Computational Science and Engineering using Python

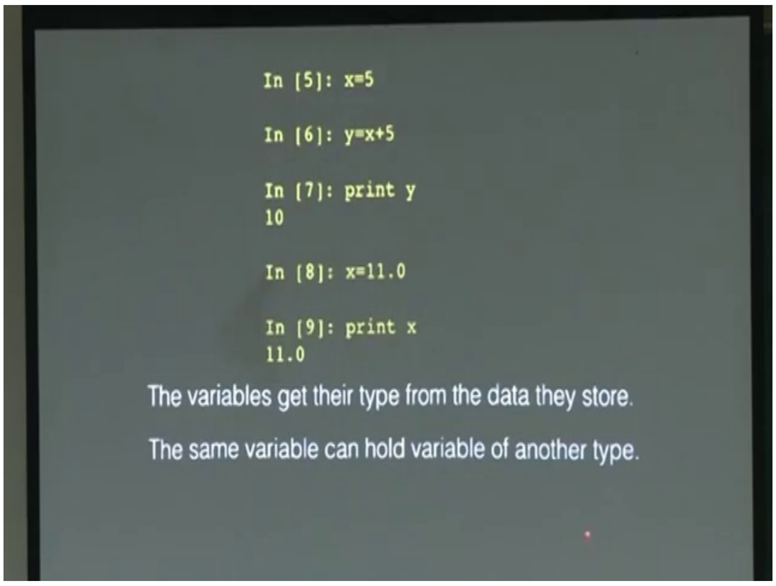
Prof. Mahendra K. Verma
Department of Physics
Indian Institute of Technology, Kanpur

Lecture - 04
Python: Control Structures

So far we discussed how to deal with data; including arrays strings. But now, we can write state of set of lines; basically write programs; how to deal with this, how to manipulate this variables.

So, there will be four types; I will cover four types of programs. So, the first simplest of them is called simple statements.

(Refer Slide Time: 00:50)



```
In [5]: x=5

In [6]: y=x+5

In [7]: print y
10

In [8]: x=11.0

In [9]: print x
11.0
```

The variables get their type from the data they store.

The same variable can hold variable of another type.

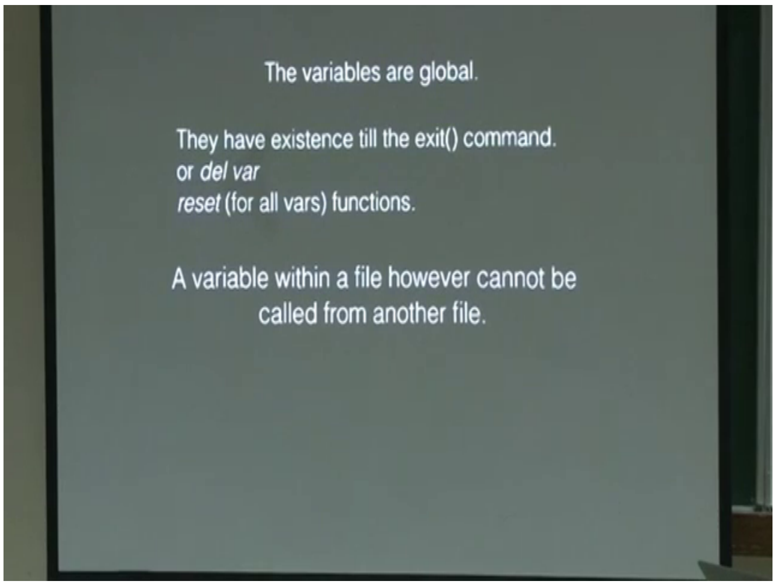
So, this is a set of statements; so, each line which I type is a statement. So, x is equal to 5, y is equal to x plus 5. So, this is a statement; I could have put them in one file, I will show you one example. I mean you can just put them in one file and can run that file name; so maybe I should just run it once; just to give you. So, you can put set of lines and all of them will run in one shot. So, if I say print y; it will give you 10; so, these are set of lines, so this is simple statements; each of them is a simple statement.

Now, few key points which you need to know; that the variables get the data type. So, every variable has a data type like float, integer or string; from the data they store, it is not a strict typing in python. Notice one more thing that x is what is type of x here?

Student: (Refer Time: 01:42).

Integer and here it is float, so I change the type of x; it has become float. It could also become an array; so a c will not allow this, but python will happily do this. So, same variable can hold data of another type like example given here

(Refer Slide Time: 02:02)



The variables are global.

They have existence till the exit() command.
or *del var*
reset (for all vars) functions.

A variable within a file however cannot be called from another file.

Variables are global; that means, it can be accessed by any us; any line or any set of program within that file. So, this is useful as well as dangerous; basically speaking about the variable. So, do not well I am right now doing example; where I could add x, y, a, b; but you should give names, specific names. Otherwise you may think that my value of x is something; but it was something else because it was sometimes the x is 3 x, then I say x is equal to 5.

So, you should give for example, temperature must be temperature of today or today's temperature; that is the name of the variable. And then, we will not mix with today's pressure, so pressure will be; you should not say x for temperature, y for pressure; you just write pressure. So, variables are global and they have existence till the program quits, but we can delete a variable; this operation called del. So, if you say del variable name; it will delete, it just goes away.

We can also reset all the variables means it will become all empty no variable will look present and you can use this function called reset which will reset all the variables. Actually, if I using multiple files which you will be doing later in the course then variables from one file cannot be accessed by another file, you have to be careful.

Student: (Refer Time: 03:28).

Anything, so this var can be integer so.

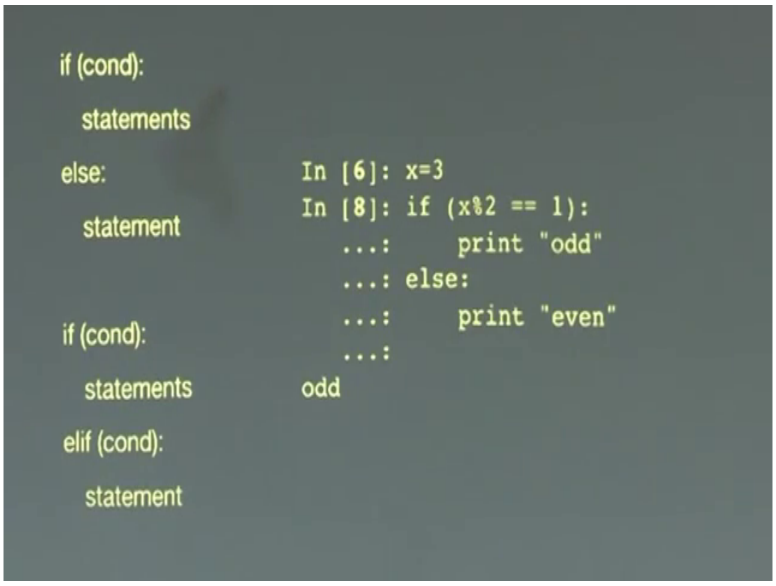
Student: (Refer Time: 03:35).

I mean all var, so, I wrote x is equal to something, y is equal to something, z all that; I have lot of variables. I want to start from scratch; reset means just delete everything.

Student: (Refer Time: 03:52).

No; variables, for all variables. It resets all variables; no I am not this is nothing do not worry about. So, it is a reset; just say reset and it will reset everything means no variable will be there inside the memory; so this one has to be careful. Now, we need another second class of statements called branching statements. So, everybody knows I mean if today is warm, then I should not wear sweater if then; you need to take decision.

(Refer Slide Time: 04:24)



```
if (cond):
  statements
else:
  statement

if (cond):
  statements
elif (cond):
  statement
```

```
In [6]: x=3
In [8]: if (x%2 == 1):
   ...:     print "odd"
   ...: else:
   ...:     print "even"
   ...:
odd
```

So, the statements look like this; so, if condition colon this is a very important; this colon. Now without colon this will throw up error and most of the time, in the beginning everybody makes a mistake that colon is missing. So, I will this program should work, but this is not working. So, the colon is a common error; then you should put some blank.

There are four blanks; not some actually four blanks. So, this is the tab this is called indentation; otherwise python will not understand what you have written. So, this is; so if condition and else between that it will execute set of lines, it could not be one line; it could be set of lines and all the lines should be shifted by four spaces, this is called indentation; indent, shift it.

And if you do not indent it; the code will not. So, this one of the errors will find start having in your first few homeworks. If you do not indent it, it is not going to work; then put else and colon again then other statement; other set of statements. So, these are simple statements here which are usually; it closer branch, this can also; so this is set of statements, I will not say branch or something.

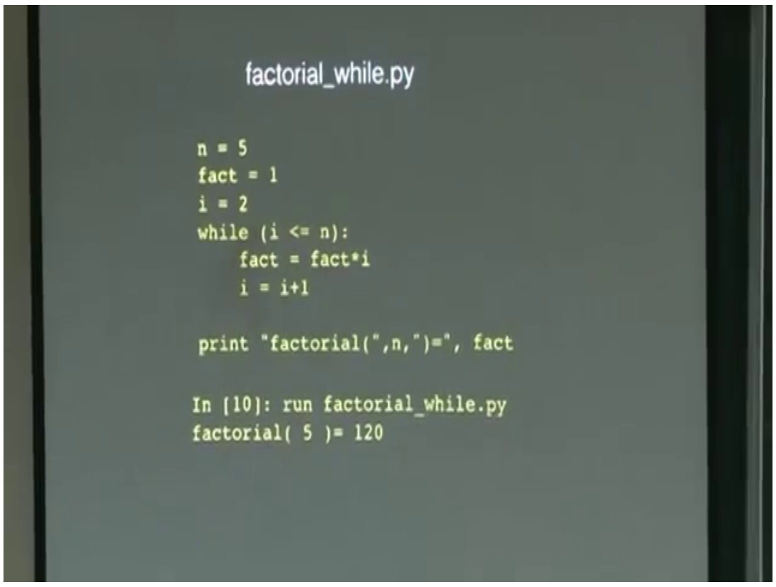
So, it is set of statement or you could have; so, this is one kind of conditional statement you may encounter. Else means anything other than the condition, but I say well if this condition is true then it should do this or some other condition is true then do this. Then you can put a another else if; so elif is called else if; else if. Python has it; this is the syntax elif; elif. So, I can continue and cover all of it, so this is for branching statement; so if else, if else like that.

So, simple example is this; so, this is written in the interpreter itself; I did not create a file. So, if I do not create a file; so, these are four blanks here, you can see this 1, 2, 3, 4; four blanks; you can see that? So, if x is divisible by 2 so reminder of x; this is reminder know reminder function. If reminder of x is equal to 1; then odd else print even. So, whatever x is there at that time in the program or in this memory; if x was not declared, it will throw up an error.

So, suppose x is already known; x is 3; so, it will tell you whether x is even or odd. So, these four lines will do the job and answer is that x is 3 and it is odd. So, answer is written here the print always gives you it does not come without; it just outputs in the first column. So, this is the example of a branching statement, so this colon is important and this colon is important. You will encounter lot of it for example; I want to find minimum number within a set of numbers.

So, you will have to use if this condition; so, you have to write this function how to write like homework has all that; how to find minimum? So, the third class of repetitive statements very useful; it is called loop. So, this is the example so, I have written a file called factorial while py; so, I am going to run it after this.

(Refer Slide Time: 07:46)



factorial_while.py

```
n = 5
fact = 1
i = 2
while (i <= n):
    fact = fact*i
    i = i+1

print "factorial(",n,")=", fact
```

```
In [10]: run factorial_while.py
factorial( 5 )= 120
```

So, let us first just understand this whole thing is a file; inside a file it is written. Set of simple statements these three and this is a loop statement. So, let us understand what is this doing; so, n is 5; so, this variable n contains 5. So, I initiate or basically I create a variable fact which is basically factorial equal to 1; initialize it. My answer will be in fact; so, I create another variable i equal to 2; I do not want to say 1 because I want to multiply 2 onwards.

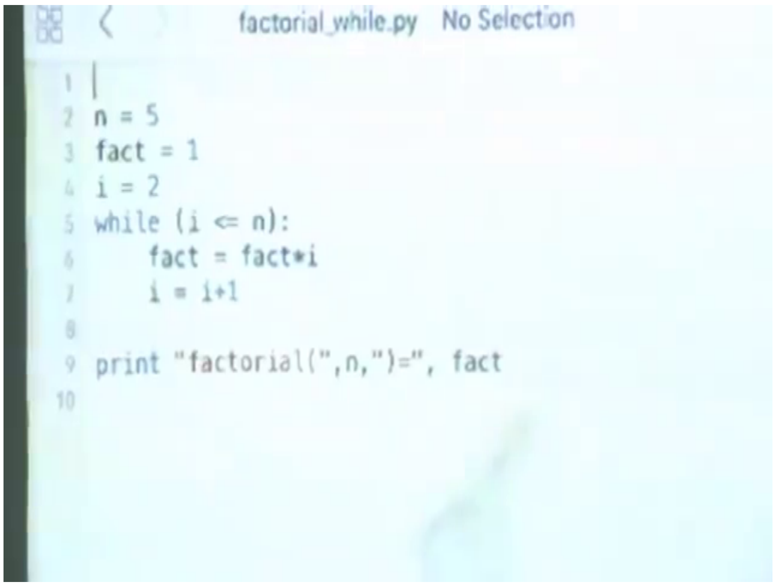
So, 2, 3, 4, 5 up to n; I just multiply in a loop; so, what will I do? So, I am going to increment i each time and I will check whether i is less than n. So, this increment is going happening here and I check whether i is less than n, equal to n. If it is less than or equal to n, then multiply the factorial by its old value. So, it is getting; so, we start with 1 then 2 then 2 multiplied by 3; multiplied by 4; so, keep multiplying; that is straight forward.

So, I am getting a product till n and that is answer once you reach n and so, fact is the global variable it is known till here. So, I say print factorial n; so, this is the print statement I am using. This is string, this answer fact; so, this is a file I want to run that file; what should I do? So, run the file name; the file name should have dot py in the end, otherwise python interpreter will not know what kind of file is it and also good idea you know you have c functions or c files or c plus plus files; photo run files.

So, each of them should have unique name and python files should have unique name dot py; unique extension. So, dot py means it is a python file, so answer is printed here once I run it. So, I will just show you this file because you need to know this for your class.

So, this is a python file; it is created in some editor; so that is why you need editor. So, the editor is up to you if you use Mac, you can use x code.

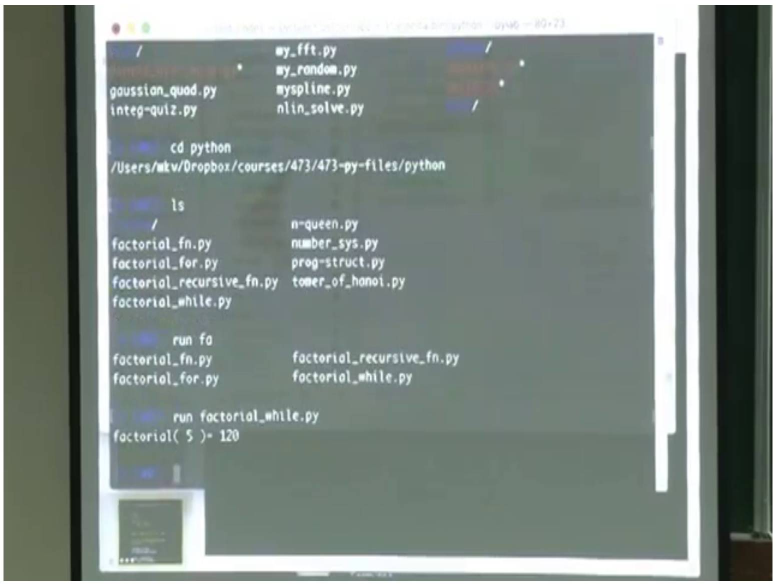
(Refer Slide Time: 10:12)



```

n = 5
fact = 1
i = 2
while (i <= n):
    fact = fact*i
    i = i+1

print "factorial(",n,")=", fact

```

If you use something else; g edit is what is in next which is convenient; very easy, so this is set of lines exactly same what I have in the keynote. So, there is indentation; this 4 is important this four is spacious here; if you do not have it, it is going to crash. You need to create this in editor; so I go to that python interpreter.

(Refer Slide Time: 10:33)



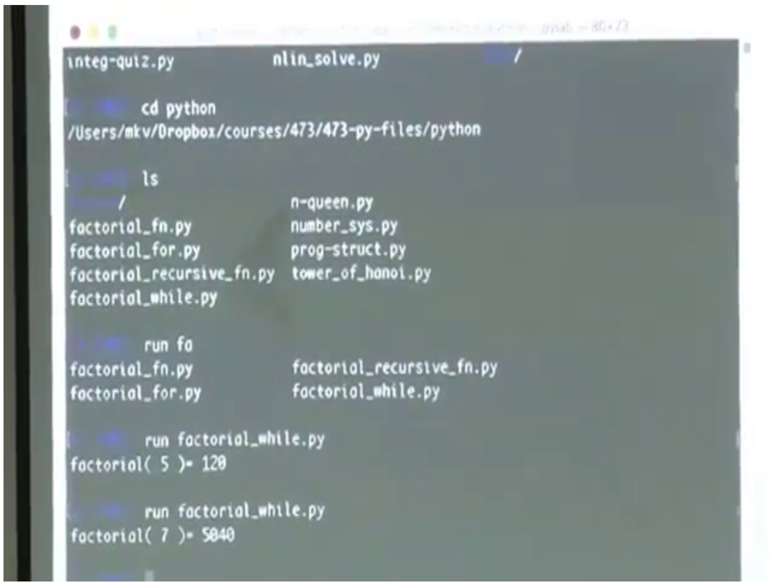
```
    /                      my_fft.py                    /
                        *  my_random.py                 *
gaussian_quad.py           myspline.py                *
integ-quiz.py              nlin_solve.py              /

        cd python
/Users/mkv/Dropbox/courses/473/473-py-files/python

        ls
     /                        n-queen.py
factorial_fn.py               number_sys.py
factorial_for.py              prog-struct.py
factorial_recursive_fn.py     tower_of_hanoi.py
factorial_while.py

        run fa
factorial_fn.py               factorial_recursive_fn.py
factorial_for.py              factorial_while.py

        run factorial_while.py
factorial( 5 )= 120
```

So, I have to be in the right directory. So, the file should be in the same directory; presently I am not in the same directory. So, I can check whether which directory I am in that is why you need to know this Unix commands; so I need to know what directory I am in PWD. So, this is not the right directory, so this is in python files.

So, I can say ls; ls is a list cd, it is not very clearly visible, but it is good for the recording that is why I put the black screen. So, here there is a file cd is change directory so that file is here; factorial while py. Now, I can say run factorial while dot py; so, you have to type it. So, this is called tab if you know this tab will just do this. So, if I run it; I will get the answer.

So, this is how you run a file; you create a file and run dot py will work. There are more ways to do it, but just stick to one when we run file name. If I want to change to some other n; what should I do? I should just go to the file and instead of n, I can just put 7; save it. So, I save, I buy some commands; you must save it and then come back and run again.

(Refer Slide Time: 11:59)



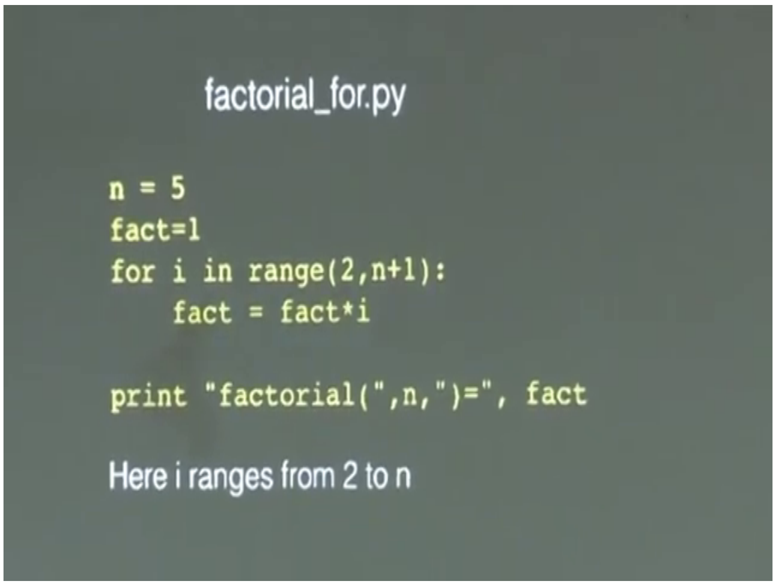
```
integ-quiz.py          nlin_solve.py          /

      cd python
/Users/mkv/Dropbox/courses/473/473-py-files/python

      ls
     /                         n-queen.py
factorial_fn.py                number_sys.py
factorial_for.py               prog-struct.py
factorial_recursive_fn.py      tower_of_hanoi.py
factorial_while.py

      run fa
factorial_fn.py                factorial_recursive_fn.py
factorial_for.py               factorial_while.py

      run factorial_while.py
factorial( 5 )= 120

      run factorial_while.py
factorial( 7 )= 5040
```

So, fact to 7; so, you do not need to retype; if you had to type it then it is like it is not a good idea; you should type all of it again. So, save it in a file just change in; it will work;

Student: (Refer Time: 12:09).

You can do that precisely; so if in file then you can say input n; what I should told you about input function that is good idea. So, you should input n that is another way to do it. So, let get back to our presentation; is it clear how to get it work with files. So, this is for factorial using while loop, but there are several other loops. So, I like while and for so that is called; so, if I know the number of index, number of times I have to iterate; then I can use a something called for loop. There is one more just wanted to say that this indentation is critical, so this is called for loop which is simpler than while loop.

(Refer Slide Time: 12:51)



factorial_for.py

```
n = 5
fact=1
for i in range(2,n+1):
    fact = fact*i

print "factorial(",n,")=", fact
```

Here i ranges from 2 to n

Now, suppose I know how many times I have to iterate then just use for loop; if I do not know till what time I need to continue; then I should use while loop. For this example this is a trivial example both are equivalent, but for testing minimum rather to find a minimum value in a array, you do not need to go all the way till you find them well; minimum we need to find sorry, for minimum we need to go all the way; to find a minimum number.

Student: (Refer Time: 13:20).

No, it will not take n plus 1.

Student: (Refer Time: 13:24).

2 to n; that is a range function know; so, what I had said range is value of the wall; it is not value of the element. So, the wall is at n plus 1; not the element. So, the range will go from 2 to n; sorry the i will go from 2 to n. And this will work; for i is the local; well i is the variable; which will be in this range. So, i will be; in fact, as a list this is the list whose values are from 2 to n and you multiply again; so this is the same idea, instead of while loop I am using for loop and print. So, if I run it again; I will get the same answer.

So, few things I will just like to remark here. So, this i is a; I cannot use i outside. If I say after this; if I just write this and try to print i; (Refer Time: 14:39) will it print? I am confused now; this is it; so, i will come. So, they see it will not come, but here it will come. So, ignore what I wanted to say; this i will, I believe will be 5; when it comes here.

Student: (Refer Time: 14:57).

I will increment automatically in for loop; you do not need to do it yourself, i will increment. So, one homework inside is print i; what to see what you get.

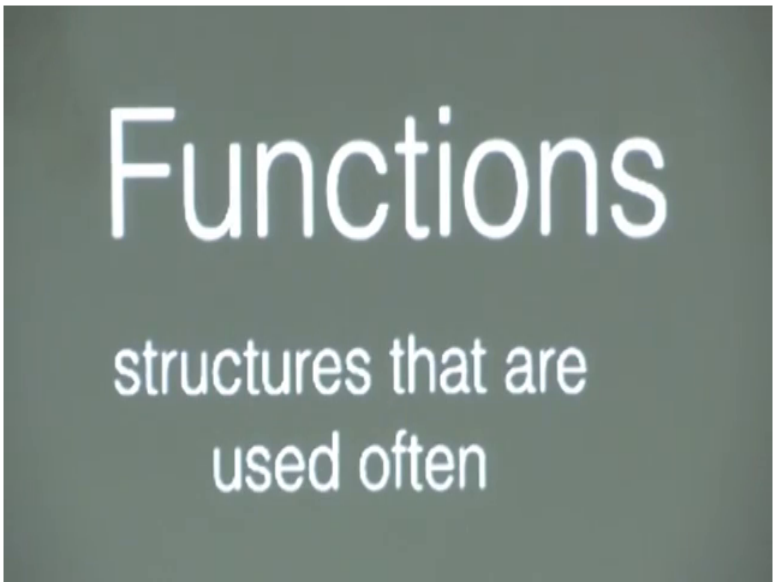
So, when do I use for and when do I use while? So, example I was trying to give; if I am trying to do is find a minimum number in a list then I have to check all the numbers. I say the first one is minimum then see whether the second one is smaller than the first. So, I have to go all the way, so there for is good, but if I am searching for a number then should I use for or while. As soon as I get the number; I should get out, I should not be doing it again after that. So, then while is good; so, you have to choose your loop properly.

Now, next and slightly more complex structure is function. So, what all I have done: simple statements, branching statements, repetitive statements. Now I am going to do functions.

Student: Sir.

Yeah.

(Refer Slide Time: 16:04)



Functions

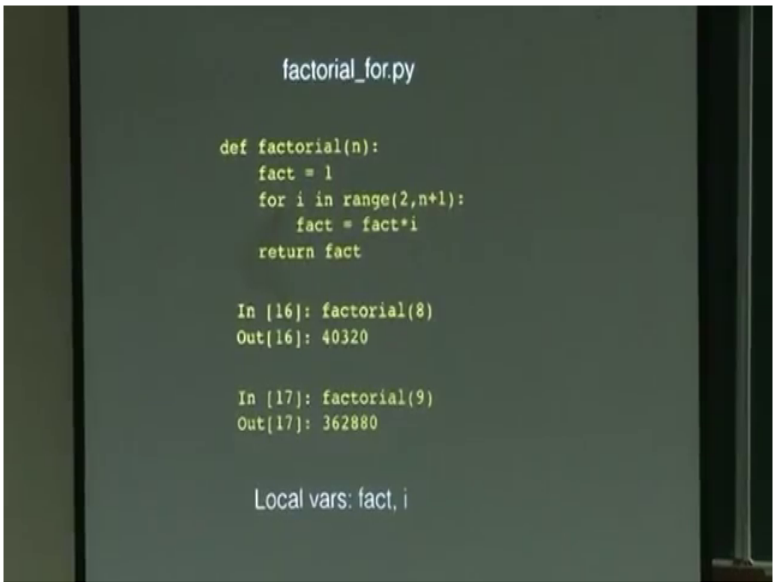
structures that are
used often

Student: (Refer Time: 16:07).

So, range has three argument array; so, I can say one; n plus 1 comma 2. So, it will increment by 1, so range has three arguments increment is the other third argument.

So function; so, suppose something has to be done again and again like factorial or sin function. So, you do not want to write set of lines, in fact you can encapsulate it or rather you can make a box. So, this will always give me my desired result. So, sin factorial; so, it will come from that function in one line. So, you should know how to write a function; example I said is factorial; can be a function itself. A sin function or minimum of an array, so you should give arrays and argument to this object to the box and should just spit out this is the minimum.

(Refer Slide Time: 17:12)



factorial_for.py

```
def factorial(n):
    fact = 1
    for i in range(2,n+1):
        fact = fact*i
    return fact

In [16]: factorial(8)
Out[16]: 40320

In [17]: factorial(9)
Out[17]: 362880
```

Local vars: fact, i

So, let us again write this factorial function, now I am going to write a function whose input is this n; for what factorial you want? For which number? And I will give that number; I will give you the answer. So, this is how you write a function, so the first word is def. So, def is for definition; so this is a definition of a function factorial. So, name of the function is factorial and it takes an argument n, so this is called argument.

So, f of x; x is an argument, now within this I have this called fact is equal to 1. So, this is a variable; so, this inside the function; fact equal to 1, then I loop; i goes from. So, it is a same for loop; to range 2 n plus 1 and fact is equal to fact star i and return fact; this is important. So, it will return a number, a function of course, you would like it to return, but sometimes we do not want it to return, you just say well print something or do something and get out.

But here I want it to give a value and the value it should return is a factorial and it is stored in fact. So, it will return this fact so these are simple example for a function called factorial and to call it is very easy. So, this is name of the function; I give a number 8; so, these are integer; 8 it will give me a number. I want to say factorial something else factorial 9; it gives me a number. So, this is very easy in fact, you can; so, the home work has some things are we just do it as write it as a program.

But then I wanted you to write like finding a minimum of an array, you can write as a loop, but then you write as a function as well. So, there are programs which may not be very clearly my homework statement will not be very clear, but that is what I want you should will right function as well as loop or statements.

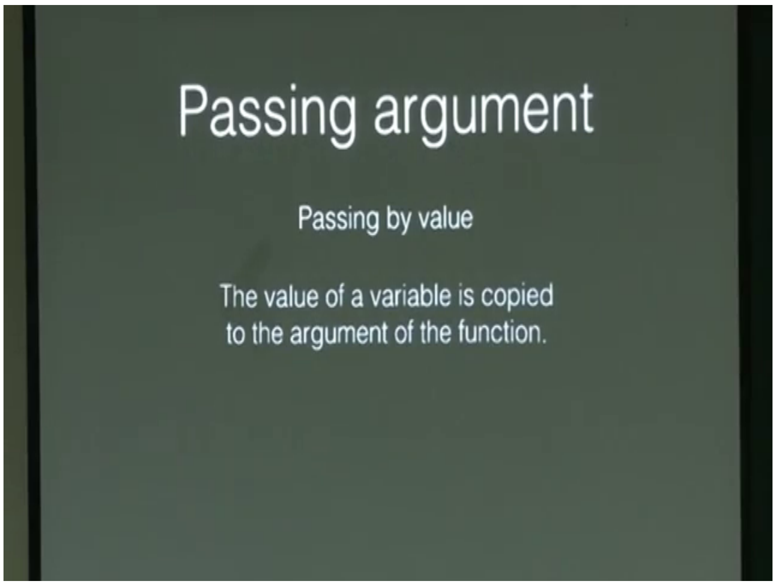
Student: Sir (Refer Time: 19:19).

Not necessary, it could do something inside a print for example, it will just print and return. Now, few important points suppose I say print fact; will it be visible? So, please imagine that this is I wrote in my interpreter and I ask for this fact; will it be visible? No fact is not visible outside. So, there is a nice thing about function, it hides some variables it uses; so, fact is called local variable.

So, we can run it actually I can just show you that this is not homework. So, this is also a file; so, factorial function. So, this is a function I just had written; now I will run function dot py. So, it gives the answer because if it can remember fact of something else; so, reset. So, this is what I meant by reset; reset everything, now I run factorial. Now I say print fact because fact is called local variable which is inside the function.

So, it is not visible to the program; in fact, it is a good thing why should many variables should be available in the memory for the user? I can track keep track of five things not thousands of things. So, that is idea; so, you want to hide them; so, factorial these are called local variables this is a standard thing in python. So, these are local variables; which are not visible outside.

(Refer Slide Time: 21:33)



Passing argument

Passing by value

The value of a variable is copied to the argument of the function.

So, how do pass arguments and important for every language has his own passing argument.

Student: (Refer Time: 21:40).

So no, the argument is in the whatever bracket; whatever you want factorial bracket you put the value. Factorial 100 you want; put 100 there, it may become very large integer.

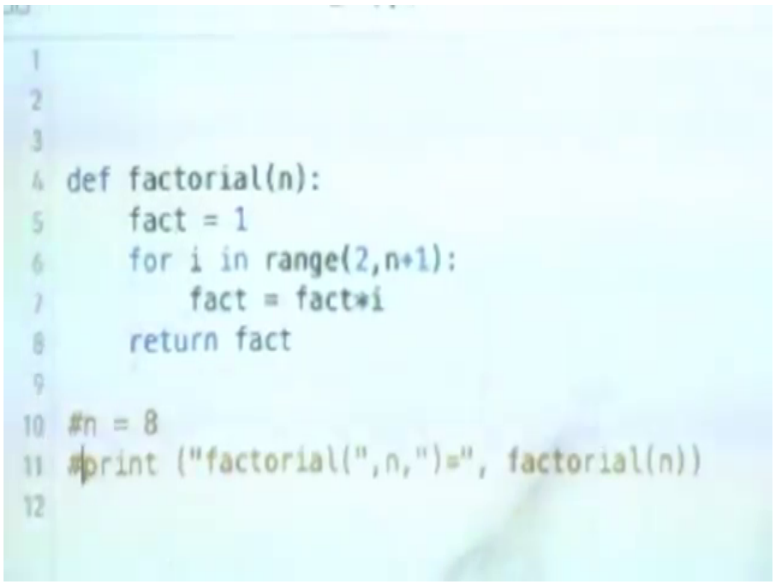
Student: (Refer Time: 22:00).

No, I am not. So, definition let us look at it. This n is a variable.

Student: (Refer Time: 22:14).

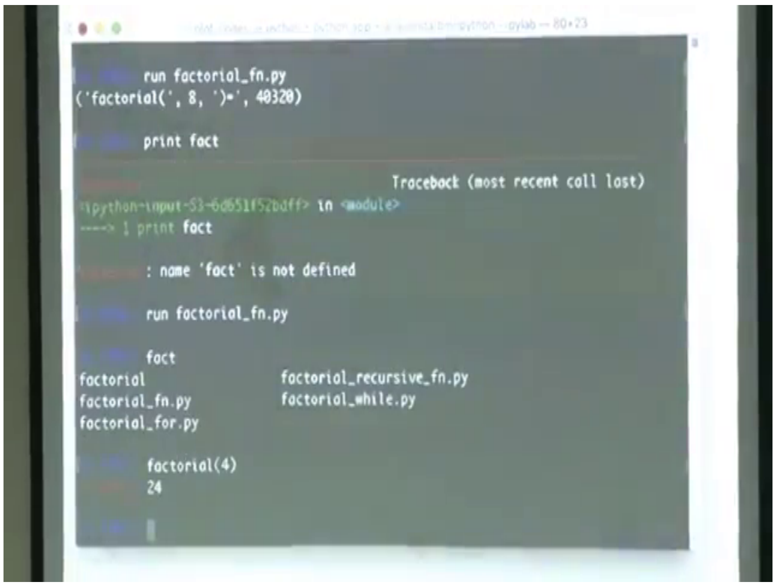
No actually no here; so, I get your point. So, actually now I will see what he is after. So, I should delete that those comment these line; factorial while inside the function I had add two lines here below that is what mid.

(Refer Slide Time: 22:44)



```
def factorial(n):
    fact = 1
    for i in range(2,n+1):
        fact = fact*i
    return fact

#n = 8
#print ("factorial(",n,")=", factorial(n))
```

(Refer Slide Time: 22:59)



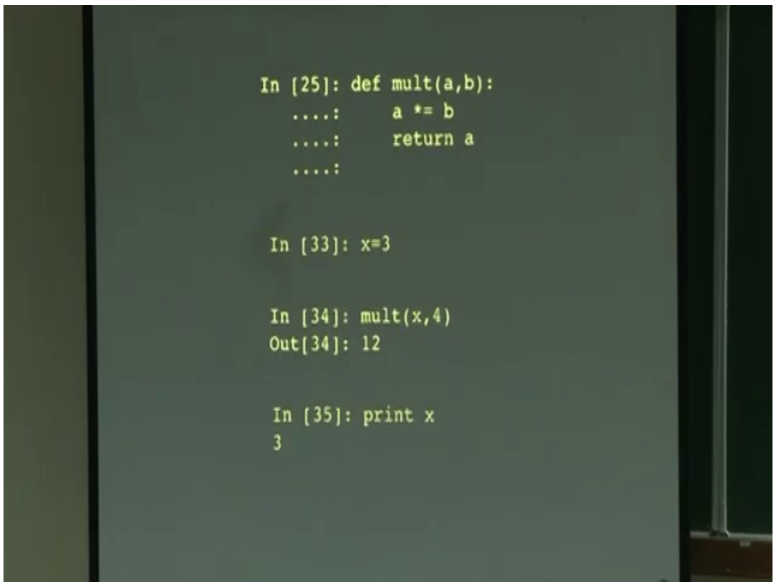
```
        run factorial_fn.py
('factorial(', 8, ')=', 40320)

        print fact
                                         Traceback (most recent call last)
<ipython-input-53-6d651f52bdff> in <module>
----> 1 print fact

        : name 'fact' is not defined

        run factorial_fn.py

        fact
factorial                 factorial_recursive_fn.py
factorial_fn.py           factorial_while.py
factorial_for.py

        factorial(4)
        24
```

So, if I comment them comment is by this hash no; so, this comment. So, it will not going to run; now I should run again. So, it is nothing is coming; now factorial function is; however, there in my memory function. If I say factorial now 4 will come; so, it runs it and it just gives it 24.

So, let us quickly just finish this aspect; so, passing argument. So, it has to be completely clear to you how are you passing the argument. So, in python is passed by value; the integer in float or string is passed by value. So, what is mean by pass a value means; it means something is a standard thing in programming languages; in computer science.

So, when you pass a value; the value of a variable is copied into the argument of the function. So, factorial n was there; n was the argument, so when I say factorial 8; 8 is copied there to n and you get an answer. But I could also have said factorial x and x had a value; so, let us say x was 8; x is equal to 8; factorial bracket x. Then you do something and you get out, so will x be modified or not? What I passed. So, remember there are two things one; so, I will give you an example. So, what you pass is not changed that is called call by value; I just put the value and let it go, where variable being passed remains unchanged.

(Refer Slide Time: 24:32)



```
In [25]: def mult(a,b):
   ....:     a *= b
   ....:     return a
   ....:


In [33]: x=3

In [34]: mult(x,4)
Out[34]: 12

In [35]: print x
3
```

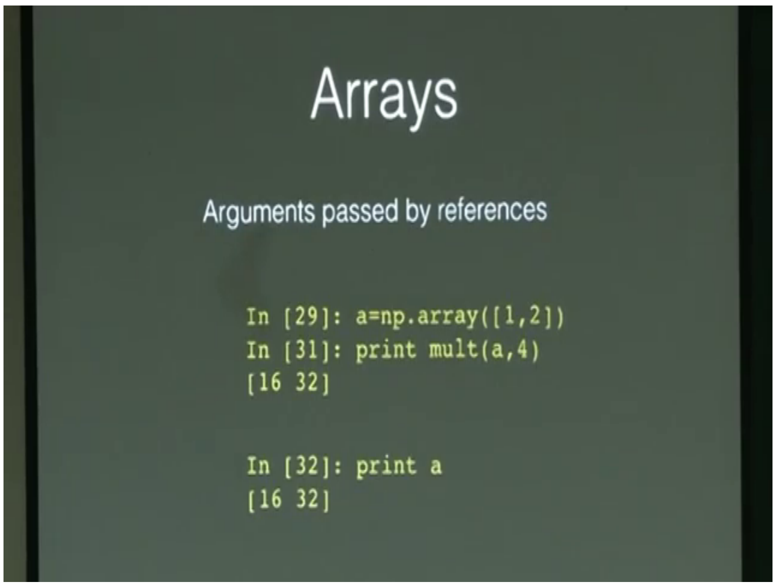
So, let us do an example; so, I do a multiplication function called mult a comma b. So, it has two arguments so, but what you will get is. So, this is a another useful function; a star is equal to b; what does that mean? This is equivalent to a equal to a star b; if my variable name is very large then I save typing. Suppose it was pressured equal to pressure star b; so, I would have said pressure star equal to b. So, return a; so, it will return the modified a, a is modified and the value a star b.

So, now I have x equal to 3 as variable now I say mult x comma 4, so 3 goes here multiply and the answer is 12; it returns 12, but what is the value of x after this?

Student: (Refer Time: 25:18).

It remains as 3; it is not that x is copied here, x is passed here; it not passed here, x is copied there. So, this x does not mean this x really; so, you have to be this x means I just copied this x into variable a; inside this function. So, print x is 3 what about if I pass arrays? I could pass instead of a; in fact, this function this is a beauty of python, you do not specify the type of the variable here; I could just pass array into this; a could be an array.

(Refer Slide Time: 25:56)



Arrays

Arguments passed by references

```
In [29]: a=np.array([1,2])
In [31]: print mult(a,4)
[16 32]

In [32]: print a
[16 32]
```

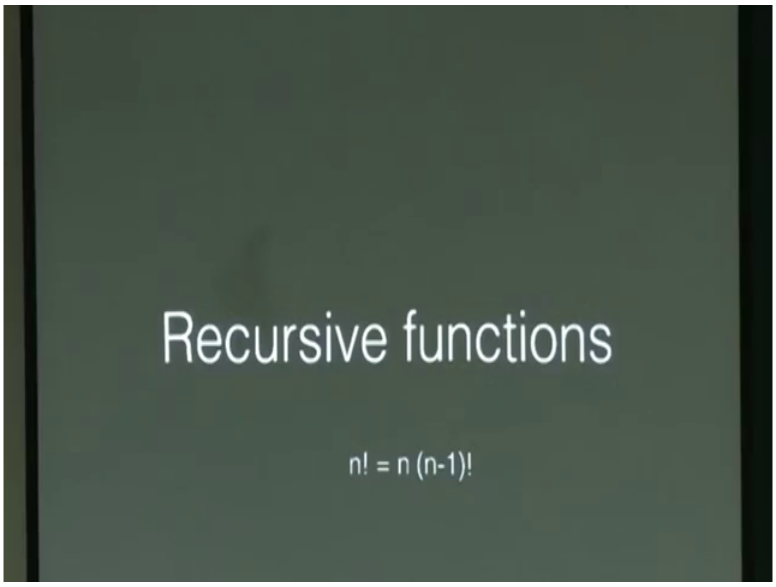
So, for array passed by references or what do I mean by this? So, array is not copied; the same arrays address is passed; the first. So, basically big arrays will be very expensive; I may deal with 100 by 100 matrix. So, 10 power 4 elements; copying will be expensive for memory as well as computer time. So, arrays are not copied neither in c; it is not copied in c.

So, you pass that reference and it is going to modify it because I am basically going to pass the same array. So, that is not copying there; it is passed by reference. So, I have an array a; which is a 1 comma 2; now I multiply this array a is passed here is argument; same function, mult function. So, what is the answer? So, multiply by 4.

Student: (Refer Time: 26:47).

So, 4 into; so, it is multiplying; so, it is giving you 16; I did a copying something else. It should be multiplied by 4; so it should be 4 comma 8; I get the answer, but what is the value of a after this? So, actually it should be 2 comma 4. So, the array has been modified; it is unlike the last example where x was unchanged, here array is modified. So, this has to be you have to keep in mind when you use function.

(Refer Slide Time: 27:22)

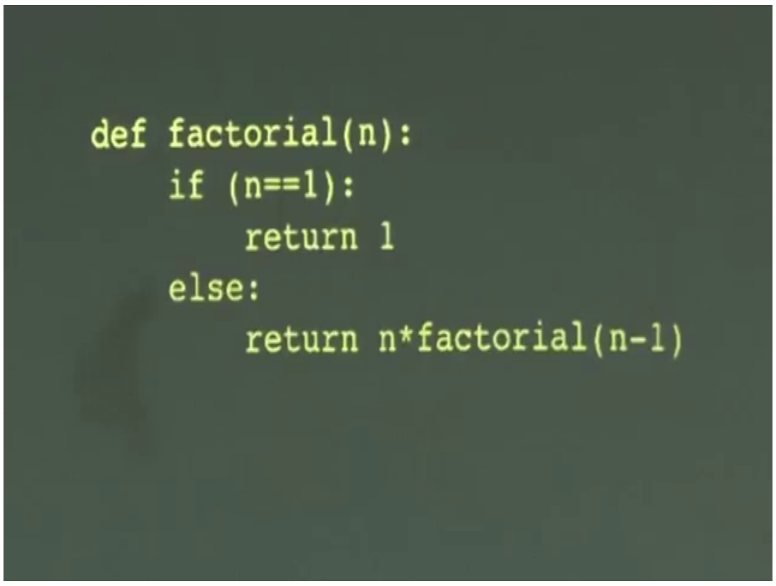


Recursive functions

n! = n (n-1)!

So, this is a very useful function; not useful for this course interesting way to solve problem using recursive. So, definition of factorial can be written like this, in fact, in math books you write like factorial n equal to n minus 1 factorial. So, this is called recursion; you define a function using itself is very useful idea in nature like factors. Factors are defined by itself, so this is allowed in python; this kind of function definition. So, I can define the same factorial function using this definition, so it looks like this.

(Refer Slide Time: 27:53)



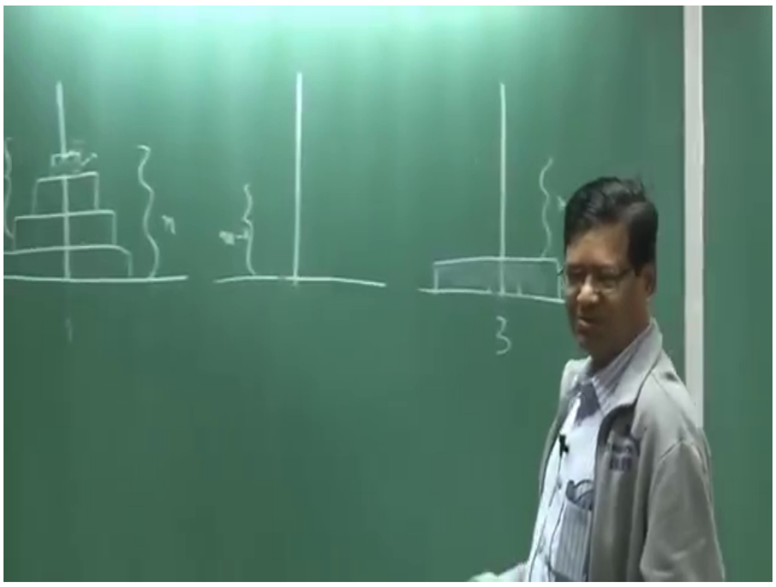
```
def factorial(n):
    if (n==1):
        return 1
    else:
        return n*factorial(n-1)
```

So, definition factorial n will be; so this n into factorial n minus 1. However, if I just write this is going to infinite loop. So, how does it compute? How does it; in fact, this will work this python it works; if I want factorial 4, it goes here; it says 4 into factorial 3. So, it comes here and calls again; it calls the same function. So, factorial 3 will be; factor 3 is equal to 3 into factorial 2, goes back 2 is factorial 2 into factorial 1. 1 will be 1 into minus 1, so it will go into infinite loop; unless I close it.

So, closing is that when n is equal to 1; I should return 1. So, as soon as it hits 1; it knows that bottom up. So, I should not go anymore down and starts going back after that. So, basically go down till that end and then come back and start multiplying; now you know the answer what is factorial 1. And you know factorial 2, factorial 3, factorial 4; answer is 1. So, this how is going to execute; this is more expensive because you have to go down and go up. So, I was on this problem Tower of Hanoi; so, how to solve this problem?

(Refer Slide Time: 29:10)

(Refer Slide Time: 29:18)



So, everybody knows the problem; so, there are 3 picks and each well the first one has a set of disk. Now, I have to transfer this disk from pick 1 to pick 3, but the rule is that smaller disk cannot be below the larger disk. So, how shall we do it; so, some I mean this is well known problem yes. So, I take; so, my objective is to send it to third. So, I take this; so, this one I do it here.

Student: (Refer Time: 30:18).

I see; so, you are getting it, but if I wanted to tell the computer. So, if you want to tell this to computer; how should I do it? I want to write a program. So, if I write that many lines; so, do this; so that is not a good program. So, if I tell this full solution it is not a good programmer; when the computer is not doing in few lines; it should also write. So, when objective of a code is that it should be short; as short as possible

Student: (Refer Time: 30:54).

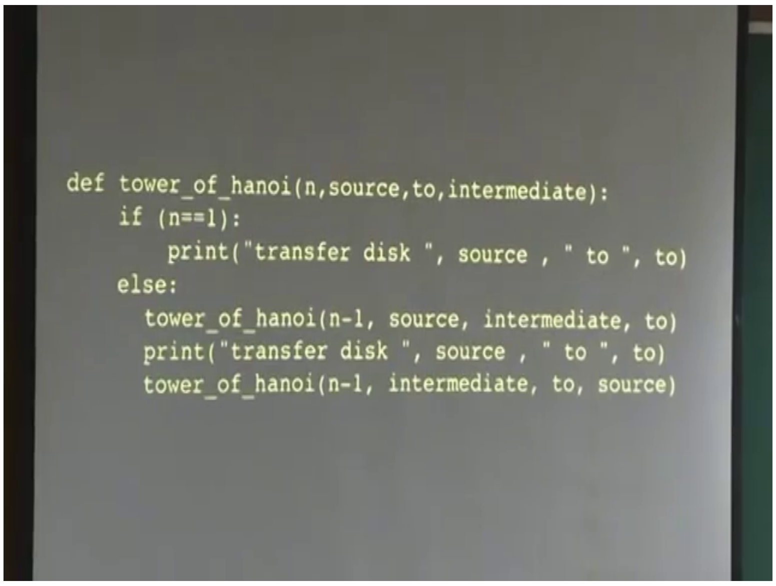
And it should be efficient; it should take least amount of time. So, I will discuss that in a while about efficiency, but you have to do better than this. So, you know how to do better I think Dinesh you know how to do it; no someone else? I have put in a code; well I have sent you the files you shall be able to see. I would say think recursively, this was not done in your scope course no done.

So, the idea is; so basically I want to transfer everything here. So, my objective is to write a function which will take n minus. So, what I should do is, I should basically say well if I can push it push n minus 1 disk here. So, this is n disk always write use n it is general do not say 3, 4. So, idea is to take n minus 1 of these and push it to intermediate. So, I have a function which can do it; suppose you are the function.

So, I send n minus 1 here n minus 1 disks here; the function which you have will do the job. Then the bottom one; I send here the bottom one here and then take this n minus 1 and put on top. So, n minus 1 top; so I transferred everything. So, if you know how to solve for n then you also solve n minus 1 and I can use it recursively. So, the way you do for factorial, so compute factorial 4; I compute fact. Well I suppose somebody give me factorial 3; then I can always compute factorial 4. So, that was the definition of recursion; so, solving it recursively; so, and is easy to code; once you know the idea.

So, what shall we do? So, is there in the code python can do it easily few lines; so, these are code.

(Refer Slide Time: 33:21)



```
def tower_of_hanoi(n,source,to,intermediate):
    if (n==1):
        print("transfer disk ", source , " to ", to)
    else:
       tower_of_hanoi(n-1, source, intermediate, to)
       print("transfer disk ", source , " to ", to)
       tower_of_hanoi(n-1, intermediate, to, source)
```

So, n is a number of this; source is you will start from I did not want to put 1, 2, 3 that is again not a good choice of variables; you should choose a variable which will tell you this is what you mean. Whether I like my code to be used by other people as well or myself, if you do not write a nice code; then you come after one week; you will not know what you have written; it happens to me.

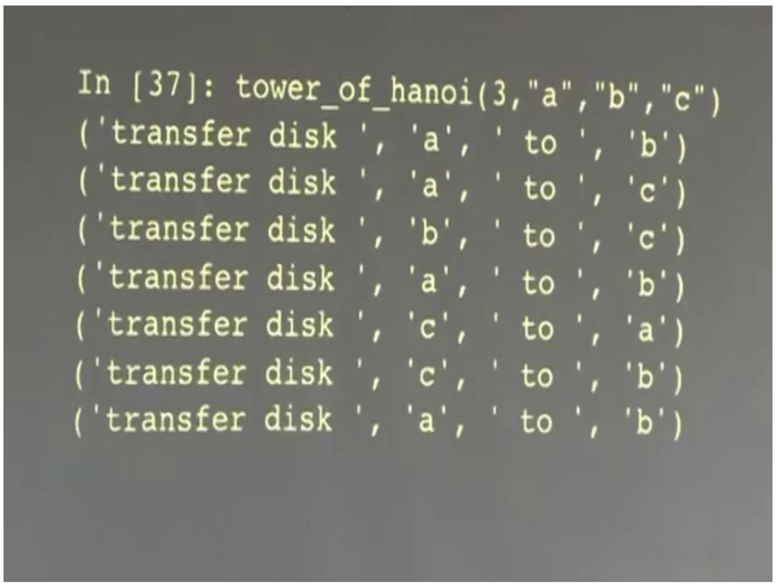
So, if I write a code what did I write? I cannot figure out from what I have written. So, you must be careful what variables we choose. So, source is a starting peg to; I want to for example, will here an intermediate. So, we are this is intermediate; if I do not intermediate I cannot solve this problem. So, what I say is if n is equal to 1; it is only one disk it is like factorial; if it is n is equal to 1, you say your answer is 1. If n equal to 1 then just say from source; so, I have to print transfer disk source to 2.

So, these are strings and this string and I send it there. For 1 it is trivial; I mean for 1 you just do it, but suppose you had more than 1; then you follow the same thing which I told transfer n minus 1 from source to intermediate. So, second argument is where you want to set; well third argument. So, this is the number of disk; the third argument where you want to send.

So, do not send to 2; send to intermediate n minus 1. So, what did I said? So, I take n minus 1 here first; that is the first step. So, I take n minus 1; this from source to intermediate, but 2 will be my intermediate for that process. If I want to send n minus 1 here, I can use this as a intermediate and then what do I do? I have left with one disk. So, it transferred this disk to the; from source to 2. So, I have this situation n minus 1 sitting here and n minus 1 sitting here and after that I take n minus 1 from intermediate to 2, but this peg becomes a intermediate now. So, source peg becomes intermediate for this process and if you do this; you solve the problem.

So, this is called recursion if you know how to solve for n, you can solve for n minus 1. If you know how to solve for n minus 1, you can solve for n. You can always try to stop somewhere, otherwise it will go for infinite loop. It will stop somewhere, otherwise it will keep calling, calling, calling and it will not quite and once I run it; this is the output.

(Refer Slide Time: 36:21)



```
In [37]: tower_of_hanoi(3,"a","b","c")
('transfer disk ', 'a', ' to ', 'b')
('transfer disk ', 'a', ' to ', 'c')
('transfer disk ', 'b', ' to ', 'c')
('transfer disk ', 'a', ' to ', 'b')
('transfer disk ', 'c', ' to ', 'a')
('transfer disk ', 'c', ' to ', 'b')
('transfer disk ', 'a', ' to ', 'b')
```

It is very similar this is what you will get; after the computer solve the problem. So, it is all the print will come and a; so, it will a take a to b transfer this from; so, I have name the peg to a string a, b, c. So, this is source and b is intermediate since of what is said; the top one you should take this and put on any this is solution; you just run it, you have the code just see.

So, this how you solve recursively I will give you one more example after some time. It turns out I will ask you to do it at home; how many steps required to solve Tower of Hanoi problem for n disk? So think about it; it turns out the number rises exponentially and there is a story that; in Banaras, there is a saint somewhere sitting there and he used to ask is the 64 disks and needs to transfer it here or here and once that is done the whole world will end.

So, how long will it take to transfer 64; you should figure it out. It is a huge number, so but regression is not efficient because it goes down, goes up if you have to go down, down, down then start coming up. So, normally we avoid recursion for physics, but some problems we have to do using recursion. In fact, all the fractal stuff; if you want to print them, you have to use recursion. A Tower of Hanoi, if you want to write sequential code; it is not easy. In fact, I do not know the solution; there maybe one, but I do not know how to solve it using iteration.