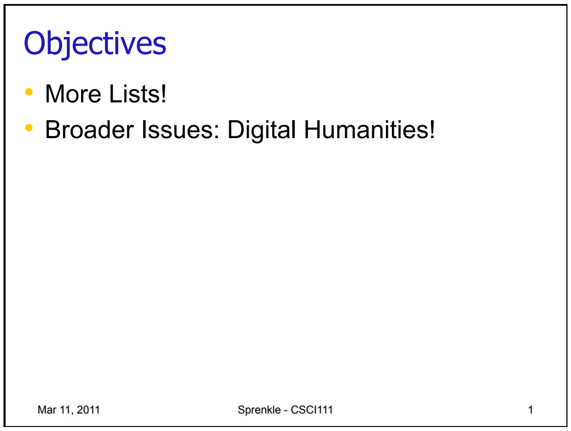
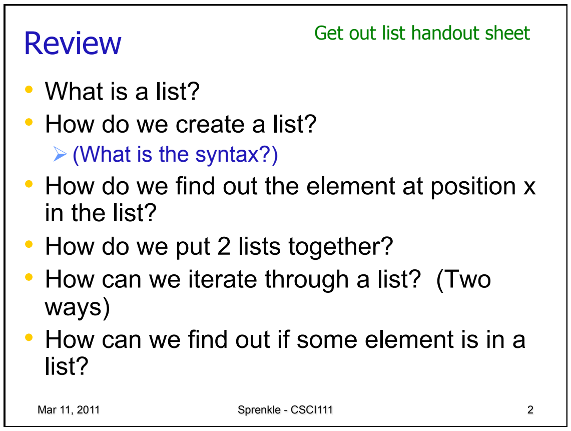


Objectives

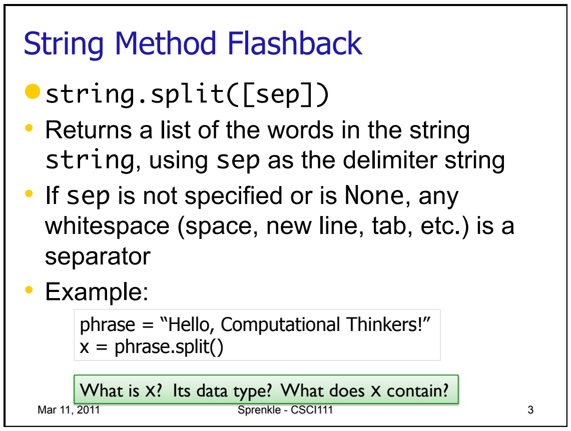
- More Lists!
- Broader Issues: Digital Humanities!



Review

Get out list handout sheet

- What is a list?
- How do we create a list?
  - (What is the syntax?)
- How do we find out the element at position x in the list?
- How do we put 2 lists together?
- How can we iterate through a list? (Two ways)
- How can we find out if some element is in a list?

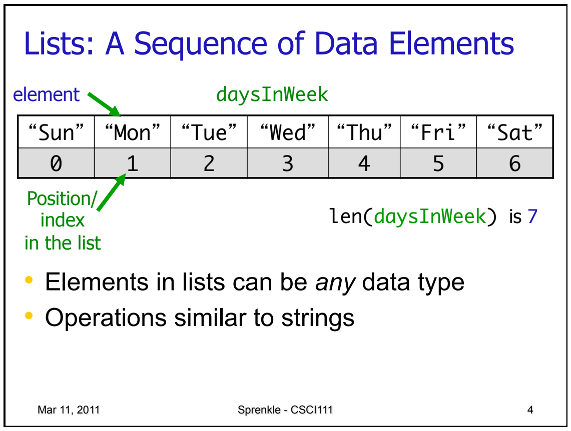


String Method Flashback

- `string.split([sep])`
- Returns a list of the words in the string `string`, using `sep` as the delimiter string
- If `sep` is not specified or is None, any whitespace (space, new line, tab, etc.) is a separator
- Example:

```
phrase = "Hello, Computational Thinkers!"
x = phrase.split()
```

What is x? Its data type? What does x contain?



Lists: A Sequence of Data Elements

element

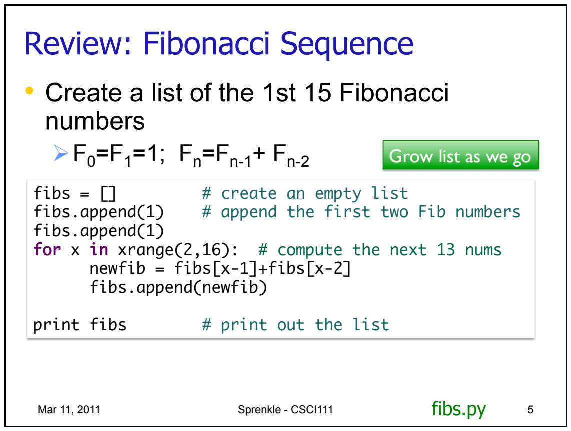
`daysInWeek`

“Sun”	“Mon”	“Tue”	“Wed”	“Thu”	“Fri”	“Sat”
0	1	2	3	4	5	6

Position/ index in the list

`len(daysInWeek)` is 7

- Elements in lists can be *any* data type
- Operations similar to strings



Review: Fibonacci Sequence

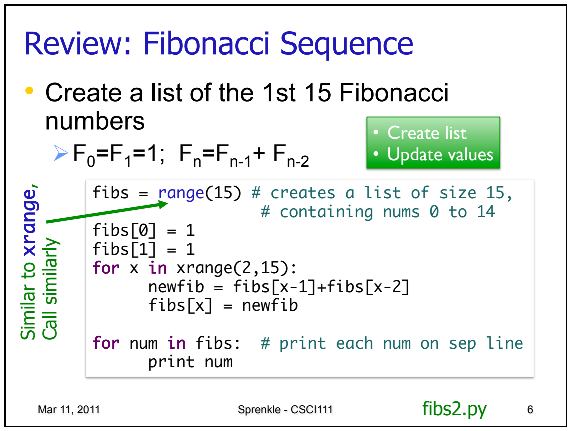
- Create a list of the 1st 15 Fibonacci numbers
  - $F_0=F_1=1$; $F_n=F_{n-1}+ F_{n-2}$

Grow list as we go

```
fibs = []             # create an empty list
fibs.append(1)        # append the first two Fib numbers
fibs.append(1)
for x in xrange(2,16):  # compute the next 13 nums
      newfib = fibs[x-1]+fibs[x-2]
      fibs.append(newfib)

print fibs            # print out the list
```

fibs.py



Review: Fibonacci Sequence

- Create a list of the 1st 15 Fibonacci numbers
  - $F_0=F_1=1$; $F_n=F_{n-1}+ F_{n-2}$

- Create list
- Update values

Similar to xrange,
Call similarly

```
fibs = range(15) # creates a list of size 15,
                 # containing nums 0 to 14
fibs[0] = 1
fibs[1] = 1
for x in xrange(2,15):
      newfib = fibs[x-1]+fibs[x-2]
      fibs[x] = newfib

for num in fibs:  # print each num on sep line
      print num
```

fibs2.py

Practice: Wheel of Fortune

- Read in all puzzles from a file, then randomly select from those puzzles
- Modify: don't allow repeats

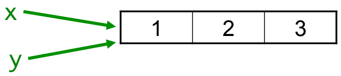
Copies of Lists

- What does the following code output?

```
x = [1, 2, 3]
y = x
y[0] = -1
print y
print x
```

List Identifiers are **Pointers**

```
x = [1, 2, 3]
y = x
```



- y is **not** a copy of x
  - Points to what x points to
- How to make a copy of y?

```
y = x + []    OR   y = []
                   y.extend(x)
```

Empty list

Lists as Parameters to Functions

If a list that is passed as a parameter into a function is **modified *in* the function**, the list **is modified *outside* the function**

- Lists are **not** passed-by-value/copied
- Different from immutable types (e.g., numbers, strings)
- Parameter is actually a **pointer** to the list in memory

Problem: Sort a list of 3 numbers, in descending order

- How with list methods?
- Can we do this using only 3 comparisons?

```
# order list such that list3[0] >= list3[1] >= list3[2]
def descendSort3Nums( list3 ):
```

Called as:

```
list = …
descendSort3Nums(list)
print list
```

descendSort.py

Descend Sort a List w/ 3 elements

```
def descendSort3Nums(list3):
   if list3[1] > list3[0]:
       # swap 'em
       tmp = list3[0]
       list3[0] = list3[1]
       list3[1] = tmp

   if list3[2] > list3[1]:
       tmp = list3[1]
       list3[1] = list3[2]
       list3[2] = tmp

   if list3[1] > list3[0]:
       tmp = list3[0]
       list3[0] = list3[1]
       list3[1] = tmp
```

```
def main():
    list = [1,2,3]
    descendSort3Nums(list)
    print list
```

Function does **not** *return* anything.
Simply modifies the `list3` parameter.

Lab 8: *Deal or No Deal* Overview

- Have 26 cases with various amounts of money
  - Amounts are known
- Player selects a case (hope has the big jackpot)
- In each round, player opens up cases
  - Reveals amounts that are not in the case they chose
- Banker makes an offer to buy the case
- Player decides if want to take the deal
  - Is the offer more than what is in the case?
  - Make decision based on amounts that haven't been opened yet
- Game ends when only one more case to open (two amounts on board) or player takes the deal.

Implementing *Deal or No Deal*

- Given: partial solution in code
  - Complete `main()` function, some additional functions
- Your job:
  - Read, understand given code
  - Fill in the functions for a complete solution

Digital Humanities

- Process large amounts of information to learn new information
  - Not necessarily numeric information
    - Text-based, images, multi-media, etc.
- Area growing in importance

Yates, Callie, Lida, Meng

Minh, Colin, Nick

Anh, Ola, Jean Paul, Will

Jan Cuny's Talk

Discussion

- What problem were the researchers solving?
- How did they solve the problem?
  - What were some of the challenges in solving it?
- Were you surprised by this use of computer science in the humanities?
- Are there any follow up questions that can be asked/answered?
- What other gains do you see possible by applying computing to the humanities?
- Will digital humanities take the "art" out of the humanities?
- Did anyone try out http://ngrams.googlelabs.com/?

Relation to Class

- How would you solve/implement …
  - Parsing the words
  - Keeping track of how many words there are
  - When there is a new word

Relation to Class

- Most concrete example of something we could write so far
- Analyzing texts, data
  - Strings
- Files containing the data
- Information retrieval, Natural Language Processing
  - Growing fields of computer science
    - E.g., Speech recognition
  - Process sentences
    - Determine subject/verb/direct object, etc.
- Aside: Phi Beta Kappa speaker discussed analyzing names