INDIAN INSTITUTE OF TECHNOLOGY MADRAS

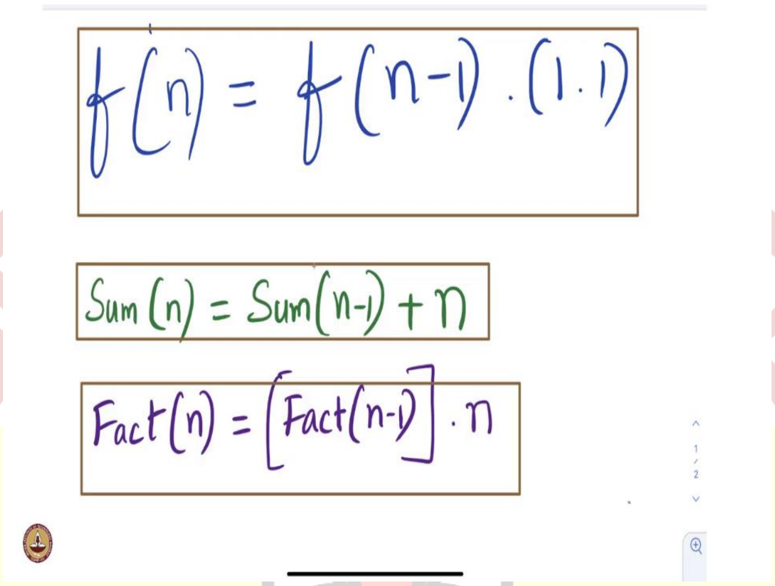
सिद्धिर्भवति कर्मजा

IIT Madras

ONLINE DEGREE

Programming in Python
Professor. Sudarshan Iyengar
Department of Computer Science and Engineering
Indian Institute of Technology, Ropar
Recursion – An Illustration

(Refer Slide Time: 00:16)



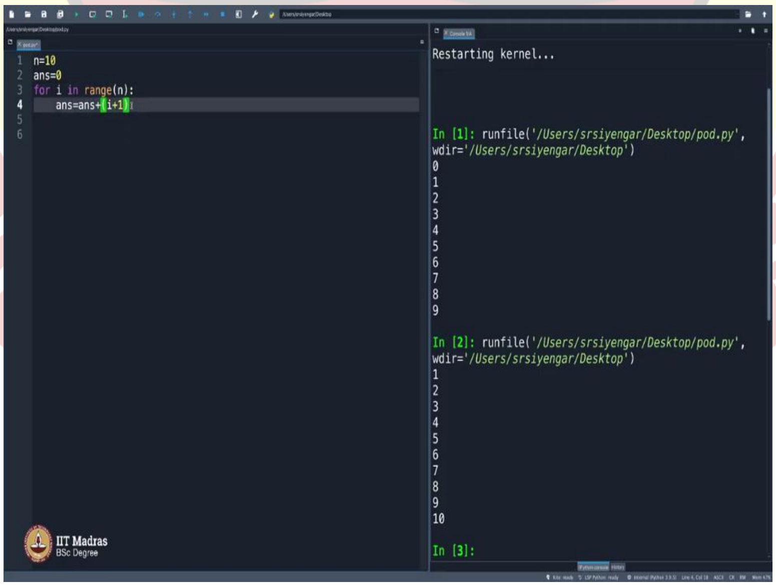
$$\boxed{f(n) = f(n-1)\,.\,(1.1)}$$

$$\boxed{Sum(n) = Sum(n-1) + n}$$

$$\boxed{Fact(n) = [Fact(n-1)]\,.\,n}$$

So, continuing our discussion. Let me try writing a piece of code, which will find out the sum of n numbers, so look at this.

(Refer Slide Time: 00:27)



```python
n=10
ans=0
for i in range(n):
    ans=ans+(i+1)
```

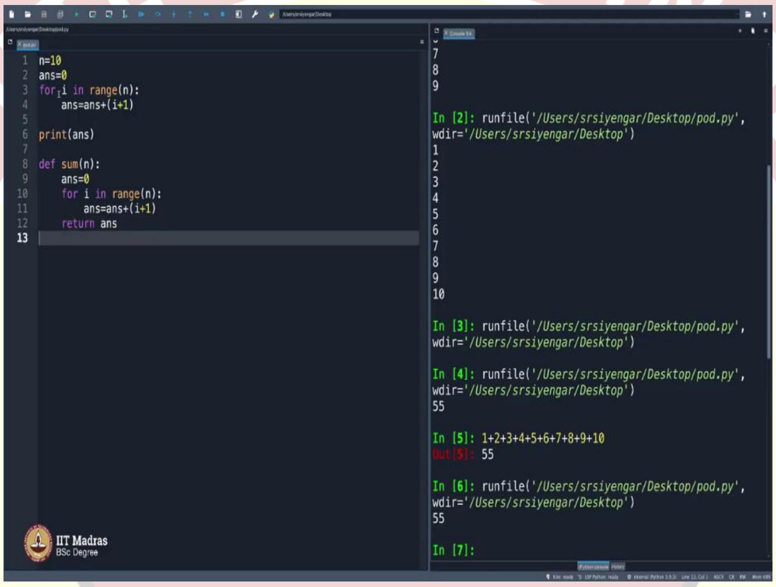
```
Restarting kernel...


In [1]: runfile('/Users/srsiyengar/Desktop/pod.py',
wdir='/Users/srsiyengar/Desktop')
0
1
2
3
4
5
6
7
8
9


In [2]: runfile('/Users/srsiyengar/Desktop/pod.py',
wdir='/Users/srsiyengar/Desktop')
1
2
3
4
5
6
7
8
9
10

In [3]:
```

Let me try coding it the nave way that we have been doing from a long time. How would that go? So, let us say, assuming n equals 10 I will not take it as an input from the user, I will change the variable here itself. So, I say for i in range n. So, what I do is, I first say answer equals 0 and i in range n will simply what will this do let me print i and so that you can see it.

It prints numbers from 0 to 9, and I am going to be a little fast now, because these things are very basic for you people. You probably should be able to do it in no time or if you think you are unable to do it in no time then you need practice.

So, print i, which means it starts from 0 and goes up to 9, but I want the first n numbers maybe I will do a plus 1 I get 1 to 10, because that is what I need to add. So, what I will do is I will say answer equals answer, plus your i plus 1. Correct. This will basically do the sum of first n numbers. Let me see if it gives me 55.

(Refer Slide Time: 01:45)

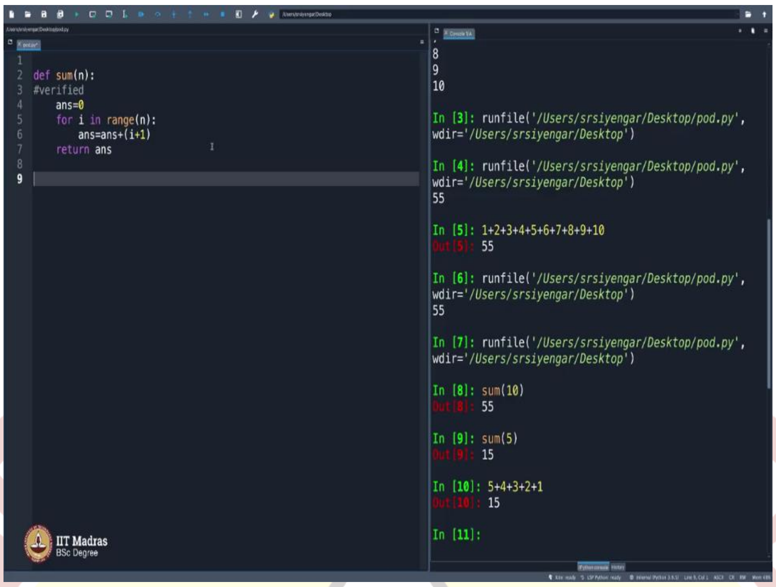


```
n=10
ans=0
for i in range(n):
    ans=ans+(i+1)

print(ans)

def sum(n):
    ans=0
    for i in range(n):
        ans=ans+(i+1)
    return ans
```

```
7
8
9

In [2]: runfile('/Users/srsiyengar/Desktop/pod.py',
wdir='/Users/srsiyengar/Desktop')
1
2
3
4
5
6
7
8
9
10

In [3]: runfile('/Users/srsiyengar/Desktop/pod.py',
wdir='/Users/srsiyengar/Desktop')

In [4]: runfile('/Users/srsiyengar/Desktop/pod.py',
wdir='/Users/srsiyengar/Desktop')
55

In [5]: 1+2+3+4+5+6+7+8+9+10
Out[5]: 55

In [6]: runfile('/Users/srsiyengar/Desktop/pod.py',
wdir='/Users/srsiyengar/Desktop')
55

In [7]:
```

Print answer, answer is 55. That in fact, is the answer for 1 plus 2 plus 3 plus 4 plus 5 plus 6 plus 7, plus 8 plus 9 plus 10 which is 55. So how do I put this inside a function? Defined sum of n numbers is simply the same thing, answer equals 0, this entire stuff should come inside. And then for i in range n you do answer equals answer, plus i actually, but then you want from 1 to n not from 0 to n minus 1 a small trick there. It should be easy on your minds then I say return answer. Execute this.

(Refer Slide Time: 02:33)

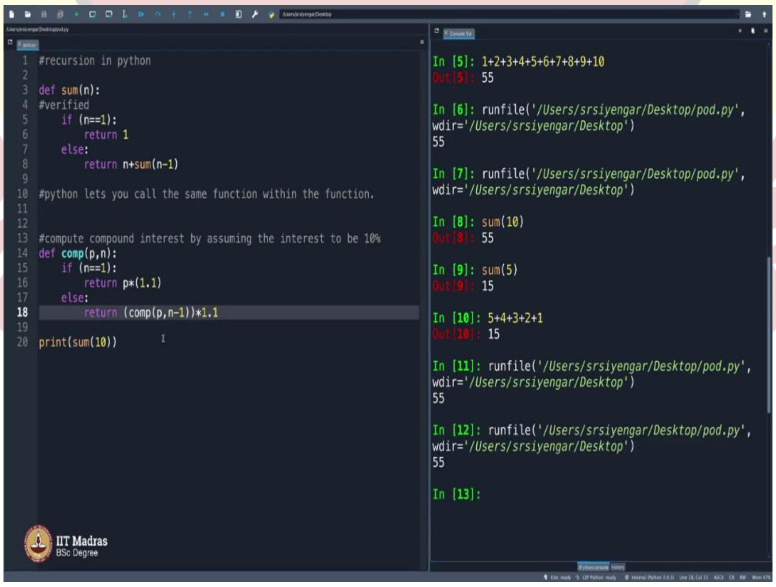


```
def sum(n):
#verified
    ans=0
    for i in range(n):
        ans=ans+(i+1)
    return ans

```

```
8
9
10

In [3]: runfile('/Users/srsiyengar/Desktop/pod.py',
wdir='/Users/srsiyengar/Desktop')

In [4]: runfile('/Users/srsiyengar/Desktop/pod.py',
wdir='/Users/srsiyengar/Desktop')
55

In [5]: 1+2+3+4+5+6+7+8+9+10
Out[5]: 55

In [6]: runfile('/Users/srsiyengar/Desktop/pod.py',
wdir='/Users/srsiyengar/Desktop')
55

In [7]: runfile('/Users/srsiyengar/Desktop/pod.py',
wdir='/Users/srsiyengar/Desktop')

In [8]: sum(10)
Out[8]: 55

In [9]: sum(5)
Out[9]: 15

In [10]: 5+4+3+2+1
Out[10]: 15

In [11]:
```

Let me now remove this. Now that I have created a function out of it let me remove this. What is the point of doing all these things you will understand in a second, I am going to introduce you people to a brand new idea in Python. So, sum of 10 is 55, sum of 5 is 5 plus 4 plus 3 plus 2 plus 1, which is 15. Perfect, my function is doing good. As and always, it is nice to say, verified, just so that you know that this function is doing good, and you need to break your head.

(Refer Slide Time: 03:24)



```
#recursion in python

def sum(n):
#verified
    if (n==1):
        return 1
    else:
        return n+sum(n-1)

#python lets you call the same function within the function.


#compute compound interest by assuming the interest to be 10%
def comp(p,n):
    if (n==1):
        return p*(1.1)
    else:
        return (comp(p,n-1))*1.1

print(sum(10))
```

```
In [5]: 1+2+3+4+5+6+7+8+9+10
Out[5]: 55

In [6]: runfile('/Users/srsiyengar/Desktop/pod.py',
wdir='/Users/srsiyengar/Desktop')
55

In [7]: runfile('/Users/srsiyengar/Desktop/pod.py',
wdir='/Users/srsiyengar/Desktop')

In [8]: sum(10)
Out[8]: 55

In [9]: sum(5)
Out[9]: 15

In [10]: 5+4+3+2+1
Out[10]: 15

In [11]: runfile('/Users/srsiyengar/Desktop/pod.py',
wdir='/Users/srsiyengar/Desktop')
55

In [12]: runfile('/Users/srsiyengar/Desktop/pod.py',
wdir='/Users/srsiyengar/Desktop')
55

In [13]:
```

Now, is there a better way to write this function? By better I mean, a different way to write this function? Probably yes. I will show you that different way, and that is called what is called recursion in Python. The word recursion. What does recurse mean? Recurse means you recursive means do it again and again. That is what it means, but in Python or in programming in general, recursion is a very important concept, which means the following.

Let me not say what that is, let me show you what that is. Look at this, define sum of n. If n is equal to 1, return 1. So sum of 1, which should be 1. I mean, so if n is equal to 1, sum of 1is the first n natural numbers, by default this should be 1. So, what is sum of 5, 15, but sum of 1 should be not 5 to 1, but just 1 alone the answer should be 1.

So, it should display 1, but if n is not 1, else means what if n is not 1, which means n is greater than 1, I am not taking any negative numbers or 0 here, so this should return what? Pause for a minute, think. Sum of n is sum of n minus 1 plus n. This must return n plus sum of n minus 1. Pause for a minute and think what this is doing. Look at the power of this piece of code. It looks very short. It is not so straightforward of the mind, but it takes time for you to digest but this is a very, very, very powerful programming idea.

You will get to know as you advance into more and more programming, you will realize that this thing called the recursion, where you call the definition here itself, the function inside the same function, this is called recursion. You may be wondering, Sir, this, the word sum is defined only here, how does it know what to do here. So, your Python is smart enough to understand that this very thing, whatever you are doing insight happens at this level also.

This is exactly analogous exactly same as this one that we were talking about a function. Remember the compound interest function, the compound interest at the nth year is compound interest with the n minus oneth here, times 10 percent more, which is 1.1. Basically 1 plus point 1, it becomes 1.1. So, to calculate this function, you use the same function and calculate a smaller number and add something to it. That is what we are saying here.

If you expect this to return an error, you are right. But Python has enabled this as a facility, it lets you it will write that down. Python lets you call the same function within itself within the function. So, what do you mean by this English? My English sounds complicated. All that I am trying to say is this a function, and it can call the same function within itself. What do you mean by this? I mean, this does not even sound like English.

This simply means you have a facility analogous to this. If you want to compute something, you can define what this function is by using the same function. This is just convenience and you can do it. So let me see if this works or not. Let me come here and say print sum of 10, this should they should print 55 here. Let us see if it prints 55. Yes, it does. 55. It says 55. It means it is working. So you can call a function within a function.

Just to mimic the very mathematical way in which we define some things. Shall we go ahead now and find the compound interest for n years have some principal amount? What is that? So, help me out. Define compound interest instead of f let me call it comp. Compound interest of a principal sum of P over n years. What is this equal to? This is equal to before that let me say verified because I am going to write two to three functions here. I am going to denote that it is done. It is doing what it is supposed to do by saying verified, this is just for convenience, and also good documentation practice.

So, define comm p comma n, and what should I do now, if n is equal to 1. If it is only one year, then return P times 1.1. I am assuming the interest to be 10 percent here. Compute compound interest, I am sorry for the typos, compound interest by assuming the interest to be 10 percent. So, it is 1.1 or you can even put that as a parameter here so let us not complicate it.

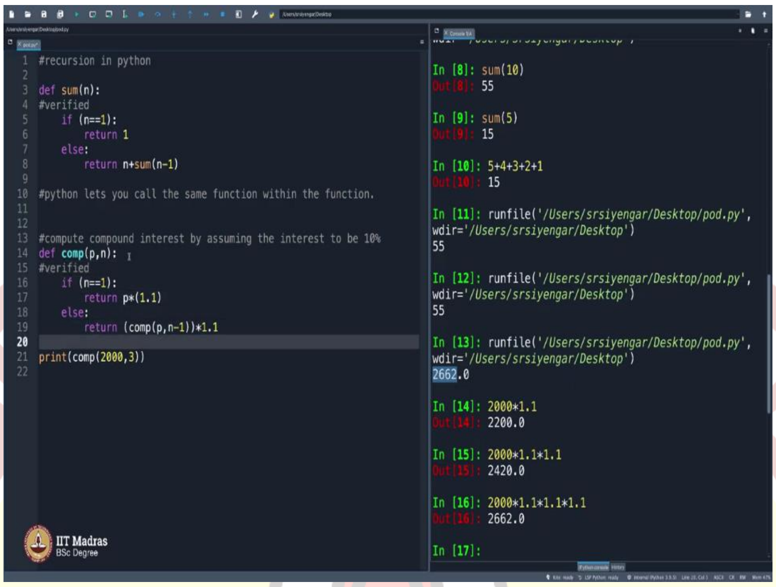
I am trying to explain what is recursion to you P but let me stick to a simple example, return p into 1 my 1.1. But what if the number of years is more than 1, then put else. Assuming n cannot be 0 or any other number, it can only be 1, 2, 3, 4, 5 I will say else means what n is greater than 1 then return whatever was the value for compound interest of a principal of P for n minus 1 years, whatever that is, you compute that.

You compute that and you multiply this by 1.1 and return that. So, what does this do to compute 1,000 comma 2, it comes here it says n is not 2, it is 1 so it does not execute this part it comes here and it returns compound interest of principle with one year times 1.1. And for that again, it comes in and then it recursively evaluates this.

This is complicated. This is not easy on your mind to see comp here and comp here, so it takes some time to get used to it. I still remember, when I was a student recursion did take me some time to even understand the very philosophy of it. Why would they put something like this, where would they use something like this.

Answer is simple. The answer is, in mathematics, we do this in math, there is something called recurrence relations, if you have not heard of it. Do not worry if you have not heard, but please note that in math we use functions like this. To define something, we use the same function with a smaller value. So, just to help you simply code these kind of functions onto a Python program, Python helps you with this kind of facility.

(Refer Slide Time: 10:27)

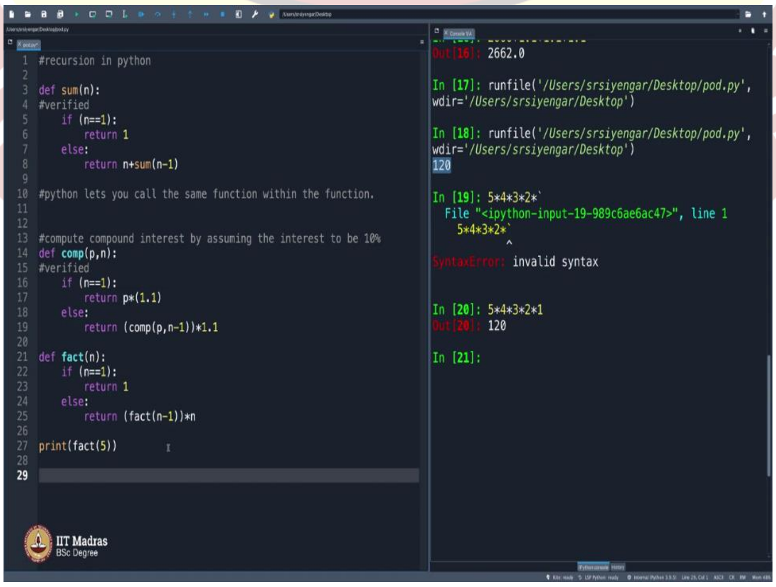


```python
#recursion in python

def sum(n):
#verified
    if (n==1):
        return 1
    else:
        return n+sum(n-1)

#python lets you call the same function within the function.


#compute compound interest by assuming the interest to be 10%
def comp(p,n):
#verified
    if (n==1):
        return p*(1.1)
    else:
        return (comp(p,n-1))*1.1

print(comp(2000,3))
```

```
In [8]: sum(10)
Out[8]: 55

In [9]: sum(5)
Out[9]: 15

In [10]: 5+4+3+2+1
Out[10]: 15

In [11]: runfile('/Users/srsiyengar/Desktop/pod.py',
wdir='/Users/srsiyengar/Desktop')
55

In [12]: runfile('/Users/srsiyengar/Desktop/pod.py',
wdir='/Users/srsiyengar/Desktop')
55

In [13]: runfile('/Users/srsiyengar/Desktop/pod.py',
wdir='/Users/srsiyengar/Desktop')
2662.0

In [14]: 2000*1.1
Out[14]: 2200.0

In [15]: 2000*1.1*1.1
Out[15]: 2420.0

In [16]: 2000*1.1*1.1*1.1
Out[16]: 2662.0

In [17]:
```

So, all the chatter, and no answer, let us see if this really works or not. So, how do I go about it? I will, comment this, this is not required I would even delete it. I will say, print compound interest of 2,000 rupees for 3 years at the rate of 10 percent that is fix it. It says 2662. Let me just check that 2,000 times 1.1. This is for first year. And this is for second year and this is for third year 2662, perfect, that is what it is saying here. So, my program is working fine. I will say, verified.

(Refer Slide Time: 11:15)



```python
#recursion in python

def sum(n):
#verified
    if (n==1):
        return 1
    else:
        return n+sum(n-1)

#python lets you call the same function within the function.


#compute compound interest by assuming the interest to be 10%
def comp(p,n):
#verified
    if (n==1):
        return p*(1.1)
    else:
        return (comp(p,n-1))*1.1

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

print(fact(5))
```

```
Out[16]: 2662.0

In [17]: runfile('/Users/srsiyengar/Desktop/pod.py',
wdir='/Users/srsiyengar/Desktop')

In [18]: runfile('/Users/srsiyengar/Desktop/pod.py',
wdir='/Users/srsiyengar/Desktop')
120

In [19]: 5*4*3*2*`
  File "<ipython-input-19-989c6ae6ac47>", line 1
    5*4*3*2*`
            ^
SyntaxError: invalid syntax


In [20]: 5*4*3*2*1
Out[20]: 120

In [21]:
```

So, let me remove this. And now if we go ahead and write the standard program, factorial of n, Whenever someone teaches you a recursion, I believe from the past 30, 40 or even 50 years, people have been using factorial as a nice example.

So, although, you must be wondering where is factorial even used, so do not worry about that. Let us go ahead and try to compute factorial using the recursion way. I will just type and you will just watch define fact and this will be if n is equal to 1 once again, I will say return 1 fact of 1 is 1 else return factor of n minus 1 times n. This should work. So, I have simply codified this part of the formula, which I explained in the previous video, this part of the formula losing programing. So, you know very well what will happen if I do not put this.

Fact n will be fact of n minus 1 into n and fact of n minus 1 here, it again, will call this function that will be a factor of n minus 2 into n minus 1 and so on you see fact of n is fact of n minus 1 into n, you expand this by using this formula, then it becomes an n minus 1 into fact of n minus 2. Where will this stop it can go on and on, as n keeps getting smaller and smaller.

I stop it by saying that when n equals 1, the answer is simply 1. And you go ahead and calculate factorial of 2 factorial of 3 factorial 4, and so on and so forth. Let we execute this, and then find out, print, factorial of 5, factorial of 5 is 120 is a true, 5 into 4 into 3 into 2 into 1, boom, stick there, 120 that is right.

So, factorial 5 is 120 and that is what is being displayed here. So, my program is running perfectly right. Again, I repeat, a word of warning, this can be heavy on your heads to begin with, but do not worry, with time you will get used to it, all you need to do is boom, look at this. It helps you codify a typical mathematical statement like this.

That is all what your recursion does. So, this is the first introduction to recursion. I suppose not many of you would appreciate it much. You will appreciate it, as we start using this in our code in the future. In many aspects of your forthcoming subjects, you will be using recursion extensively, in fact, so get used to it, but do not break your head much if you cannot understand it completely or appreciate it completely.

So, in fact, I would like to tell you people that at this stage, as we stop discussing one part of functions. Of course, we will go ahead and talk about a little more advanced aspects of functions in the forthcoming videos, but then, at this stage, I would like to stop and tell you that forget even recursion for that matter.

Go up to the matrix multiplication, which was the previous topic that we discussed. This much is more than enough for you to start coding, start coding and type some very advanced Python programming. In fact, you are recruitable right now, you can call yourself a Python literate, you can do good amount of programming with just this much whatever we have taught so far up to let us say, matrix multiplication.

Whatever we are going to go and teach ahead will be slightly advanced, which you can always learn all by yourself. So, you should feel very confident right now. You should feel more confident as we teach you more concepts. Alight, then. All the best. Let me, I will now take a break here and your, our instructor Omkar our friend Omkar will now come go ahead and teach you people more codes, more examples, a little more advanced topics and we will meet in the next week. Thank you all very much.