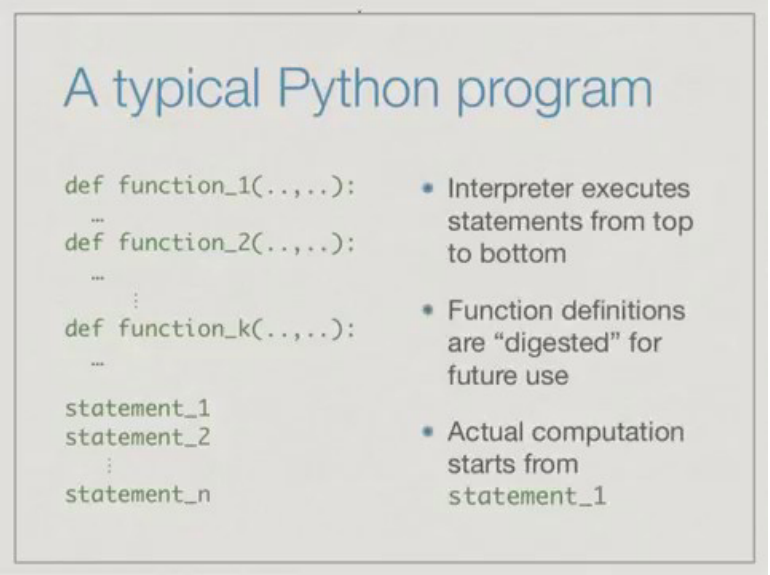
Programming, Data Structures and Algorithms in Python
Prof. Madhavan Mukund
Department of Computer Science and Engineering
Chennai Mathematical Institute,Madras

Week - 02
Lecture - 01
Assignment statement, Basic types – int, float, bool

Last week, we were introduced to notation of algorithms using the gcd example. We also saw informally some python code, which we could understand but we have not actually been introduced to formal python syntax. Let us start with some real python step.

(Refer Slide Time: 00:17)

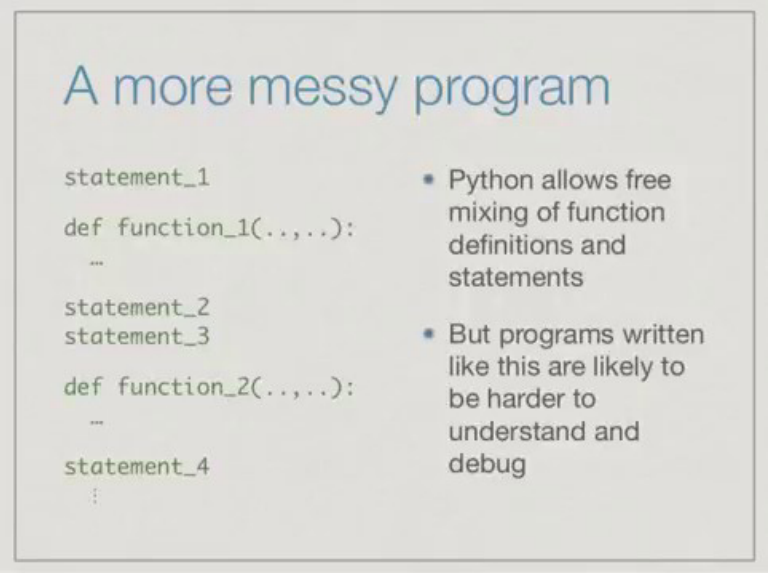


A typical python program would be written like this, we have a bunch of function definitions followed by a bunch of statements to be executed. So remember that we said python is typically interpreted, so an interpreter is a program, which will read python code and execute it from top to bottom. So, the interpreter always starts at the beginning of you of python code and reads from top to bottom.

Now, function definition is a kind of statement, but it does not actually result in anything happening, the python interpreter merely digests the function kind of remembers the function definition. So that later on if an actual statement refers to this function it knows what to do. In this kind of organization the execution would actually start with the statement which is called statement 1. So, first you will digest k functions and then start

executing 1, 2, 3, 4 up to statement n.

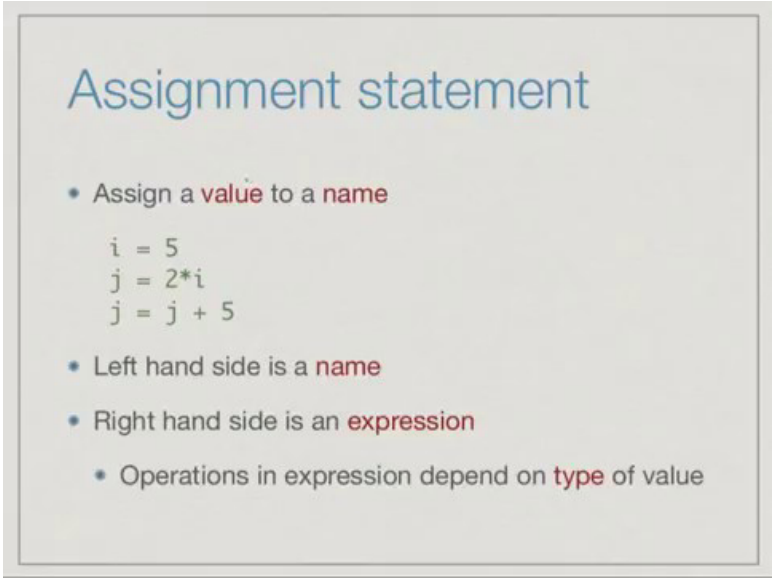
(Refer Slide Time: 01: 09)



Now there is no reason to do this. So, python actually allows you to freely mix function definitions and statements, and in fact, function definitions are also statements of a kind its just they do not result in something immediately happening, but rather in the function been remembered.

But one of things that python would insist is that if a function is used in a statement that has to be executed that function should have been defined already; either it must be a built in function or its definition must be provided. So, it may use this kind of jumbled up order, we have to be careful that functions are defined before they are used. Also jumbling up the order of statements and function definitions in this way, makes it much harder to read the program and understand what it is doing. Though it is not required by python as such as, it strongly recommended that all function definition should be put at the top of the program and all the statements that form the main part of the code should follow later.

(Refer Slide Time: 02:07)



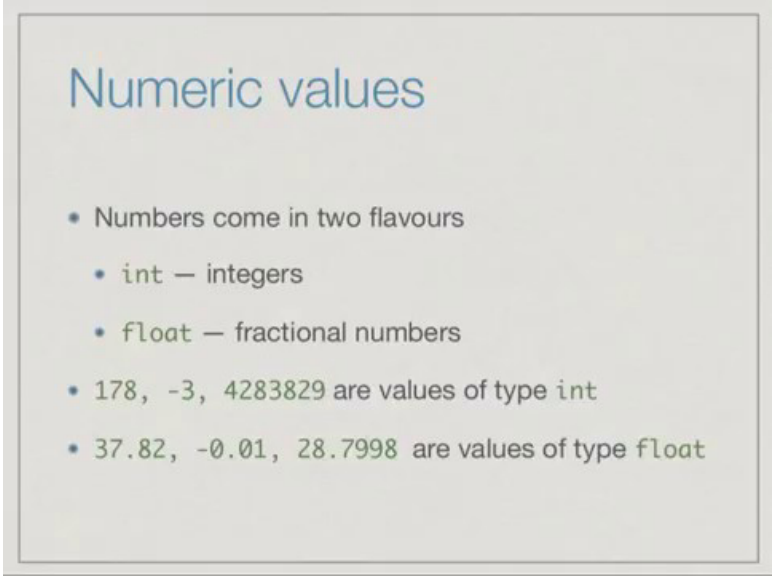
What is a statement ? The most basic statement in python is to assign a value to a name. So, we see examples and we have seen examples and here are some examples. So, in the first statement i is a name and it is assigned a value 5; in the second statement, j is a different name and it is assigned an expression 2 times i. So, in this expression, the value of i will be substituted for the expression i here. So, if i have not already been assigned a value before, python would not know what to substitute for i and it would be flagged as an error.

When you use a name of the right hand side as part of an expression, you must make sure that it already has a valid value. And as we saw, you can also have statements which merely update a value. So, when we say j is equal to j plus 5 it is not a mathematical statement, where the value of j is equal to the value of j plus 5. But rather that the old value of j which is on the right hand side is updated by adding 5 to it and then it gets replaced as a new value j. This is an assignment statement, this equality assigns the value computed form the right hand side given the current values of all the names if the name given on the left hand side, with the same name can appear on both sides.

The left hand side is a name and the right hand side in general is an expression. And in the expression, you can do things which are legal, given the types of values in the expression. So, values have types; if you have numbers, you can perform arithmetic operations; if you have some others things, you can perform other operations. So, what

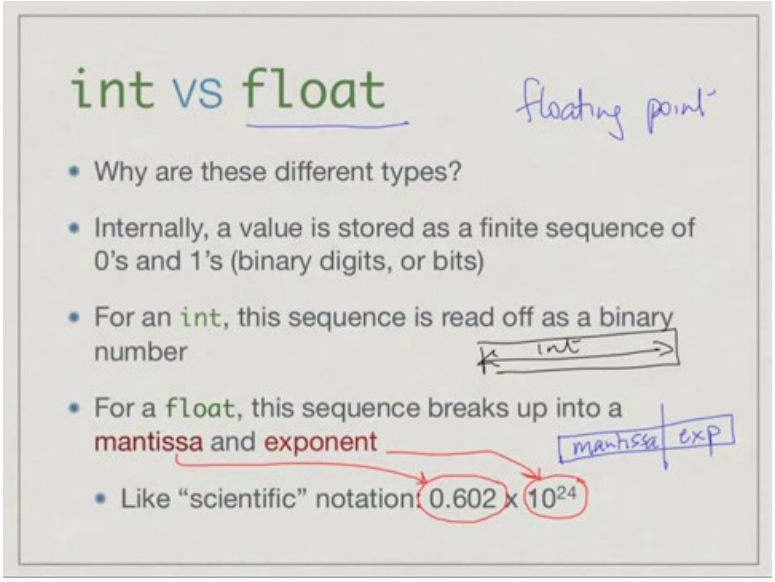
operations are allowed depend on the values and this is given technically the name type. So, when we said type of values it is really specifying what kinds of operations are legally available on that class of values.

(Refer Slide Time: 04:01)



So the most basic type of value that one can think of are numbers. Now in python and in most programming languages numbers come in two distinct flavours as you can call them integers and numbers which have fractional parts. So, in python these two types are called int and float. So, int refers to numbers, which have no decimal part, which have no fractional part. So, these are whole numbers they could be negative. So, these are some examples of values of type int. On the other hand, if we have fractional parts then these are values of type float.

(Refer Slide Time: 04:43)



Normally in mathematics we can think of integers as being a special class of say real numbers. So, real numbers are arbitrary numbers with fractional parts integers are those real numbers which have no fractional. But in a programming language there is a real distinction between these two and that is, because of the way that these numbers are stored. So, when python has to remember values it has to represent this value in some internal form and this has to take a finite amount of space.

If you are writing down, say a manual addition sum you will write it down on a sheet of paper and depending on a sheet of paper and the size of your handwriting there is a physical limit to how large a number you can add on that given sheet of paper. In the same way any programming language, will fix in advance some size of how many digits it uses to store numbers and in particular as you know almost all programming languages will internally use a binary representation. So, we can assume that every number whether an integer or real number is stored as a finite sequence of zeroes and ones which represents its value.

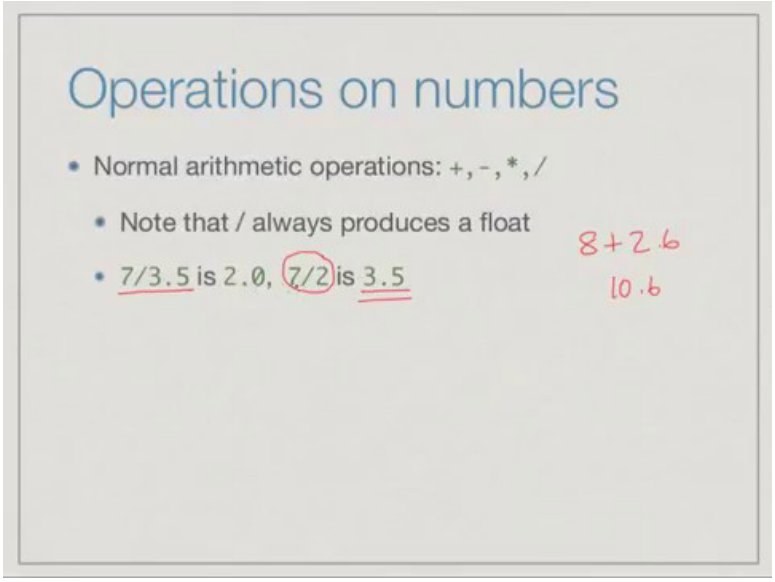
Now, if this happens to be an integer you can just treat that binary sequence as a binary number as you would have learnt in school. So, the digits represent powers of 2, usually there will be one extra binary digit 0 or 1 indicate whether it is plus or minus and they may be other more efficient ways of representing negative numbers, but in particular you can assume that integers are basically binary numbers.

They are just written as integers in binary notation. Now when we come to non integers then we have two issues one is we have to remember the value which is the number of digits which make up the fractional part and then we have to remember the scale. So, think of a number in scientific notation right, so, your normally have two parts when we use things in physics and chemistry for instance, we have the value itself that is what are the components of the value and we have how we must shift it with respect to the decimal point. So, this says move the decimal point 24 digits to the right.

So, this first part is called the mantissa right and this is called the exponent. So, when we have the number in memory if it is an int, then the entire string is just considered to be one value where as if we have block of digits which represents a float. Then we have some part of it, which is the mantissa, and the other part, which is the exponent.

The same sequence of binary digits if we think of it as an int has a different value and if we think of it as a float has a different value. So, why float you might ask. Float is an old term for computer science for floating point; it refers to the fact that this decimal point is not fixed. So, an integer can be thought of as a fixed decimal point at the end of the integer a floating point number is really a number where the decimal point can vary and how much it varies depends on the exponent. So there are basically fundamental differences in the way you represent integers and floating point numbers inside a computer and therefore, one has to be careful to distinguish between the two. So, what can we do with numbers?

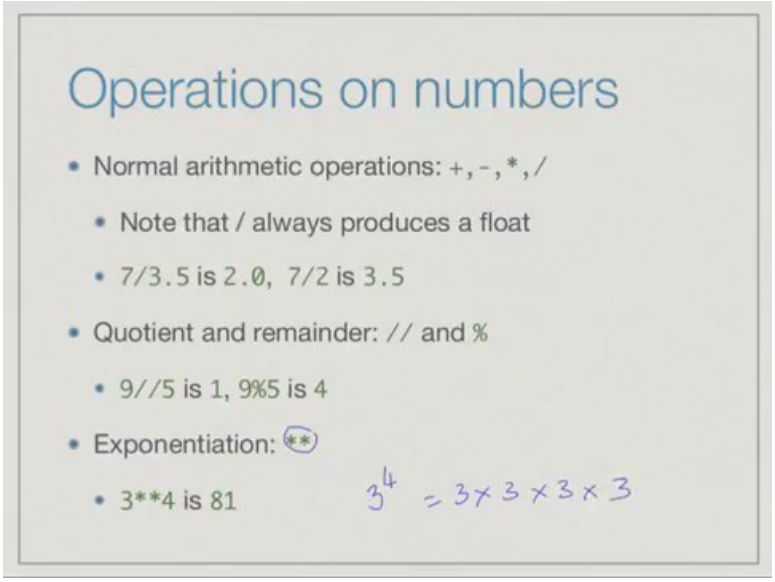
(Refer Slide Time: 07:59)



Well we have the normal arithmetic operations plus, minus, multiplication, which has a symbol star modern x and division, which has a symbol slash. Now notice that for the first three operations it is very clear if I have 2 ints and I multiply them or add them or subtract them I get an int. But I have 2 floats I will get a float, division, on the other hand will always produce a float if i say 7 and divided by 2 , for instance where both are ints I will get an answer 3 point 5.

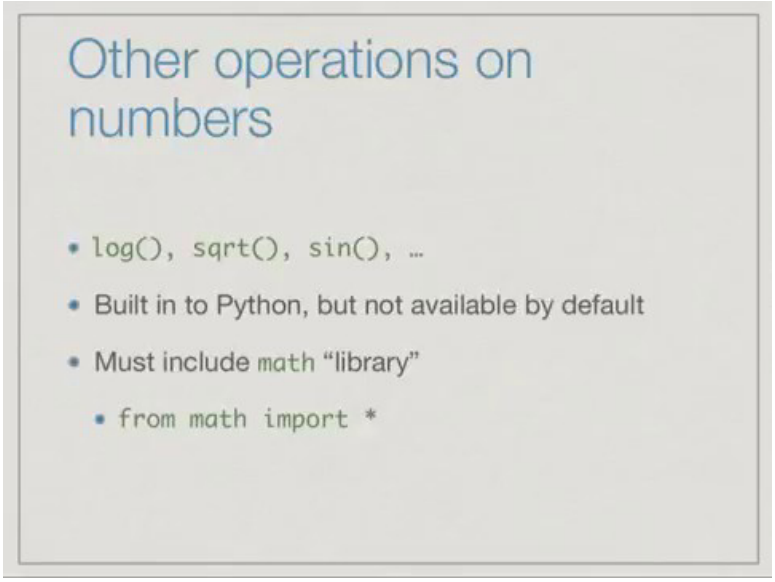
Now in general python will allow you to mix ints and floats, so i can write 8 plus 2 point 6 even though the left is an int and right is a float and it will correctly give me 10 point 6. In that sense python respects the fact that floats are a generalized form of int. So, we can always think of an int as being a float with a point 0 at the end. So, we can sort of upgrade an int to a float if you want to think of it that way and incorporate with an expression, but division always produces floats. So, 7 divided by 3 point 5 as an example of a mixed expression, where I have an int and float and this division results in 2 point 0 and 7 by two results in 3.5.

(Refer Slide Time: 09:15)



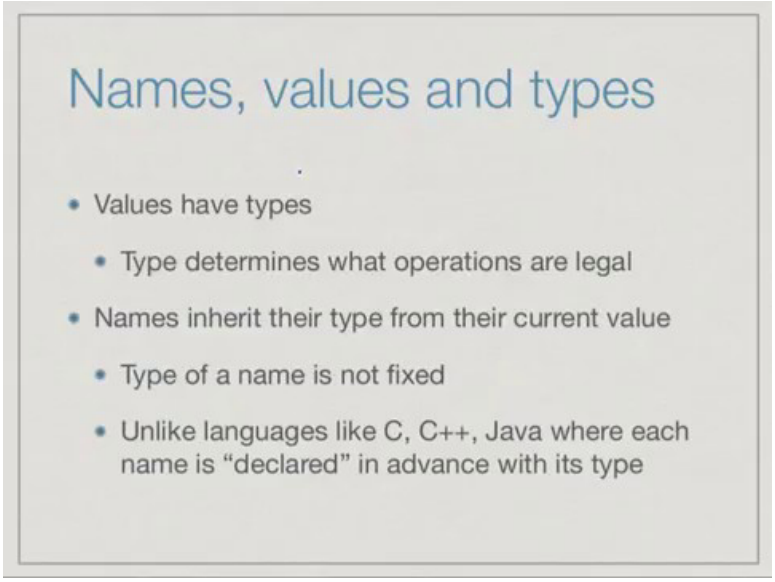
Now there are some operations where we want to preserve the integer nature of the operands. We have seen one repeatedly in gcd which is the modulus operator, the remainder operator. But the req corresponding operator that go through the reminder is the quotient operator. So, if I use a double slash it gives me the quotient. So, 9 double slash 5 says how many times 5 going to 9 exactly without a fraction and that is 1 because in a 5 times 1 is 5 and 5 times 2 is 10 which is bigger than 9 and the remainder is 4. So, 9 percent 5 will be 4. Another operation which is quite natural and common is to raise one number to another number and this is denoted by double star. 3 double star 4 is what we would write normally as 3 to the power 4 is 3 times, 3 times, 3 four times right and this is 81.

(Refer Slide Time: 10:12)



Now there are more advanced functions like log, square root, sin and all which are also built into python, but these are not loaded by default. If you start the python interpreter you have to include these explicitly. Remember we said that we can include functions from a file we write using this import statement. There is a built in set of functions for mathematical things which is called math. So, we must add from math import star; this can be done even within the python program it does not have to be done only at the interpreter. So, when we write a python program where we would like to use log, square root and sin and such like, then we should add the line from math import star before we use these functions.

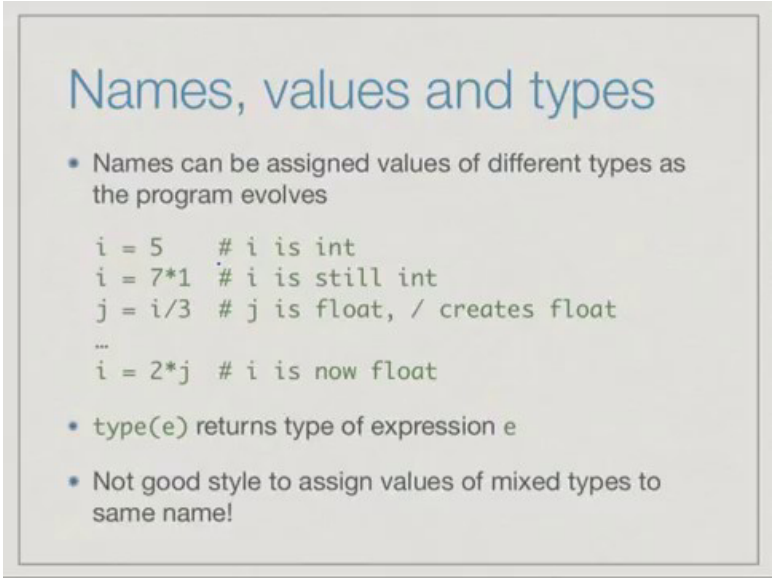
(Refer Slide Time: 10:58)



We have seen three concepts - names which are what we use to remember values, values which are the actual quantities which we assign to names and we said that there is a notion of a type. So, type determines what operations are legal given the values that we have. So, the main difference between python and other languages is that names themselves do not have any inherent type. I do not say in advance that the name i as an integer or the name x is a float. Names have only the type that they are currently assigned to by a value that they have.

The type of a name is not fixed. In a language like C or C++ or Java we announce our names in advance. We declare them and say in advance what type they have. So, if we see an i in an expression we know in advance that this i was declared to be of type int this x was declared to be of type float and so on. Now in python this is not the case.

(Refer Slide Time: 12:00)



So, let us illustrate this with an example. So, the name main feature of python is that a name can be assigned values of different types as the program evolves. So, if we start with an assignment i equals to 5 since 5 is an int i has a type int. Now if we take an expression, which produces an int such as 7 times 1, i remain an int. Now if we divide the value of i by 3. So, at this point if we had followed the sequence i is 7. So, 7 by 3 would be 2.33 and this would be a float.

Therefore, because the operation results in a float at this point j is assigned the value of type float. Now if we continue at some later stage we take i and assign it to the value 2 times j, since j was a float i now becomes float. In the interpreter there is a useful function called type. So, if you type the word type and put an expression and either a name or an expression in the bracket, it will tell you actually type of the expression.

Now although python allows this feature of changing the type of value assigned to a name as the program evolves, this is not something that is recommended. Because if you see an i and sometimes its a float and sometimes its an int it is only confusing for you as a programmer and for the person trying to understand your code. The same way that we said before that we would like to organize our python code so that we define all functions before we execute statements, it is a good idea to fix in advance in your mind at least, what different names stand for and stick to a consistent way of using these either as ints or as floats.

(Refer Slide Time: 13:54)

Let us execute some code and check that what we have been saying actually happens. So, supposing we start the python interpreter and we say i is equal to 5, then if we use this command type i it tells us type of i. So, it returns it in the form which is not exactly transparent, but it says that i is of class int. So, you see the word int, if i say j is equal to 7 point 5 and i ask for the type of j then it will say j is of class float. So, the names int and floats are used internally to signify the types of these expressions. Now if I say i is equal to 2 times j as we suggested i has a value 15 point 0, because j was a float and therefore, the multiplication resulted in a float and indeed if we ask for the type of i at this point it says that i is now a float.

The point to keep in mind is that the name is themselves do not have fixed types they are not assigned types in advance. It depends on the value that is currently stored in that name according to the last expression that was assigned.