Lecture 7

Lists are mutable.

Ivys[x] = y

This wont generate an error iff the position x is occupied by some other element of the list but in case the list does not have anything under x then it will generate an error.

Ivys[1]=’Yale’ is fine only if Ivys[1] is something else before and we are simple removing that binding and binding it to something else now.

If L1 is a list and then we have a statement like L2 = L1 and we change some element of L1 then the element will change identically in L2 also. It is like L2 and L1 are pointing to the same object. It is like having two paths to the same object and we can refer to it or change it by using any one list.

This is not the case with immutable things like variables and strings.

**A new type: DICTIONARIES**

* Immutable as in lists
* They are mutable
* But elements are not ordered
* They have generalized Indexing
* Every element can be thought of as a <key> value pair
* and the key can be thought of as the index of an element

ETof {‘one’:1}

ETof[‘one’] is 1

NtoS = {‘one’:1,’two’:2}

dictionaries are implemented using a technique called HASHING. Which we will be seeing soon enough in the term.

##Over to Professor Grimson

Pseudo code=description of the steps, not in any particular language.

* get into the habit of a pseudo code
* use it to find the flow of control
* and what are the modules required and how the flow of control is in a program

**EFFICIENCY – Orders of growth**

* how to choose a good algorithm
* and how to map your problem into the selected class of algorithms

**How to measure efficiency?**

* space

how much memory do I need to complete a computation? – not our focus

* time

the question that we need to ask is – what is the number of basic steps needed as a function of the input size?

this removes the following specs:

* + machine specs
  + which programming language?

the assumptions:

* basic steps – built in primitives – arithmetics,comparisons
* we will be using the random access model, ie reaching any part of the memory is done in a constant time.

we have to count the number of basic steps as a function of the input given

* best case input – minimum
* worst case -maximum
* expected case – average time

we will focus on worst case:

* no surprises. we know the upper bound of time