type	ASSIGNMENT 2: Design and develop complex console applications using a scripting language				
Date Set	20/12/2022	Deadline Date	13/01/2023		
Student Name		ID Number		Class / Group	

Assessment Criteria	Maximum Mark
KU1.1 Identify the ideal paradigm that best suits the application to solve a given problem	5
KU1.3 Present an application that demonstrates the key features of a scripting language	5
KU3.1 Present a program with functions and modules to minimise maintenance	5
KU3.2 Present an error-free application by using exception handling	5
KU4.1 Record how file manipulation was used in a given scenario	5
KU4.2 Reproduce the script application using doc strings and comments to minimise maintenance	5
SE4.4 Evaluate a created script for effectiveness, particularly whether interactivity can help to improve the script	10
Total Mark	40

Notes to Students:

- This assignment brief has been approved and released by the Internal Verifier through Classter.
- Assessment marks and feedback by the lecturer will be available online via Classter (http://mcast.classter.com) following release by the Internal Verifier
- Students submitting their assignment on VLE will be requested to confirm online the following statements:

Student's declaration prior to handing-in of assignment
- ❖ I certify that the work submitted for this assignment is my own and that I have read and understood the respective Plagiarism Policy

Student's declaration on assessment special arrangements
- ❖ I certify that adequate support was given to me during the assignment through the Institute and/or the Inclusive Education Unit.
- ❖ I declare that I refused the special support offered by the Institute.

MCAST

UNIT IICT4002 – Fundamentals of Scripting

ASSIGNMENT 2: Design and develop complex console applications using a scripting language

MCAST ADVANCED DIPLOMA (MQF LEVEL 4)

Deadline: 13th January 2023

General Guidelines

- Any reference from class notes, books or internet should be included in the appendix.
- Plagiarism is strictly **prohibited** and will be penalised in accordance with the college's disciplinary procedures.
- All tasks should be attempted.
- This is a home assignment. To complete these tasks, you are expected to do some research and work from home.
- You may be called for an interview to clarify your work.

Assignment Brief

Scenario

You have been assigned the duty to implement a contact list application to store contact details of persons. The details include the name, surname, mobile number, date of birth and locality. You are therefore required to follow the tasks below to ensure that the final application conforms well with the requirements.

Task 1 Main Menu

Create a new file called *name_surname_4.1x_Task1.py*, for example *John_Borg_4.1A_Task1.py*.

In the main section of your program write code to implement a menu which displays as shown below:

```
-------------
Main Menu
-------------
1 - Show all contacts
2 - Add new contact
3 - Find contact
4 - Delete contact
5 - Quit
Enter your choice:
```

For this task the user must be able to select an item from the menu and for each item a short message is to be displayed, showing which item has been selected as shown:

```
Item 2 was selected:  Add new contact
```

The menu must keep on repeating until the user chooses the last option (Quit), where the program must exit with an appropriate message as shown below:

```
Exiting...
```

If the user's choice is not in the menu an appropriate message must be displayed as shown below and the menu must be displayed again to allow the user to make another choice:

```
-------------
Main Menu
-------------
1 - Show all contacts
2 - Add new contact
3 - Find contact
4 - Delete contact
5 - Quit
Enter your choice: 8
Please enter a number between 1 and 5
-------------
Main Menu
-------------
```

Save your file using the name indicated above.

Task 2 Load / Save Contacts

Make a copy of the Python file that you have created in Task 1 and name it in the format *name_surname_4.1x_Task2.py*. Make sure that you keep the original Task 1 file and include it in your submission of the assignment for grading purposes.

In this task you are required to add functionality to your script to make it load the contacts from disk as soon as the script is run and save the updated contacts before the script quits (last menu option).

Use of any module (e.g. json, csv, etc) that uses JSON or any other format to store and retrieve data from a file is NOT allowed in this assignment. Marks will only be awarded to the use of traditional file and data handling in Python.

Each contact is to be stored in Comma Separated Values (CSV) format in a file called *contacts.csv*. For each contact, the name, surname, date of birth, mobile number and locality must be saved. The example below shows the typical content of such a file:

```
Peter, Abdilla, 22/02/1986, 79811526, Zabbar
John, Borg, 12/04/1982, 99887654, Paola
```

Each contact is to be loaded into memory and stored as a **dictionary**. All the contacts must be stored in memory using a list, i.e., a list of dictionaries. For this, at the start of your script, you need to create a global empty list called `contacts`, which is to be populated with contacts after loading them from the CSV file.

Create a function called `loadContacts(fileName)`, which when called, opens the file whose name is `fileName` and loads and parses the CSV data into the global `contacts` list of dictionaries. This function needs to be called when your script starts, before displaying the menu. Below find a representation of how a dictionary would be used to store the contacts in memory.

```
[
 {'name': 'Peter', 'surname': 'Abdilla', 'DOB': '22/02/1986', 'mobileNo':
'79811526', 'locality': 'Zabbar'},
 {'name': 'John', 'surname': 'Borg', 'DOB': '12/04/1982', 'mobileNo':
'99887654', 'locality': 'Paola'}
]
```

HINTS: You can use the `str.split()` function to split the contact details form each line of CSV text. You can use the `dict()` and `zip()` functions to quickly create the dictionary from the split data.

Create a function called `saveContacts(fileName)`, which when called, opens the file whose name is `fileName` and saves each contact from the `contacts` list to the file in CSV format as shown further up.

When the user chooses the last option from the menu, before quitting, the script must call the `saveContacts` function to save the contacts and then display the message shown below:

```
Enter your choice: 5
Saving contacts and exiting...
```

Save your file using the name indicated above.

Task 3 Functionality Implementation

Make a copy of the Python file that you have created in Task 2 and name it in the format *name_surname_4.1x_Task3.py*. Make sure that you keep the original Task 2 file and include it in your submission of the assignment for grading purposes.

You are now required to implement the first four menu items by following the steps below:

1 - Show all contacts

Define a new function called `showAllContacts()` which will display all the contacts from the dictionary list in the format shown below:

```
Enter your choice: 1
Item 1 was selected:  Show all contacts

name : Peter
surname : Abdilla
DOB : 22/02/1986
mobileNo : 79811526
locality : Zabbar

name : John
surname : Borg
DOB : 12/04/1982
mobileNo : 99887654
locality : Paola
```

2 - Add new contact

Define a new function called `addNewContact()` which will ask the user to enter the name, surname, date of birth, mobile number and locality for a new user which is then inserted into the dictionary list in memory:

```
Enter your choice: 2
Item 2 was selected:  Add new contact
Enter name: Alex
Enter surname: Brincat
Enter DOB: 12/06/1992
Enter mobileNo: 99887445
Enter locality: Bormla
```

3 - Find contacts

The find contact functionality will allow the user to find the contact details only by first name. For this functionality you need to define two functions:

> `findContactsByName(name)` receives the name of a person as a parameter which is used to find contacts from the dictionary list in memory. The function should then **return** a list of dictionaries holding the details of each of the found contacts after the search.
>
> `displayContactsByName(name)` receives the name of a person as a parameter which is passed as an argument to the `findContactsByName(name)` function. Once this function receives the list of found contacts from the return value of `findContactsByName(name)` function, it must display the list of contacts on the screen. At the end of this funcion must also **return** the list of found contacts to be used later on by the next functionality (see 4 - Delete Contacts). The total number of contacts found must also be displayed. The output must be as shown below:

```
Enter your choice: 3
Item 3 was selected:  Find contact
Enter name of contact to find: Peter

name : Peter
surname : Abdilla
DOB : 22/02/1986
mobileNo : 79811526
locality : Zabbar

name : Peter
surname : Briffa
DOB : 15/04/1990
mobileNo : 79811526
locality : Attard

name : Peter
surname : Decelis
DOB : 22/02/1986
mobileNo : 79811526
locality : Zejtun

3 contacts found.
```

If no matching contacts are found, an appropriate message should be printed on screen as shown below:

```
Enter your choice: 3
Item 3 was selected:  Find contact
Enter name of contact to find: Angelo
No contacts were found with name 'Angelo'
```

4 - Delete contacts

The delete contact functionality will allow the user to enter a contact's first name and then allow the user to delete all the contacts whose name matches the name entered by the user. As soon as the user chooses this item from the menu, the script should first ask the user to input the name of a contact. The name is then passed as an argument to a function called `deleteContactsByName(name)` which you need to define.

The function will then pass the name as an argument to the `displayContactsByName(name)` function (see previous functionality) which will display the found contacts and their count. The user is then asked to confirm whether they want to delete the found contacts (see the screenshot below).

If the user confirms, the script must delete the found contacts from the dictionary list in memory. The output must be as shown below:

```
Enter your choice: 4
Item 4 was selected:  Delete contacts
Enter name of contact to delete: Peter

name : Peter
surname : Abdilla
DOB : 22/02/1986
mobileNo : 79811526
locality : Zabbar

name : Peter
surname : Briffa
DOB : 15/04/1990
mobileNo : 79811526
locality : Attard

name : Peter
surname : Decelis
DOB : 22/02/1986
mobileNo : 79811526
locality : Zejtun

3 contact(s) found.

Are you sure you want to remove the above 3 contact(s) (Y/N)? y
Contacts removed.
```

If the user opts not to delete the matching contacts the script must display a message that confirms the user’s choice as shown below:

```
Are you sure you want to remove the above 3 contacts (Y/N)? n
No contacts were removed.
```

If no contacts are found matching the given name, the script must display an appropriate message as shown below. Note that this functionality is already implemented in the previous functionality, so no extra code is required here if the `displayContactsByName(name)` function is called from this function:

```
Enter name of contact to delete: Angelo
No contacts were found with name 'Angelo'
```

Task 4 Further Improvements

Make a copy of the Python file that you have created in Task 3 and name it in the format *name_surname_4.1x_Task4.py*. Make sure that you keep the original Task 3 file and include it in your submission of the assignment for grading purposes.

In this task you are required to add the following improvements to your application script:

Improvement 1

When the script starts, if the contacts.csv file is not found, your program will crash. Update the appropriate function so that when the CSV file is not present the program simply starts with an empty contact list without crashing.

Improvement 2

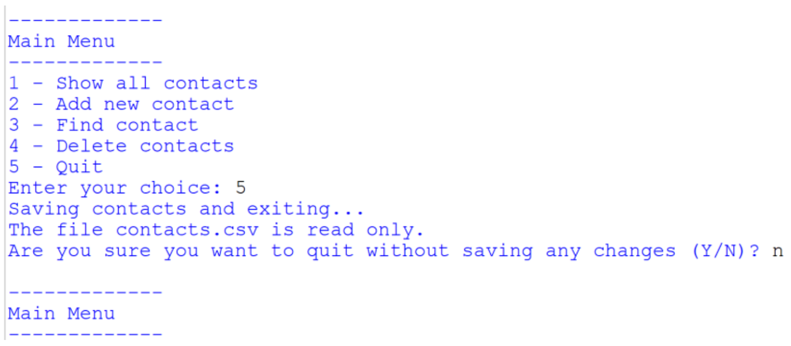
In the main menu, if the user enters anything that is not a number, e.g. “abc”, the script crashes. Upgrade your code such that when such an input is provided by the user the script displays an appropriate massage and displays the menu again as shown below:

```
-------------
Main Menu
-------------
1 - Show all contacts
2 - Add new contact
3 - Find contact
4 - Delete contacts
5 - Quit
Enter your choice: abc
Please enter a number between 1 and 5

-------------
Main Menu
-------------
```

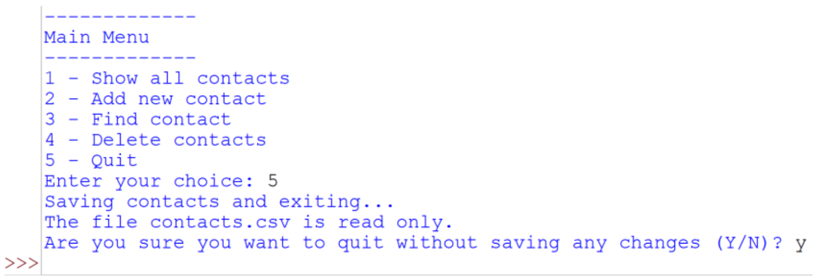
Improvement 3

When opening the CSV file for reading or writing, an IOError may occur, for example if the disk is not available (IOError) or the file is read only (PermissionError). This will cause your application to crash. Upgrade your code such that when such exceptions occur your script informs the user and waits for confirmation whether they want to quit without saving or not as shown below.



```
-------------
Main Menu
-------------
1 - Show all contacts
2 - Add new contact
3 - Find contact
4 - Delete contacts
5 - Quit
Enter your choice: 5
Saving contacts and exiting...
The file contacts.csv is read only.
Are you sure you want to quit without saving any changes (Y/N)? n

-------------
Main Menu
-------------
```



```
-------------
Main Menu
-------------
1 - Show all contacts
2 - Add new contact
3 - Find contact
4 - Delete contacts
5 - Quit
Enter your choice: 5
Saving contacts and exiting...
The file contacts.csv is read only.
Are you sure you want to quit without saving any changes (Y/N)? y
>>>
```

For this update you need to modify the `saveContacts(fileName)` function to make it return a Boolean value. It should return True if the script should quit and false otherwise.

Improvement 4

In this application there are points where the user needs to enter their input:

- when saving contacts on error
- when confirming contact deletion
- when searching by name

All the input from the user is currently case sensitive, i.e., an upper-case letter is different from a lower-case one. For this improvement you need to update the code so that all user input is made case insensitive. It should not make any difference in the program behaviour whether the user enters “Y” or “y” or whether the user enters “Peter” or “peTer”.

Task 5 Documentation

Using docstrings, include 3-4 lines of text to describe each of the functions explained in the previous tasks. Your docstrings should include a description of the purpose of the function, and a description the parameters and any return value for each function.

You also need to add comments to explain each section of your code.

Task 6 Theory

This task must be answered in a MS Word document and saved as `name_surname_class.docx`, e.g., john_borg_4.1L.docx

You are required to do some research and describe in at least 150 words, **two** of the following programming paradigms. **Your description should include excerpts from your code as examples.** If and only if a paradigm you choose is not used in your code, you must include other Python example code sample that explains the chosen paradigm.

- Imperative
- Functional
- Procedural
- Object-oriented programming

KU1.1	Identify the ideal paradigm that best suits the application to solve a given problem

To achieve KU1.1 [5 marks], in Task 6:

a) 2 programming paradigms well described [3 marks]
b) Code excerpts (or examples if code does not follow the chosen paradigm) given [2 marks]

KU1.3	Present an application that demonstrates the key features of a scripting language

To achieve KU1.3 [5 marks]:

a) Task 1 correctly implemented as per instructions [4 marks]
   - 0 marks if Task 1 does not run
   - 1 mark if Task 1 runs but does not function properly
   - 2 marks if Task 1 runs and functions properly with some missing requirements
   - 4 marks if Task 1 implemented according to all the requirements

b) All Python script, Word doc and data files are properly named as per given instructions [1 mark]

KU3.1	Present a program with functions and modules to minimise maintenance.

To achieve KU3.1 [5 marks], in Task 3:

a) Correctly define the 5 functions mentioned in Task 3, including parameters and any return values (0.5 marks for each function)
   - showAllContacts
   - addNewContact
   - findContactsByName
   - displayContactsByName
   - deleteContactsByName

b) Correctly call each function with correct arguments and access the returned values where applicable [0.5 marks for each function]
   - showAllContacts
   - addNewContact
   - findContactsByName
   - displayContactsByName
   - deleteContactsByName

KU3.2	Present an error-free application by using exception handling

To achieve KU3.2 [5 marks], in Task 2:

a) Exception handling is applied to correctly implement improvement 1 [1 marks]
b) Exception handling is applied to correctly implement improvement 2 [1 marks]
c) Exception handling is applied to correctly implement improvement 3 [3 marks]

KU4.1	Record how file manipulation was used in a given scenario

To achieve KU4.1 [5 marks], in Task 2:

a) Correctly open the file once for reading and another time for writing at the right place in the script [1 mark]
b) Correctly read and parse the file content into a dictionary [1.5 mark]
c) Correctly write the file content in CSV format [1.5 mark]
d) Correctly close the file [1 mark]

KU4.2	Reproduce the script application using doc strings and comments to minimise maintenance

To achieve KU4.2 [5 marks], in Tasks 5:

a) At least 4 functions must be well documented using docstrings which should include the following: [1 mark for each function]
   - Description of the purpose of the function
   - Description of the parameters if applicable
   - Description of the return value if applicable
b) Sections of the code well documented using # comments [1 mark]

SE4.4	Evaluate a created script for effectiveness, particularly whether interactivity can help to improve the script.

To achieve SE4.4 [10 marks]:

a) Each of the 4 menu functionalities in task 3 work flawlessly and implemented correctly in functions as per requirements [2 marks each]
b) Task 4, Improvement 4 - All user input is case insensitive (when saving contacts on error, when confirming contact deletion, when searching by name) [2 marks]

---END OF ASSIGNMENT---