One-minute responses

- I definitely need more time to practice with the problems outside of class.
- The Python section builds well on what we've done previously.
- It helps me a lot when we go through various pitfalls/common mistakes–it was good to stop after the first sample program. (x2)
- I am finally beginning to see how one can layer functions and modules in Python to achieve complex tasks.
- I really appreciate that only 1 or 2 new Python concepts are being
- I usually get distracted during Python explanations because I am trying to fix silly errors in my program instead of listening and taking notes.
- Programming section was more challenging than I thought it would be.

- Today's class the pace was great, although it might have been better if I had been encouraged to start the second problem immediately after completing the first–I wasn't sure if we should have waited to review before moving on. *In general, feel free to move on if you are done.*

One-minute responses

- Did not really get what you were saying about modules or problem 3 in class.
- I'm still confused about how to use a module.
- *A module is a collection of variables and functions in a file. The module is like a bag of useful tools. It doesn't do anything itself, but it provides access to those tools.* `modulename.toolname` *refers to one tool within the bag; for example* `math.log()` *is the log function offered by the math module.*
- *We might write a module with all kinds of useful functions to handle the FASTA database format, and call it the* `fasta` *module. Then, when our program needed to handle FASTA data we would* `import fasta` *and use the functions in it, such as* `myname = fasta.getspeciesname(myline)`.

Sorting

- Basic sorting
- Sorting different kinds of containers
- Comparison functions for more complex sorting

How to swap two variables

- Suppose I have a drawer of shirts and a drawer of pants
- I'd like to switch the two drawers
- Generally I need some temporary place to store the things I'm moving

```
drawer1 = "shirts"
drawer2 = "pants"

# swap shirts and pants
temp = drawer1
drawer1 = drawer2
drawer2 = temp
```

Swapping in a list

```
clotheslist = ["shirts","pants","socks"]
print clotheslist[0]
'shirts'
print clotheslist[1]
'pants'

# swap shirts and pants
temp = clotheslist[0]
clotheslist[0] = clotheslist[1]
clotheslist[1] = temp

print clotheslist
['pants','shirts','socks']
```

`sort()`

- The `sort` method modifies a list in-place
- It normally sorts in ascending order

```
mylist = [3,2,1]
print mylist
[3, 2, 1]

mylist.sort()
print mylist
[1, 2, 3]
```

Sorting of strings in lexicographic order

```
mylist = ["Mary", "Joe", "Steve"]
mylist.sort()
print mylist
['Joe','Mary','Steve']

# case matters!
mylist.append("kevin")
mylist.append("bill")
mylist.sort()
print mylist
['Joe', 'Mary', 'Steve', 'bill', 'kevin']
```

How to sort a tuple?

- `sort` changes a list in place
- tuples are immutable and can't be changed in place
- `mytuple.sort()` is therefore a Python error
- To sort a tuple, make a list copy:

How to sort a tuple?

```
mytuple = (3,2,1)
mytuple.sort()
AttributeError: ’tuple’ object has no attribute ’sort’

mylist = list(mytuple)
mylist.sort()
print mylist
[1, 2, 3]
mytuple = tuple(mylist)
print mytuple
(1, 2, 3)
```

How to sort a dictionary?

- Dictionaries are kept in an order Python finds convenient
- You aren't allowed to sort them
- However, you can sort the keys, which is nearly the same:

```
mydict = {"Mary":"1023", "Jon":"2324", "Fred":"0023"}
sortkeys = mydict.keys()
sortkeys.sort()
for key in sortkeys :
  print key, "--", mydict[key]
```

How to sort a dictionary?

What if we want to sort by entry, not by key? One solution is to make a reversed dictionary:

```
mydict = {"Mary":"1023", "Jon":"2324", "Fred":"0023"}
# want to sort by number, not name
keylist = mydict.keys()
reversedict = {}
for key in keylist :
  reversedict[mydict[key]] = key
sortkeys = reversedict.keys()
sortkeys.sort()
for key in sortkeys :
  print key, "--", reversedict[key]
```

More complicated sorting problems

- What if we want to sort by a different rule than ascending order?
- We need to write a comparison function
- `mylist.sort(mycomparison)` will use the function

Comparison function

- Must take 2 arguments
- Return -1 if the first argument should first
- Return 0 if there is a tie
- Return 1 if the first argument should come second

Comparison function: sort in descending order

```
mylist = [10, 17, 12]
mylist.sort()
print mylist
[10, 12, 17]
def reverseCompare (first, second):
  if (first > second) :
    return (-1)
  elif (first < second) :
    return 1
  else :
    return 0

mylist.sort(reverseCompare)
print mylist
[17, 12, 10]
```

Practice problem 1

- Write a function which compares two strings, ignoring upper/lower case
- Return -1 if the first string should come first
- Return 0 if the two strings are tied
- Return 1 if the second string should come first
- "Mary" and "maRY" should give a 0

Importing a function

- Suppose our function was called caselessCompare and was in file nocase.py.
- We could use it in a different file by importing it:

```
# note that there is no ".py" here; just the bare filename
# the filename becomes the module name
import nocase

# note that the name of an imported function
# begins with the name of its module
mylist.sort(nocase.caselessCompare)
```

Practice problem 2

- Write a program which:
  - Reads in a whole file
  - Separates the file into a list of words
  - Sorts the words using your comparison function
  - Prints the sorted words
- Try it on file sample.txt

Practice problem 3

- Modify your previous program so that if a word appears several times, it is only printed once
- Hint: don't try to change the list in place
- Make a new list holding only one copy of each word

Problem 1 solution

```
def caselessCompare(first,second) :
  first = first.lower()
  second = second.lower()
  if (first < second) :
    return (-1)
  elif (first > second) :
    return (1)
  else :
    return 0
```

Problem 2 solution

```
import sys
filename = sys.argv[1]
filehandle = open(filename,"r")
# get the whole file as a big string
filestring = filehandle.read()
# split into words
wordlist = filestring.split()
# sort
import nocase
wordlist.sort(nocase.caselessCompare)
for word in wordlist :
  print word
```

Problem 3 solution

```
import sys
filename = sys.argv[1]
filehandle = open(filename,"r")
filestring = filehandle.read()
wordlist = filestring.split()
import nocase
wordlist.sort(nocase.caselessCompare)
# make a list containing the first word
uniquewords = [wordlist[0]]
for index in range(1,len(wordlist)) :
  # if it's a new word, add it
  if wordlist[index].lower() != wordlist[index-1].lower() :
    uniquewords.append(wordlist[index])
for word in uniquewords :
  print word
```

Issues with these solutions

- If you test these solutions, you will find that punctuation confuses them
- They think "students," is a different word than "students"
- A good take-home problem: how to fix this?