Week 11, Lecture 1

Today's topic is LIST COMPREHENSION.

Most of what we cover is described in the pdf given in this folder. It's best to start with the pdf and when you have finished reading it, then look at the examples.

Example 1:

First a very brief review of lists, and then some simple code that shows you how to implement a data structure called a STACK.

You can put ANY objects onto a STACK. But what is a stack? Think of a stack of plates. Say each plate is an item.

You "push" an item onto a stack s using a function: push(s, item). Notice that the item can only go on TOP of the stack.

You "pop" an item off a stack using a function pop(s). So x = pop(s) gets the item returned in variable x. This actually removes the item from the stack.

These are the two main stack functions. Other functions are possible, but not important. For example you can have a function top(s) that returns the value of the item at the top of the stack, but does not delete it. And another

function that tells you the size of the stack, I.e., how many items.

The stack is implemented using a list. So we are free to use list methods to help us.

https://isaaccomputerscience.org/concepts/dsa_datastruct_stack?examBoard=all&stage=all

Applications:

This simple stack data structure has MANY uses in computer science applications.

https://www.tutorialride.com/data-structures/applications-of-stack-in-data-structure.htm

Examples 2 and 3 start by defining LIST COMPREHENSION and proceed from simple examples to other examples. The examples are set-explanatory.

Example 4 shows you three "advanced" functions in Python. These are very useful functions and not difficult to master. They are NOT crucial — meaning that you can always write code without using these functions, but it will mean using loops. These advanced functions can make your code small and compact. Use them if you see a clear need.

The **OOP (object-oriented programming)** folder contains **EXAMPLE 5**, which is your very first look at how to construct your first Python object.

We read a list of lines from a file, where each line has: name, height, weight. [We assume height is in feet and weight is in pounds]. Each line represents a volunteer for a local basketball team.

1. Read in one line at a time
2. From each line construct a basket-ball player OBJECT.
3. In writing the CLASS from which INSTAN CES (i.e., OBJECTS) can be constructed, we also write METHODS that can be called on objects in this class.
4. Once the file is read, we have a LIST of all the volunteers, where each list item is a basket-ball player object.
5. We can't use all the volunteers, so we rank them in descending order based on (1) height, (2) weight and (3) height*weight. That is, 3 different rankings.
6. The point of doing the rankings is to show that the list sort() method can be used to sort objects based on any instance variables that we like, but we must pass to the sort() method a function name which it can use. This is called a "key", i.e., it does the sort based on a key, and we can choose the key.
7. You'll notice that when you think in terms of objects (meaning, you write classes and methods) your Python code starts to look so much more structured, simple and clean.