G00007918
Applied Databases Project Innovation Section

I wrote 9 extra functions for the python programming section of the project. Apart from some error messages included in the functions that returned data from each database, there wasn't any validation or sanitisation of user input included in the project specification. The input validation functions below improve the user experience of interacting with the program.

The functions below check the input to make sure that the user input is the correct data type, format and length, where appropriate. I also added conditions that ask the user to reenter their information if there were no records returned from the database.

The logic related to input numbers 6 and 7 has been modified to show the results of the first time the countries are read from the database *and* return valid records. If no results are returned then the user can input their information again, and only if results are returned, then they are stored in the program and used any subsequent times the user chooses 6 or 7, as is outlined in the submission specification.

sanitiseMenuInput function:
The sanitiseMenuInput function validates the user's choice of menu input. If the user enters anything other than a number between and 7 an error message is shown, otherwise if the input is valid then the valid user input is returned.(1) I also incorporated the exit program functionality in this function.

```
def sanitiseMenuInput():
  choice = input('Choice: ')
  print('')
  if (choice == 'x'):
    print('Goodbye')
    return exit(0)
  elif (choice.isdigit() == False or int(choice) < 1 or int(choice) > 7):
    print('Please choose a number from 1 to 7. Press x to exit.')
  else:
    return choice
```

sanitiseComparisonInput function:
The sanitiseComparisonInput function checks to make sure that the user inputs either <, >, or =, and if not it calls itself recursively to allow the user to try again.

```
def sanitiseComparisonInput():
  compare = input('Enter < > or = : ')
  if (compare == '<' or compare == '>' or compare == '='):
    return compare
  else:
    print('Invalid input')
    return sanitiseComparisonInput()
```

sanitiseNumericInput function:
The sanitiseNumericInput function makes sure that the user input is a number.(1) If not it prints an error message and calls itself recursively, allowing the user to try again.

```
def sanitiseNumericInput(inputType):
  inputData = input("Enter {} : ".format(inputType))
  if (inputData.isdigit() == False):
    print('Please enter numbers not letters or other characters..')
    return sanitiseNumericInput(inputType)
```

```
  else:
    return inputData
```

sanitiseAlphabeticInput function:

The sanitiseAlphabeticInput function makes sure that user input is alphabetic. If not it prints an error message and calls itself recursively, allowing the user to try again. If the user input is for the country code input, it adds an extra check to make sure that the input is both alphabetic and that it isn't longer than 3 characters.(1)

```
def sanitiseAlphabeticInput(inputType):
  inputData = ''
  inputData = input("Enter {} : ".format(inputType))
  if (inputData.isalpha() == True and inputType == 'country code' and len(inputData) > 3):
    print('Please enter 3 letters max.')
    outputData = sanitiseAlphabeticInput(inputType)
  elif (inputData.isalpha() == False):
    print('Please enter letters not numbers or other characters.')
    outputData = sanitiseAlphabeticInput(inputType)
  else:
    outputData = inputData
    return outputData
```

sanitiseFloatInput function:

The sanitiseFloatInput function checks to see if the user unput can be converted into a float. If not the error message is shown and the function called recursively. If the input can be converted to a float then another condition checks to make sure the input is in the correct range and has only 1 decimal place.

```
def sanitiseFloatInput():
  engineSize = input('Enter engine size: ')
  try:
    engineSizeFloat = float(engineSize)
  except Exception as e:
    print(e, 'Please enter a floating point number with no more than 1 decimal place.')
    return sanitiseFloatInput()
  if (0.5 <= engineSizeFloat < 10 and round(engineSizeFloat,1) == engineSizeFloat):
    return engineSizeFloat
  else:
    print('Please enter a floating point number between 0.5 and 10 with no more than 1 decimal place.')
    return sanitiseFloatInput()
```

sanitiseRegPlateInput function:

The sanitiseRegPlateInput function uses regexes to ensure that user input matches the correct registration plate format.(2) If not it prints an error message and calls itself recursively, allowing the user to try again.

```
def sanitiseRegPlateInput():
  regNumber = input('Enter registration: ')
  plate_format = compile('^[0-9]{2,3}-[a-zA-z]{2}-[0-9]{2,3}$')
  if (plate_format.match(regNumber) is not None):
    return regNumber
  else:
    print('Please enter a registration number in the format nn-ll-nn or nnn-ll-nnn.')
    return sanitiseRegPlateInput()
```

countryInputResults function:

The countryInputResults function for menu item 6 checks to make sure that the user's input returns records from the database, if not it calls itself recursively allowing the user to add input again. This function is only called the first time the user chooses menu item 6.

```
def countryInputResults():
  print('Countries by name')
  print('-----------------')
  countryName = sanitiseAlphabeticInput('country name')
  countries = connectToDb.countryNames(countryName)
  while (not countries):
    print('No results for this input')
    print('')
    return countryInputResults()
  else:
    return countries
```

countryLoop function:

The countryLoop function is called both when the user enters valid input the first time they choose the number 6 menu option, and after all subsequent choices of option 6.

```
def countryLoop(countries):
  for country in countries:
    print(country['Name'],'|', country['Continent'],'|', country['Population'],'|', country['HeadOfState'])
```

populationInputResults function:

The populationInputResults function for menu item 7 checks to make sure that the user's input returns records from the database, if not it calls itself recursively allowing the user to add input again. This function is only called the first time the user chooses menu item 7.

```
def populationInputResults():
  print('Countries by population')
  print('-----------------------')
  compare = sanitiseComparisonInput()
  pop = sanitiseNumericInput('population')
  countryPopulations = connectToDb.countryPopulations(compare,pop)
  if (not countryPopulations):
    print('No results for this input')
    print('')
    return populationInputResults()
  else:
    return countryPopulations
```

countryPopLoop function:

The countryPopLoop function is called both when the user enters valid input the first time they choose the number 7 menu option, and after all subsequent choices of option 7.

```
def countryPopLoop(countryPopulations):
  for populations in countryPopulations:
    print(populations['Name'],'|', populations['Continent'],'|', populations['Population'],'|', populations['HeadOfState'])
```

Returned records checks:

The below code shows a typical query check on a MongoDB collection for to see if records/documents were found that match the user's input. An error message is shown if no records are found.(3)

```
elif (choice == '4'):
  print('Show cars by engine size')
  print('------------------------')
  engineSize = sanitiseFloatInput()
  engines = connectToDb.engineSize(engineSize)
  print('engines ', engines)
  if (engines.count() > 0):
    for engine in engines:
```

```
    print(engine)
  else:
    print('No results for this input')
```

Similarly to the code above, the below code for menu item 2 returns an error message if there are no records returnde form the MySQL query based on the user's input.

```
elif (choice == '2'):
  print('Cities by population')
  print('--------------------')
  compare = sanitiseComparisonInput()
  pop = sanitiseNumericInput('population')
  populations = connectToDb.population(compare,pop)
  if (not populations):
    print('No results for this input')
  else:
  for population in populations:
    print(population['ID'], '|', population['Name'],'|', population['CountryCode'],'|', population['District'],'|', population['Population'])
  display_menu()
```

References:

1 W3Schools Python string methods, accessed 15 May 2019, <https://www.w3schools.com/python/python_ref_string.asp>

2 Stack Overflow, accessed 15 May 2019, <https://stackoverflow.com/questions/47163143/how-to-make-a-format-check-for-registration-plates>

3 MongoDB manual, accessed 15 May 2019, <https://docs.mongodb.com/manual/reference/method/db.collection.count/>