Data Structures: Lists

UW CSE 190p

Summer 2012

Motivating data structures

Problem: Compute the number of unique words in a document

first attempt:

```
doc = list_of_words
number_of_words = len(list_of_words)
print number_of_words
```

Motivating data structures

Problem: Compute the number of unique words in a document

```
doc = list_of_words
number_of_words = 0
for word in doc:
  if not already_seen_it(word):
    number_of_words = number_of_words + 1
```

remember: first write the program you wish you had!

Motivating data structures

Problem: Compute the number of unique words in a document

```
doc = list_of_words
number_of_words = 0
scratchpad = []
for word in doc:
  if not already_seen_it(word, scratchpad):
    save(word, scratchpad)
    number_of_words = number_of_words + 1
```

Motivating data structures

Problem: Compute the number of unique words in a document

```
doc = list_of_words
number_of_words = 0
scratchpad = []
for word in doc:
  if not (word in scratchpad):
    number_of_words = number_of_words + 1
    scratchpad.append(word)
```

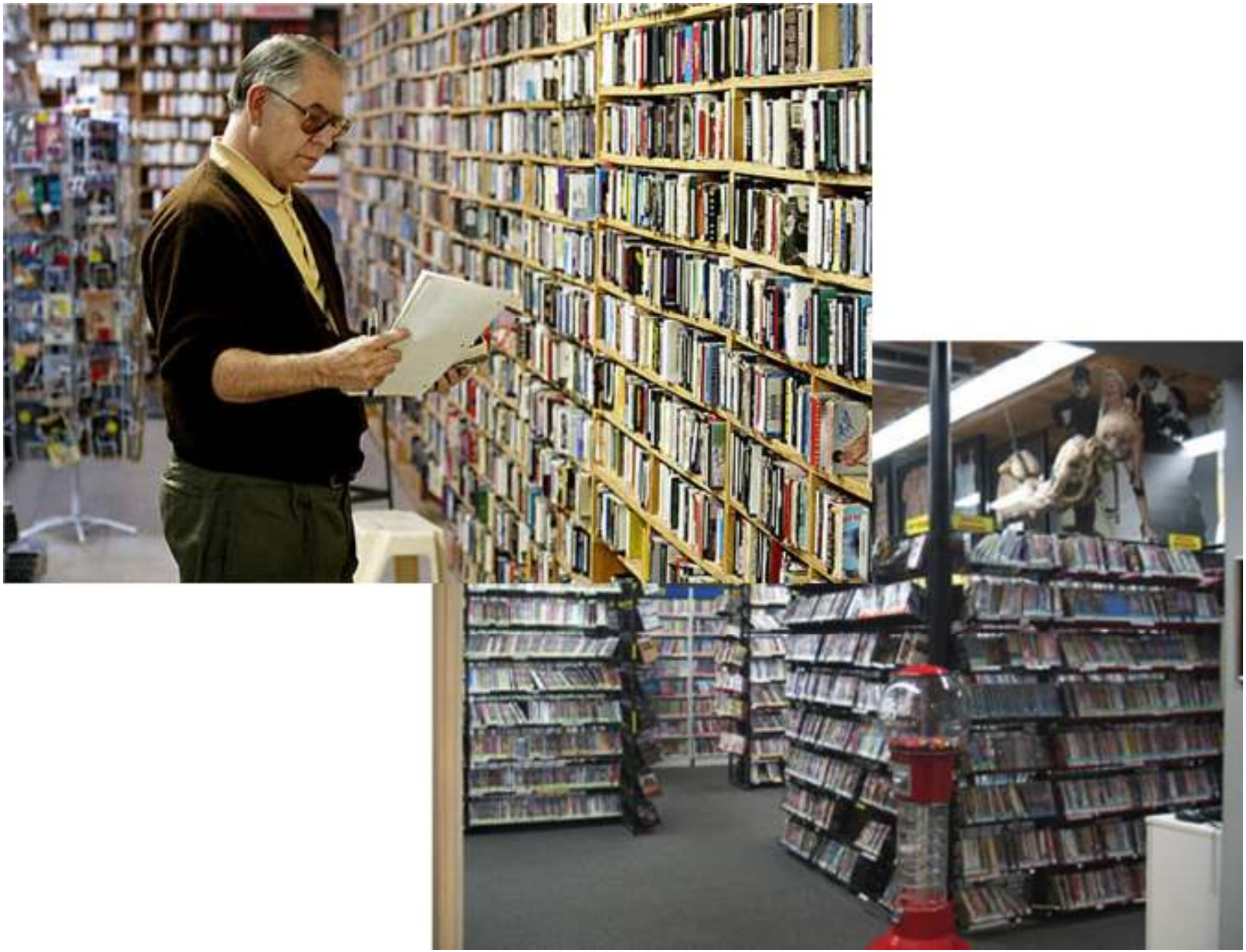
Motivating data structures

Problem: Compute the number of unique words in a document

```
doc = list_of_words
number_of_words = 0
scratchpad = set() # a set!
for word in doc:
  scratchpad.add(word)
print len(scratchpad)
```

Data Structures

- A *data structure* is way of organizing data such that certain operations are convenient or efficient
- Example: What operations are efficient with
  - a file cabinet sorted by date?
  - a shoe box?




BLOCKBUSTER
3 Movie Rentals
$15

Exercise

Write a function to compute the position of the first occurrence of the element `value` in a list `somelist`

```
def index(value, somelist):
    i = 0
    for c in somelist:
      if c == value:
        return i
      i = i + 1
    return None
```

Lists

- A list is a sequence of elements
- What operations should a list support efficiently and conveniently?

List Basics (demo)

“method syntax”

```
>>> mylist = [10,20,30]
>>> mylist.append(4) # think “append(mylist, 4)”

>>> mylist.index(10)
0
>>> mylist.sort()
>>>
```

List methods (demo)

Other list methods

- list.index(x)
  - Return the index in the list of the first item whose value is x. It is an error if there is no such item.
- list.sort()
  - Sort the items of the list, in place.
- list.reverse()
  - Reverse the elements of the list, in place.
- list.insert(i, x)
  - Insert an item at a given position. The first argument is the index of the element before which to insert, so a.insert(0, x) inserts at the front of the list, and a.insert(len(a), x) is equivalent to a.append(x).

More list methods

- list.extend(L)
  - Extend the list by appending all the items in the given list; equivalent to a[len(a):] = L.
- list.count(x)
  - Return the number of times x appears in the list.
- list.remove(x)
  - Remove the first item from the list whose value is x. It is an error if there is no such item.
- list.pop([i])
  - Remove the item at the given position in the list, and return it. If no index is specified, a.pop() removes and returns the last item in the list. (The square brackets around the i in the method signature denote that the parameter is optional, not that you should type square brackets at that position. You will see this notation frequently in the Python Library Reference.)

Exercise: Convert Units

```
ctemps = [17.1, 22.3, 18.4, 19.1]
```

```
ftemps = []
for c in ctemps:
  f = celsius_to_farenheit(c)
  ftemps.append(f)
```

```
ftemps = [celsius_to_farenehit(c) for c in ctemps]
```

```
doubles = [i*2 for i in range(5)]
```

Sets

- What's a set?
- What's the difference between a set and a list?
- What operations should a set support efficiently and conveniently?