Lecture 14: Lists (cont'd)

CS 51P

October 30, 2023

Lists

- a list is an ordered collection of arbitrary elements:

```
a_list = [3, 6.5, True, “a”, [5, 3]]
b_list = []
c_list = "a b c d".split()
```

- a list is a sequence, so can index into, loop over, check for membership, slice, etc

- lists are mutable
  - Add, remove or modify elements

List Operations

adding to a list (updates original list)

- a_list.extend(*list*)
- a_list.append(*elem*)
  - Different than extend – e.g. [5, 1]
- a_list.insert(*index*, *elem*)
  - Insert elem at index, shifts down

modifying a list

- direct assignment
  - a_list[0] = 2

removing from a list

- a_list.remove(*elem*)
  - removes 1st instance of elem
  - error if *elem* not in a_list
- del a_list[*slice*])
  - removes the slice from the list based on the given index
- a_list.pop()
  - returns (and removes) a_list[-1]
- a_list.pop(*index*)
  - returns (and removes) a_list[index]

List Operations

other

- min(a_list), max(a_list), sum(a_list)
- len(a_list)
- a_list.index(*elem*)
  - returns index of 1st instance of elem or error
- a_list.count(elem)
  - returns the number of *elem* in the list
- a_list.copy()
  - Returns a copy of list

+ and * operators

- Works on lists, but creates a new list
  - >>> a_list = [1, 2, 3]
  - >>> new_list = a_list + a_list
  - >>> new_list
  - [1,2,3,1,2,3]
  - >>> another_list = a_list * 2
  - >>> another list
  - [1,2,3,1,2,3]

Example

- Define a function word_list that takes a filename as an argument and returns a list of all the words in that file.

Exercise

- Define a function count_words that takes a filename as input and returns the total number of unique words in that file

List alias

```
>>> sports = ["basketball", "soccer", "tennis"]
>>> my_sports = sports
>>> my_sports.append("swimming")
>>> my_sports
>>> ?
>>> sports
>>> ?
```

- The same list that have two different names
- Changes made with one list will affect the other

List.copy()

- list.copy() – returns a copy of the list

- >>> sports = ["basketball", "soccer", "tennis"]
- >>> my_sports = sports.copy()
- >>> my_sports.append("swimming")
- >>> my_sports
- ???
- >>> sports
- ???

Example

- Can we define a function `capitalize_colors` that takes in a list of rainbow colors as input, and modify the colors in place?

list comprehension (filter + map)

```
new_list = []
for i in old_list:
     if filter(i):
          new_list.append(map(i))
```

```
new_list = [map(i) for i in old_list if filter(i)]
```

- Examples:
  - write a function `double` that takes a list of ints and returns a list with every number doubled
  - write a function `odds` that takes a list of ints and returns a list of the odd elements

Exercise

- Use list comprehension to write a function `square_positive` that takes a list of ints and returns a list that contains the square of all those that are positive. For example, square_positive([-1, -2, 3, 4, -5]) will return [9, 16].

Tuples

- a tuple is an ordered set of elements:

```
(3, 6, 2, 1)
```

- ways to create a tuple:

```
tup = (3, 6, 2, 1)
tup1 = ()
tup2 = tuple(["a","b","c"])
```

- a tuple is a sequence, so can index into, loop over, check for membership, slice, etc

```
>>> tup[1]
>>> 6
```

- operators: + and *
- tuples are immutable

Tuples are immutable

- tuples are immutable (can not be changed in place)

```
tup = (3, 6, 2, 1)
tup[0] = 4
```

- TypeError: 'tuple' object does not support item assignment

Tuple unpacking

- Can use tuples to assign multiple variables at the same time

  - >>> (x, y) = (5, 1)
  - >>> x
  - 5
  - >>> y
  - 1

- Number of variables on left hand side needs to be the same as the right hand side

Why Tuples?

- More restrictive because it is immutable
- Tuples are more memory efficient than lists
- Execution speed of using tuples is faster than using lists

Example/Exercise

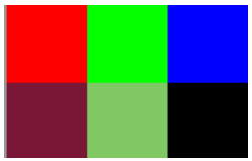
- Use list comprehension to write a function `average_pairs` that takes a list of pairs (two-element tuples where both elements are integers), and returns a list consists of the average value of each pair. For example average_pairs([(1, 2), (2, 3), (3, 4)]) will return [1.5, 2.5, 3.5].

Pixels and RGB

- A PPM image consists of
  - Header
  - Body that consists of rows of pixels
- Each pixel holds RGB values
  - Red, Green, and Blue
  - Each value is the brightness for the color
  - Can make any color from RGB

```
P3
3 2
255
```

header



```
255 0 0      0 255 0      0 0 255
122 23 55    128 200      100 0 0 0
```

body