pythonDataStructures.py

```python
from random import randint
aList = [1,2,3,4]
nElements = 10

# List
print("List")
myList = [1,2,3,4]
print(myList)

print("\nAdding 5 to list")
myList.append(5)
print(myList)

print("\nRemoving 5 from list")
myList.remove(5)
print(myList)

# Stack
print("\nStack")
myStack = []
for i in aList:
    myStack.append(i)
print(myStack)

print("\nAdding 5 to stack")
myStack.append(5)
print(myStack)

print("\nPopping all from stack")
for i in range(5):
    print(myStack.pop())

# Queue
print("\nQueue")
from collections import deque
myQueue = deque(aList)
print(myQueue)

print("\nAdding 5 to queue")
myQueue.append(5)
print(myQueue)

print("\nServing all from queue")
for i in range(5):
    print(myQueue.popleft())

# Dictionary
print("\nDictionary")
myDict = { 'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5 }
print(myDict)

print("\nIs 'a' in dictionary?")
print('a' in myDict)

print("\nDeleting 'a' from dictionary")
if 'a' in myDict:
    del myDict['a']
print(myDict)

print("\nSetting value of 'a' to 10")
myDict['a'] = 10
print(myDict)
# myDict.keys(), myDict.values()

# DefaultDict
from collections import defaultdict
```

```
print("\nDefault Dict mapping to a list")
myDict = defaultdict(list)
myDict["a"].append(0)
print(myDict.items())

print("\nDefault Dict mapping to an int")
myDict = defaultdict(int)
myDict["a"] += 1
print(myDict.items())

# Set
print("\nSet")
mySet = set([1,2,3,4])
print(mySet)

print("\nAdding 5 to set")
mySet.add(5)
print(mySet)

print("\nDiscard 5 from set")
mySet.discard(5)
print(mySet)

# Min Heap
from heapq import heappush, heappop, heapify
print("\nMin Heap")
myMinHeap = [1,2,3,4]
heapify(myMinHeap)

for i in [randint(1,100) for _ in range(nElements)]:
    heappush(myMinHeap, i)
print(myMinHeap)

print("\nPopping all from min heap")
for i in range(len(myMinHeap)):
    print(heappop(myMinHeap))

# Max Heap
print("\nMax Heap")
class MaxHeapObject:
    def __init__(self, value):
        self.value = value

    def __lt__(self, other):
        return self.value > other.value

    def __eq__(self, other):
        return self.value == other.value

    def __int__(self):
        return self.value

def maxheappush(maxHeap, item):
    heappush(maxHeap, MaxHeapObject(item))

def maxheappop(maxHeap):
    return heappop(maxHeap).value

myMaxHeap = []
for i in [randint(1,100) for _ in range(nElements)]:
    maxheappush(myMaxHeap, i)
print([int(i) for i in myMaxHeap])

print("\nPopping all from max heap")
for i in range(len(myMaxHeap)):
    print(maxheappop(myMaxHeap))
```