# More Examples Using Lists Tuples and Dictionaries in Pythom 

CS 8: Introduction to Computer Science, Winter 2018
Lecture \#11

Ziad Matni
Dept. of Computer Science, UCSB

## Administrative

- Homework \#6 is DUE on MONDAY (3/5)
- Lab \#4 due Today!
- Project \#2 will be issued by the end of the week


## Lecture Outline

- 2 more example with lists...
- Tuples
- Dictionaries
- Multiple examples


## Calculating Means and Medians Using Lists

- Mean (Average) $=(\max -\min ) /$ sum
- Median (middle item) is more complex...
- This isn't in any list function, so we have to develop it ourselves

Example:

| 1 | 5 | 2 | 10 | 8 | 7 | 7 | 6 | 3 |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |

sort it first and then find the middle value...

| 1 | 2 | 3 | 5 | 6 | 7 | 7 | 8 | 10 |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | Median = 6

If there's an even number of entities, then employ an average calc...

| $\mathbf{1}$ | $\mathbf{2}$ | $\mathbf{3}$ | 5 | 6 | $\mathbf{7}$ | $\mathbf{7}$ | $\mathbf{8}$ |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |$\quad$ Median $=5.5$

## "Find the Median" Algorithm

1. Sort the list first
2. Determine the length of the list (why?)
3. Find the middle of the list (length/2)
a) If the length is an odd number, then there's only 1 middle
b) If the length is an even number, then identify the middle 2 and get their average

## "Find the Median" Function

```
def median(alist):
    # Make a copy so we won't change "alist" itself
    copylist = alist
    copylist.sort() # guess what this does??
    if len(copylist)%2 == 0: # if length of list is even, identify the middle 2 numbers
            rightmiddle = len(copylist)//2
        leftmiddle = rightmiddle - 1
        median = (copylist[leftmiddle] + copylist[rightmiddle])/2
    else: # if length of list is odd, just find the middle number
        index_of_middle = len(copylist)//2
        median = copylist[index_of_middle]
    return median
```


## Measuring Statistical Dispersion

- How much do values vary from the average?
- Differences from mean: x[i] - mean(x)

$$
s d=\sqrt{\frac{\sum_{i=0}^{n-1}(x[i]-\operatorname{mean}(x))^{2}}{n-1}}
$$

- Includes positive and negative differences
- Take the square the difference: (x[i] - mean(x))**2
- Variance = sum of squared differences (for all i), divided by $n$ - 1
- Standard deviation $=$ square root of variance


## Let's Program This Using Lists!

## Tuples

- Another type of Python data structure
- Like a list, EXCEPT:
- It's immutable
- You cannot add elements to a tuple (they are a fixed size)
- Example: ('CS8', 125) is a tuple
- Note the use of ( ), instead of [ ]


## Tuples

- You can't add elements to a tuple.
- Tuples have no append method, like lists do.
- You can't remove elements from a tuple.
- Tuples have no remove or pop method.
- You can find elements in a tuple
- Using the index method (same as in lists)
- You can also use the in operator to check if an element exists in the tuple
- Again, same as in lists


## Dictionaries

- Popular data structures in Python
- Unordered associative collections
- Basically lists, but you can access each value by a key
instead of an index position
- Use curly braces, \{ \} to define a dictionary

$$
\text { ages }=\left\{\text { 'sam': 19, 'alice':20 }^{\prime}\right.
$$

NOTE THE SYNTAX
and the use of the colon
key:value

## Dictionaries - Key/Value Pairs

- Use the familiar [ ] to access, set or delete by key

```
ages['alice'] >>> 20
ages['pete'] = 24 # adds new item in this case
del(ages['pete']) # bye bye pete
```


## Let's try it!

- In Dictionaries, we don't use indexing like we did with lists
- That's because values are not stored in a discernible order
- How do find things in a dictionary? We go by the key.


## Useful Functions for Dictionaries

Assume: MyDict = \{'Britta':33, 'Annie':20, 'Jeff':42 \}

## Show all the keys

- MyDict.keys() = ['Britta', 'Annie', 'Jeff']

Show all the values

- MyDict.values() = [33, 20, 42]

FYI: Although these look like lists, they are actually different kinds of data types: dict_keys and dict_values

You can always try using type() to figure out the data type you're using!

## Another Useful Dictionary Function

Assume: MyDict = \{'Britta':33, 'Annie':20, 'Jeff':42 \}

Show all the items in the dictionary as a list of tuples

- MyDict.items() =
[('Britta', 33), ('Annie', 20), ('Jeff', 42)]


## Application Example: Finding the Mode

- Number that occurs most often within a set of numbers
- Example:

Consider the set of numbers: $1,3,2,3,5,1,6,1$ The mode is 1.

- Given a list nums $=[1,3,2,3,5,1,6,1]$, how do I find the mode?
- I'll have to make a count of all the elements
- The element with the highest count is the "mode"


## Find the Mode of a List: The Algorithm

## Simple (no coding detail) algorithm/plan:

We'll create a dictionary to store all the numbers in the list WITH their frequency counts (i.e. how often they appear):

- Go thru each number in the list
- Put it in the dictionary (as key) and mark the count (as value) as 1
- If you see that number again, increment the value
- When this is done, look at all the values you've collected and search for the BIGGEST one (why?)
- Now that you have the maximum value, look for the key that it's associated with THAT'S YOUR MODE! ©
- Careful: there may be cases where you have MORE than 1 mode!


## Finding The Mode Of A List

$$
\begin{aligned}
& \text { def mode(alist): \# see Listing } 4.6 \text { (and start of } 4.7 \text { ) } \\
& \text { countdict }=\{ \}
\end{aligned}
$$

for item in alist:
if item in countdict:

$$
\text { countdict[item] += } 1
$$

else:

$$
\text { countdict[item] }=1
$$

- Continued next slide


## Finding mode (cont.)

## Continued...

```
countlist = countdict.values()
maxcount = max(countlist)
modelist = [ ] # in case there is more than one
for item in countdict:
    if countdict[item] == maxcount:
        modelist.append(item)
```

    return modelist
    
## YOUR TO-DOs

[ Do Homework6 (due Monday 3/5)

- Turn in Lab4 today
- Get some sleep

