Lecture 6

**SQUARE ROOT BY BISECTION METHOD**

Continuing on the square root using the bisection method. There is a bug in the code.

We generally like to test using all possible values. this is called regression testing.

In bisection method, we look for answers between 0 and the number itself. so we change the code, if we need the square root of fractions.

but there are methods which are better do exist.

**NEWTON RAPHSON METHOD**

while finding the square root, we are finding the root of: guess\*\*2-16=0

we take a guess and then find the tangent to the curve at that point. and the next guess is where this tangent cuts the x axis.

We need the derivative now, for the square root is 2\*x.

F(x)=x\*\*2-16

F’(guess)=guess\*2

guess(i+1)=guess(i)-(F(guess)/(2\*guess(i)))

This method does not work if the first guess is 0.

As the problems get complex the newton raphson is much better than the bisection method.

**Most important lesson we will talk about the whole semester:**

**ANSWERS CAN BE WRONG**

**When you get an answer from the computer ask yourself why I should believe the answer that has been given by the computer.**

Non scalar data types:

* tuples – written with ()
* strings – written with single quotes ‘ ‘

Both of these are immutable. That is these can’t be changed.

We will look at some that are mutable. Values in a list need not be characters. They can be integers, characters, even other lists.

**LISTS**:

They are written with square brackets. [ ]

to append two lists we use: univs.append(techs) where techs is also a list. then the list univs has one list as an element. here append is a method(like in classes).

And if we write code to again append another list then this list has two lists as its elements.

Flattening the list: We can use the + sign to simply add the strings and assign these strings as elements to the list and not the lists as elements.

Techs = [‘MIT’,’CalTech’]

Ivys = [‘Harvard’,’Yale’,’Brown’]

Univs.append(Techs)

Univs.append(Ivys)

now Univs looks like this:

Univs = [[‘MIT’,’CalTech’], [‘Harvard’,’Yale’,’Brown’]]

a statement like the following:

for e in Univs:

print e

for c in e:

print c

now e is a list of the list Univs. First it is Techs. So it will print the List Techs.

The nested loop will print the elements in the list Techs as strings.

And so on for the list Ivys.

When we remove something from a list what we do is mutation. As against slicing where we took a string and made a changed copy of it. in lists we just took the list and changed it at the basic level.