Abstract Data Type Set

A Set is an abstract data type representing an unordered collection of distinct items.

Sets appear in many problems: All the words used by Shakespeare. All correctly spelled words. All prime numbers. All the pixels of a given color. All the bomb locations in MineSweeper.

We could represent a set as an array or a list, but that is not natural (and often not efficient): Lists are ordered sequences of not necessarily distinct elements.

The Set ADT

- `set(elements)` Create new set with given elements.
- `len(s)` Return size of set.
- `x in s` Is $x \in s$?
- `s == t` Are sets equal?
- `s.issubset(t)` Is $s \subseteq t$?
- `s.issuperset(t)` Is $s \supseteq t$?
- `s.add(el)` Add `el` to set.
- `s.remove(el)` Remove existing element `el` from set.
- `s.discard(el)` Remove `el` from set.
- `s.union(t)` Return $s \cup t$.
- `s.intersection(t)` Return $s \cap t$.
- `s.difference(t)` Return $s \setminus t$.
- `for el in s:` Iterate over set elements.

A simple spell checker

```
def read_words():
  s = open("words.txt", "r")
  words = set()
  for w in s.readlines():
    words.add(w.strip())
  return words

def spell():
  words = read_words()
  while True:
    w = input("Tell me a word> ").strip().lower()
    if w in words:
      print("'%s' is a word" % w)
    else:
      print("Error: '%s' is not a word" % w)
```


Applications

- A spell checker.
  (Use set of correctly spelled words.)
- Measuring similarity between texts.
  (Consider set of words of each text, look at the size of their intersection and union.)
- Computing prime numbers.
  (Sieve of Erathosthenes).
- Remembering visited positions in a maze.
- Storing bomb positions in MineSweeper.

Implementing the Set ADT

Let's try to implement the Set ADT ourselves, using a Python List to store the elements.

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

def __contains__(self, el):
  return el in self._data

def __length__(self):
  return len(self._data)

def add(self, el):
  if el not in self._data:
    self._data.append(el)
```

Our implementation works, but it is significantly slower for large sets than the Python implementation.