CSC 148: Introduction to Computer Science

Week 7

Recursion (continued)

List comprehensions, recursion efficiency

University of Toronto Mississauga,

Department of Mathematical and Computational Sciences

New Lists from Old

- Suppose L is a list of the first hundred natural numbers

```
L = list(range(100))
```

- If I want a new list with the squares of all the elements of L,

  I could:

```
new_list = []
for x in L:
   new_list.append(x * x)
```

- Or I could use the equivalent list comprehension

```
new_list = [x * x  for x in L]
```

Filtering with [...]

- I can make sure my new list only uses specific elements of the old list ...

```
L = ["one","two","three","four","five","six"]
```

by adding a condition ...

```
 New_list = [s * 3
             for s in L
             if s <= "one"]
```

- Notice that a comprehension can span several lines, if that makes it easier to understand

General comprehension pattern

```
[expression for name in iterable if condition]
```

- *expression* evaluates to a value
- *name* refers to each element in the iterable (list, tuple, dict, ...)
- *condition* (optional) evaluates to either True or False

Practice … worksheet

Recall: Sum of Elements in Nested List

- L = [1, [5,3], 8, [4,[9,7]]]

```
def sum_list(L):
    if  isinstance(L, list) :
        s = 0
        for elem in L:
            # calculate the sum of the sublist "elem" recursively
            s += sum_list(elem)
        return s
    else:
        return L
```

recursive step

base case

⇨ We could rewrite the recursive step using list comprehensions:

```
if  isinstance(L, list) :
    return  sum([sum_list(elem)  for elem in L])
```

Sum of list elements – nested lists

- L = [1, [5,3], 8, [4,[9,7]]]

```
        def sum_list(L):
recursive  if  isinstance(L, list) :
   step        return sum([sum_list(elem)  for elem in L])
  base     else:
  case         return L
```

- Can we make this even more compact?

Sum of list elements – nested lists

- L = [1, [5,3], 8, [4,[9,7]]]

```
def sum_list(L):

    return  sum( [sum_list(elem)  if  isinstance(elem, list)  else  elem
                 for elem in L])
```

More Complex: `semi_homogeneous`

- A single integer and empty list are semi-homogeneous.
- In general, a list is semi-homogeneous if and only if:
    - (all of its sub-nested-lists are integers) OR (all of them are lists)
    - all of its sub-nested-lists are semi-homogeneous