NPTEL MOOC

PROGRAMMING, DATA STRUCTURES AND ALGORITHMS IN PYTHON

Week 7, Lecture 1

Madhavan Mukund, Chennai Mathematical Institute
http://www.cmi.ac.in/~madhavan

Data structures

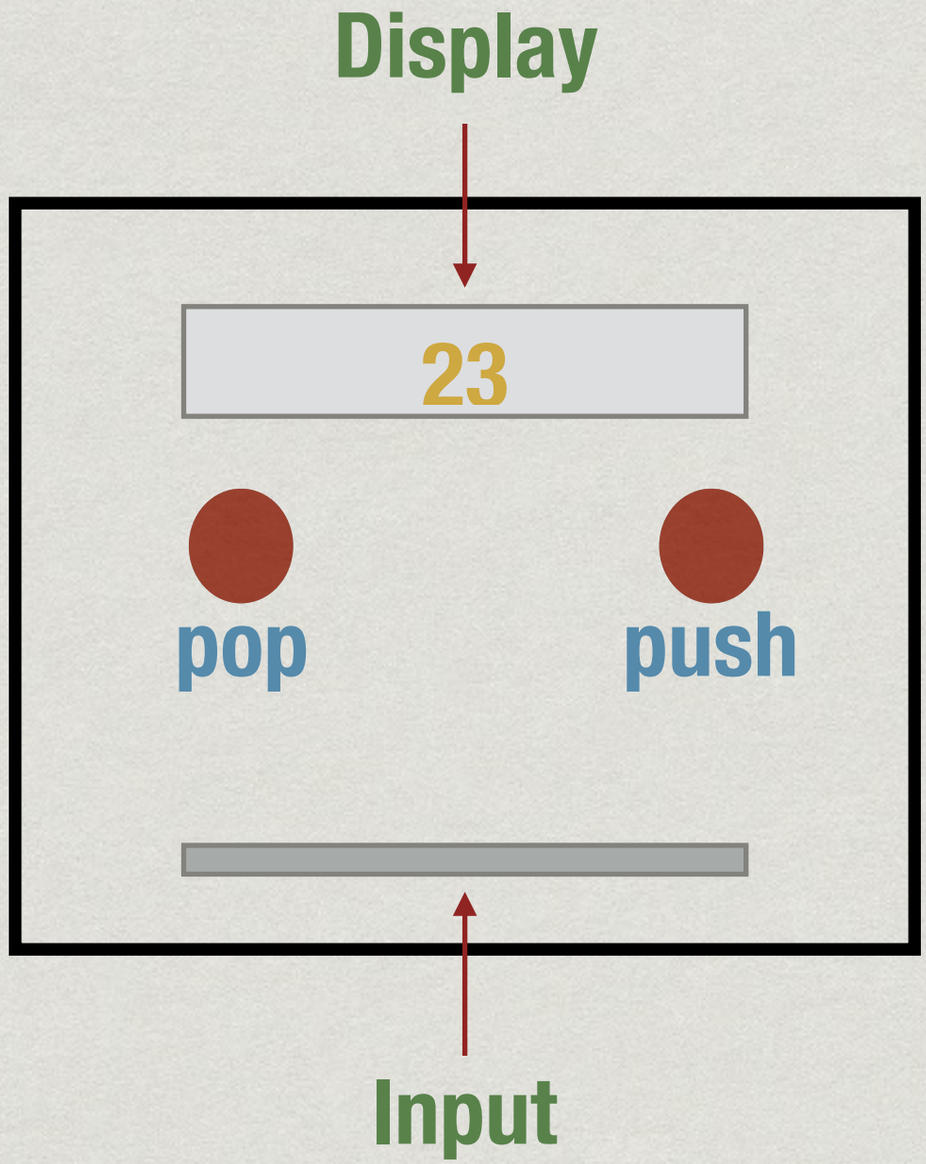
- Behaviour defined through interface
  - Allowed set of operations
- Stack: `push()` and `pop()`
- Queue: `addq()` and `removeq()`
- Heap: `insert()` and `delete_max()`
  - Heap implemented as a list `h`, does not mean `h.append(7)` is legal

Abstract datatype

- Define behaviour in terms of operations
  - `(s.push(v)).pop() == v`
  - If `q.isempty()` then
    `((q.addq(u)).addq(v)).removeq() == u`
- No reference to implementation details
- Implementation can be optimized without affecting functionality

Black box view

- Imagine the data structure as a black box
- Designated buttons to interact
- Slot for input
- Display for output
- No other manipulation allowed



Built in datatypes

```
l = []
```

- List operations `l.append()`, `l.extend()` permitted
  - … but not dictionary operations like `l.keys()`
- Likewise, after `d = {}`, `d.values()` is OK
  - … but not `d.append()`
- Can we do this for stacks, queues, heaps, …?

Object Oriented programming

- Data type definition with
  - Public interface
    - Operations allowed on the data
- Private implementation
  - Match the specification of the interface

Classes and objects

- Class
  - Template for a data type
    - How data is stored
    - How public functions manipulate data
- Object
  - Concrete instance of template

Classes and objects

```
class Heap:
  def __init__(self,l):
    # Create heap
    # from list l

  def insert(self,x):
    # insert x into heap

  def delete_max(self):
    # return max element
```

```
# Create object,
# calls __init__()
l = [14,32,15]
h = Heap(l)

# Apply operation
h.insert(17)

h.insert(28)

v = h.delete_max()
```

Summary

- An abstract data type is a black box description
  - Public interface — update/query the data type
  - Private implementation — change does not affect functionality
- Classes and objects can be used for this
- More details in the next lecture