COMP10001 Foundations of Computing
Iteration, Lists and Sequences

Semester 2, 2016
Chris Leckie

July 31, 2016

Announcements

- Projects are us
- Worksheers 3 & 4 due 23:59pm Monday 15/8
- Revision lecture Wednesday

Lecture Agenda

- Last lecture:
  - Functions
  - Iteration
- This lecture:
  - Iteration (cont. from previous lecture)
  - Lists
  - Mutability

Lists: An Introduction

- To date, we have discussed data types for storing single values (numbers or strings), and tuples for storing multiple things. There is another way to store multiple things: a “list”.

```
["head","tail","tail"] # list of strings
[5,5,30,10,50] # list of ints
[1,2,"buckle my shoe",3.0,4.0] # allsorts
```

- As with all types, we can assign a list to a variable:

```
fruit = ["orange","apple","apple"]
```

List Indexing and Splitting

- To access the items in a list we can use indexing (just like we do with strings and tuples):

```
>>> listOfStuff = ["12", 23, 4, 'burp']
>>> listOfStuff[-1]
'burp'
```

- We can similarly slice a list:

```
>>> listOfStuff[:2]
['12', 23]
```

and calculate the length of a list with `len()`

```
>>> len(listOfStuff)
4
```

Class Exercise

- Write code to extract the middle element from the list `l`:

```
>>> l = [1,2,3]
>>> middle(l)
[2]
>>> l = [1,2]
>>> middle(l)
[]
```

- What are the values of `l1` and `l2` after execution of the following code:

```
l1 = [1,2,3,4]
l2 = l1[::-1]
```

But what's the difference?

It seems that tuples and lists are the same, why have both? Important difference: **mutability**

```
>>> mylist = [1,2,3]
>>> mytuple = (1,2,3)
>>> mylist[1] = 6 ; print(mylist)
[1,6,3]
>>> mytuple[1] = 6 ; print(mytuple)
TypeError: 'tuple' object does not support ite
```

- Tuples are immutable - they cannot be changed once created
- Lists are mutable - individual elements can be changed

Mutability

Types in Python can be either:

- "immutable": the state of objects of that type cannot be changed after they are created
- "mutable": the state of objects of that type **can** be changed after they are created

Quiz

- Are strings mutable?
- Are lists mutable?
- Are tuples mutable?

Function Arguments I

A key place where mutability is important is when passing arguments to functions.

```python
def f(l):
   l[1] = 6

mylist = [1,2,3,4,5]
f(mylist)
print(mylist)

mytuple = (1,2,3,4,5)
f(mytuple)
print(mytuple)
```

Function Arguments II

```
def f(l):
   if type(l) is list:
      l = l + [6]
   else:
      l = l + (6,)

mylist = [1,2,3,4,5]
f(mylist)
print(mylist)

mytuple = (1,2,3,4,5)
f(mytuple)
print(mytuple)
```

Function Arguments III

```
def f(l):
    if type(l) is list:
        l.append(6)
    else:
        l = l + (6,)

mylist = [1,2,3,4,5]
f(mylist)
print(mylist)

mytuple = (1,2,3,4,5)
f(mytuple)
print(mytuple)
```

Lecture Summary

- What is a list?
- What are mutable types?