Build a Queue Using a Stack

May 17, 2020

1 Build a Queue From Stacks

In this exercise we are going to create a queue with just stacks.

Code

In [18]:
```
# Here is our Stack Class

class Stack:
    def __init__(self):
        self.items = []

    def size(self):
        return len(self.items)

    def push(self, item):
        self.items.append(item)

    def pop(self):
        if self.size()==0:
            return None
        else:
            return self.items.pop()

class Queue:
    def __init__(self):
        # Code here
        self.in_stack = Stack()
        self.out_stack = Stack()

    def size(self):
         # Code here
        return self.in_stack.size() + self.out_stack.size()

    def enqueue(self,item):
        # Code here
        self.in_stack.push(item)
```

```
        def dequeue(self):
            # Code here
            if not self.out_stack.items:
                while self.in_stack.items:
                    self.out_stack.push(self.in_stack.pop())
            return self.out_stack.pop()
```

Test Cases

```
In [19]: # Setup
         q = Queue()
         q.enqueue(1)
         q.enqueue(2)
         q.enqueue(3)

         # Test size
         print ("Pass" if (q.size() == 3) else "Fail")

         # Test dequeue
         print ("Pass" if (q.dequeue() == 1) else "Fail")

         # Test enqueue
         q.enqueue(4)
         print ("Pass" if (q.dequeue() == 2) else "Fail")
         print ("Pass" if (q.dequeue() == 3) else "Fail")
         print ("Pass" if (q.dequeue() == 4) else "Fail")
         q.enqueue(5)
         print ("Pass" if (q.size() == 1) else "Fail")
```

```
Pass
Pass
Pass
Pass
Pass
Pass
```

Hide Solution

```
In [ ]: # Solution

        # Here is our Stack Class

        class Stack:
            def __init__(self):
                self.items = []
```

```
    def size(self):
        return len(self.items)

    def push(self, item):
        self.items.append(item)

    def pop(self):
        if self.size()==0:
            return None
        else:
            return self.items.pop()

class Queue:
    def __init__(self):
        self.instorage=Stack()
        self.outstorage=Stack()

    def size(self):
         return self.outstorage.size() + self.instorage.size()

    def enqueue(self,item):
        self.instorage.push(item)

    def dequeue(self):
        if not self.outstorage.items:
            while self.instorage.items:
                self.outstorage.push(self.instorage.pop())
        return self.outstorage.pop()


# Setup
q = Queue()
q.enqueue(1)
q.enqueue(2)
q.enqueue(3)

# Test size
print ("Pass" if (q.size() == 3) else "Fail")

# Test dequeue
print ("Pass" if (q.dequeue() == 1) else "Fail")

# Test enqueue
q.enqueue(4)
print ("Pass" if (q.dequeue() == 2) else "Fail")
print ("Pass" if (q.dequeue() == 3) else "Fail")
print ("Pass" if (q.dequeue() == 4) else "Fail")
```

```
q.enqueue(5)
print ("Pass" if (q.size() == 1) else "Fail")
```