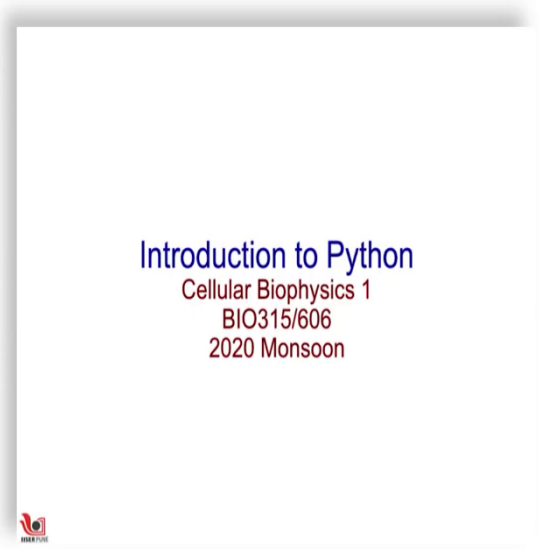
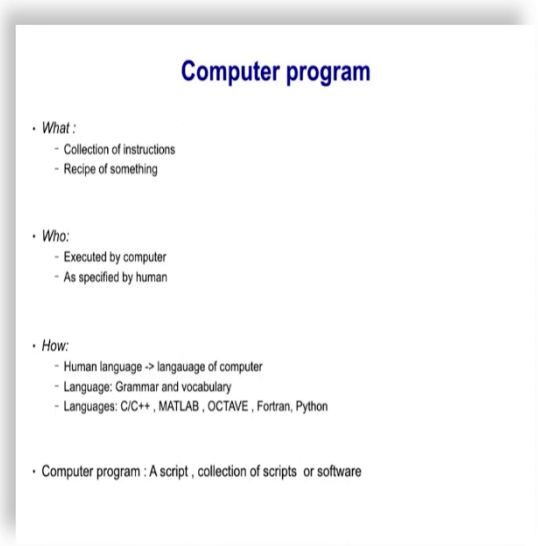
Cellular Biophysics
Doctor Chaitanya Athale
Department of Biology
Indian Institute of Science Education and Research, Pune
Python Programming – Part - 01

(Refer Slide Time: 00:16)



Hi, welcome to BIO315. Today, we are going to make a slight interlude, a slight diversion from what we have been talking about between order of magnitude estimates, numbers, quantification, technique, microscopy and on the way to starting with the cytoskeleton. So, today, I want to talk to you a little bit about an introduction to someone whom you already know, Python. Those of you who have done your BSMS coursework here, have already been exposed to Python programming. And I am going to very briefly touch upon this, just as a reminder to you of what you have relatively done.

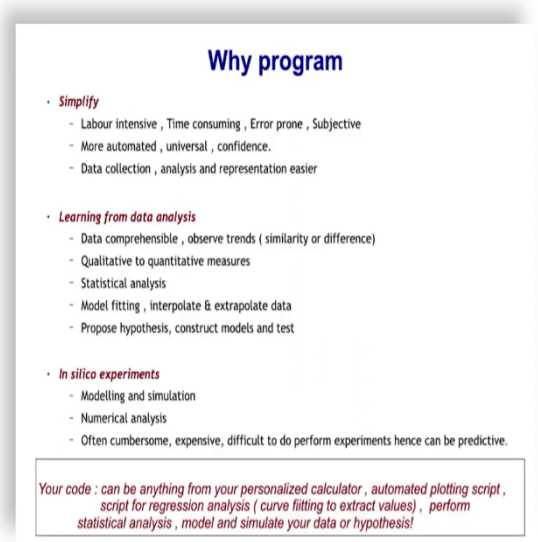
(Refer Slide Time: 01:08)



So, in a sense it is always important to ask this question? What? Who? How? What is a computer program? So, typically every child now learns computer programming. So collection of instructions, recipe of something, who executes it, by the computer, specified by a human. How is it done? Human language to language of computer, a conversion, which has grammar and vocabulary. And there are some examples of such languages. And you are familiar with C, C++. Some of you have used MATLAB, Octave, Fortran, Python, new languages keep getting created.

As you know, languages are defined as either high level or low level depending on whether the difference between human language and computer language is greater or smaller. In other words, computers are stupid logic objects. They need to be told how to do it, we have a more complex set of logical instructions, we need to convert those from our human language into computer language. And in that sense, Python is a high-level language. Programs could indeed be a script or a collection of scripts.

(Refer Slide Time: 02:25)



So, one of the most obvious reasons to program is indeed to simplify your task. This is especially relevant in quantitative biology where a lot of what we do can be labour intensive, time consuming. and much more importantly, because we are not lazy, is that those measurements or quantifications might be error prone or subjective. Meaning, if person A does it and then person B does it, you get different answers.

So, programming is critical in automating tasks, providing in some sense universality to the method, at least transiently, and improving our confidence in the reproducibility of the method that has been used. You, at the same time, you will also need to program in order to collect data to analyse it and make clear representations.

Now, data analysis is one of the first few things that we are going to be doing. And the purposes of data analysis are to make data comprehensible to observe trends, find out whether there are, find out what the quantitative and qualitative measures are, perform statistical analysis, and finally do what many of us implicitly do in our minds as humans, that is we fit a model.

In other words, when we see that the sun has risen, we expect the intensity of light to increase and then stay steady for at least 5 to 10 hours. This is our day-to-day experience and this is based on a model that we have developed of how the sun increases, stays steady and goes away. So, for that, we always need to propose a hypothesis, construct it and test it.

And this is often the feature of statistical testing. But interestingly enough, one can extend this to the larger question of physical methods and the role of computation in it. So, for today

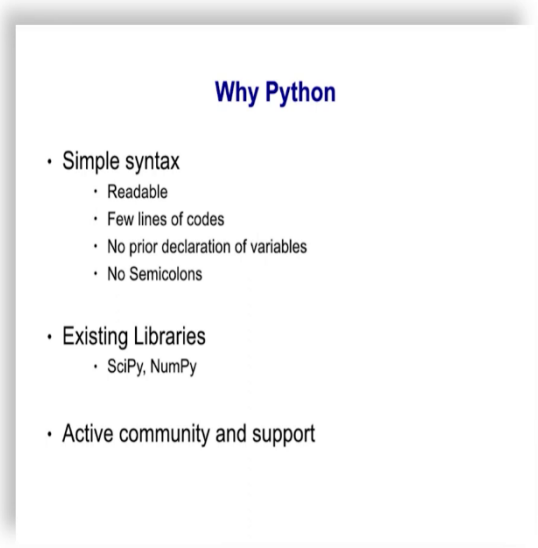
we are going to, in a way, also motivate a little bit of what we really want to do with this which is even beyond model fitting, beyond simple statistics to predict what is going to happen in the future.

And to do this we are going to model something, we are going to model the simplest cases, obviously, to start with. And we can even, in future, do numerical analysis and use those model predictions as we have been sort of off and on talking in the course, to verify by comparing against data.

So, what is your code? It can be anything. It can be your personalized calculator that you set up to do a shortcut. It can be an automated plotting script, it can be a regression analysis code, it can perform statistical analysis or it can be much more sophisticated like it could predict the birth, growth, division and death of bacterial cells, for instance.

All of that modelling and simulation is based on a hypothesis. And in some senses, that is one of the crucial aspects which form, in many places, the foundation of entire courses on the nature of modelling or mathematical and computational modelling. We will not have time to delve into that much depth. But I think that you will see the connection between what we do in the theoretical aspects of quantitative biology and biophysics, cellular biophysics, and what we can do to connect it to what you learn in programming.

(Refer Slide Time: 06:19)



So, I chose Python, and I mentioned this also in one of my posts on the course. There are many reasons to choose a particular programming language. And one can always say why

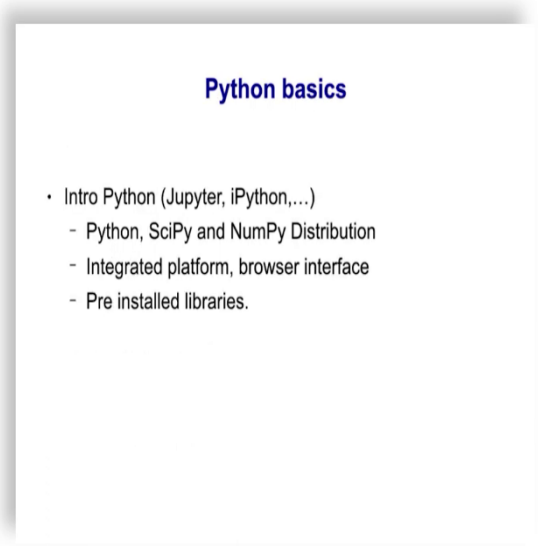
Python? Because one can of course easily justify this question, simple syntax, readable, few lines of code to do things. It is powerful in that sense.

Declarations are on the fly. In other words, variables do not, unlike in C or Fortran that maybe some of you have used before, and even if you have not used, declaration of variables is something like saying, I pre-allocate a bucket in which I am going to put some data. And you can imagine that today if I tell you, one week from now you have a quiz, you are more comfortable, so is a compiler.

But analogies apart, there are also some minor issues about syntax that are different from Python, like those semicolons. And the more interesting advantage is not just simplicity but power because linked with two very prominent scientific and numerical computing libraries SciPy and NumPy, respectively, Python has become a very serious tool for scientific computing over the last 20 years. In addition to this, there is an active community and a support system and there are forums, there are help files. So, getting started is a lot easier.

And remember the purpose of introducing Python, then bringing this into a cellular biophysics course is not to make you programmers that Infosys or Wipro or IBM is going to hire, it is as a tool that you can use to answer scientific questions. You can of course become a better programmer. That is, your upper bound is not limited by anything we do here, but this is to ensure that the minimum is satisfied.

(Refer Slide Time: 08:17)

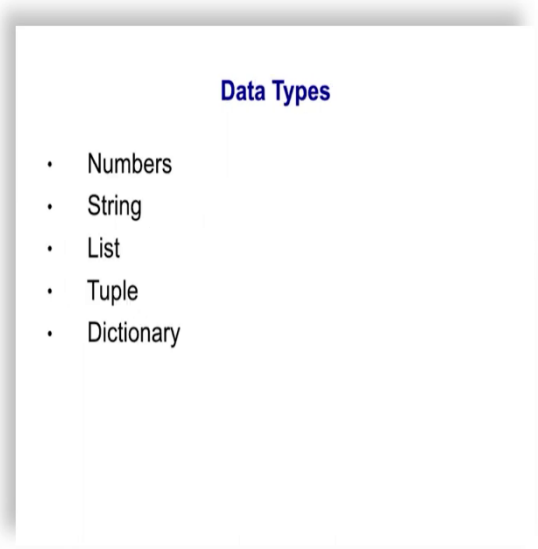


So, a few basics about Python, we are going to be using the so called Jupyter platform. Now because of the current conditions you are going to be using the Google code collaboratory for

testing and running your codes and saving it and submitting it. But ordinarily, one would normally be installing it locally on your drive, and there are distributions such as Anaconda and command line, Jupyter interfaces.

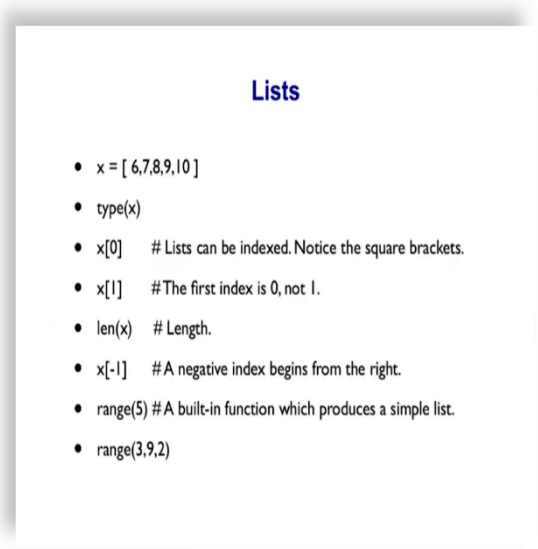
Suffice to say that Jupyter bundles by SciPy NumPy and MATLAB, the three tools that you will need to actually get your work together for this course. The interesting part is that the interface is through the browser. And with pre-installed libraries, Python really becomes very powerful.

(Refer Slide Time: 09:09)



So, let us start off with some very, very simple things. What are the data types for numbers, strings and what are lists tuples and a dictionary.

(Refer Slide Time: 09:18)

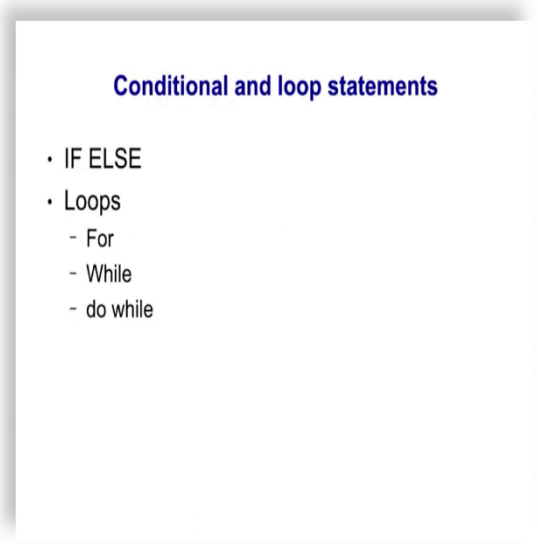


So, lists are these equivalences in square bounding boxes of numbers. They could be integers, they could be floats, they could be decimal numbers in other words. Calling that x, x is equal to square bracket something, tells you the type of variables that are stored in it. You must notice that the index, in other words, the counting scheme like in most parts of India and in many parts of the western world goes from left to right. Not in Japan, not in Arabia, left to right.

The smallest number is on the left, 6 in this case in this list. It is indexed, in other words, it is called, referred to, its address is at position 0. First number is 0. First position is 0. It is also sometimes called base 0. This is emphasized by if you were to actually type in this array list and call x square bracket 0 versus x square bracket 1, you get two different answers.

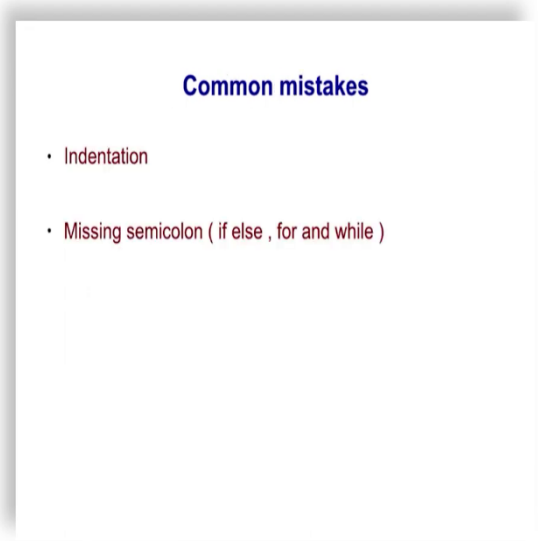
Try them out for yourself. len is a short form for length. This command calls up the length of the array. If you do minus 1, the negative index begins from the right. So you go from instead of right to left, you go from left to right. Range is a built-in function which produces a simple list.

(Refer Slide Time: 10:52)



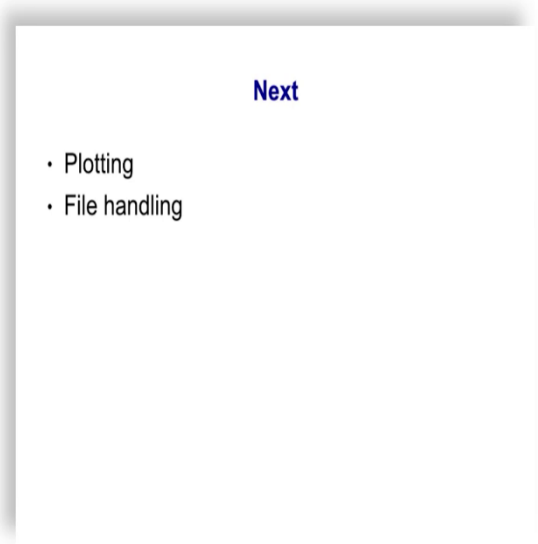
On the other hand, conditional statements, typical being IF ELSE, allow you to set up a condition that says if some status condition is satisfied, perform such and such task, else do not perform it or go ahead. The classic loops that you see in programming languages are also found in Python, for, while and some slightly unusual, called do while.

(Refer Slide Time: 11:19)



Some of the most common mistakes that you will find yourself making when your code will not run, relate to indentation meaning to say, how many spaces you give from the left margin to your code so that a block of code falls into a common point. In addition, you may also have missing semicolons, after if statements or else statements, false statements and while statements.

(Refer Slide Time: 11:51)



And we are going to talk about a few of these things by running through some examples because given the simplicity of Python, the nicest thing we can do is actually learn by doing. Thank you very much, and we will continue with the next module.