CSE 331 **Spring 2018**

Project 4

100 points
Due March 1st 8:00pm
This is not a team work, do not copy somebody else's work.

Reference: Authors of this project are Cyndy Ishida and Ikechukwu Uchendu

Assignment Overview

In the case that we are utilizing a queue of fixed size, it is more effective to represent it as a circular array. Using a linked list, one would have to allocate new memory and delete memory for insertions and deletions. With a circular array, we can adjust pointers in the array to insert and delete.

Your job is to implement a queue using a circular array. In addition to that, at any given time we should be able to request the **average** of the values currently in the queue.

Assignment Deliverables

CircularQueue.py

Be sure to use the specified file name(s) and to submit your files for grading **via D2L Dropbox** before the project deadline.

Assignment Specifications

This project would be trivial using methods on Python built-in container type. Therefore, **the use of any python built-in method, not including the index "[]" method, outside of the resize function will result in a zero.**

Your task will be to complete the methods listed below.

- __str__ (Provides a string representation of a CircularQueue)
  - This method will allow you to easily see the contents of your queue
  - Return a string representation of the queue
  - Return (type string)
  - O(n) time complexity, O(n) space complexity

- enqueue(self, number)
    - Add a number to the back of the queue.
    - Return (type None) no return statement will also return None
    - O(1)* time complexity requirement, O(1)* space complexity

- dequeue(self)
    - Remove an element from the front of a queue but do nothing if the queue is empty.
    - Return (type None) no return statement will also return None
    - O(1) time complexity requirement, O(1) space complexity

- get_average(self)
    - Returns the average of the elements contained in the queue
    - Return (type float)
    - O(1) time requirement, O(1) space complexity
- resize
    - Resizes the queue to be twice its previous size
    - Return (type None) no return statement will also return None
    - O(n) time complexity requirement, O(n) space complexity

* is to denote an amortized time and/or space complexity.

We have provided the file "main.py" that will feed commands to your circular queue class. It contains 2 test cases that will also be used when grading your solution. Remember, we will also use the equality operator provided in the Queue Class definition to check for correctness. You may edit main.py as you please, since it isn't being submitted.

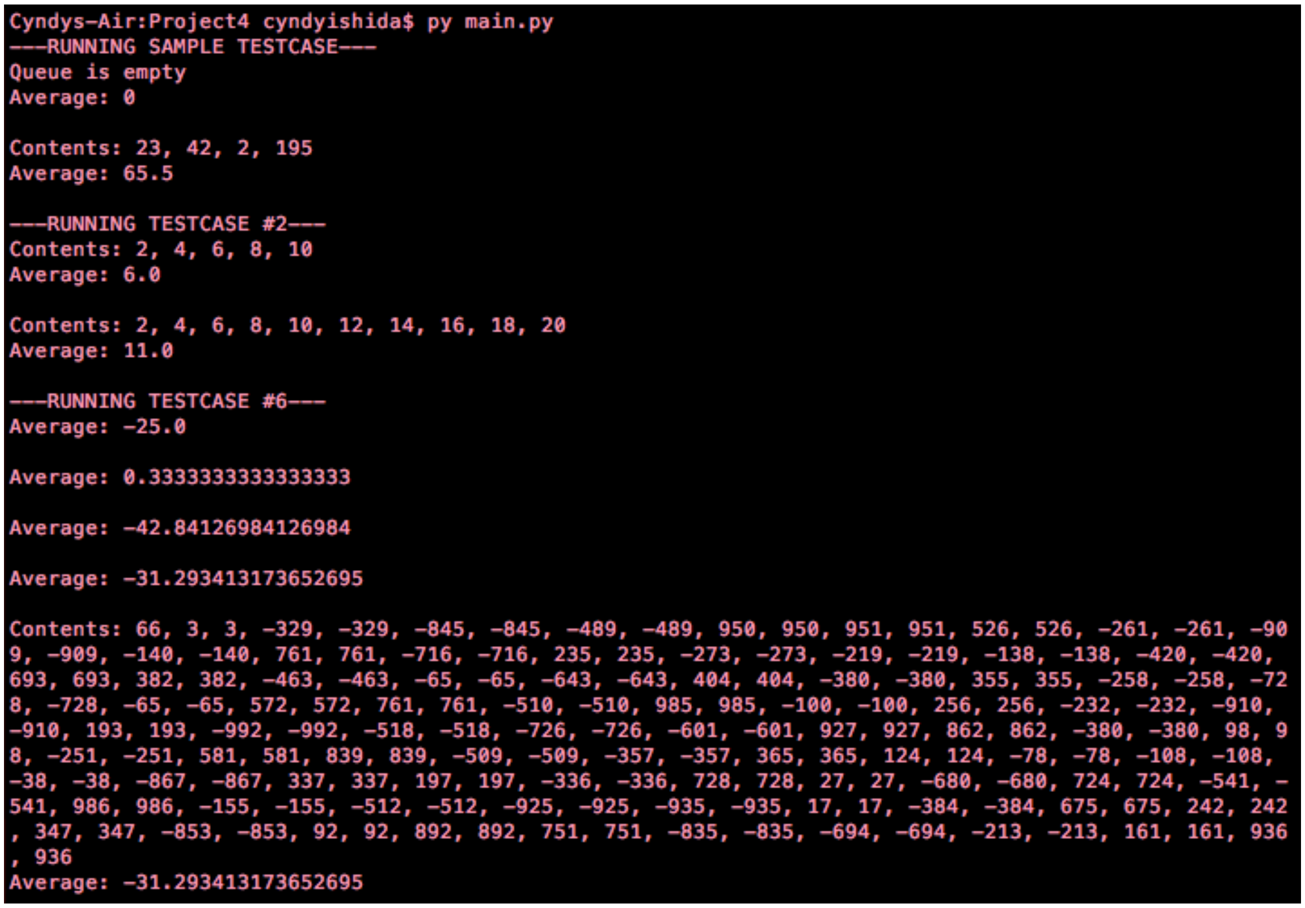
Assignment Notes

Points will be deducted if your solution has any warnings of type:

- It is the assumption you are using the python 3.6 interpreter.
- You must implement the queue using a circular array.
- You are provided with skeleton code for the CircularQueue class. Your job is to complete the methods given. You may add more functions than what is provided, **but may not alter the function signatures in anyway.**
- You **may not use any** python list methods outside of the resize method (you can use the index operator wherever you like!)
- Every method and function should have docstrings associated for submission.
- Please refer to https://d2l.msu.edu/d2l/le/content/482170/viewContent/5369895/View for a detailed tutorial on how to run and debug a python project within PyCharm.

Testing your work

Run your project on Pycharm see sample run below



```
Cyndys-Air:Project4 cyndyishida$ py main.py
---RUNNING SAMPLE TESTCASE---
Queue is empty
Average: 0

Contents: 23, 42, 2, 195
Average: 65.5

---RUNNING TESTCASE #2---
Contents: 2, 4, 6, 8, 10
Average: 6.0

Contents: 2, 4, 6, 8, 10, 12, 14, 16, 18, 20
Average: 11.0

---RUNNING TESTCASE #6---
Average: -25.0

Average: 0.3333333333333333

Average: -42.84126984126984

Average: -31.293413173652695

Contents: 66, 3, 3, -329, -329, -845, -845, -489, -489, 950, 950, 951, 951, 526, 526, -261, -261, -90
9, -909, -140, -140, 761, 761, -716, -716, 235, 235, -273, -273, -219, -219, -138, -138, -420, -420,
693, 693, 382, 382, -463, -463, -65, -65, -643, -643, 404, 404, -380, -380, 355, 355, -258, -258, -72
8, -728, -65, -65, 572, 572, 761, 761, -510, -510, 985, 985, -100, -100, 256, 256, -232, -232, -910,
-910, 193, 193, -992, -992, -518, -518, -726, -726, -601, -601, 927, 927, 862, 862, -380, -380, 98, 9
8, -251, -251, 581, 581, 839, 839, -509, -509, -357, -357, 365, 365, 124, 124, -78, -78, -108, -108,
-38, -38, -867, -867, 337, 337, 197, 197, -336, -336, 728, 728, 27, 27, -680, -680, 724, 724, -541, -
541, 986, 986, -155, -155, -512, -512, -925, -925, -935, -935, 17, 17, -384, -384, 675, 675, 242, 242
, 347, 347, -853, -853, 92, 92, 892, 892, 751, 751, -835, -835, -694, -694, -213, -213, 161, 161, 936
, 936
Average: -31.293413173652695
```