CIND830 - Python Programming for Data Science

Assignment 2 (15% of the final grade)

Due on April 12, 2023 11:59 PM

This is a Jupyter Notebook document that extends a simple formatting syntax for authoring HTML and PDF. Review this website for more details on using Jupyter Notebooks.

Consider using a Jupyter Notebook platform to complete this assignment. Ensure using **Python 3.7 release or higher** then complete the assignment by inserting your Python code wherever seeing the string `#INSERT YOUR ANSWER HERE.`

You are expected to submit the notebook file (in IPYNB format) and the exported version (either in PDF or HTML) in the same Assignment link in D2L. Use these guidelines to submit **both** the IPYNB and the exported file (HTML or PDF). Failing to submit both files will be subject to mark deduction.

Please be advised that you cannot get more than 100% in this assignment, and the **BONUS** question will only be graded if all other questions have been submitted.

Coverage:

1. Lists, Tuples and Dictionaries
2. Functions and Classes
3. Searching and Sorting
4. Arrays and Grids
5. Stacks, Queues, and Lists

Question 1 [20 pts]:

Question 1.a [10 pts]: The `iArray` class in the course materials does not support the sorting functionality. Define a `sort` method that **implements the Selection Sort algorithm** to sort an array object's elements.

After modifying the `iArray` class, run the code below to verify your answer.

```
myArray = iArray(10)
myArray[0] = "C"
myArray[1] = "B"
myArray[2] = "A"
myArray[3] = "Z"
myArray[4] = "K"
myArray[5] = "Y"
myArray[6] = "O"
myArray[7] = "P"

myArray.sort()
print(myArray)
# If you design the iArray class correctly, the output should be as follows:
# ['A', 'B', 'C', 'K', 'O', 'P', 'Y', 'Z', None, None]
```

Assumption: In the `sort` method, you can assume that all array elements precede the `None` value.

Note: Please consider using the `iArray` and `selectionSort` algorithms from your course materials; however, you need to modify the selection sort algorithm so that it can perform on any `iArray` object.

```
#INSERT YOUR ANSWER HERE.
```

Question 1.b [10 pts]:

Define a `search` method that **implements the Binary Search algorithm** and add it to the `iArray` class you modified in Question 1. a.

After modifying the `iArray` class, run the code below to verify your answer.

```
# DO NOT MODIFY THIS PART
myArray = iArray(10)
myArray[0] = "C"
myArray[1] = "B"
myArray[2] = "A"
myArray[3] = "Z"
myArray[4] = "K"
myArray[5] = "Y"
myArray[6] = "O"
myArray[7] = "P"

myArray.sort()
searchCharacter = "C"
foundPosition = myArray.search(searchCharacter)
print(myArray)
print(f"The searched character {searchCharacter} is found at index {foundPosition}")

# If you design the iArray class correctly, the output should be as follows
# ['A', 'B', 'C', 'K', 'O', 'P', 'Y', 'Z', None, None]
# The searched character C is found at index 2
```

Assumption: In the `search` method, you can assume that all array elements precede the `None` value.

Note: The `binarySearch()` in your lecture materials has been declared with two parameters: `target` and `lyst`. However, your `search` method does not need the `lyst` parameter as it will be associated with the array object.

```
#INSERT YOUR ANSWER HERE.
```

Question 2 **[30 pts]**:

The data file used in this question is a sample extracted from the (https://archive.ics.uci.edu/ml/datasets/Online+Retail+II) and converted to a CSV file. The `online_retail_II.csv` is a delimited text file containing all the transactions that occurred for a UK-based online retail store in June 2011, and the CSV data file is given on the course shell along with the assignment notebook. Please consider downloading it. The data file contains the following columns separated by the comma character `,`

- InvoiceNo: Invoice number. Nominal. A unique 6-digit integral number is assigned to each transaction. If this code starts with the letter 'c', it indicates a cancellation.
- StockCode: Product (item) code. Nominal. A unique 5-digit integral number is assigned to each particular product.
- Description: Product (item) name. Nominal.
- Quantity: The quantities of each product (item) per transaction. Numeric.
- InvoiceDate: Invoice date and time. Numeric. The day and time when a transaction was generated.
- UnitPrice: Unit price. Numeric. Product price per unit in sterling (Â£).
- CustomerID: Customer number. Nominal. A 5-digit integral number is uniquely assigned to each customer.
- Country: Country name. Nominal. The name of the country where a customer resides.

Note: Only Python built-in functions or standard modules are allowed to be used in this question. Do not use a third-party library for any part of the code, and importing any third-party library will result in marks deduction.

Question 2.a [10 pts]: Write a Python function that reads the transactional data into a tuple of lists. Each list represents a single row/line of the file, and the elements of the list are the values of the columns. Ensure converting the Quantity and UnitPrice values to integer and float data types, respectively.

Note: The first line of the CSV contains the column headings; therefore, there is no need to store it in the tuple.

```
#INSERT YOUR ANSWER HERE.

from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Question 2.b [10 pts]: Write a function that reads the tuple of transactions you created in the previous question and returns the total number of transaction for each country as a dictionary.

Here is an example of the structure of the output: (the numbers given below may not be correct):

```
{'United Kingdom': 317, 'Netherlands': 32, 'Belgium': 23, 'Germany': 67, ... }
```

```
#INSERT YOUR ANSWER HERE.
```

Question 2.c [10 pts]: Write a function that reads the list of transactions you created in the previous question and returns the total quantity of units sold for each product stock code as a dictionary. Then print out only the three product stock codes with the highest number of units sold.

```
#INSERT YOUR ANSWER HERE.
```

BONUS [10 pts] Write a function that reads the tuple of transactions you created in Q2.a and returns the total revenue for each product stock code as a dictionary. Then print out only the ten product stock codes with the highest revenue.

```
#INSERT YOUR ANSWER HERE.
```

Question 3 [30 pts]:

Assume the following class implements the STACK abstract data type (ADT) using the array ADT.

```
class aStack(iArray):
    def __init__(self, capacity = 5):
        self._items = iArray(capacity)
        self._top = -1
        self._size = 0
    def push(self, newItem):
        self._top += 1
        self._size += 1
        self._items[self._top] = newItem
    def pop(self):
        oldItem = self._items[self._top]
        self._items[self._top] = None
        self._top -= 1
        self._size -= 1
        return oldItem
    def peek(self):
        return self._items[self._top]
    def __len__(self):
        return self._size
    def __str__(self):
        result = ' '
        for i in range(len(self)):
            result += str(self._items[i]) + ' '
        return result
```

Question 3.a [10 pts]: Emulate the stack behavior using the Python list data structure rather than the Arrary ADT.

```
#INSERT YOUR ANSWER HERE.
```

Question 3.b [10 pts]: Redefine the Stack class methods to push and pop two items rather than one item at a time.

For example, if the stack includes numbers from one to ten: `[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]`, then invoking the `pop()` method twice will remove the last four elements and modify the stack elements to be: `[1, 2, 3, 4, 5, 6]`

```
#INSERT YOUR ANSWER HERE.
```

Question 3.c [10 pts]: Define and implement a function that reverses the items of a given stack using only the methods defined in the Stack class.

For example, if the stack includes the elements from one to ten: `[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]`, then calling the new function will modify the stack elements to be from ten to one: `[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]`

```
#INSERT YOUR ANSWER HERE.
```

Question 4 [20 pts]:

Question 4.a [10 pts]: The following two algorithms sort a list of integers in ascending order.

```
def Algorithm_1(lyst):
    i = 0
    while i < len(lyst) - 1:
        minIndex = i
        j = i + 1
        while j < len(lyst):
            if lyst[minIndex] > lyst[j]:
                minIndex = j
            j += 1
        if minIndex != i:
            lyst[minIndex], lyst[i] = lyst[i], lyst[minIndex]
        i += 1
    return(lyst)


def Algorithm_2(lyst):
    n = len(lyst)
    while n > 1:
        i = 1
        while i < n:
            if lyst[i] < lyst[i - 1]:
                lyst[i], lyst[i - 1] = lyst[i-1], lyst[i]
                count += 1
            i += 1
        n -= 1
    return(lyst) # return(lyst,count)
```

Modify each algorithm to display the total number of swap operatioins. For example, if the user called `Algorithm_1` with a list of `[5, 6, 8, 3, 2, 1, 9, 4]` the number of swaps would be `6`, however the output would be `16` when calling `Algorithm_2` with the same list.

```
#INSERT YOUR ANSWER HERE.
```

Question 4.b [10 pts]: Would you conclude that `Algorithm_1` is more effecient that `Algorithm_2`? Justify your answer.

```
#INSERT YOUR ANSWER HERE.
```

Question 5 [20 pts]:

Question 5.a [10 pts]: Write an algorithm that creates a Ragged Grid with four rows. The first row has 3 positions, the second row has 4 positions, the third row includes 6 positions, and the fourth row contains 1. You can modify the iArray ADT defined below or use any standard or third-party library to implement your algorithm.

```
class iArray():
    def __init__(self, capacity, fillValue = None):
        self._items = list()
        for count in range(capacity):
            self._items.append(fillValue)
    def __len__(self):
        return len(self._items)
    def __str__(self):
        return str(self._items)
    def __iter__(self):
        return iter(self._items)
    def __getitem__(self, index):
        return self._items[index]
    def __setitem__(self, index, newItem):
        self._items[index] = newItem
```

```
#INSERT YOUR ANSWER HERE.
```

Question 5.b [10 pts]: Write an algorithm to create and display a two-dimensional array or a regular Grid of three rows and four columns. The Grid's elements are selected randomly from an inclusive interval of `[-100, 100]`. You can install and use any standard or third-party library to report your answer to this question.

```
#INSERT YOUR ANSWER HERE.
```

Question BONUS [10 pts]: Define and implement a Double-ended queue (Dequeue) ADT. The Dequeue is a sequence of items where addtions and deletions of items can be done at both ends of the sequence. The Dequeue can be implemnted using the iArray ADT with the following methods:

```
constructor, append_left, append_right, pop_left, pop_right, peek_left, peek_right, __len__, and __str__.
```

```
class Deque(iArray):
  def __init__(self, capacity = 10):
    #INSERT YOUR ANSWER HERE.

  def append_left(self, newItem):
    #INSERT YOUR ANSWER HERE.

  def append_right(self, newItem):
    #INSERT YOUR ANSWER HERE.

  def pop_left(self):
    #INSERT YOUR ANSWER HERE.

  def pop_right(self):
    #INSERT YOUR ANSWER HERE.

  def peek_left(self):
    #INSERT YOUR ANSWER HERE.

  def peek_right(self):
    #INSERT YOUR ANSWER HERE.

  def __len__(self):
    #INSERT YOUR ANSWER HERE.
  def __str__(self):
    #INSERT YOUR ANSWER HERE.
```

```
#INSERT YOUR ANSWER HERE.
```

This is the end of assignment 2