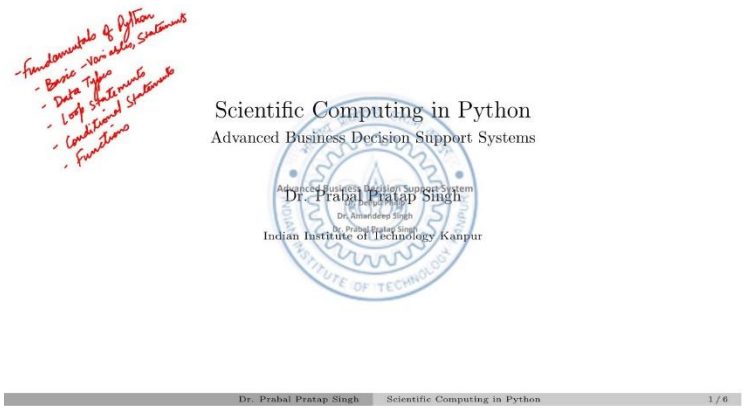
Advanced Business Decision Support Systems
Professor Deepu Philip
Department of Industrial Engineering and Management Engineering
Indian Institute of Technology, Kanpur
Professor Amandeep Singh
Imagineering Laboratory
Dr. Prabal Pratap Singh
Indian Institute of Technology, Kanpur
Lecture 33
Scientific Computing in Python

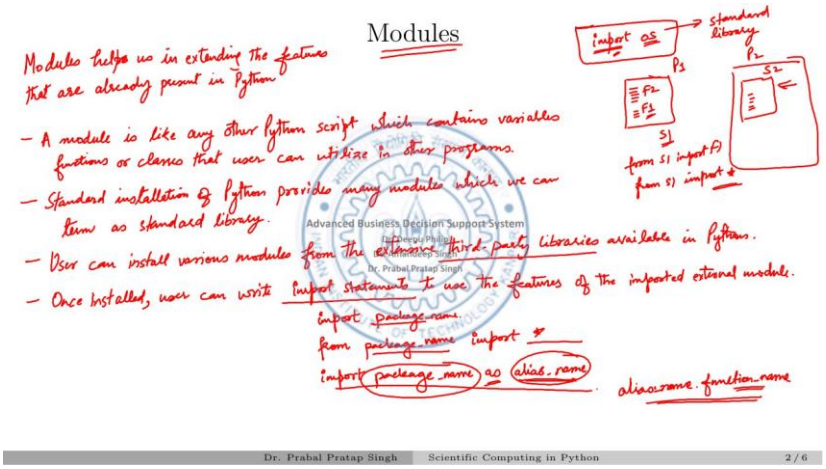
Hello everyone, I welcome you all to the lecture series on Advanced Business Decision Support Systems. I am Prabal Pratap Singh from IIT Kanpur and we are learning how to create different Decision Support Systems.

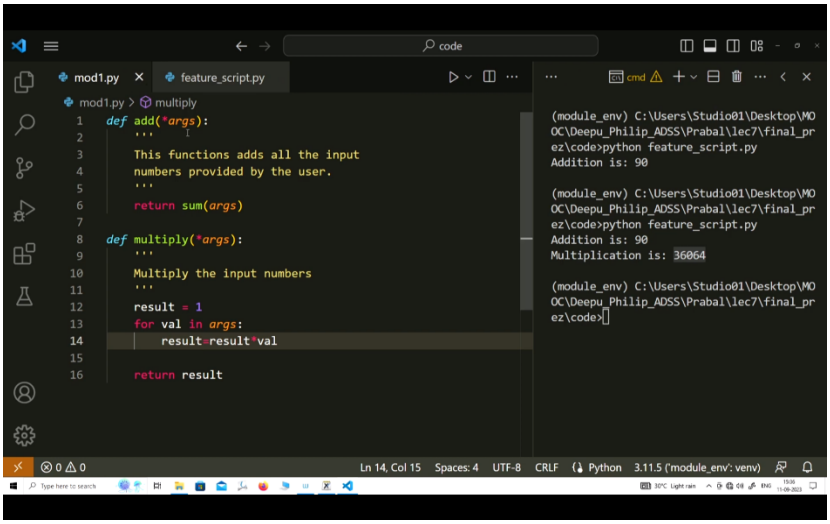


So, till now, we have covered the fundamentals of python like basics, variables, statements, how to write these things. Then, we learnt about data types, then we also learnt about different loop statements, conditional statements and we also looked upon how to create functions in python. So, today let us start looking at how we can perform various computations using the python programming language.

So, there are various kinds of scientific computing that we need to create different kinds of applications and even if we are creating a general type of Decision Support Systems, we need various computations that can give us different results to showcase on our Decision Support Systems. So, to do that, we need to use various functions or create them to perform different kinds of scientific computations.

So, we will learn today how to do these computations, but before that we need to see how to use various functions or routines that are pre-built in the python standard library or in the third-party extensions that python consist of.





So, let us move forward and try to learn about what Modules are. So, Modules help us in extending the features that are already present in the python standard in python. So, by using different basic functions or different standard libraries, we can create more complex or advanced functions and we can store them to use these advanced functions in our different kinds of programs.

So, a Module is like any other python script which contains variables, functions or classes that users can utilize in other programs. So, you have already seen that I was using this statement 'import OS' in my earlier codes when we were learning different types of functions or data types, we are using this statement. So, what this statement is doing is, it is using this Module, this OS Module which is pre built in the standard library. So, the OS is available in the standard library of python. So, there are many other kinds of pre-built Modules available with the standard library and standard installation of python.

So, how do these Modules work and how do this import statement works, is what we are going to see today. So, let us say you are just creating your simple python script and it contains different kinds of statements and functions and all these things, but now after one day, two days or a week, you start developing a different project. So, let us say you created some function F1 here and you know that this function works very beautifully and it is highly optimized for your use case. So, now you want to perform the same operation in a different project as well. So, let us say this was project 1, and this whole directory is for project 2.

Now, in project 2, you are creating a different script, this is Script 2, this is Script 1, and now you want to use this F1 in S2. So, the basic thing you can do is, one way to do this is like you create this F1 again in this S2, you can write here, but it will be cumbersome and it is like reinventing the wheel. So, why do that? Instead of that python provides this Module functionality which will use this import statement, you can write this import statement in this S2 and it will import all the functionality of this F1 into this S2 and now you can use the same highly optimized function you have created just a week before to use in your current project. So, this is how this module functionality works. We will also see how to code this. So, some other general information about Modules is like the standard installation of python provides many Modules which we can term as standard library.

So, for these functions you usually do not require to import other things, it is just easily directly available in your script or your interpreter. Users can install various Modules from the extensive third-party libraries available in python. So, these third-party libraries are being created by the community developers or some major organizations also create these libraries and they maintain these libraries. So, that users can do their own projects and these are all open source libraries that they are using. So, we can directly import these libraries and we will see how to do scientific computation using one of the very famous libraries which is called NumPy (Numeric Python). So, once installed, users can write import statements to use the features of the imported external Module.

Now, these import statements are of different kinds. So, there is one simple statement like this import and your package name. You can also use from package name, package name is nothing but your module name, you can say it as a package name or module name. From package name import, either you can import a particular function or you can use star operator to import everything from that Module. So, lets say, if this is Script 1 and you just want to import this F1. So, you can write from S1 import F1, but what if you want all the functions or variables from this Module to get imported in your S2.

So, you can write from S1 import star. Now, what it will do is, in your S2 Module, you can directly use everything like F1 or let us say there is another function F2. So, you can just by writing this star, it is available in S2. And, one more feature available with the import statement is, import package_name as alias name. So, here what we are doing is, we are using some alias for this package name. So, usually a developer, who is creating this package or module, has the freedom to use any name, but what if you do not want to use that name while writing your own code in your script.

Then, you can use this alias name by using this as operator and this will perform all the actions on this Module using this alias name. Now, you can assess any function in this package name using alias name.function name. So, any function name that is defined in this Module can be assessed by this statement in your script. So, this is how you can use different kinds of Modules. So, let us create a basic Module and we will write a complete Module, and then we will import it.

So, let me open my coding environment. So, let us start by creating a new environment for our module functionality. Now, we have initialized our newly created environment, and we can now start writing the code. So, you can see in my file explorer that I just created this one file till now.

This is ‘mod1.py’, and this is the environment directory ‘module_env.’. Now, let us create some two basic functions which are ‘def add’, and we have already learned the star operator in the as an argument which can capture all the arguments that is provided by the user. And, let us write the '. string' which will say that this function adds all the input numbers provided by the user and let us write our only single return statement which will add all these numbers. We can add one more function which is ‘def multiply’ and we can use our ‘star args’ argument again and provide our '. string' that says multiply the input numbers and to multiply all the numbers, this ‘star args’ will create a list of all the provided numbers.

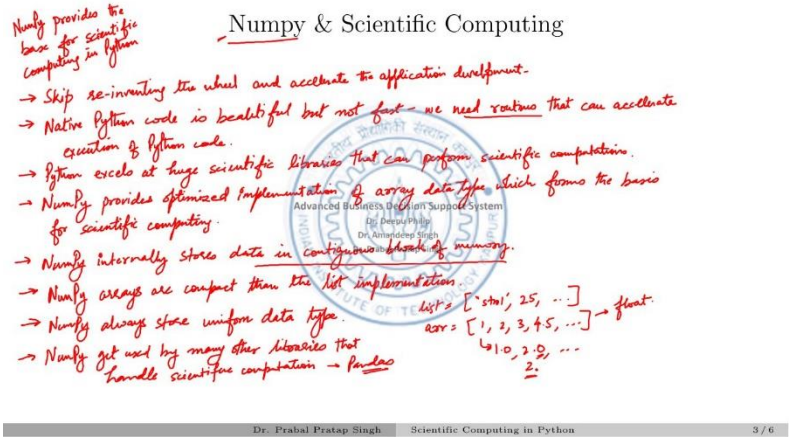
So, we can start by saying the ‘result = 1’ and then for ‘val in args’. So, this ‘args’ is a list here and we can write ‘result equals to result into val’. This will multiply all the numbers, and then we can return the result. Now, our very simple 'mod1.py' Module is ready. We can run this Module by saying python 'mod1.py'.

So, it is saying that there is a syntax error here. Now, it should be ok. So, now let us create a different script which is ‘feature script.py'. So, let us say you start writing your code in this file and you want to perform addition and multiplication multiple times in this script. So, either you can write this function here, these functions in this script, and then try to invoke these functions or what you can do is, you can just import this file ‘import mod1'.

And, now you can assess these ‘mod1.’. So, your coding environment is showing that there are two functions that are available which are 'add’ and ‘multiply’. So, you can use this function ‘add’ and you can provide any number of arguments like 56, 7, 23, 4. So, let us use just 4 numbers and we can print. This function will return the addition of all these numbers. So, we can store this in ‘add1’ and we can print the result by using the F string ‘addition is’.

So, now if you run this script, then this script will go to this 'mod1' which should be in the same file system your same directory, and then it will fetch this add function, provide the 4 arguments to that function, perform all the operations provided here which is a single statement of sum, and then it will return this sum and this statement should print the result. Now, let us see whether it will happen or not. So, this is showing that addition is 90. Now, in this Module, in this single feature 'script.py', you have not written this ‘add’ function, you have just imported this function. The same thing you can do by using the other function as well like the multiply function which you have created.

So, now I have just used this multiply function on these same numbers and it will now print two print statements, one for addition and one for multiplication. Let us run this file again, and now it is showing that the addition is 90 and multiplication is 36064. So, this is how we can use different kinds of Modules in our scripts and we can also write a very extensive list of functions or other features in this Module that we can use in our other projects. So, this feature is important because now we are going to learn different kinds of scientific libraries and how these libraries work in our own scripts is well-defined, by what I have just shown you.



So, let us move forward. So, now let us start with the first scientific library which is NumPy. Here, NumPy provides the base for scientific computing in python. This library provides various kinds of arrays, matrices, and vectors, so that we can model our real life situations, and then try to compute the real life situations, and then get the results. So, why do we need these kinds of libraries because we do not want to reinvent the wheel.

So, we will skip reinventing the wheel and this way we can accelerate the application development. For example, we all know how to perform logarithmic calculations by hand or square root calculations, but if we need to perform all these calculations and try to code these basic fundamental calculations every time, then it may leave us with less development time for our features. So, that is why we need to use these kinds of libraries. So, native python code is beautiful. We have all seen that it provides us with indentation throughout the code, so that the code becomes more readable, but what happens is, it is beautiful, but it is not fast.

In scientific computing, we need to handle large arrays, large vectors or large matrices of data. So, to handle these kinds of data, we need to perform fast computations. So, to perform these computations, we need pre-built routines that can accelerate execution of python code. Since python is an interpreted language, it is usually not that fast as compared to the other compiled languages. So, these pre-built routines, if provided in an already compiled form, then these routines can become very useful.

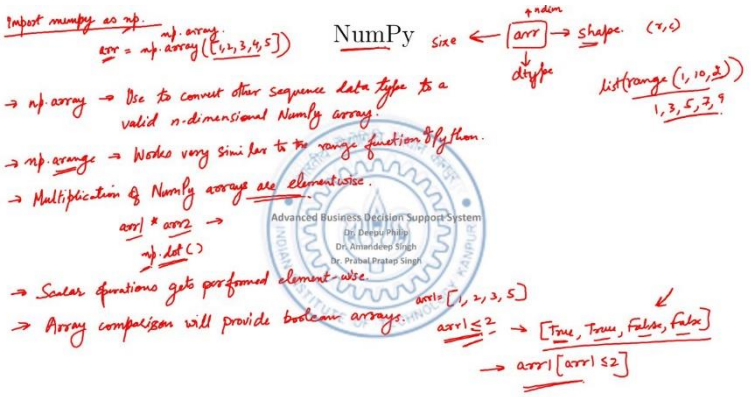
And, another reason is the python excels at huge scientific libraries that can perform scientific computations. Further, this NumPy library provides optimized implementation of 'array' data type which forms the basis for scientific computing.

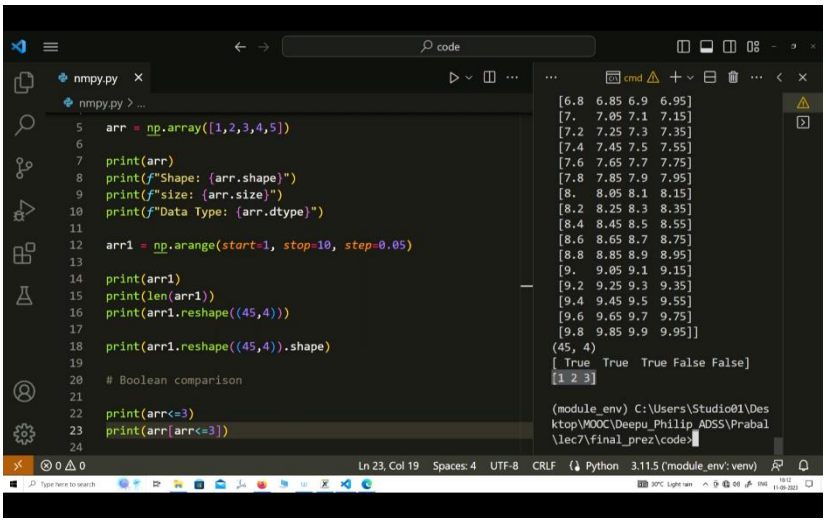
And, NumPy internally stores data in contiguous blocks of memory. So, we will see the 'array' data type provided by this library and this 'array' data type is different from what we have seen in the native python code as list. So, a list can handle different kinds of data types in it, simultaneously, but it is not as fast as this array provided by NumPy. So, what happens is, this library stores the data in contiguous blocks of memory, so that the operations performed on any data in a "NumPy array" is much faster than the operations handled by list. So, due to this, 'NumPy arrays' are more compact than the list implementation.

Also, NumPy will always store uniform data types. So, we have already seen that a list can store something like 'string 1', then it can have a number and any other data type as a complete list, but if it is a 'NumPy array', then it will either

store ‘int’ data type, 'float' data type, ‘string’ data type, but not a mixture of those. And, even if you are writing all the values as 'int' like 1, 2, 3, but you provided one 'float' type like 4.5.

Then, what NumPy will do, this array will completely turn towards 'float'. So, every other integer will get converted to 1.0, 2.0 like that and NumPy will only show 2.s, it will not show this 0 by default. 2.s means it is a 'float' value. Now, the beautiful thing about NumPy is that it is a base for many other libraries. So, NumPy gets used by many other libraries that handle scientific computations. So, one more famous library is Pandas, that is built upon NumPy.





So, now let us first try to initialize a 'NumPy array'. So, to initialize, we first need to import our library, we can import this, by using import NumPy. Now, you can just write this and to use any of the functions provided in this Module, you can use NumPy.array or something like that, but instead of using that the community follows this alias import NumPy as ‘np’. So, this is the most followed alias and you should also follow it. With this, you can use any function from the NumPy Module by using 'np.array’. To initialize a simple array, we can write ‘arr’. This is a simple variable which I am creating and we can write 'np.array’ and let us provide a simple list of 1, 2, 3, 4, 5. So, this is a list of 5 integers and after writing this 'arr' will hold a different kind of data type which is a ‘NumPy array' data type and this is an n-dimensional array.

So, here it is a single dimensional, but it can hold n-dimensions in it and this works differently than what list can do. So, once this gets created the 4 major characteristics of this array are, it has a shape, it has a size, it has a data type which is available as ‘dtype’ and you can also use ‘ndim’ to find the number of dimensions. So, we will perform all these operations in our code when we start coding, but let us look at other major characteristics of this 'NumPy array'. So, 'np.array', this is the function we can use to convert other sequence data types to a valid n-dimensional 'NumPy array'. Next there is one more function 'np.arrange’ which is if you remember we have used this range function very much in our previous coding exercises where we use the start number like 1 to 10 with a step of 2.

So, if you remember that this range function will create a list of numbers from 1 to 10 with the step of 2. So, the first number will be 1, then 3, then 5, then 7, then 9, like that. Now, this will create a list of numbers if you write this in the under list otherwise it is a generative expression. And then, a similar thing we can do in our NumPy Module also to create an ‘nd’ array. So, using this arrange function. So, this works very similar to the range function of python and it can also perform all these start and end step operations inside it.

You will see it when we code and simple multiplication of 'NumPy arrays’ are element wise. So, this is an important property, why? Because, when we start using this NumPy library, you can see that it contains vectors and matrices. Now, we have learned in our previous classes that vector multiplication or matrix multiplication follows a different rule, but here if you just multiply, let us say, ‘arr1 * arr2’, then it will perform element wise multiplication. So, if these arrays have the same shape, shape is the number of rows and number of columns, if it has a same shape, both the arrays have same shape, then each value gets multiplied by its corresponding value in the different array, and to perform a matrix multiplication, we have a different function here which is 'np.dot()’.

So, dot is the function, 'np' is the alias, which we have defined here. So, I am trying to make this as frequent as possible while writing this, so that you should start using these aliases while coding your scripts. Another thing is that scalar operations get performed element wise as well. So, if you multiply this first array with 5 or 7, then every element of this array gets multiplied by this scalar. And, the last major important thing is that array comparison will provide 'Boolean arrays’.

So, let us say we have a vector ‘NumPy array’ which is 1 comma 2 comma 3 comma 5. Now, if we perform a simple comparison which we have learned earlier, let us say ‘arr1’ is less than ‘equal to 2’, so only two values are less than which follows this criteria which is 1 and 2 others are greater than 2. So, what will be the output of this is, this will be like that true comma true comma false comma false. So, the output is showing that the value which was present in the array at this location is less than ‘equal to 2’.

The same holds for the second value, but the third and fourth value does not hold this criteria. So, if you want to just get the values that are less than 'equal to 2'. Then, you have to again use this function and try to use this 'Boolean array' which we have obtained from this conditional expression. So, we can write ‘arr1’ less than 'equal to 2'. So, this expression will give you the actual values. But, if you just write this conditional expression, then it will provide a complete 'Boolean array' of the answer of this statement. So, let us start coding and see these properties which we have just discussed.

So, let me close this and create a new file. And, to use this NumPy, we need to first install it. And, to install that, we will use our pip install command. So, let me clear the screen. So, let me install the NumPy library first by writing our command ‘python hyphen-m pip install NumPy’. So, it will start collecting the package, and then it will install it. So, now you will see that it has successfully installed ‘NumPy version 1.25.2’.

So, this will provide us functionality of all the scientific computations that we need in a script. So, let us start writing this script by first importing the newly installed library by writing importing ‘NumPy as np’. So, instead of 'np' you can write anything you can write my name, your name, anything, but we will always try to compress this alias name so that we have to write a short number of characters every time whenever we try to fetch any function from these libraries. So, let us first try to initialize a very basic variable, very basic array. So, we can write 'np . array’ and provide any sequence data. I am using list data, so 1, 2, 3, 4, 5.

So, let us try to run this much code only and you can see here that it is showing 1 2 3 4 5. Now, you can ask what is the difference between this 'NumPy array' and a simple list. So, to get that, we need to fetch all the four characteristics that we have discussed earlier. So, let us start getting these values from this. So, the first value should be ‘array.shape’. It can be anything, but I am trying to get the shape of the array.

So, now it is showing that the array has a shape of 5 comma which means it is a one-dimensional array which contains five values. You can also type the size of the array. So, the size is also 5 and the main thing, which is the data type, so the data type is 'int32’. This 32 means there are 32 bits and there are different kinds of integers which are available in 'NumPy array'. This is of the type 'int32’, we can have 'float32’, 'int64’, 'float64’, 'float8’, 'float16’, like that.

So, let us move forward and try to use another function which we wrote and discussed earlier which is the ‘arr1’, a range function. So, it will generate the values from the start which you provide, let us say 1, then stop is, let us say 100, and you can provide this step as well. So, instead of 100, we can use 10 to see the functionality of the step and let me just write that step = 0.05.

So, starting from 1, the next value will be 1.05, then it will be 1.10 like that. So, let us print this. Now, you can see that we have generated a large amount of data by writing a simple statement and this started from 1, And, since we have used a 'float' value in step, it converted the complete "NumPy array" into a 'float' value. So, this is 1 1.05 1.1 and like that until the 9.95 because the last value 10 is exclusive, same like other range functions. Now, this is a simple single one dimensional array, but we can convert this array into different dimensions by writing arr1. reshape.

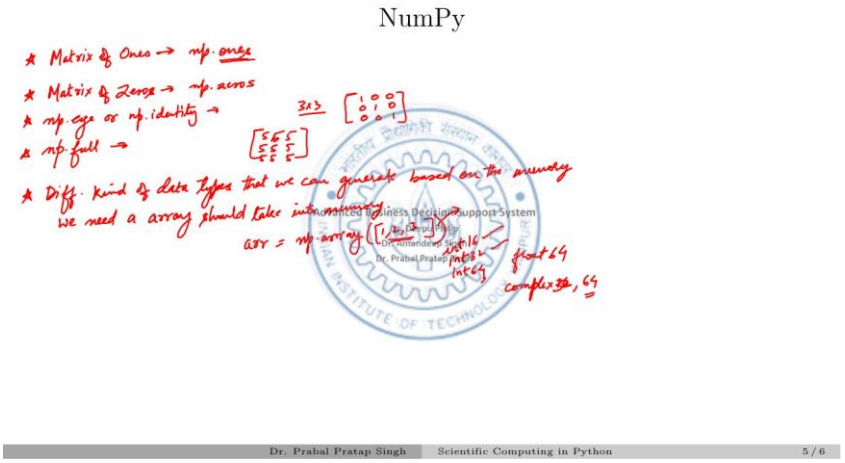
So, these functionalities are available in this NumPy Module which we have imported and you do not have to keep writing these basic functionalities again and again in your script. So, let us reshape into it. So, now what we have done here is, we first checked the length of the array which was 180. So, now by getting the number of data points available in this array, we can now convert this into a reshaped array. So, by getting 180, I have converted this into 45 rows with 4 columns.

So, after reshaping, the array looks like this. The first 4 values get into 1 row, and then the other data types get into other different rows. So, this is how you can reshape your array and the shape of this array will be different now. Initially, you have seen that this array was of 5 comma shape, but the array shape you can also combine various

operations on the same thing like this. So, what will this statement do? It will first fetch the ‘arr1’ array, then it will reshape into this, and then it will find the shape of this.

So, let us run our code again. Now, you can see the shape is 45 comma 4. So, let us again move forward and we can perform the Boolean comparison. So, we can write a basic Boolean check as we discussed earlier by writing 'arr' ≤ 3 and let us see what this will print. So, you can see here it is printing the Boolean values true, true, true, and then 2 values are false. So, 4 and 5 is not less than 3. Now, to get the values from this array, you can use this like using the square bracket notations. We can perform the same comparison inside these square brackets.

So, now you can see here that only those values that were true gets printed here 1, 2, 3, not the false values. So, this is a very efficient way to fetch only those values that hold your comparison criteria. So, now let us again move forward to what other functionalities are available in NumPy.



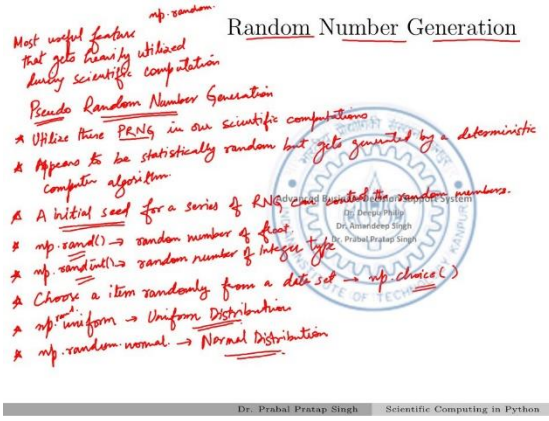
So, as I have discussed, NumPy provides various kinds of scalars and matrices and it provides some functions that can help you generate large kinds of matrices which are very standard in scientific computing. Like, if you just want to create a matrix of 1 you can use ‘np.1’. So, this is a function that will ask you the shape of the matrix you want, and then it will create a complete matrix of 1's.

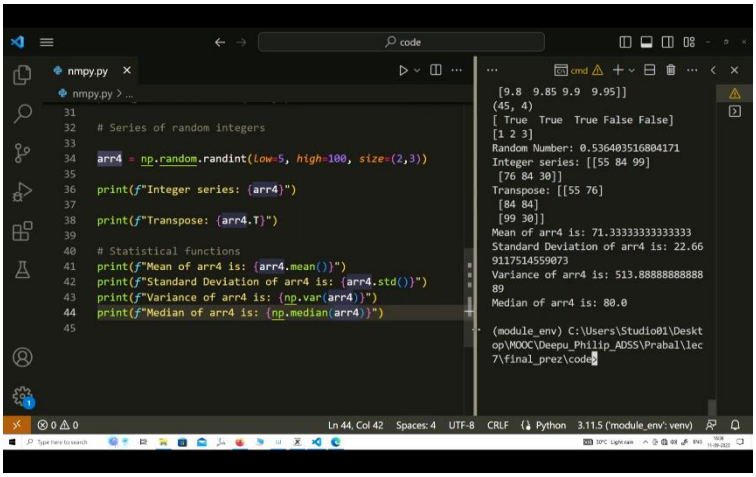
You can also create a matrix of 0's for that the function is ‘np.0's’, then you can create an identity matrix as well by using ‘np.i’ or ‘np.identity’. So, a 3 * 3 identity matrix looks like that. Only the diagonal elements are 1, others are 0. So, you can generate any n-dimensional identity matrix here, we are using 3 into 3. You can also use ‘np.full’. So, what if instead of 1's or 0's you want any other number that contains all the elements in that matrix.

If you try to generate a ‘np.full’ for 5, it will create a 3 into 3 matrix, then it will create 5, 5, 5, 5, 5. So, this will be a full matrix. And, as we have seen in our last coding exercise, there are different kinds of data types that we can generate based on the memory we need an array to take into memory. So, when you start doing large computations on very large data sets, then what happens is, your memory starts getting full.

So, you need to perform various modifications in your script, so that at each statement, you must ensure that the number of memories that gets used by the particular array is known to you. So, if you are just initializing an array by writing this ‘np.array’ and using a list and with some numbers, then it will automatically find the best possible data type for these values. Like if this is 1 2 3, then it will be an integer value.

So, the system can use 'int16’, 'int32’, 'int64’, or something like that. But, if you want to have a complete control, then you can also specify these values here also by providing the ‘dtype’ argument and tell the interpreter that I only want to use 'int62’ or 'int32’ or anything else or 'float'. The same things are available with 'float' as well 'float64’. But, complex values are I think only available from 16 or 32 and 64 like that. So, these things are also available in NumPy.





Now, we can move to our next concept which is Random Number Generation and this is the most useful feature that gets heavily utilized during scientific computations. So, in-real world, when we try to model the real world situations, we need to develop a few criteria that perform things that need to generate random numbers, so that we can completely model the real world. So, since we do not have any kind of natural Random Number Generation process, so the developers have created their own deterministic Random Number Generation algorithms and they are deterministic, but they look random. So, these types of deterministic algorithms, that were form Random Number Generations, are

known as Pseudo Random Number Generation, so they generate random numbers. But they are not completely random, they are Pseudo Random Numbers.

So, we can utilize these PRNG in our scientific computations. These generated numbers appear to be statistically random, but are generated by a deterministic computer algorithm and you can control these random numbers, what if you start running your script after one or two days and you start getting another series of random numbers.

So, to control that thing, we can provide an initial seed for a series of Random Number Generation that can control the random numbers. So, a particular seed will always provide the same series of random numbers. And, these functionalities are providing a seed value and generating these pseudo random numbers, these are prebuilt in our NumPy library.

So, we will try to see how to generate these things. So, why are we discussing all these things is because we need to create a Decision Support System and you have already seen that professor Deepu Philip has shown you that Monte Carlo simulation needs Random Number Generation. Now, to perform all these calculations, we need to scientifically generate our random number series each time in a particular loop.

So, at that time when we are trying to model the Monte Carlo simulations, then we will generate these random numbers. Now, let us also look at some of the functions that are available in NumPy. So, we can write 'np.rand'. This will give you a random number of 'float' type and it follows the uniform distribution.

The other function that is useful is 'rand int'. So, this will give a random number of integer type and we can generate a series of random numbers by providing different argument values here in these functions. So, we will see how to perform those things. One more thing which is possible here is that we can choose an item randomly from a data set. So, we can perform this by using the function 'np.choice'. And in this function, we can also provide the probabilities for choosing a particular item from a sequence data type. Also, there is a function 'np.uniform' that will generate the random numbers from uniform distribution.

The other function is 'np.random . normal'. So, these are all 'np.random . rand int'. So, inside NumPy, there is another function that is random and inside that, these different functions are available. So, I will be writing here 'np.random'. So, this will fetch the random numbers from normal distribution.

Let us try to generate some random numbers today. So, now here you can see that in our last print statement, we are trying to print the 'arr3' variable which generates a random number and this random number is 0.0346. So, this is the precision of python which it is using in this machine.

So, basically you can see that this is a 0.03 random number which gets generated by this. If you again run this file without editing anything in your code, then you will find that this number gets changed. So, let us run this file again and now you can see that this random number is 4191. But what if you want to control this output? So, to do that, you can write 'np.random.seed', and you can use anything like 108.

So, now run this file, this generated 0.233 some number. If you run this again, then this again generates the same number and you can keep on repeating the same thing, like I have cleared the screen, and now again repeating the execution of this file and it shows that the number remains the same. So, with this random seed, you will keep on

getting the same number. So, to pin your random series, we often use this random seed functionality in scientific computations. Now, moving further ahead, we can generate a series of random integers. To do that let us store this series in the variable here ‘arr4’ and we can write 'np.random . rand int' and it will ask you a few arguments, like the lowest value, from which it will fetch the random number, let us say it is 1 or let us say it is 5.

Then, the high value until which you wanted to do so, let us say it is 100 and let us just write this much and try to print the value of ‘arr4’. So, integer series will give the output as 22. Now, if I just comment this line and try to run this again by saving this file, and then running it again and now you can see that the number changed, again it changed, but it will not move ahead of 100 because our bucket contains the low value of 5 and high value of 100.

Now, if you want to generate a series of these values, you can write here size equals and let me write a 2 comma 3 array. So, what it will do is it will give you 6 random values from a bucket of 5 to 100 and it is in the shape of 2 comma 3.

So, now let us run this again and you can see here that this array which is being given by this integer series is 56, 55, 9, 31, 11 and 79 and these two brackets are showing that it contains two rows and three columns because there are three values in each row right. You can run this again to get a different output 5, 50, 93 and no value is getting ahead of 100. There is one more property which we frequently use which is the transpose. So, we can transpose this array of 2 comma 3 into 3 comma 2 by using the function 'arr4.T’.

So, this T functionality will change this 2, 3 into 3, 2. So, you can see this. Now, the new array contains three rows and only two columns. So, this has been transpose. Now, to do the same thing in other languages, you need to write different kinds of ‘for loops’ to swap these values, and then perform this transpose operation or they do have their own libraries, but here NumPy provides these operations and these are very easy to use. So, NumPy also provides the statistical functions and we can use these operations very easily.

Let us find the mean of the complete ‘arr4’ matrix by just using the mean function. So, the meaning of ‘arr4’ is ‘arr4.mean’. So, the mean is 48. Now, there are all the operations. So, we can write the standard deviation by using the function std and we can find the variance by using ‘arr4.var’ or we can also write 'np.var arr4’ same thing. We can also find the median by using 'np.median arr’.

Now, we can see here that the mean of the array is this standard deviation. So, these are all the functions which are available and there are many other functions that are available. This is a very vast library and you can easily switch to this NumPy documentation and find whatever you want to use like you can also create a histogram or you can use this quantile function to get all the quantiles for your array.

So, these things are available and this has a very vast library of functions that can be used and many other libraries use these functions from this documentation only. So, this is all for today and in the next lecture, we will meet to create a complete Decision Support System from scratch and there we will use a new library, and then we will try to see how to start coding in that library so that we can perform and create a new web application and after that we will try to generate a Monte Carlo simulation, and then we will provide the complete interface on the web using that new library. Thank you.