

One minute responses

- Not really sure how the loop, for, while, and zip work. I just need more practice problems to work on.
- More practice problems please!
- Comparing the different loops clarified how they work.
- Is my love affair with `while` loops a dangerous road to travel? *Just be careful of infinite loops and you should be fine! Control-C is always available if your program gets out of control.*
- Today, analyzing a loop problem in several ways helped very much. (x2)
- Today's Python work was easier to digest.
- I'm slowly getting this.

- This time I felt like there was enough time to do the practice problems and they were less difficult. (x2)
- I understand the sample problems after help from the solutions, but I am still having problems finding solutions on my own.
- I liked how we walked through different options for writing a program and evaluated how good they are based on extreme cases.
- Not much new programming material. The analysis of different loops for the same task was informative.
- I didn't get the zip command. *See solutions to last session's practice problem 2, in this lecture.*

Tuesday's practice problem 2

```
seq1 = "ATCGTGTGCAATG"  
seq2 = "ATGGAGTGTAATG"  
count = 0.0  
for (b1, b2) in zip(seq1, seq2) :  
    if b1 != b2 :  
        count = count + 1  
count = count / len(seq1)  
print count
```

Note: if you read seq1 and seq2 from a file, you'd need to knock off the trailing newline character!

Tuesday's practice problem 2 without zip

```
seq1 = "ATCGTGTGCAATG"
seq2 = "ATGGAGTGTAATG"
count = 0.0
seqlength = len(seq1)
for index in range(0,seqlength) :
    if seq1[index] != seq2[index] :
        count = count + 1.0
count = count / seqlength
print count
```

Dictionaries

- A dictionary organizes linked information
- Examples:
 - word and definition
 - name and phone number
 - name and DNA sequence
 - username and password
- If you know the first entry, you can immediately get the second one

Rules for dictionaries

- The first item is a “key”
- Each key can only appear once
- A key must be an immutable object: number, string or tuple
- Lists cannot be keys (they are mutable)
- The key should be the item you’ll use to do look-ups

Key example: name and telephone number

- Phone book: we have a name, we want a number
- Name is the key
- Crank call prevention: we have a number, we want a name
- Number is the key

Creating a dictionary

```
#create an empty dictionary  
myDict = {}
```

```
#create a dictionary with three entries  
myDict = {"Mary": 4123, "Jon":2057, "Fred":1122}
```

```
#add another entry  
myDict["Joe"] = 2232
```

```
#change Mary's phone number  
myDict["Mary"] = 4040
```

```
#delete Mary from dictionary  
del myDict["Mary"]
```


Using a dictionary

```
>>> myDict["Mary"]
4040
>>> myDict.keys()
['Mary', 'Joe', 'Fred', 'Jon']           # in no particular order
>>> myDict.haskey('Bill')
False
>>> 'Jon' in myDict
True
```

Using a dictionary

```
birthdays = {"Mary": "July 6", "Jon": "May 21", "Gwynne": "Sept 1"}
for person in birthdays.keys() :
    print "Send ", person, " a card on ", birthdays[person]
```

Sorting a dictionary

```
sortkeys = birthday.keys()
sortkeys.sort()
for person in sortkeys:
    print "Send ", person, " a card on ", birthdays[person]
```

Question: why not `for person in birthday.keys().sort()`?

Sorting a dictionary

```
sortkeys = birthday.keys()
sortkeys.sort()
for person in sortkeys:
    print "Send ", person, " a card on ", birthdays[person]
```

Question: why not `for person in birthday.keys().sort()`?

Python says: Iteration over non-sequence

Problem: `sortkeys.sort()` returns `None`, not `sortkeys`!

Practice problem 1

- Make a file with several lines like this:
- Mary 0121
- Joe 1432
- Paul 8476
- Write a Python program to read this file in and create a dictionary where name is the key
- Print the dictionary to make sure it works

Solution to 1

```
import sys
filename = sys.argv[1]
filehandle = open(filename,"r")
linelist = filehandle.readlines()
phonedict = {}
for line in linelist :
    wordlist = line.split()
    phonedict[wordlist[0]] = wordlist[1]
print phonedict
```

Comment on 1

- The phone "number" is a string
- This is probably a good thing:
- Phone number "0121" should not turn into 121

Practice problem 2

- Building on the previous program:
- Read in a second command line argument which is a name
- Print the phone number corresponding to that name

Solution to 2

```
import sys
filename = sys.argv[1]
filehandle = open(filename,"r")
linelist = filehandle.readlines()
phonedict = {}
for line in linelist :
    wordlist = line.split()
    phonedict[wordlist[0]] = wordlist[1]
targetname = sys.argv[2]
print phonedict[targetname]
```

Practice problem 3

- Building on the previous program:
- Print out a telephone directory sorted by name
- Format like this:
- Mary extension 0121
- William extension 3243

Solution to 3

```
import sys
filename = sys.argv[1]
filehandle = open(filename,"r")
linelist = filehandle.readlines()
phonedict = {}
for line in linelist :
    wordlist = line.split()
    phonedict[wordlist[0]] = wordlist[1]
keylist = phonedict.keys()
keylist.sort()
for person in keylist:
    print person, "extension", phonedict[person]
```

Commands covered in this lecture

Example	Meaning
<code>myDictionary.keys()</code>	gets a list of keys
<code>myKeys.sort()</code>	sort keys
<code>myDictionary[maryKey]</code>	retrieve entry for key 'maryKey'
<code>myDictionary.haskey[maryKey]</code>	test for presence of key in dictionary
<code>maryKey in myDictionary</code>	test for presence of key in dictionary