In the following, we investigate two data structures—a queue and a stack—constructed using an underlying data structure like a Python-style list.

A queue is a container (like a list) that holds many values. There are two important operations: enqueue and dequeue. enqueue adds a value to the queue. dequeue removes that element of the queue that was added first. It is a first-in, first-out or FIFO structure. If we wait in line, that’s a queue: the person waiting the longest is the next to get attention.

1a. Inside the queue class, we’ll use a Python list to hold the queue values:

```python
class queue(object):
    __slots__ = ['_l']
    def __init__(self):
        """Initialize the queue."""
        self._l = []
```

Write the enqueue method for this class:

```python
def enqueue(self, v):
```

1b. Write the dequeue method for this class (remember: it returns the value removed from the queue).

```python
def dequeue(self):
    # assume the queue holds at least one value
A stack is a container that also offers two operations: push and pop. push adds a value to the stack. pop removes that element of the stack that was added last. It is a last-in, first-out or LIFO structure. Dining hall trays are stored in a stack: the tray on the top is used frequently while the tray at the bottom is rarely used.

2a. Inside the stack class, we'll use a list class to hold the stack values:

```python
class stack(object):
    __slots__ = ['_l']
    def __init__(self):
        """Initialize the stack."""
        self._l = None
```

Write the push method for this class:

```python
def push(self,v):
```

2b. Write the pop method for this class, returning the popped value:

```python
def pop(self):
    # assume the stack holds at least one value
```