List Recursion:

Practice & Examples

One-Slide Summary

• Writing recursive functions that operate on recursive data structures takes **practice**. There are **standard approaches** to such problems.
• **length**, **member**, **sumlist**, **intsto**, **map** and **filter** are all important recursive functions that operate on lists. You should know **what they do and how to write them**.
• Python allows **list comprehensions**, in which a new list is created from a **filtered mapping** of an existing.

Outline

• Review: Procedure Problem Solving
• Review: [a,b,c], [elt]+lst, lst[0], lst[1:]
• **length**
• **member**
• **sumlist**
• **intsto**
• **map**
• **filter**
• List comprehensions

Teamwork!

• PS2 Partners Posted
  - Meet @ lab hours?
• PS1 Written Grades Posted
  - Holding Fee
  - Pick them up
• Do the readings!

How To Write A Procedure

• Find out what it is supposed to do.
  - What are the **inputs**? What types of values?
  - What is the **output**? A number? Procedure? List?
• Think about some example inputs and outputs
• **Define your procedure**
  - More on this next slide
• **Test** your procedure
## Defining A Procedure
- **Be optimistic!**
- **Base case:** Think of the simplest input to the problem that you know the answer to.
  - For number inputs, this is often zero.
  - For list inputs, this is often the empty list (null).
- **Recursive step:** Think of how you would solve the problem in terms of a smaller input. Do part of the work now, then make a recursive call to handle the rest.
  - For numbers, this usually involves subtracting 1.
  - For lists, this usually involves cdr.

## Procedure Skeleton
- The vast majority of recursive functions look like this:

```python
def my_procedure(my_input):
    if is-base-case?(my_input):
        return handle-base-case (my_input)
    return combine(first-part-of(my_input), \n                   my_procedure(rest-of(my_input)))
```

## Pairs and Lists
- 
  - **[a,b]** makes a pair of two things (“cons”)
    - [1, 2] --> [1, 2]
    - isinstance([1,2], list) --> True
  - **[0]** and **[1:]** get the first and rest
    - [1, 2][0] --> 1
    - [1, 2][1:] --> [2]
  - A list is either [] (null) or a pair where the rest is also a list
    - [1] + [2] + [3] --> [1,2,3]
    - [1,2,3] --> [1,2,3]
    - [1,2] == [] --> False
    - [1,2] + [3,4] --> [1,2,3,4]

## length
- The length function takes a single list as an argument and returns the number of elements in that list.
  - Recall: a list is either null or a pair where the second element is a list
  - length([]) --> 0
  - length([9,8]) --> 2
  - length([1] + []) --> 1
  - length([1,2,3][1:]) --> 2
  - length(5) --> error
- Write it now on paper. Base case? Recursion?

## More Power Needed!

## length Hint
- Here's a hint:

```python
def length(lst):
    if lst == null:
        ..
    ..
```
**Liberal Arts Trivia: Economics**

- This 1930 Tariff Act raised US tariffs on imported goods to record levels. Over 1000 US Economists signed a petition against it, and after it passed many others contributed increased their tariffs in retribution. US exports and imports dropped by half and many view this Act as a major catalyst for the Great Depression.

---

**Liberal Arts Trivia: German Lit**

- This tragic closet play is considered by many to be one of the greatest works of German literature. It centers on a man who makes a pact with the Devil in exchange for knowledge in his quest to discover the essence of life (“was die Welt im Innersten zusammenhält”). The man’s name officially means “Lucky” in Latin, but now has negative connotations.

---

**Definition of member**

```python
def member(elt, lst):
    if lst == []:     # empty list contains nothing
        return False
    if lst[0] == elt:
        return lst    # we found it!
    return member(elt, lst[1:]) # keep looking
```

- Where is the base case? Where is the inductive step?

---

**sumlist**

- Write a procedure `sumlist` that takes as input a list of numbers. It returns the sum (addition) of all of the elements of the list. It returns 0 for the empty list.
  - `sumlist([1,2,3])`   --> 6
  - `sumlist([])`        --> 0
**Definition of sumlist**

- And here it is …

```python
def sumlist(lst):
    if lst == []:  # base case
        return 0
    return lst[0] + \  # add current element
    sumlist(lst[1:])  # to rest of list
```

**intsto**

- The function `intsto` takes a single non-negative integer as an argument. It produces a list of all of the integers between 1 and its argument.
  - `intsto(3) -> [1,2,3]`
  - `intsto(7) -> [1,2,3,4,5,6,7]`
  - `intsto(0) -> []`

```python
def intsto(x):
    if (x < 1):  
        return []             # base case
    return [x] + \  # this number
        intsto(x-1)     # recursive result
```

**Correct Definition of intsto**

```python
def intsto(x):
    if (x < 1): 
        return []             # base case
    return [x] + \  # this number
        intsto(x-1)     # recursive result
```

**Higher-Order Functions: map**

- The `map` function takes two arguments: a work function and a list. It applies the work function to every element of the list in order and returns a list of the result.
  - `map(sqrt, [9,16,36])` -> `[3,4,6]`
  - `map(square, [1,2,3])` -> `[1,4,9]`
  - `map(abs, [2, -3, 4])` -> `[2,3,4]`
  - `map(len, [“I”, “Claudius”])` -> `[1,8]`
  - `map(sqrt, [])` -> `[]`

**Mission Impossible: Write map**

- You can do it!
  - `map(square,[1,2,3])`  
    - (1 4 9)
  - `map(abs,[2, -3, 4])`  
    - (2 3 4)
  - `map(sqrt,[])`  
    - []
Definition of map

Let's look in detail:

```python
def map(workfun, lst):
    if lst == []:
        return []  # base case
    return [workfun(lst[0])] +  # make a list
              map(workfun, lst[1:])  # recursive
```

Alternate map

```python
• map(abs, [1, -2, 3]) -> [1, 2, 3]
• [ abs(x) for x in [1, -2, 3] ] -> [1, 2, 3]
• map(sqrt, [1, 4, 9]) -> [1, 2, 3]
• [ sqrt(i) for i in [1, 4, 9] ] -> [1, 2, 3]
• map(len, []) -> []
• [ len(elt) for elt in [] ] -> []
• Either way is full credit!
```

Liberal Arts Trivia: Philosophy

• This branch of philosophy deals with the theory, nature and scope of knowledge. Key questions include “what is knowledge?”, “how is knowledge acquired?”, “what do people know?”, “how do we know what we know?”, “what is the relationship between truth and belief?”.

Liberal Arts Trivia: Norse Myth

• In Norse Mythology, this god is associated with light and beauty. His mother made every object on earth vow never to harm him, but she did not ask mistletoe. The other gods made a new pastime of hurling objects at him and watching them bounce off. The trickster Loki heard of this, fashioned a spear from mistletoe and had it thrown a him, with fatal results.

Liberal Arts Trivia: Music

• This musical instrument of the brass family produces sound when the player’s vibrating lips cause the air column inside the instrument to vibrate. It is usually characterized by a telescopic slide with which the player varies the length of the tube to change the pitch. Glenn Miller, famous for his “big band” and songs like *In the Mood* and *Chattanooga Choo Choo*, played this instrument.

Map and iteration ...

• I want to print the squares of numbers 0 to 5.
```python
    for number in [0, 1, 2, 3, 4, 5]:
        display(number * number)
```
```python
    for number in range(6):
        display(number * number)
```
```python
    map(display, [x*x for x in range(6)])
```
• All three are equal! Expect last one on tests.
The `filter` function takes two arguments: a predicate and a list. A predicate is a function that returns True or False. Filter returns the sublist consisting of those elements that satisfy the predicate.

- `filter(is_odd, [1,2,3,4])` -> `[1,3]`
- `filter(is_five, [1,5,5,"hi"])` -> `[5,5]`
- `filter(lambda (x) : x < 5, [1,9,2,0])` -> `[1,2,0]`
- `filter(is_five, ["susan","b","anthony"])` -> `[]`
- `filter(is_odd, [])` -> `[]`

### Alternate filter

```python
filter(is_odd, [1,2,3,4])
```

```python
[ x for x in [1,2,3,4] if is_odd(x) ]
```

```python
filter(is_five, [1,5,5,"hi"])
```

```python
[ x for x in [1,5,5,"hi"] if is_five(x) ]
```

```python
filter(lambda (x) : x < 5, [1,9,2,0])
```

```python
[ x for x in [1,9,2,0] if x < 5 ]
```

Either way is full credit!

### Map and Filter Combined

```python
[ x * x for x in [1,2,3,4,5] if is_odd(x) ]
```

```python
[ x/2 for x in [11,22,33,44] if is_odd(x+1) ]
```

```python
[ x/3 for x in [11,22,33,44] if is_odd(x+1) ]
```

```python
[ transform for name in list if predicate ]
```

```python
[ map for elt in list if filter ]
```
Homework

• Problem Set 2
• Problem Set 2 -- Reading