COMP10001 Foundations of Computing
Mutability and Function Debugging

Semester 2, 2016

Chris Leckie

July 31, 2016

Announcements

- Workshops 5 & 6 due 23:59pm Monday 22/8

Lecture Agenda

- Last lecture:
  - Lists
  - Mutability
- This lecture:
  - More on mutability and functions
  - How to "debug"?

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.

```
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,)
    return(l)

mylist = [1,2,3,4,5]
list2 = f(mylist)
print(mylist) ; print(list2)

mytuple = (1,2,3,4,5)
tuple2 = f(mytuple)
print(mytuple) ; print(tuple2)
```

Local Variables and Mutability I

- When you pass a mutable object to a function and locally mutate it in the function, the change is preserved in the global object:

```
def changeList(lst):
    lst = []
    return lst
def changeListItem(lst):
    lst[0] = "Changed, hah!"
```

```
>>> mylist = [1,2,3]
>>> changeList(mylist)
[]
>>> mylist
[1, 2, 3]
>>> changeListItem(mylist)
>>> mylist
['Changed, hah!', 2, 3]
```

Local Variables and Mutability II

- In fact, there is nothing specific to functions going on here; it is consistent with the behaviour of mutable objects user assignment/mutation:

```
>>> list1 = [1,2,3]
>>> list2 = list1
>>> list2[0] = "Changed, hah!"
>>> list2
['Changed, hah!', 2, 3]
>>> list1
['Changed, hah!', 2, 3]
```

Python Tips: Placement of Function Definitions

- Functions in Python must be defined before they are called (i.e. the definition must precede any code that calls them)
- This may seem curious, until you realise that function names are just variables, and the behaviour is identical to that of other variables

Function Practice

- A "pangram" is a string that contains all the letters of the English alphabet at least once, for example:

  `"The quick brown fox jumps over the lazy dog"`

  Write a function `pangram` to check whether a string `gram` is a pangram or not.

Function Debugging Practice

- What is wrong with the following, and how would we fix it?

```
def last_vowel(word):
    """Find the index of the last vowel
    in 'word'"""
    for i in range(len(word)):
        if word[i] in "aeiou":
            print(i)
        else:
            print(None)
```

- A great visualisation tool: `http://www.pythontutor.com` (not in Safari)