**Module 8 Notes (Lists)**

*Part 1*

Algorithms: set of rules or steps used to solve a problem

Data structures: A particular way of organizing data in a computer

Lists: the first and most simple data structure

-What is not a collection?

- Most variables have one value in them, when we put a new value in the variable, the old value is overwritten, not “collected” and stored

- A list of collections: allows us to put many values in a single variable

- Nice because we can carry all many values around in one convenient package.

- List Constants: Surrounded by square brackets and the elements in the list are separated by commas

- A list element can be any python object, even another list, brackets [] inside of brackets [ x[x]x ]

- A list can be empty

- list and definite loops are “best pals”: Going through each item in the list in order

- Looking inside list: Just like strings, we can get at any single element in a list using an index specified in square brackets

- list have positions (order) they start at zero and go up from there [x, y, z] x would be 0 y would be 1 z would be 2

- List are mutable:

- Strings are “immutable” or we cannot change the contents of a string, we must make a new string to make any change

- List are “mutable” or we can change an element of a list using the index operator (what position it is in the order of the list. i.e) if x = 22 in a list, and x is the 3rd position, you could change the value of x within the list by saying: list[3] = 25 and the value of x would change within the list

- You can tell the length, or number of elements within a list using the len() function

- Using the range function:

- The range function returns a list of numbers that range from zero to one less than the parameter, so if the parameter is 4, it will return zero through three

- We can construct an index loop using for and an integer iterator

- Using range and len together with your variable can be useful to go through a list in the order of the list i.e) for I in range(len(friends):

*Part 2*

-Concatenating list using +:

* We can create a new list by adding two existing list together

-List can be sliced using (:) :

- Just like a string, the second number is “up to but not including”

-Building a list from scratch:

- We can create an empty list and then add elements using the append method

- The list stays in order and new elements are added at the end of the list

- when making a list) stuff = list() this indicates to python your making a new list, the lists name being “stuff”

- stuff.append(x) append is how you add

-Asking if something is in a list:

- you can ask if it is in) 10 in 5, or ask not in, either way returns true of false

- does not modify list

- List are in order, and can hold many items and keep those items in the order until we do something to change that order.

- a list can be sorted, the sort() method means “sort yourself”

*Part 3*

-“Split” breaks a string into parts and produces a list of strings. We think of these as works, we can access a particular word or loop through all the words

- When you do not specify a delimiter, multiple spaces or tabs are treated like one delimiter

- You can specify what delimiter character to use in the splitting i.e) thing = line.split(‘ ; ’) would split when it ran into a semicolon

- useful when parsing things like emails

- Double split pattern:

- Sometimes we split a line one way, and then grab one of the pieces of the line and split that piece again, so you do you initial split and assign the values to a list, assign one of the values from the list to a variable, then you can do another split function to split from that new variable and create a new list based on whatever you specified for you delimiter in your split function i.e) you can split a email that was pulled from a list at the @ sign and break it into two parts and either have that new list or store one of the values from that new list into a new variable

*Strings,files, list, and the guardian pattern*

* You can basically protect code that may blow up by introducing if statements that look for possible things that may cause your search to fail, such as blank entries when looking for words in a line. You do this by introducing if statements before your main search i.e) if line == ‘ ‘ : continue would skip all lines that were blank and continue to the next line or you could do) if len(wrds) < 0: continue it would do the same thing. But to make it stronger you could say < 3 and then you know that you are only looking at lines where the data is reasonable
  + The other way to do it is with a compound statement, which is done in order i.e) if len(wds) < 3 or wds[0] != ‘from’
    - Order is important, if we were to flip that, it may not work
    - Your basically putting a guardian before your main search but putting it in one statement
* Add print statements before and after the line that blew up to debug