Data Structures for Classes

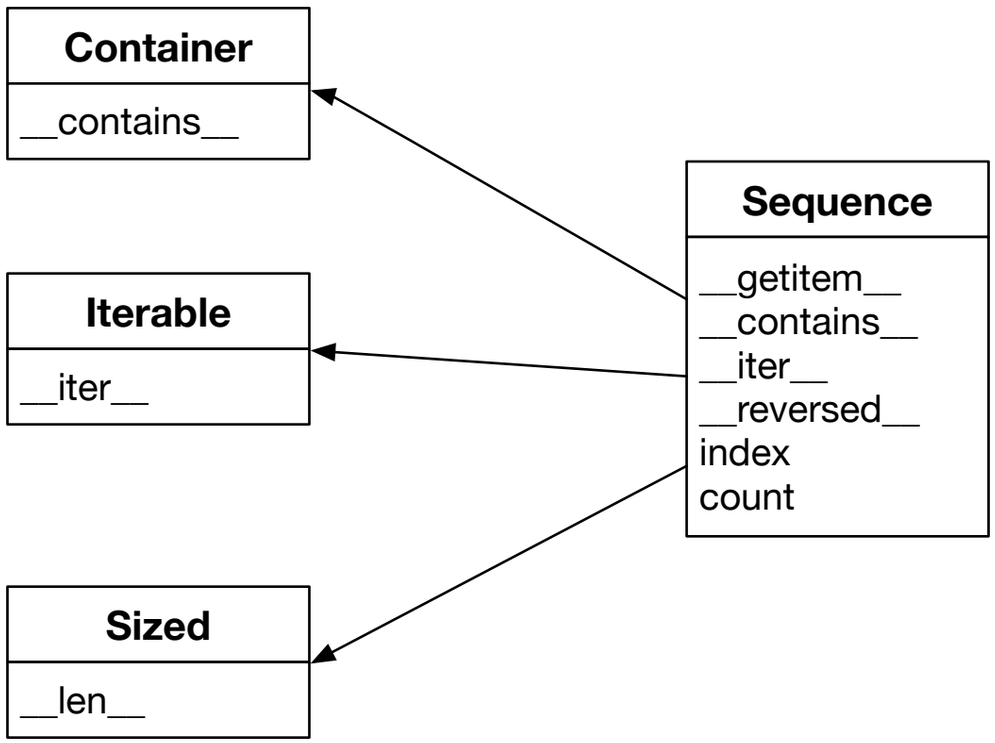
Thomas Schwarz

Interfaces

- Interfaces encapsulate how a user can use a certain set of classes
- Python does not need interfaces and only implemented them as Abstract Base Classes (ABC) in 3.4

Interfaces

- Example: Sequences



- An interface describes what can be invoked

Interfaces

- Example: Sequences
  - Some missing methods can be implemented via other methods
    - in still works even without __contains__ and __iter__

Interfaces

- ABC: Abstract Base Class
  - A class that does not have any methods implemented
- If you derive a class from an ABC:
  - You have to implement these methods
  - You make a public declaration that these methods are in your class

Interfaces

```
class FrenchDeck(collections.MutableSequence):
    ranks = [str(n) for n in range(2, 11)] + list('JQKA')
    suits = 'spades diamonds clubs hearts'.split()

    def __init__(self):
        self._cards = [Card(rank, suit) for suit in self.suits
                       for rank in self.ranks]

    def __len__(self):
        return len(self._cards)

def __getitem__(self, position):
    return self._cards[position]

def __setitem__(self, position, value):
    self._cards[position] = value

def __delitem__(self, position):
    del self._cards[position]

def insert(self, position, value):
    self._cards.insert(position, value)
```

Interfaces

- Here we have to implement methods that do not make sense for a deck of cards because MutableSequence demands them
- But now we get a whole lot of other methods that are implemented in terms of these methods