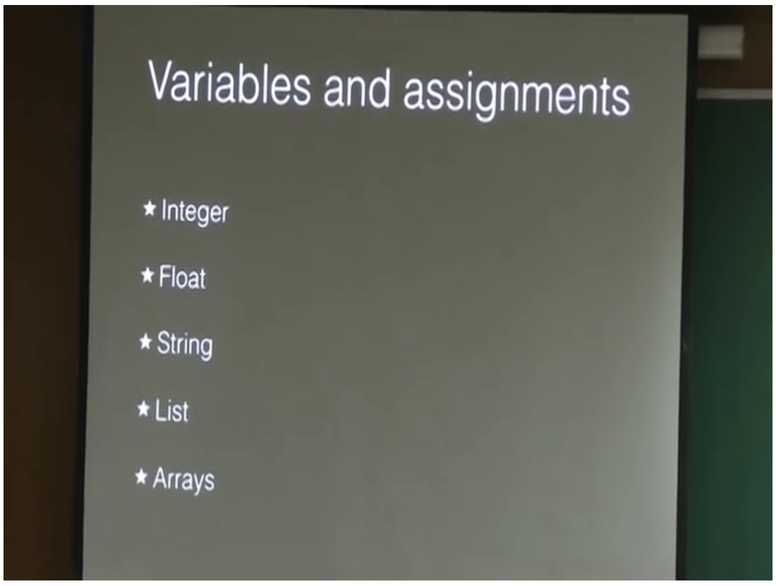
Computational Science and Engineering using Python

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

Lecture - 02
Phython: Variables & Assignments

So, I repeat some of the things which I did last time.

(Refer Slide Time: 00:21)

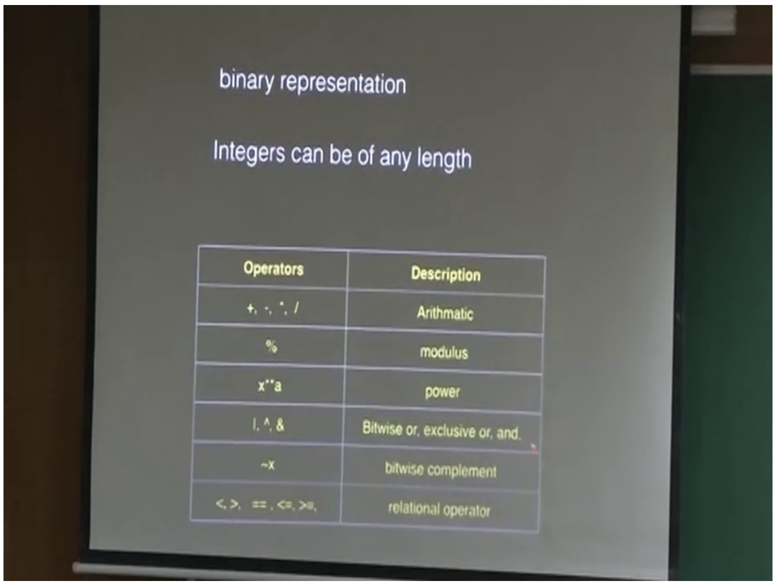


Variables and assignments

- Integer
- Float
- String
- List
- Arrays

So, variables and assignment, so python has 5 types of variables, we will not use much of string and list. So, we will use integer float and arrays; now integers just recap quick recap.

(Refer Slide Time: 00:32)



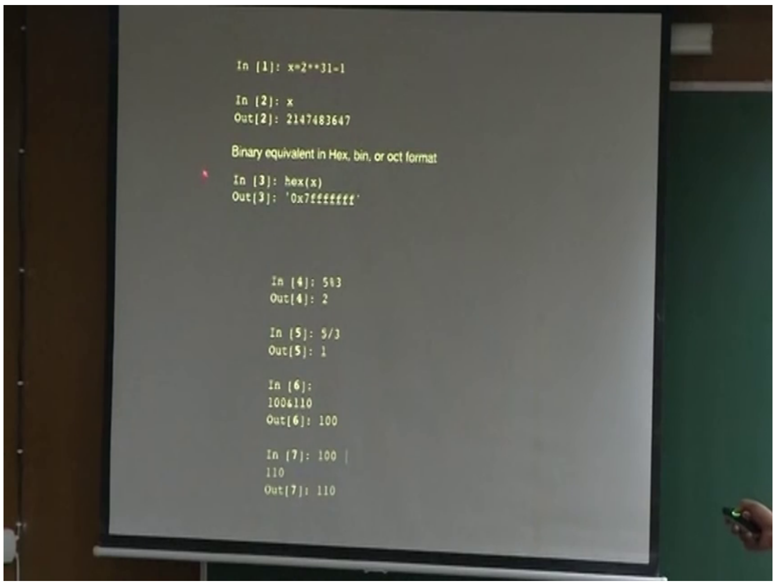
binary representation

Integers can be of any length

Operators	Description
+, -, *, /	Arithmatic
%	modulus
x**a	power
|, ^, &	Bitwise or, exclusive or, and.
~x	bitwise complement
<, >, ==, <=, >=,	relational operator

So, it is a binary representation in computer with in computer, and integers can be of any length unlike other languages it allows any length and these are the operations. So, we will again not use this bitwise stuff in this course most of the time.

(Refer Slide Time: 00:47)



```
In [1]: x=2**31-1

In [2]: x
Out[2]: 2147483647
```

Binary equivalent in Hex, bin, or oct format

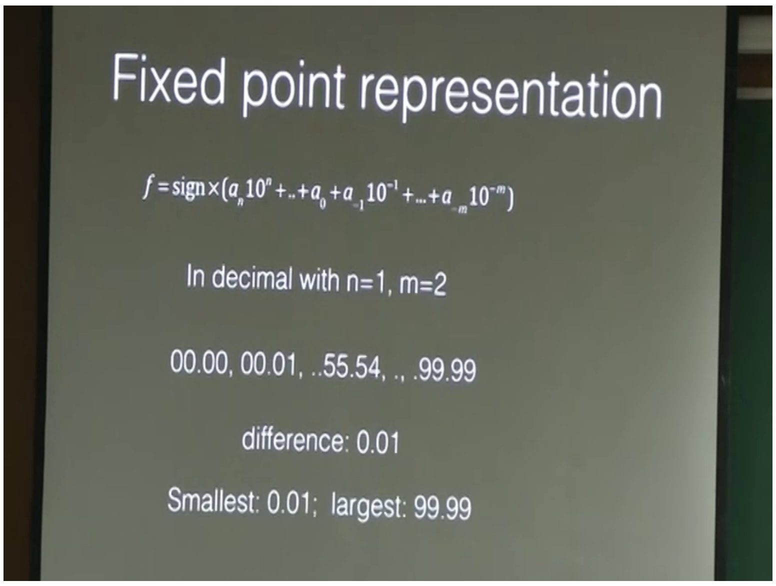
```
In [3]: hex(x)
Out[3]: '0x7fffffff'

In [4]: 5%3
Out[4]: 2

In [5]: 5/3
Out[5]: 1

In [6]:
100&110
Out[6]: 100

In [7]: 100 |
110
Out[7]: 110
```

So, these are examples which had put in the last class. So, x to the 31 minus 1 will be this large number in decimal. So, you have to be careful today I will make a contrast between binary and decimal. So, this is in decimal now if you want in binary or hex. So, these are representation you be in the homework I think everybody is happy with it now. You can also do negative numbers, but python will just say minus of this, the division and reminder.

So, this is a division is an integer division. So, right now treating 5 in 3 is integer and dividing it is a division, if you want refloat division then you can put 5.0 by 3. So, you put one of them is float, then it will convert the other and also to float and it will give you 1.666. These are reminder these are bit operations I think I will skip you can just. So, this and these or is not I think it got cut this has come here 100 or 110 will give you 110 or operation in and operation you probably know from binary operation. So, let us look at real numbers which we deal most of the time.

So, the 2 ways to represent real numbers; now this is not computer right now I just tell you the idea in fact, this very interesting line of thought went into how to represent real numbers in computer. So, the 2 is which we are aware of we use the first one very often, but. So, this called fixed point representation the decimal point is fixed.

(Refer Slide Time: 02:28)



Fixed point representation

$f = \text{sign} \times (a_n 10^n + .. + a_0 + a_{-1} 10^{-1} + ... + a_{-m} 10^{-m})$

In decimal with n=1, m=2

00.00, 00.01, ..55.54, ., .99.99

difference: 0.01

Smallest: 0.01; largest: 99.99

So, if you write in terms of formula then is going to be looking like this. So, the less than one numbers will. So, this is one and.

Student: (Refer Time: 02:39).

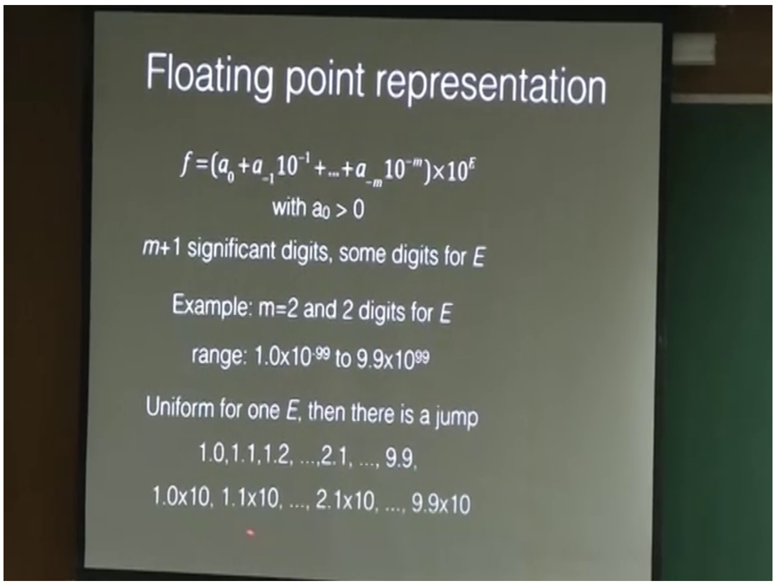
This is 1 by 10, 1 by 10 to power m and this 10 power n. So, with this you can represent any real number, but here my decimal point is somewhere here right here. So, this called fixed point representation we use this most often in our daily life. So, let us do an example. So, in decimal number if you have n equal to 1.

So, this is n equal to 1and m equal to 2. So, what are the numbers can I represent and I am I am limited. So, if I just have this then I cannot represent all numbers, a very limited set of numbers and the numbers you can represent will be these I am focusing only on positive, so 0.00, so m is 2. So, these are the 2 digits and n equal to 1. So, I can go up to up to 99. So, that is 99. So, these are the numbers very limited.

Student: (Refer Time: 03:33).

Right I mean this what you can do and what is the spacing between 2 numbers is point o one, but they are uniformly spaced. So, the difference is point o one and, but it turns out that they can represent all these very small set of numbers, then largest number is close to hundred as smallest number is point o one it is smallest non zero number it is point o one, but you can do better. So, you can do better and that is what the computer sent is thought about and they said let us do called floating point representation, it is a fascinating in interpreter representation.

(Refer Slide Time: 04:07)



Floating point representation

$f = (a_0 + a_{-1}10^{-1} + \ldots + a_{-m}10^{-m}) \times 10^E$

with $a_0 > 0$

$m+1$ significant digits, some digits for E

Example: m=2 and 2 digits for E

range: 1.0×10^{-99} to 9.9×10^{99}

Uniform for one E, then there is a jump

1.0, 1.1, 1.2, ..., 2.1, ..., 9.9,

1.0x10, 1.1x10, ..., 2.1x10, ..., 9.9x10

Now, how is it is really represent it? It is written as the decimal part is here, but then you multiply have a 10 power E.

So, I am right now working with decimals, but the idea of whatever you do in decimal is same as what you do in binary not much difference. So, we write this, so I had ensure that a naught is non zero, you make a not non zero and then any number can be represented like this, you need you need large m, but you represent any number. Now it turns out he can increase a range with this representation, I can get more set up, but a range of numbers not more set to range of numbers. So, let us do an example I will go quickly this is very obvious, but it is very interesting thing to notice. So, it has m plus 1 significant digits right because this is m plus 1 significant digit and we need to keep some digits for exponential.

So, let us do an example with m equal to 2.

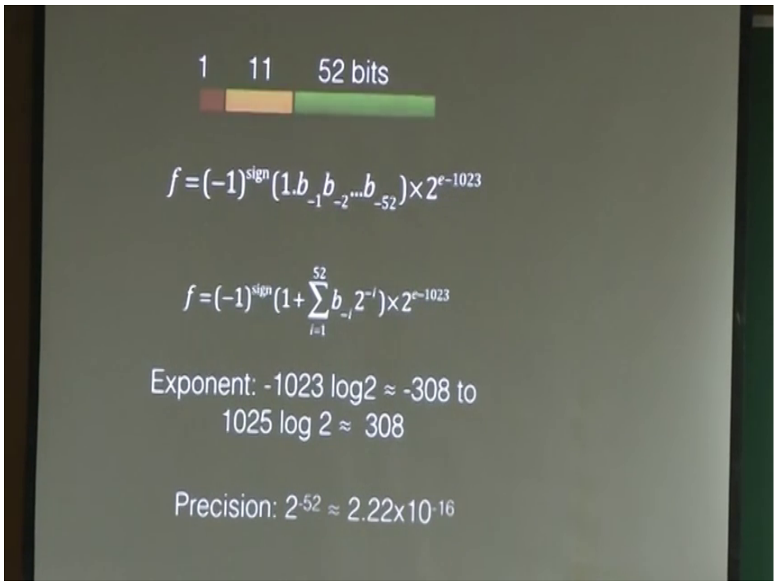
Student: (Refer Time: 05:03).

And 2 digit for E. So, I am again using four digits. So, what is a set of numbers I can represent? So, now, my E is 2 digits means minus 90 to 99 range increased from minus 2 to minus 99 for my number is 1.0. So, that this part is 1.0 and 9.9 and it is obviously, this now let us look at as the numbers uniformly spaced in this representation they are not uniformly spaced. So, if they are uniformly spaced first one single E, but then you jump to E go to the next E then this a jump. So, to see an example I like this idea. So, I wanted to communicate to this part 1.0.

So, here I am making exponential E is 1. So, it is spacing it is 0.1 clearly, now when you go to E equal to 1 that means, I am multiplying with 10. So, multiply 10 then what happens. So, I have 1.0 times 10. So, it is 10 this is 11, 12. So, which spacing has become 1. So, my spacing is not uniform in floating point representation, but my increase my range. Now if you have the choice which one will you choose of course, for computer this is the best well I want to represent huge number of numbers very small and very large, but even for realistic like for physics, this is more natural you look at nuclear scale, then go to molecular scale, then you go to biological scale, then human scale extra physical scale we always do this we do not use in fact, we do for our daily lives decimal as the fixed point number, but really the nature is built with this a different scale you need this representation. So, nature is doing this.

So, is in fact, no wonder computer we need this well I mean this is a somewhat same phenomena when computer and real life or in nature, actually I would say not real life nature. Now in computer how do you represent a real number in computer? Now this is idea, but instead of 10 you replace by 2. So, that is binary.

(Refer Slide Time: 07:32)


1 11 52 bits

$$f=(-1)^{sign}(1.b_{-1}b_{-2}...b_{-52})\times 2^{e-1023}$$

$$f=(-1)^{sign}(1+\sum_{i=1}^{52} b_{-i}2^{-i})\times 2^{e-1023}$$

Exponent: -1023 log2 ≈ -308 to 1025 log 2 ≈ 308

Precision: $2^{-52} \approx 2.22\times10^{-16}$

So, this is for double. So, one bit for sign 11 bits for explanation, and 52 bits for precision the decimal number. So, it will look like this one for sign. So, if it is 0 then it is positive and one is negative and this is b minus 1 to minus 52 this is 52 bits and exponential is represented whatever number is here.

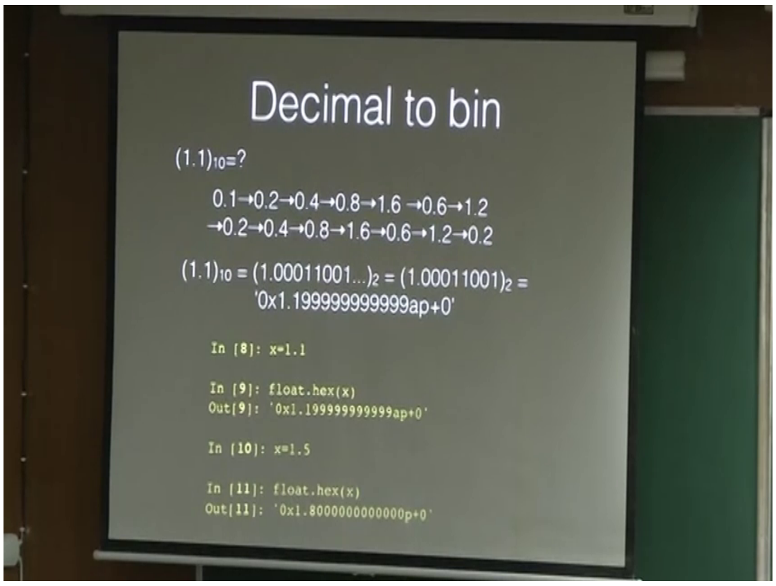
You subtract minus 1 0 2 3. So, there will give negative exponent there will positive exponent.

So, I this very simple algebra which I will not belabor on. So, you can go from minus 3 0 8 exponent to 3 0 8 which is a big number. So, in a in meters, inverse, length inverse size will be less than this or in centimeter it will be less than this, but you have to still have to careful. Suppose you want to exponentiate inverse size what power 10, then you will exceed this. So, you have to be careful when you do exponential that is where we may exceed a exponent and the precision how do I know the precision. So, a precision will be the difference between 2 consecutive numbers. So, all the bits are zeros except the last bit when I subtract 2 consecutive numbers is that correct 52 and that means, my number the difference between 2 consecutive number is 2 to power minus 52 this is binary.

So, I have to take power with 2 and that happens to be 10 power minus 16. So, my precision is only 16 digits, anything beyond I cannot represent in my computer. If you need more precision which sometimes people ask for it then you need to code separately. So, there are some libraries available. So, you can use you can go to I mean you can double this. Now the present computation standard is double, now there is enough ram there is enough communication.

So, people use standard double if you want float then instead of 64 bits this is 64 bits you use 32 bits and 32 bits the number decrease. I think this is this probably 8t and this number is different. So, the range will decrease, but double is standard. So, use double it cost more the time taken is roughly double of what you do it for float, but this more accurate most people will say well I mean compute let computed do the work you relax and I mean let the computer is there is enough power in computer. So, you can just let it work unless there is a strong limitation of ram or computer time now.

(Refer Slide Time: 10:22)



Decimal to bin

$(1.1)_{10}=?$

$0.1 \rightarrow 0.2 \rightarrow 0.4 \rightarrow 0.8 \rightarrow 1.6 \rightarrow 0.6 \rightarrow 1.2$
$\rightarrow 0.2 \rightarrow 0.4 \rightarrow 0.8 \rightarrow 1.6 \rightarrow 0.6 \rightarrow 1.2 \rightarrow 0.2$

$(1.1)_{10} = (1.00011001...)_2 = (1.00011001)_2 =$ '0x1.199999999999ap+0'

```
In [8]: x=1.1

In [9]: float.hex(x)
Out[9]: '0x1.199999999999ap+0'

In [10]: x=1.5

In [11]: float.hex(x)
Out[11]: '0x1.8000000000000p+0'
```

So, this is able yeah.

Student: (Refer Time: 10:24).

Sure sure.

Student: Before tends to be 22 (Refer Time: 10:27) operating.

Right.

Student: (Refer Time: 10:30).

Right right right.

Student: Then MATLAB for (Refer Time: 10:39) minus 60 now machines are 64 bit. So, does it (Refer Time: 10:40).

No, what it showed here is (Refer Time: 10:42) is double precision 10 power minus 16 with.

Student: (Refer Time: 10:45).

No, no this for 64 bit machine, these for 60 bit 4 bit operation now what has happened now this is technical part.

Student: (Refer Time: 10:54).

Presently the number of wires which come from the CPU to RAM, the 16 wires are coming. So, it can access 16 bits at a time earlier the number of wires which you come out was 8 you know.

So, you could get only 8 number sorry the 32 (Refer Time: 11:14) not 8 32. So, should you do has more convenient and fast, but now 64 wires are coming. So, say double is standard that is why they say we support double operations. If you doing 32 bit then this precision is 36 or 37 this number is 37 not 3.08.

Student: Sir even if 52 bit is used to be (Refer Time: 11:35).

No well MATLAB is doing some work additional work. So, python allows you to use as many numbers as many accuracy you like you get.

Student: (Refer Time: 11:49) anything extra (Refer Time: 11:50).

Yeah MATLAB is using some c libraries within and which is 64 bit. So, this is standard python will be doing this, but if you want more higher precision python will take care of it and that is why python is slower c will optimize for a uniform set of data.

So, python and MATLAB happens to be slow because it allows you flexibility, but then it cannot optimize it. So, now, everyone knows how to convert this, this is simple operation, but let us just look at one number 1.1 in decimal, I want to represent in binary now how do I do it? So, one is I will keep. So, one is remember the first a naught is 1, one is that should be non zero and only one non zero number for binary is one. So, you do not store one this one is not stored in the computer this one first one. Second one is stored. So, you multiply point into 2. So, this operation 0.1 into 2, so 0.2 is less than 1. So, it is 0 the first bit is 0.

So, the standard stuff know. So, suppose I want to get the decimal of 1.35.

(Refer Slide Time: 13:06)

So, I get this off multiply this by 10. So, 3 will go to the left of decimal and you pick that that is the first digit multiplied by 10 after that. So, that will be 0.5 this is taken. So, this multiplied by 10 then you get 5. So, that is how you get the digits. So, you do the same thing here, you multiplied by 0.2 to then 0.2 0.4. So, this is all zeros this is all zeros here I get one is that, but I request you to please do this these are simple stuff, but it should be able to convert one to other this was done thousands of years back.

Student: (Refer Time: 13:49).

So, you get one here there 0.6 is remaining, the again keep multiplying they get one here.

Student: (Refer Time: 13:57).

And do this and that is how the number I will get is 3 zeroes, 3 zeroes here then one here one here then again this one here, one here then 2 zeroes. So, you do this. So, this is the binary representation of 1.1. It turns out it will not stop that means, 1.1 cannot be represented accurately in a computer. Decimal yes 1.1 is again 1.1 is possible to represent using 2 digits 2 binary to decimal digits, but not with binary computer here you see is a recurring digit. So, this will recur after 0.2 it will recur know.

So, it is a recurring and it cannot be represented accurately in computer. So, these are problems many. So, there is called round off error. So, computer has rounded off number 1.1, I thought was could be represented accurately inside the computer it cannot. 1 by 3 of course, we cannot represent accurately in decimal itself and the same thing happens for binary. So, you can play around with this numbers it is fun, but I will not spend time here. So, this how you get the float x. So, 1.1 float x will be exactly same number and 1.5 can be represent accurately inside the computer, it is 1.001 sorry 1.100, 1.1. So, you see is 1.1 in binary well 1 by 0.1 will be 1 by 2, and 1 by 2 is 0.5 yeah.

Student: (Refer Time: 15:32).

This 1 plus ap power I think power is 0, this power is 0. So, you have this power is 0 exponent is 0 exponent this exponent part how do you convert. So, let us look at it here this is simpler. So, how do you get eight? So, eight is coming basically. So, the number is 1.1 all zeroes, like this with the 52 of them. So, here 52 I had to start fading. So, how many 4? So, x is 4 know x is 4 bits. So, 4 into 12 is 48 right. So, 48 bits will be unit, so 4. So, the last 3 is not units of four three. So, this should not be there. So, there will be lot of zeros here. So, this is. So, here a 14 into; 13 into 4 is 52. So, it comes in bunch of 4. So, this is 8, this is 82 to the power 3, 2 to the power 0 2 to the power 1 2 to the power 2 2 to the power 3 and rest all zeroes that is why we have 1.8 now here why I am writing one.

So, here you see this first four this is one. So, that is one now rest all is this is 9 right this is 9. So, 9 and this repeats nine will repeat.

Student: (Refer Time: 17:17).

So, this is x, this one is x and this is binary. So, I write 2 that is binary, computer will print x x is more convenient no otherwise 52 digits you do not want to write is that clear?

Student: (Refer Time: 17:36).

Ok.

Student: (Refer Time: 17:39).

So let us do it 3.0. So, 3.0 you have to write. So, 0.3 into 10, 3.0. So, let us just convert. So, actually you have to 3 point in no 3 point zeros. So, they have to now I have to convert to binary so that. So, this is 2 to the power. So, basically we want to write 3.0 as some number times to the power exponent, now you have to. So, 3 will be. So, I have to I can of course, use the computer.

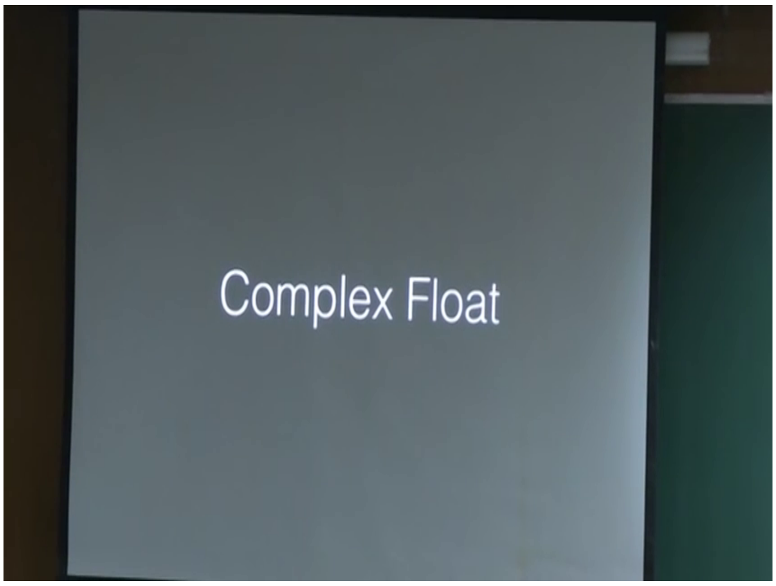
Student: 101.

101 is it.

Student: (Refer Time: 18:30) 11 (Refer Time: 18:31).

One, one but I need the float representation. So, float representation I need to do, so 11. So, this is going to be. So, to the power 1, 2 to the power 1 you take it out and then is going to be 1.5, so 1. So, this is how you will represent. So, this is going to be 1.1 into exponent will be 1, 2 to the power 1. So, check in a in your computer this how it should have. So, the exponent is a power of 2.

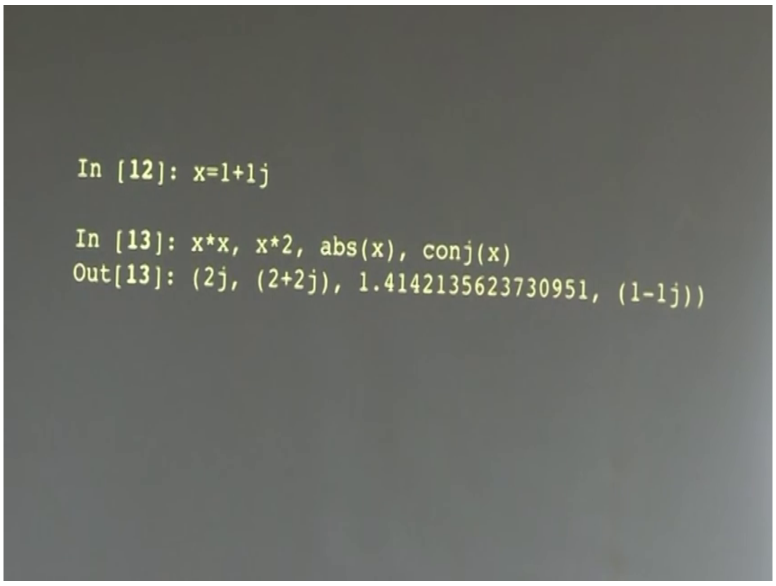
(Refer Slide Time: 19:15)



Complex Float

So, complex float. So, python will allow float operations a complex float representation is very simple, you write the number as real plus imaginary j, 1 plus 1 j and you can do operations with that.

(Refer Slide Time: 19:23)

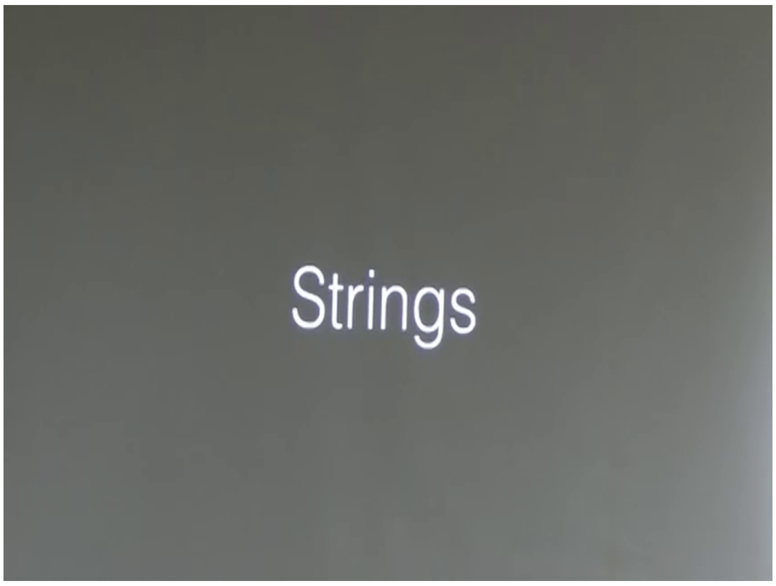


```
In [12]: x=1+1j

In [13]: x*x, x*2, abs(x), conj(x)
Out[13]: (2j, (2+2j), 1.4142135623730951, (1-1j))
```

So, I can square a number. So, x square, this x square x square is 2 j you can verify this is a now multiply by a number a absolute of x a conjugate of x ok.

So, it is very useful if you are doing quantum mechanics, it is very useful to do complex operations and is trivial ok.

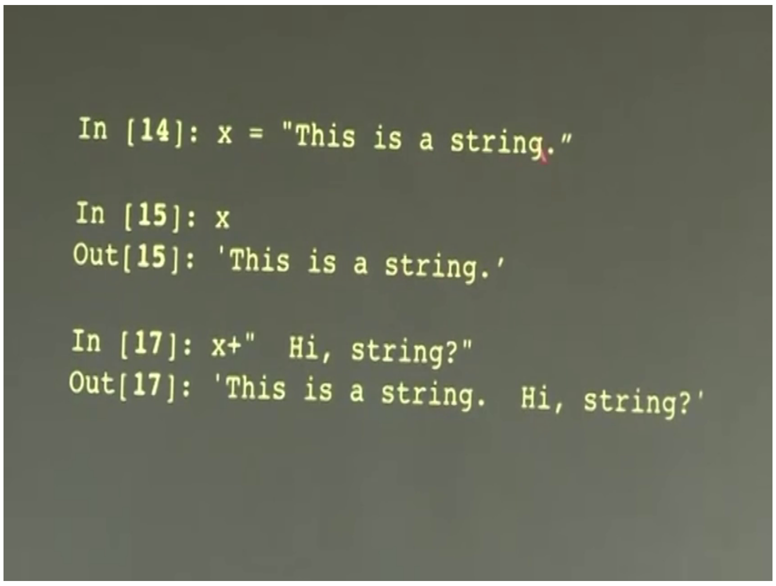
(Refer Slide Time: 19:56)



Strings

Next is strings we do not use very often, but computer scientists use all the time. In fact, everything is manipulation of strings. I feeding my program right now this computer is very building strength, displaying it and all that is string manipulation. So, how a string represented? So, take variable x and you just say within double quotes or single quote single quote double quote are interchangeable. So, x is this string ok.

(Refer Slide Time: 20:21)



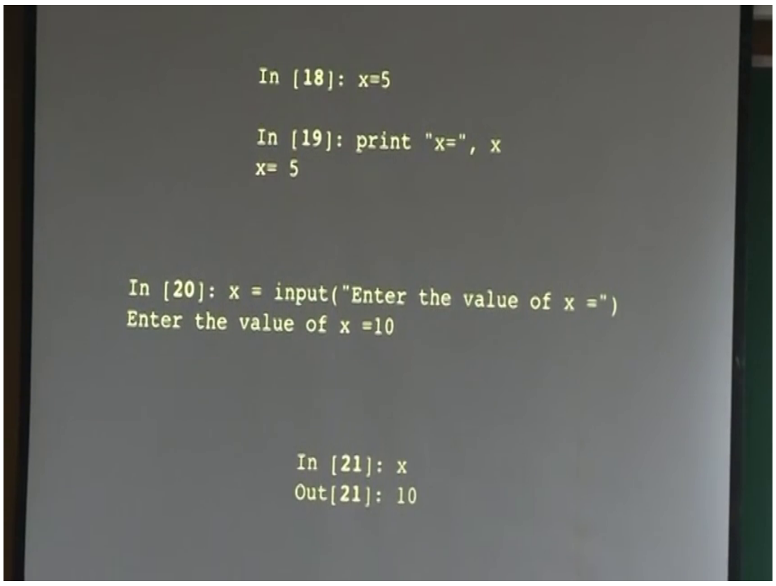
```
In [14]: x = "This is a string."

In [15]: x
Out[15]: 'This is a string.'

In [17]: x+"  Hi, string?"
Out[17]: 'This is a string.  Hi, string?'
```

So, these is a variable x has the value this string is stored in x. So, please keep in mind if you are not used to computing when I say x equal to something it does not mean I am equating it, this means I am assigning to x this is not equal operation in math the assignment operation. So, x will get this assignment of the full string and store it we no need to worry about how it stores it stores now when I say. So, in this is straight from python. So, I just cut pasted. So, in is input and out is output, I do not have different colors on this on this thing I just cut pasted in this key node.

So, this will you will get exactly same on your computer. So, when I say x means I will I am asking what is the x, and the answer is this is string we will come within quotes saying that it is a string. Now I can add something to that string this called concatenationing you can concatenate. So, I add something to string. So, x plus you just say I need string. So, now, the new string which is bigger string is this the string high string ok. So you can add string I can delete string part of it. So, all that operation can be done, we will not do this in your course we know except for input output.

So, input output is we need some communication with the computer. So, it is very easy. So, one standard way to output variable is print, you can print it on the screen and also you can print to a file that I will not discuss right now, you can print numbers to the file or put numbers to the file that part I will discuss later I mean you have larger arrays whether data you know. So, you want to put it in a file and that will be done later right now you can output a number x equal to 5 is a print x x of course, one simple way is to just say x it will give the value of x, do not need to print.

(Refer Slide Time: 22:23)

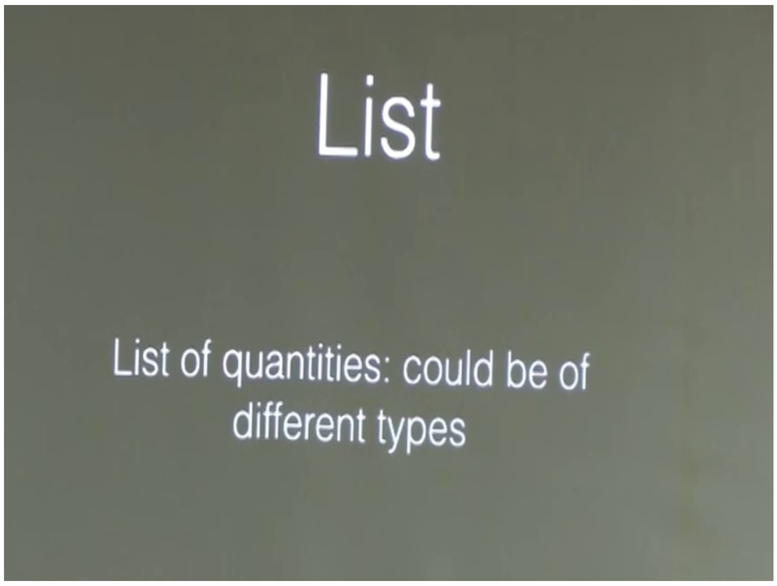


```
In [18]: x=5

In [19]: print "x=", x
x= 5


In [20]: x = input("Enter the value of x =")
Enter the value of x =10


In [21]: x
Out[21]: 10
```

So, output is for print as well as you can store it some file, but file part we will do later for input some homework may have this kind of exercise. So, you can in. So, this in, so you can put in the computer x equal to input this and. So, what will come out at the bottom will be this without this equal to, then you can put a number 10 and it will go in and store in x, but of course, there are simple ways to do what put x equal to 10.

So, in python you do not need to worry about input output, c you need to worry about it, but it is a interpreter language. So, I just say x equal to 10. What really is meaningful is to operate on data? Large data input well we are not very large data, but data let us say thousand numbers in that you need to read from a file, but that we will do later. So, x is already there. So, this part you have to enter, you enter it from a keyboard. So, you may give homework saying well put some number and do something on it. So, that is how you can do it ok.

(Refer Slide Time: 23:44)

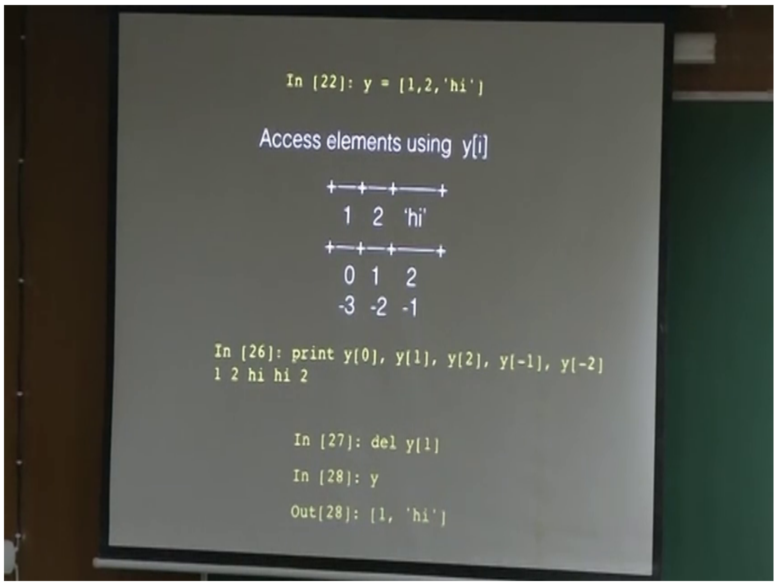


List

List of quantities: could be of different types

The next is important thing which I will just cover for completeness is called list, which again we do not use in physics, but computer scientist’s use it more often. This is a collection of objects a list is collection of objects and this objects could be of different type. So, some could be integer, some could be real, some could be string.

(Refer Slide Time: 24:12)



```
In [22]: y = [1,2,'hi']
```

Access elements using y[i]

```
+--+--+----+
 1  2 'hi'
+--+--+----+
 0  1  2
-3 -2 -1
```

```
In [26]: print y[0], y[1], y[2], y[-1], y[-2]
1 2 hi hi 2

In [27]: del y[1]

In [28]: y

Out[28]: [1, 'hi']
```

So, one example is I missed it file is it. So, i, so 1 2 hi, so this is 3 numbers one 2 hi and actually I am did seemed yeah. So, this is the string this is a list. So, my list is y equal to one. So, you put within brackets within this square brackets 1 2 and the 3 elements 1 2 and hi. Now 2 integers and one string is allowed, c will not allow this, but python allows it and how are they stored? There are 3 elements 1 2 hi and how do I access it. So, I can access by saying this is zeroth element. So, python indexing start from 0 not from 1. So, FORTRAN starts from one, but c does from 0 and python also. Python and c have very similar array operation. So, it starts from 0.

So, the indices are index know you know to catch something you need to get the roll number. So, these are roll numbers for this object, but roll number starts from 0 why 0 anyone can I mean is very obvious why should not we start from one.

Student: (Refer Time: 25:19).

No no no is in 0 is more natural.

Student: (Refer Time: 25:30).

Ok.

Student: (Refer Time: 25:34).

Ok.

Student: (Refer Time: 25:38).

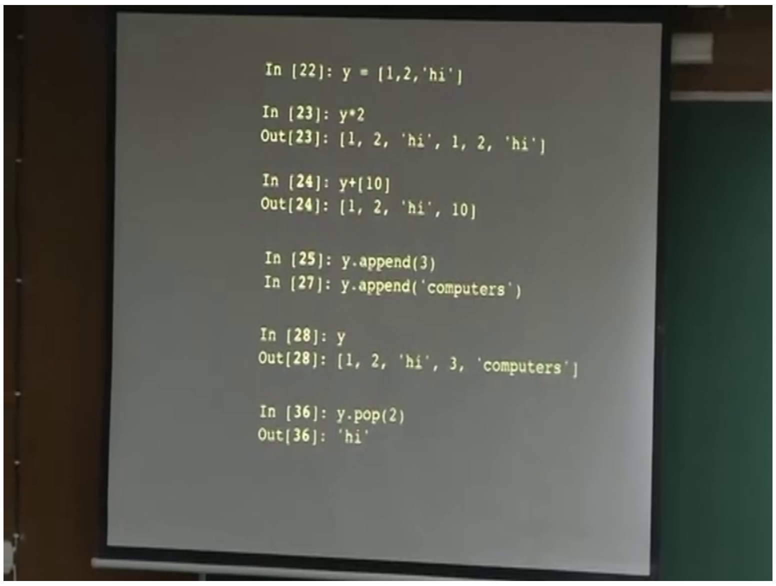
So, addresses are generated like addresses again a binary number. So, the first number binary is 0 0 0 all zeros. So, that is in address it is a valid address, my house number is 0 which is a valid address. So, that is a, so is your, you when you start that creating the addresses you start with 000 and 0001 like that. So, 0 is the first number for address and which is that is how we start. So, c started this notation and the address is 0 plus 0. So, 0 then one and 2 you can also say from minus. So, minus 1 is the last element. So, minus 1, minus 2, minus 3.

Student: (Refer Time: 26:23).

So, you can go from the right to the left. So, I can access it accessing y i. So, these are what I printed. So, y 0 is 1, y 1 is 2, y 2 is hi and y minus 1 is hi again. So, these are illustration it works. So, you can access if you the last element is y minus 1, you do not need to say if you do not know the size is very useful. You can also get the size of this array not array. So, the list y is the list call len length, len of hi will you the length now here this is in a square bracket. So, access is by square bracket and when I say. So, all the function calls are with small brackets. So, arguments will go in small brackets.

So, you have to be careful with what you are using you can delete an element. So, I can delete y 1. So, y 1 will be second element not the first element. So, if you delete the y 1 then what is left is one hi. So, you can do this operations with list, but list is since it has any kind of elements are allowed any type of elements it is not very fast.

(Refer Slide Time: 27:42)

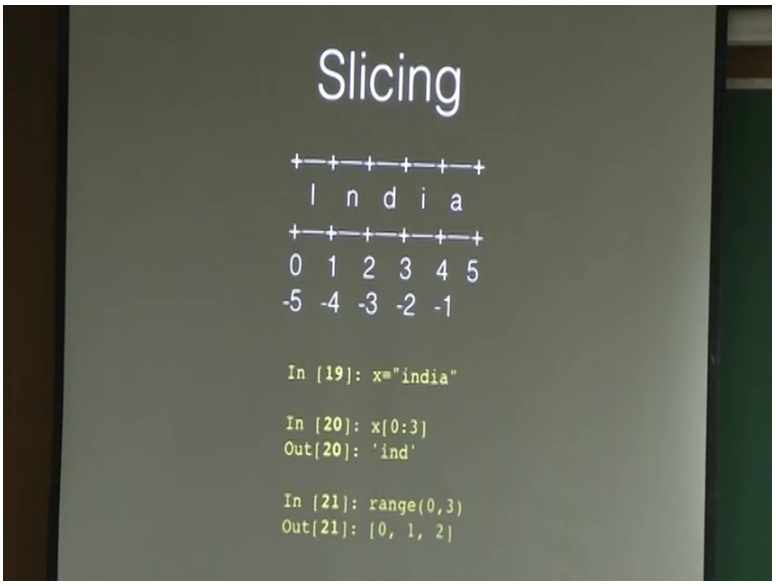


```
In [22]: y = [1,2,'hi']

In [23]: y*2
Out[23]: [1, 2, 'hi', 1, 2, 'hi']

In [24]: y+[10]
Out[24]: [1, 2, 'hi', 10]

In [25]: y.append(3)
In [27]: y.append('computers')

In [28]: y
Out[28]: [1, 2, 'hi', 3, 'computers']

In [36]: y.pop(2)
Out[36]: 'hi'
```

So, what people have done? So, I will just tell you few more operation the list, you can if you say star 2 it does not multiply because it is it is not integer or float. So, it will just double it. So, you can see that there is 6 elements and this double, you can add a new element 10 is added in this append. So, append is I appended 2 elements 3 in computer. So, append is very nice operation and you can pop you can basically access. So, this pop is access. So, pop 2 third element pop 2 is third na.

So, this is a confusion you have to get these 2 0 1 2 that is. So, these are operations for the list, now we can do slice.

(Refer Slide Time: 28:28)


Slicing

```
+—+—+—+—+—+
 I  n  d  i  a
+—+—+—+—+—+
0  1  2  3  4  5
-5 -4 -3 -2 -1
```

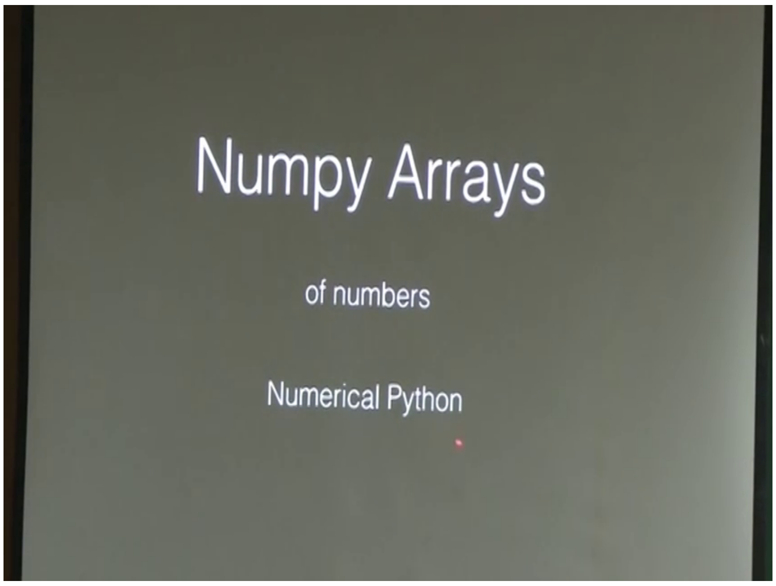
```
In [19]: x="india"

In [20]: x[0:3]
Out[20]: 'ind'

In [21]: range(0,3)
Out[21]: [0, 1, 2]
```

So, we can get part of the list not full list, but part of the list which is useful. So, this is tricky part this is confusing. So, I just want to spend bit of time on this. So, India has five characters. So, I am storing here is the string. So, it has 5, so x equal to India. So, it has 5. So, I if I say x 0 it will be 1 I, x 4 will be a. Now if I. So, I can access it like what I told in the earlier slide, but I want to get a bunch of them one short.

So, if I say 0 colon 3 what should I expect. So, you should expect 0 to 3 or 0 2 and the way to understand this is for the range this called range operation. Range operation is not starting from the middle range operation starting from the wall. So, India is stored this stored with these are boxes and these are wall of the box you know. So, this index is the wall index not the index of the house says the wall of the house all of the rooms. So, this is zeroth 0 is the first wall, second wall, third wall, fourth wall. So, I say what is contained between 0 wall and third wall and that will be ind. So, if you think like this you will get it right.

So, you have to just. So, the range object is the slightly different than indexing and that is the convention followed in python; I mean why they did it I do not know, but this something one starts and then this continues you do not want to change it. So, range is also in operation a range is also function range. So, you can give 0 3 it will give you one 2 3, you can also says the range 3 that is the same thing is 0 1 2 range 3 is not 0 to 3 0 to 2. So, these about list now we were come to arrays and this called Numpy arrays.

(Refer Slide Time: 30:48)



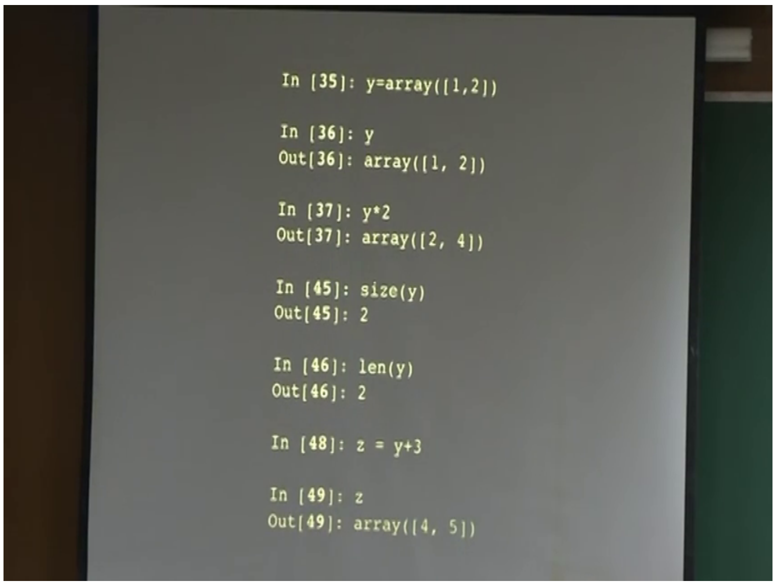
Numpy Arrays

of numbers

Numerical Python

So, what is Numpy? So, Numpy is numerical python we focus on numpy can do various things, but we will just focus on numpy is a array of numbers integer and real and this is what we will use in our course we are going to focus on numpy arrays. So, this is example of numpy. So, it how we write numpy array now you should say y equal to one bracket one comma 2 there will be a list. So, array is a library now I will just make one more remark that when we say pylab, ipython minus minus pylab. So, pylab inputs various libraries.

(Refer Slide Time: 31:22)



```
In [35]: y=array([1,2])

In [36]: y
Out[36]: array([1, 2])

In [37]: y*2
Out[37]: array([2, 4])

In [45]: size(y)
Out[45]: 2

In [46]: len(y)
Out[46]: 2

In [48]: z = y+3

In [49]: z
Out[49]: array([4, 5])
```

So one of the libraries reports is numpy numerical python. So, it has very rich operation on arrays which I will show you some of it. So, if I say one colon 2 that is a list that is part of standard python, but numpy has you have to say array within brackets. So, this is the argument of a function call array, array is the function we.

Student: (Refer Time: 31:52).

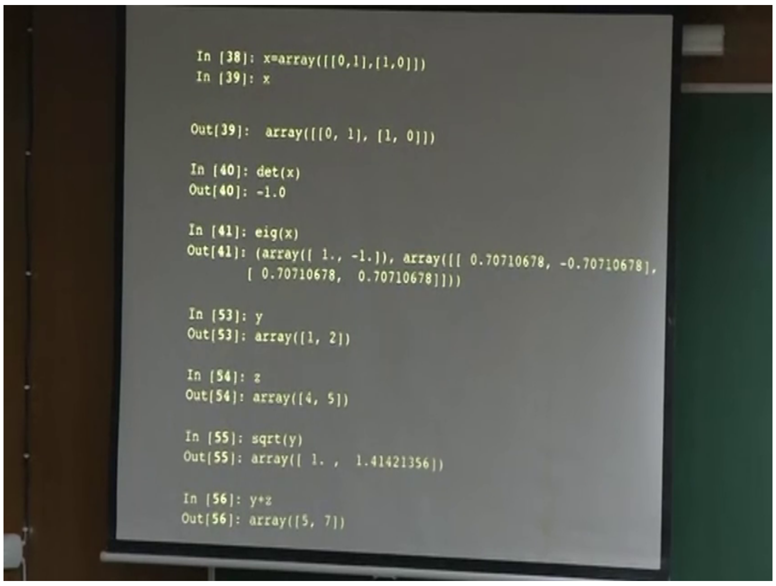
(Refer Time: 31:53) mean square in a small bracket.

Student: (Refer Time: 31:56).

Fine. So, this is it will this is same objective as list except it will be fast, and is limited to numbers for our course we will just stick to numbers. So, you should use this not one comma 2. For many operations will not be allowed on list like if I say y star 2 we saw that it is not multiplying the by the elements by 2, I wanted to multiply by 2, but this will do it.

So, if I just say y star 2 then it will multiply each element by 2. So, this is quick know if you wrote in c you have to make a loop. So, this does in one shot because multiply by 2. You can get the size of the array or you can say length, the I prefer this is preferred len len y, you can add a number to each element. So, y plus 3 is adding to each element of y and number 3. So, it will give you 4 comma 5. So, this is useful operation which you need to do later you can also create a 2 d arrays. So, 2 d array will be you put bracket.

(Refer Slide Time: 32:59)



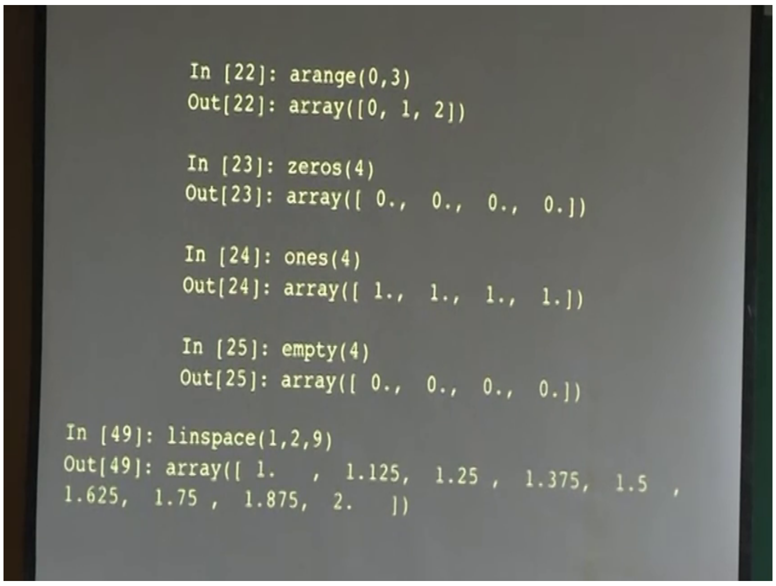
```
In [38]: x=array([[0,1],[1,0]])
In [39]: x


Out[39]:  array([[0, 1], [1, 0]])

In [40]: det(x)
Out[40]: -1.0

In [41]: eig(x)
Out[41]: (array([ 1., -1.]), array([[ 0.70710678, -0.70710678],
        [ 0.70710678,  0.70710678]]))

In [53]: y
Out[53]: array([1, 2])

In [54]: z
Out[54]: array([4, 5])

In [55]: sqrt(y)
Out[55]: array([ 1. ,  1.41421356])

In [56]: y+z
Out[56]: array([5, 7])
```

Then another bracket within is that clear. So, this is 2 d array. So, there is a square bracket here and here then there are 0 1 and 1 0. So, this is what a matrix is a not identically.

Student: 0 0 1.

0 11 0. So, it is a 2 d matrix, 0 1 0 11 0 it is an array for computer it is an array it has elements 0 11 0. So, if I say x then I will. So, this I should not have done x is if I want to know what this x is an array with these entries I can get the determinant of x. So, x it will treated as a matrix one short and determinant is minus one I can get the Eigenvalues of x an eigenvector. So, say eig x. So, Eigenvalues of 1 minus 1 is correct, and the eigenvectors are 2 of them are this 1 minus 1 and 11. So, let us do another example y is array one comma 2, z is 4 5 I can take the square root of I. Square root of I is it will take square root of each element. So, is going to give us one square root 2 and I can add these 2 arrays just do plus. So, is adding a element by element most of the operations are element by element. So, it will do this operation element by element slicing can be done with numpy arrays as well.

(Refer Slide Time: 34:34)



```
In [22]: arange(0,3)
Out[22]: array([0, 1, 2])

In [23]: zeros(4)
Out[23]: array([ 0.,  0.,  0.,  0.])

In [24]: ones(4)
Out[24]: array([ 1.,  1.,  1.,  1.])

In [25]: empty(4)
Out[25]: array([ 0.,  0.,  0.,  0.])

In [49]: linspace(1,2,9)
Out[49]: array([ 1.   ,  1.125,  1.25 ,  1.375,  1.5  ,
1.625,  1.75 ,  1.875,  2.   ])
```

And you can also create a element array with 4, but sometimes I do not know the entries, but I just want to create a array with size 4. So, I can create. In fact, I use most of other and zeros 4. So, you will create an array with 4 entries which are zeros a range 0 to 3 is very useful.

So, it is going to create from 0 to 1 2, this like range exactly a range, but is a numpy function you will put a in front ones is all ones empty is the zeros these are same as this we can also very useful function called linspace linear spacing. So, this is example, I want to create numpy array with nine elements, which are equispaced from 1 to 2 inclusive of one and 2. So, what is the difference between that consecutive numbers? 1 by 8. So, these are the entries 1, 1.125. So, linspace is a very useful function I can create an x axis with linspace points so.

Student: (Refer Time: 35:33).

Append.

Student: (Refer Time: 35:39).

Yeah, yeah appending here numpy arrays a size is fixed a you cannot append as far as I know numpy arrays you cannot append, numpy arrays append will not work for numpy array you cannot add an element in that.

So, you have to start you can start with the list now I may be wrong in this. So, you can start with the list append, then you can convert it to numpy array. So, you can convert a list to a numpy array and so that is what I did at one point. So, that I will show you in the next class I will do that. So, is that ok.

Student: (Refer Time: 36:29).

Just one. So, let us answer his question first. So, linspace is. So, the 3 three arguments first is the starting number second is the last number and how many points will be there including these 2 points, so the 9 points. So, how many. So, you create a size of the array create an array of size 9. So, that is the third argument. So, I want a size 9 array with 1 and 2 and all the elements must be equispaced. So, this is the answer some more one more question.

Student: (Refer Time: 37:13).

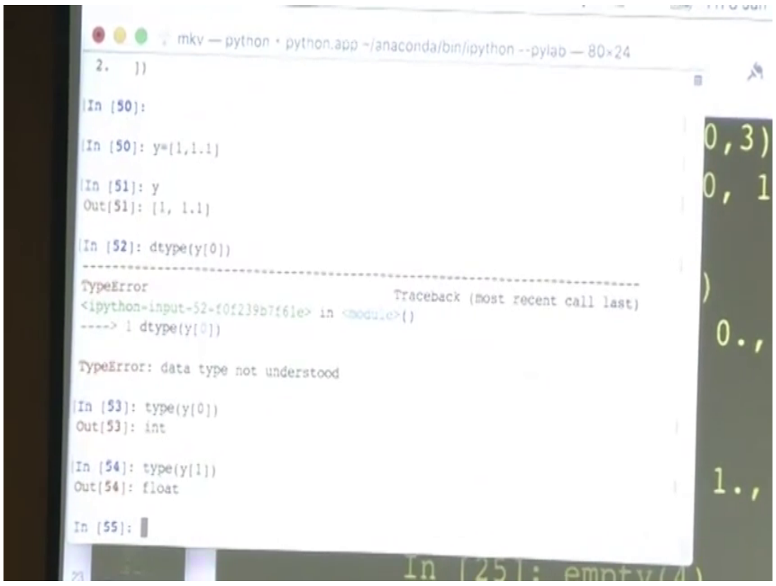
Yeah yeah. So, you can put let for example, here this will treat as integer, but it will converted to float if required.

Student: I am telling (Refer Time: 37:27).

For a for numpy array not string, string is well string is possible, but I have not worked with that, but you can put. So, I can say y equal to one comma 1.1, this is like what you want. So, this is fine, but you will convert one to 11.0 it does some things inside you can do the type. So, I am not sure what that is d type. So, to get a type you will get this is integer float.

So, right now it is storing as re integer and float, but we required it will.

(Refer Slide Time: 38:14)


mkv — python • python.app ~/anaconda/bin/ipython --pylab — 80×24

```
 2.    ])

In [50]:

In [50]: y=[1,1.1]

In [51]: y
Out[51]: [1, 1.1]

In [52]: dtype(y[0])
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-52-f0f239b7f61e> in <module>()
----> 1 dtype(y[0])

TypeError: data type not understood

In [53]: type(y[0])
Out[53]: int

In [54]: type(y[1])
Out[54]: float

In [55]:
```

So, a python is not strict typing. So, type will be changed according to the need same variable y or let us say. So, this is the example I think I will not start the next set of slides, I can say 5 the same x I can put 11.0. So, right now x is integer, but now it will become float same x which is not allowed in c, we cannot store a float number in integer I could also do a string. So, it is not a strict typing. So, that is why it is easy to program you do not need to worry about types, but also is inefficient and prove to errors also know.

So, some variable you thought it is 5, but somewhere you made it 11.0 or string. So, you need to be careful. Typing is very useful for large codes stick typing a one for efficiency a second is errors, you do not do errors now this part I will not delve into it. So, following typing is useful, but python the power is because you do not type it, means you understand what integer float string they are called type what type of variables they are and python says well I do not care I will whatever is assignment it will the type will of the variable will be of that type. So, next class we will start working on the structure of the control structures in python.

So, you can do if loops actually I have 2 minutes. So, I, but I am going slow actually I should go bit faster. So, you can do if while loop for loop you can write functions now there will be homework based on that. So, I will assign the homework, but Monday I will cover hopefully all of it, is very simple you can do if statements for statements. So, you need. So, every any language will have 3 structures. So, one thing is of course, these are called simple structures. So, set of statements one after the other, second is branching statements. So, we need to take a decision know. So, if something is true then you do something else, do something else. So, python will have this branching thing and repetitive.

So, something repeating, so you should allow. So, I should mean not be writing for I equal to one to this for I equal to 2 to this, for I equal to 3 to that. So, this is a, for I equal to 1 to 10 do this. So, this called loops and the others important thing is called function. So, something you want to use again and again like sign function. So, I do not want to be writing that set of lines to compute sign of 5, then for sign 15 or sign 1.5, it should not be writing this again and again. So, you write the function which you can call whenever you need it. So, these are constructs we need it for this course. So, this I will do in the next class.