QUEUE

Abstract Data Type

TAXI

FEED THE QUINTS!
One Dollar
AND FOR A FEW DOLLARS MORE,
YOU CAN FEED THEM FISH STICKS.

tail
head

QUEUE ADT

A List with a restriction:

QUEUE ADT

`insert` is done at one end, whereas `delete` is performed at the other end.

QUEUE ADT

FIFO

First In, First Out

ONE WAY
LAST IN, LAST OUT
FIRST IN, FIRST OUT

QUEUE ADT Operations

enqueue

equivalent to

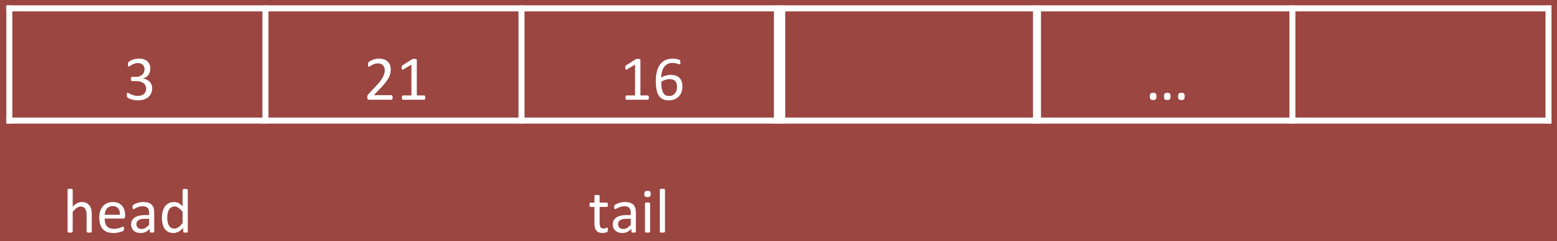
insert

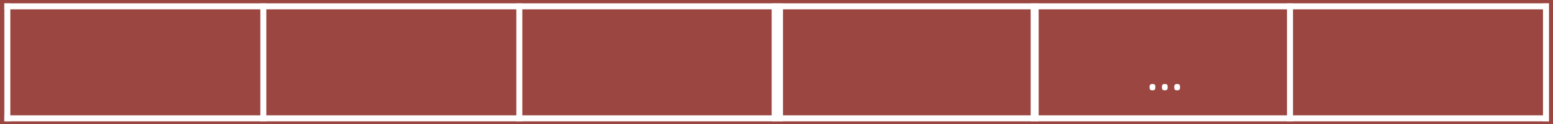
QUEUE ADT Operations

`dequeue`

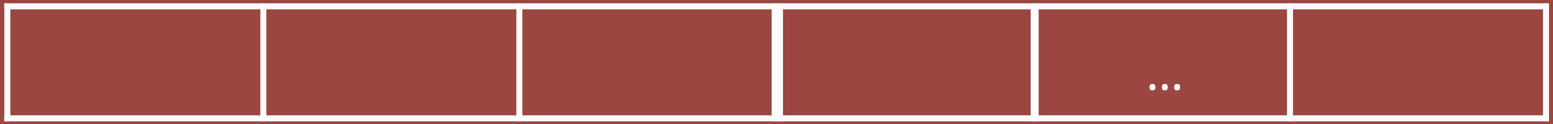
equivalent to

delete


3
21
16
…
head
tail



```
enqueue(45);
enqueue(6);
x = dequeue();
enqueue(123);
x = dequeue();
x = dequeue();
```



head

tail

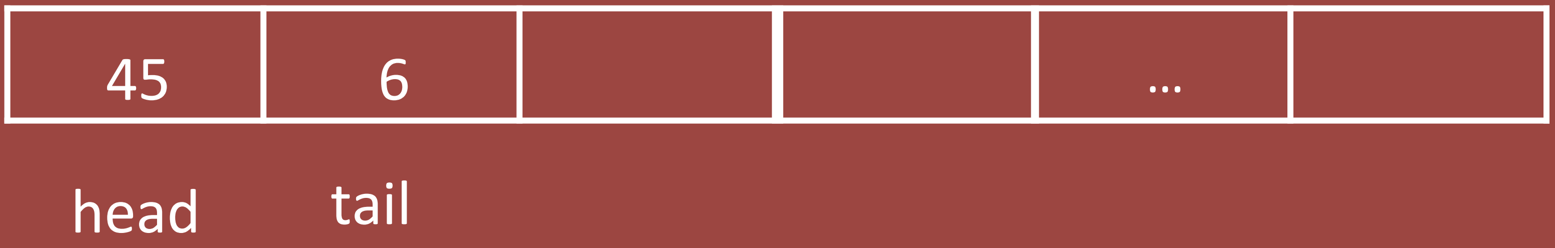
```
enqueue(45);
enqueue(6);
x = dequeue();
enqueue(123);
x = dequeue();
x = dequeue();
```

45				…	

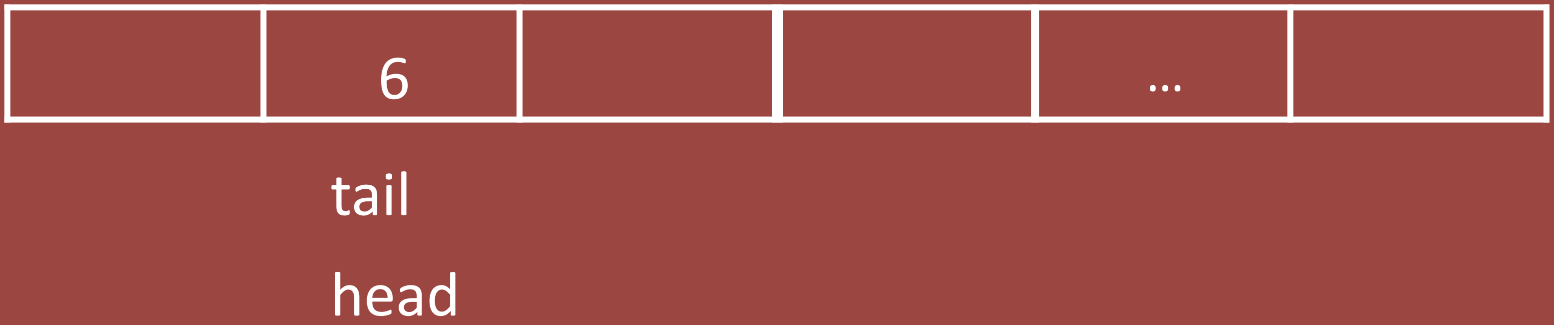
head

tail

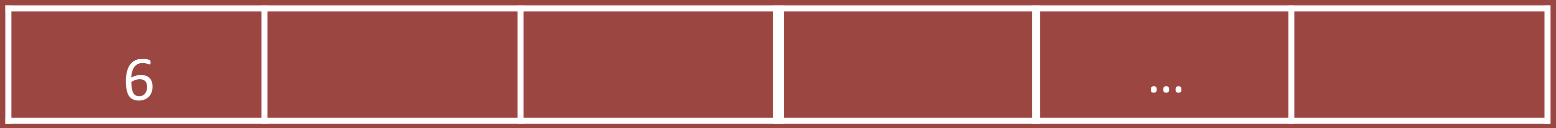
```
enqueue(45);
enqueue(6);
x = dequeue();
enqueue(123);
x = dequeue();
x = dequeue();
```



```
enqueue(45);
enqueue(6);
x = dequeue();
enqueue(123);
x = dequeue();
x = dequeue();
```



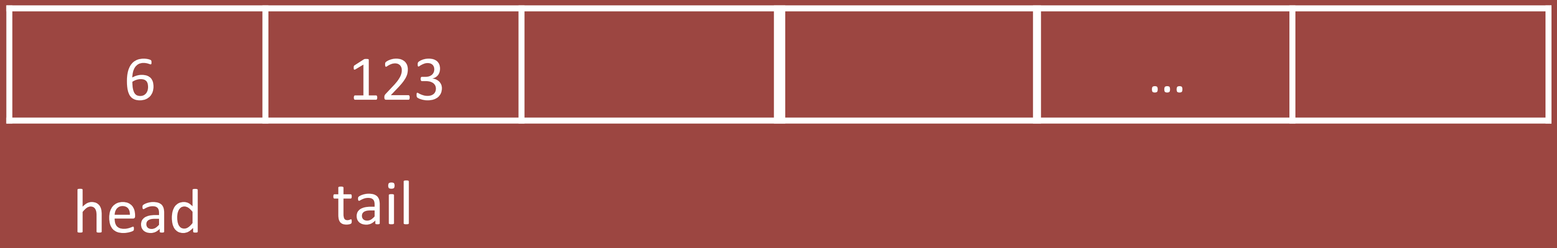
```
enqueue(45);
enqueue(6);
x = dequeue();
enqueue(123);
x = dequeue();
x = dequeue();
```



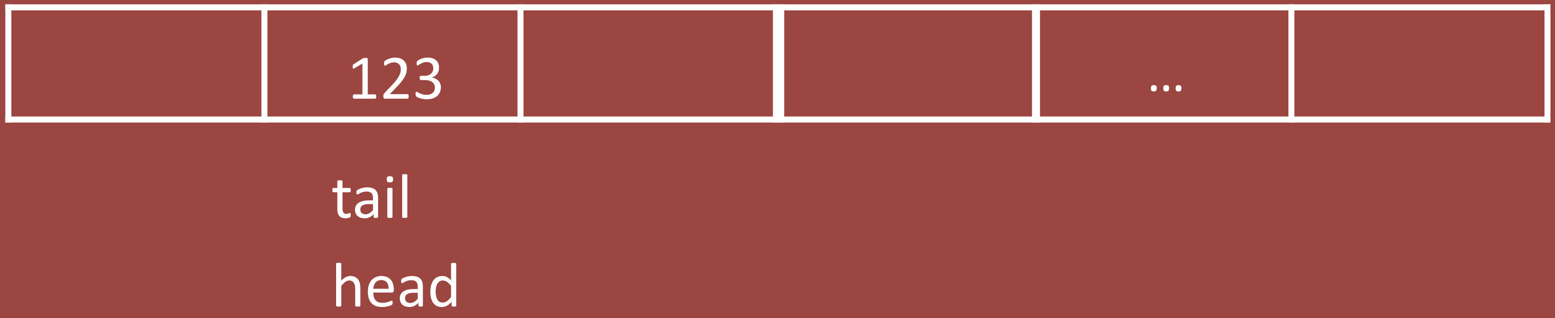
head

tail

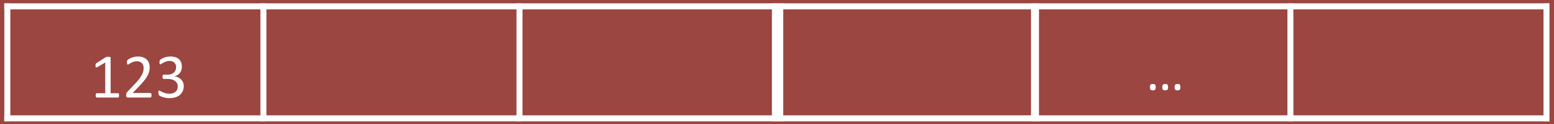
```
enqueue(45);
enqueue(6);
x = dequeue();
enqueue(123);
x = dequeue();
x = dequeue();
```



```
enqueue(45);
enqueue(6);
x = dequeue();
enqueue(123);
x = dequeue();
x = dequeue();
```



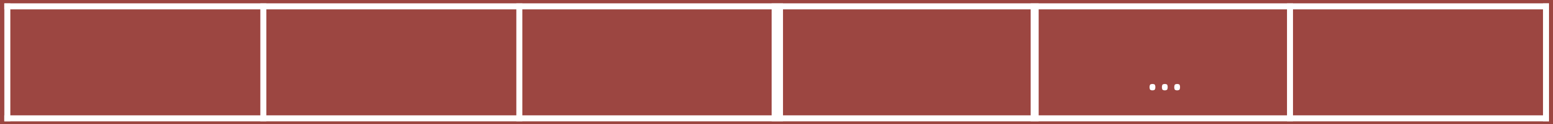
```
enqueue(45);
enqueue(6);
x = dequeue();
enqueue(123);
x = dequeue();
x = dequeue();
```



head

tail

```
enqueue(45);
enqueue(6);
x = dequeue();
enqueue(123);
x = dequeue();
x = dequeue();
```



head

tail

```
enqueue(45);
enqueue(6);
x = dequeue();
enqueue(123);
x = dequeue();
x = dequeue();
```

QUEUE

POSSIBLE ERRORS

Queue Underflow

attempt to **dequeue** a value from an **empty** queue.

Queue Overflow

attempt to **enqueue** a value into a **full** queue.

QUEUE

Array IMPLEMENTATION

```
def __init__(self):
    self.queue = []

#Add an element
def enqueue(self, item):
    #add element at the end
    self.queue.append(item)
```

Array IMPLEMENTATION

```
# Remove an element
def dequeue(self):
    if len(self.queue) < 1:
        return None
    #removes the first element (index 0)
    return self.queue.pop(0)
```

Array IMPLEMENTATION

QUEUE
Singly-Linked List
IMPLEMENTATION

```
class Node:
  # constructor
  def __init__(self, data):
     self.data = data
     self.next = None
```

Singly-Linked List
IMPLEMENTATION

```
# insert at the tail of the linked list
def enqueue(self, data):
    newNode = Node(data)
    if(self.head):
        current = self.head
        while(current.next):
            current = current.next
        current.next = newNode
    else: #for the first element/head
        self.head = newNode
```

Singly-Linked List
IMPLEMENTATION

```
# delete an element from the head of the linked list
def dequeue(self):
    temp = self.head
    if (temp is not None):
        element = temp.data
        self.head = temp.next
        temp = None
        return element
```

Singly-Linked List
IMPLEMENTATION