# Computational Mathematics with SageMath

**Prof. Ajit Kumar**
**Department of Mathematics**
**Institute of Chemical Technology, Mumbai**

## Lecture – 09
## Creating Modules and Introduction to NumPy

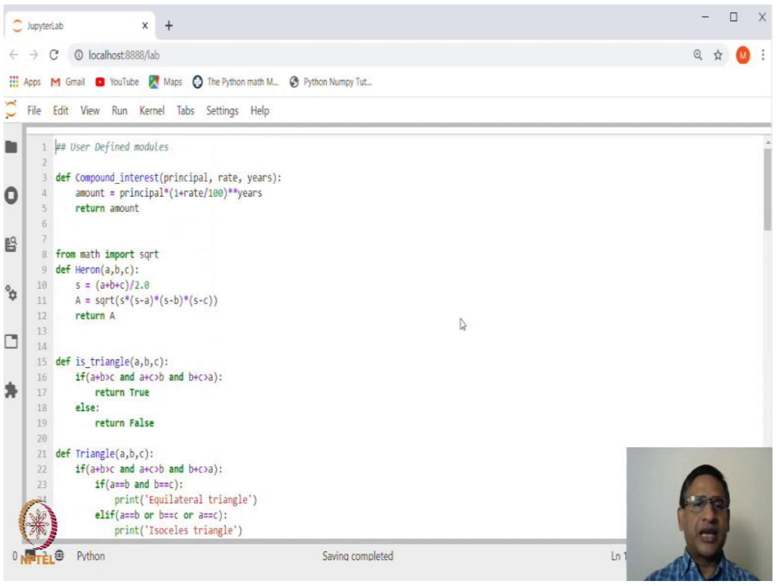
Hello and welcome to this course on Computational Mathematics with Sagemath. In the previous lectures, we have looked at how to use Python as an advanced calculator, we also explored lists, tuple, dictionaries, etcetera.

We also saw how to use the if-else condition, that is branching, and then we also created several user-defined functions and we have also seen writing loops.

So, now since we have already created several user-defined functions and we know that from a module how to import any function.

So, suppose we want to create our own module or library of functions. How do we do that? So, first, let me explain how to create our own list of functions.

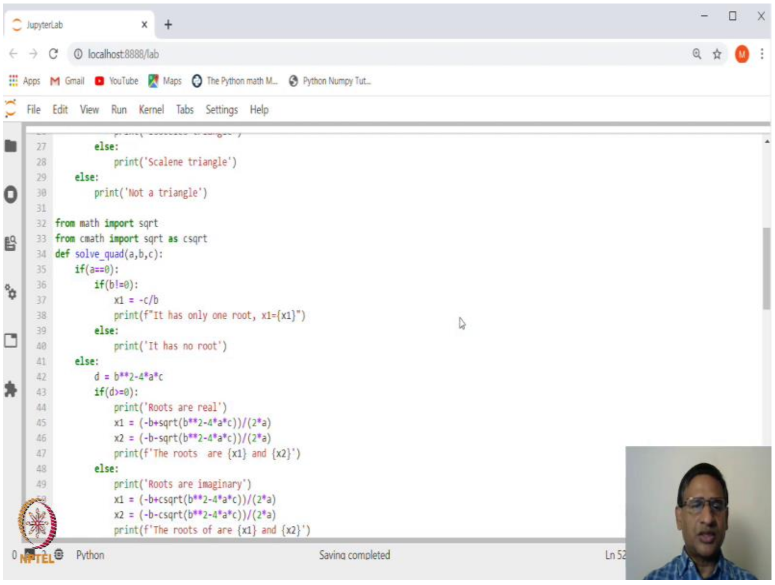
So, for that what you need to do is you simply create a user-defined function and put it in some file one after another, and save it as a dot py file. So, let me just show you we already have created several user-defined functions. (Refer Slide Time: 01:52)



```
## User Defined modules

def Compound_interest(principal, rate, years):
    amount = principal*(1+rate/100)**years
    return amount


from math import sqrt
def Heron(a,b,c):
    s = (a+b+c)/2.0
    A = sqrt(s*(s-a)*(s-b)*(s-c))
    return A


def is_triangle(a,b,c):
    if(a+b>c and a+c>b and b+c>a):
        return True
    else:
        return False

def Triangle(a,b,c):
    if(a+b>c and a+c>b and b+c>a):
        if(a==b and b==c):
            print('Equilateral triangle')
        elif(a==b or b==c or a==c):
            print('Isoceles triangle')
```

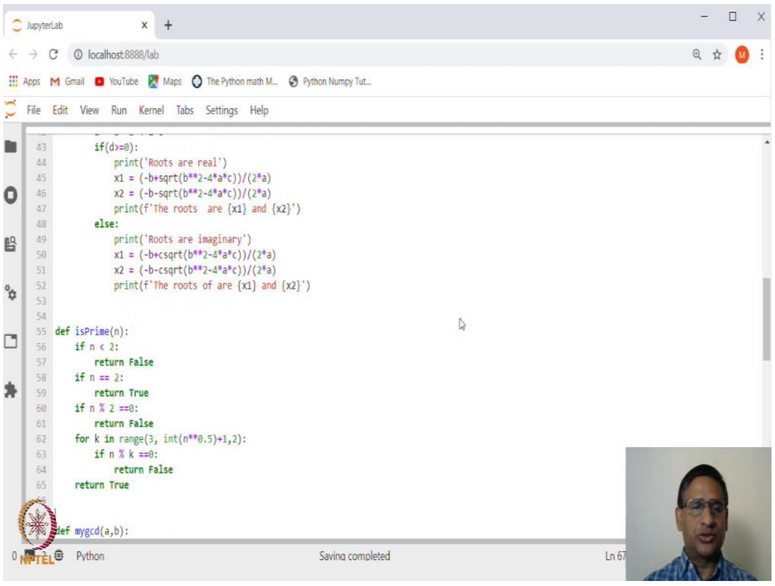
So, what I have done is I have copied these user-defined functions which we have created so far.

So, for example, we created a compound interest function, we created one function for Heron's formula for something for checking. (Refer Slide Time: 02:13)



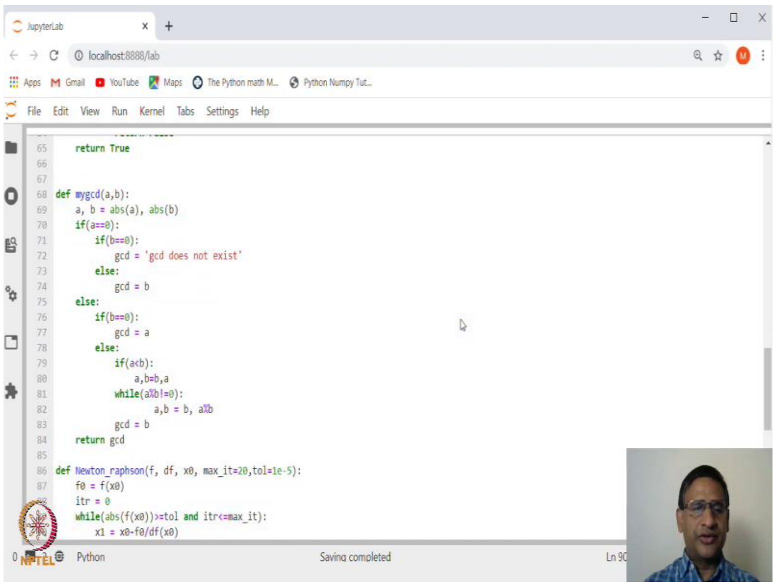
```
        else:
            print('Scalene triangle')
    else:
        print('Not a triangle')

from math import sqrt
from cmath import sqrt as csqrt
def solve_quad(a,b,c):
    if(a==0):
        if(b!=0):
            x1 = -c/b
            print(f"It has only one root, x1={x1}")
        else:
            print('It has no root')
    else:
        d = b**2-4*a*c
        if(d>=0):
            print('Roots are real')
            x1 = (-b+sqrt(b**2-4*a*c))/(2*a)
            x2 = (-b-sqrt(b**2-4*a*c))/(2*a)
            print(f'The roots  are {x1} and {x2}')
        else:
            print('Roots are imaginary')
            x1 = (-b+csqrt(b**2-4*a*c))/(2*a)
            x2 = (-b-csqrt(b**2-4*a*c))/(2*a)
            print(f'The roots of are {x1} and {x2}')
```

If there is 'a' if 'abc' forms a triangle, if it is a triangle then what kind of triangle it is, we have also created a user-defined function to find the roots of a quadratic. (Refer Slide Time: 02:21)



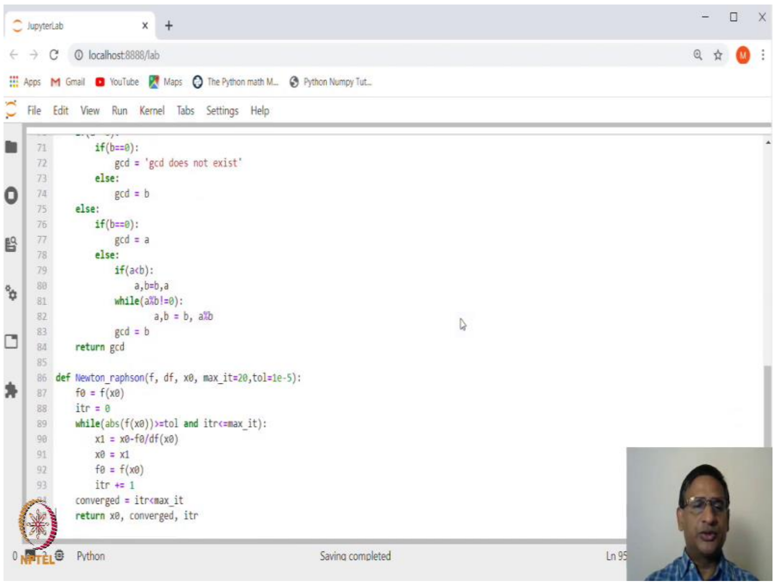
```
        if(d>=0):
            print('Roots are real')
            x1 = (-b+sqrt(b**2-4*a*c))/(2*a)
            x2 = (-b-sqrt(b**2-4*a*c))/(2*a)
            print(f'The roots  are {x1} and {x2}')
        else:
            print('Roots are imaginary')
            x1 = (-b+csqrt(b**2-4*a*c))/(2*a)
            x2 = (-b-csqrt(b**2-4*a*c))/(2*a)
            print(f'The roots of are {x1} and {x2}')


def isPrime(n):
    if n < 2:
        return False
    if n == 2:
        return True
    if n % 2 ==0:
        return False
    for k in range(3, int(n**0.5)+1,2):
        if n % k ==0:
            return False
    return True


def mygcd(a,b):
```

We also created a user-defined function to check if some positive integer is prime or not. (Refer Slide Time: 02:26)



```
    return True


def mygcd(a,b):
    a, b = abs(a), abs(b)
    if(a==0):
        if(b==0):
            gcd = 'gcd does not exist'
        else:
            gcd = b
    else:
        if(b==0):
            gcd = a
        else:
            if(a<b):
                a,b=b,a
            while(a%b!=0):
                    a,b = b, a%b
            gcd = b
    return gcd

def Newton_raphson(f, df, x0, max_it=20,tol=1e-5):
    f0 = f(x0)
    itr = 0
    while(abs(f(x0))>=tol and itr<=max_it):
        x1 = x0-f0/df(x0)
```

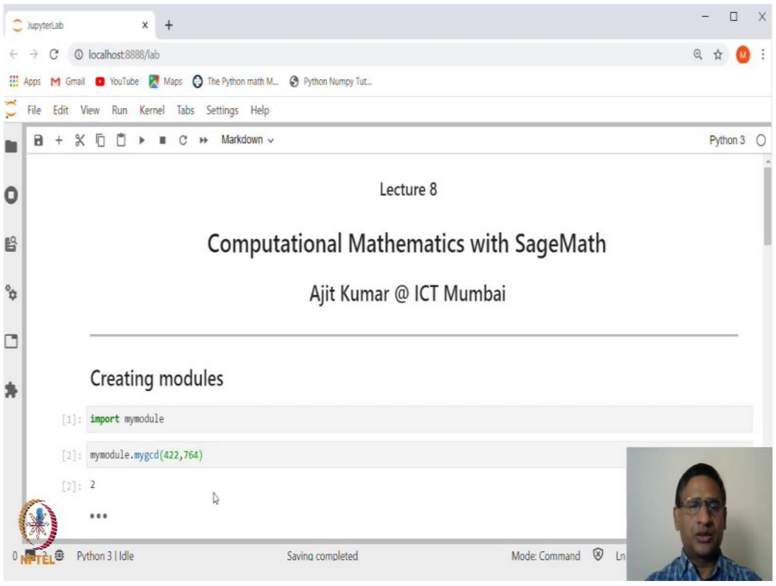
We also have created a user-defined function to find gcd of 2 integers and with the name 'mygcd'. (Refer Slide Time: 02:35)



```
        if(b==0):
            gcd = 'gcd does not exist'
        else:
            gcd = b
    else:
        if(b==0):
            gcd = a
        else:
            if(a<b):
                a,b=b,a
            while(a%b!=0):
                    a,b = b, a%b
            gcd = b
    return gcd

def Newton_raphson(f, df, x0, max_it=20,tol=1e-5):
    f0 = f(x0)
    itr = 0
    while(abs(f(x0))>=tol and itr<=max_it):
        x1 = x0-f0/df(x0)
        x0 = x1
        f0 = f(x0)
        itr += 1
    converged = itr<max_it
    return x0, converged, itr
```

And we also created a user-defined function for the Newton-Raphson method.

So, all I have done is just copied all these user-defined functions alone in a file and I have named this file as 'module.py'. You can just copy these codes in any note-book and then just save that as '.py'. So, let us see and of course, you need to save that file in the directory in which all your files are there or you can save it in fact anywhere, but then while importing you need to give the path. (Refer Slide Time: 03:24)



Lecture 8

Computational Mathematics with SageMath

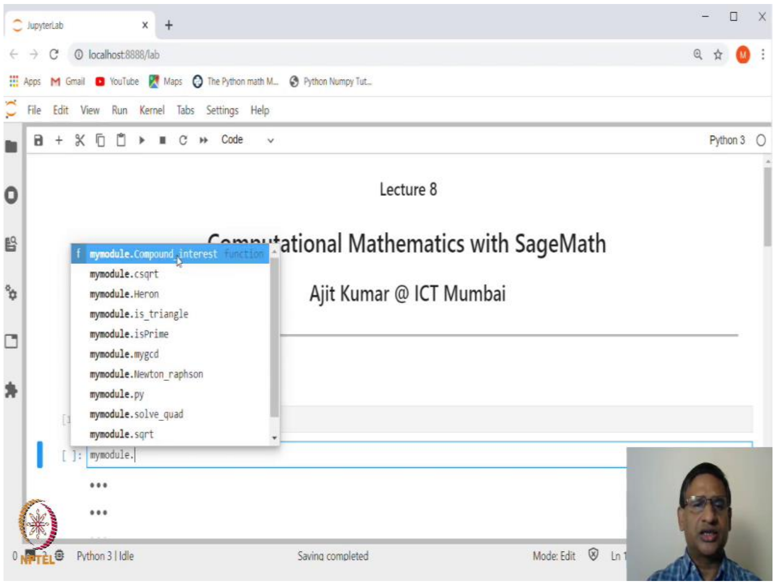
Ajit Kumar @ ICT Mumbai

Creating modules

```
[1]: import mymodule
[2]: mymodule.mygcd(422,764)
[2]: 2
```

So, let us come back to this. So, we have already saved that file which contains the list of user-defined functions and let us see how do we make use of that. So, what we can do is, let me simply say import ‘mymodule’ this is the file name I have given ‘mymodule.py’ and this is saved in the same directory in which I am working.

Now let me  execute this. So, it has imported that module and all the functions inside that module will be available using dot tab. (Refer Slide Time: 04:13)



Lecture 8

Computational Mathematics with SageMath

Ajit Kumar @ ICT Mumbai

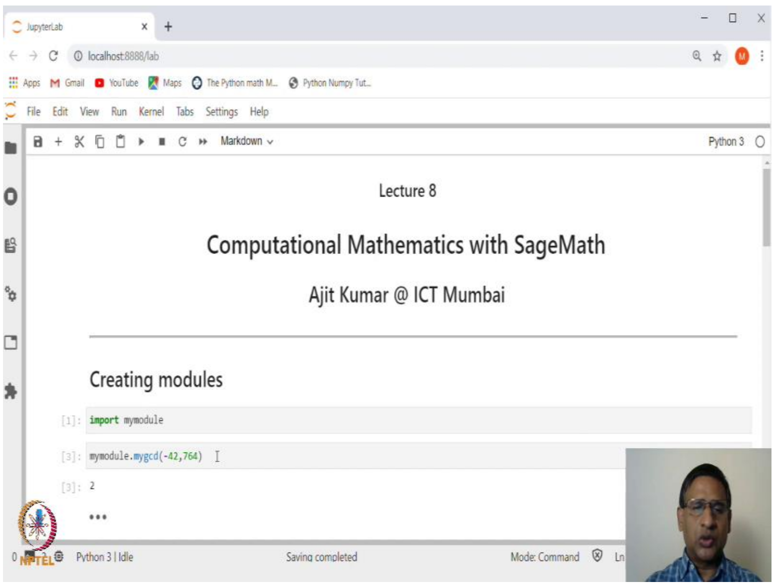
```
[ ]: mymodule.
```

```
f mymodule.Compound_interest   function
  mymodule.csqrt
  mymodule.Heron
  mymodule.is_triangle
  mymodule.isPrime
  mymodule.mygcd
  mymodule.Newton_raphson
  mymodule.py
  mymodule.solve_quad
  mymodule.sqrt
```

So, if I say mymodule and then press dot and tab, you will see all the functions that we have created, you have functions like compound interest, Herons, istriangle, isPrime, mygcd, Newton Raphson, solve quad and is yes if you look at sqrt and csqrt these are the functions we imported inside that module.

So, those will also be available right and how do we make use of this? So, for example, if I want to find gcd; so, I can say mymodule.g mygcd: gcd of 2 integers let us say 422 and 764  the answer is 2.

(Refer Slide Time: 05:00)



Lecture 8

Computational Mathematics with SageMath

Ajit Kumar @ ICT Mumbai

Creating modules

```
[1]: import mymodule

[3]: mymodule.mygcd(-42,764)

[3]: 2
```

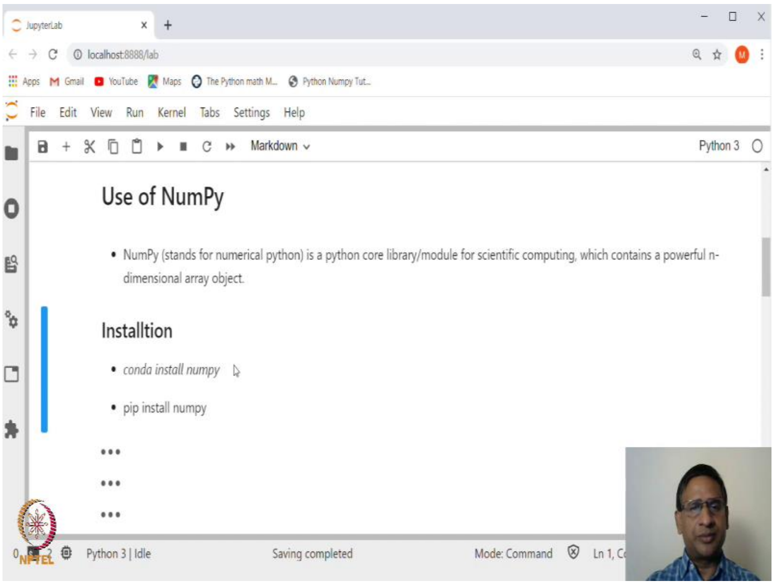
If I make it here minus of the let us say 42  and this the say the answer is 2.

So, either you can access it by typing dot mymodule dot the function name, or if you want you can simply import some functions or all the functions. You already know how to do that, you can simply say from my module import this function comma this function comma this function, or if you want you can say from my module import star, so it will import all the functions.

However, whenever you want to import all the functions generally it is not a good idea because unnecessarily all these functions will be sitting in the memory. So, we should try to import only those functions which are needed in that particular session. So, creating our own user-defined function is very easy, right?

So, all of you should just look at whatever functions we have created you just copy-paste those functions in a file and give the name whatever name you want you can call mylib.py, you click on mymodule.py, you can simply call modules dot py any name you can give.

(Refer Slide Time: 06:24)



## Use of NumPy

- NumPy (stands for numerical python) is a python core library/module for scientific computing, which contains a powerful n-dimensional array object.

### Installtion

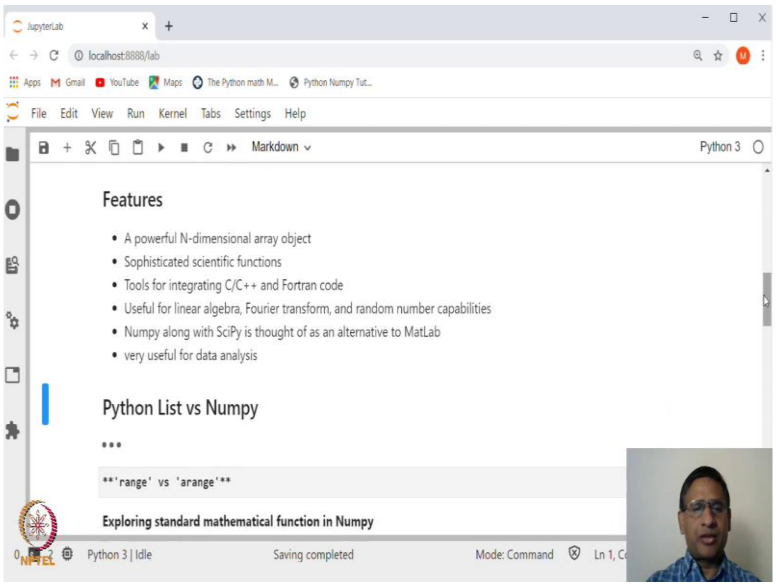
- *conda install numpy*
- pip install numpy

Now, in this lecture, we are also going to look at a very important library which is called Numpy. Numpy stands for numerical python and it is a python core library or module for doing scientific computation and it contains a powerful n-dimensional array object. Let us see what are other features in this.

So, before that let me also tell you in case you are working with simple python. If you have installed just python program then Numpy may not be available. So, you need to install Numpy. However, in case you have installed Anaconda, Numpy, etc. will be available.

So, you do not have to worry about it. If in case you want to install Numpy using conda then you can simply say conda install Numpy or you can use pip: pip install Numpy. This will install Numpy. And once you have installed Numpy you can import this and make use of all the functions that are available inside Numpy, we will see that this function is very useful.

(Refer Slide Time: 07:46)



Features

- A powerful N-dimensional array object
- Sophisticated scientific functions
- Tools for integrating C/C++ and Fortran code
- Useful for linear algebra, Fourier transform, and random number capabilities
- Numpy along with SciPy is thought of as an alternative to MatLab
- very useful for data analysis

Python List vs Numpy

**'range' vs 'arange'**

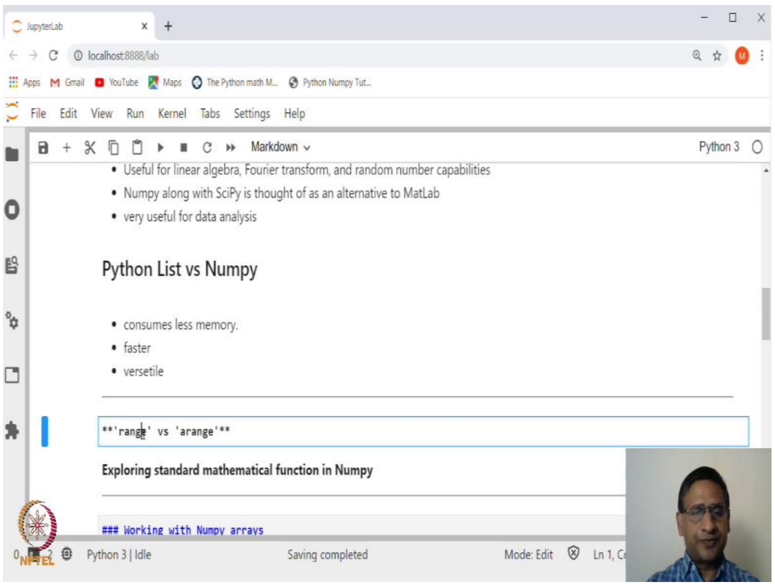
Exploring standard mathematical function in Numpy

Now, let us just look at what are some of the features of this Numpy library. So, as I said it is a powerful N-dimensional array object, it provides several methods that can be applied on an N-dimensional array, it has sophisticated scientific functions, it can be thought of as a tool for integrating C, C++, and Fortran codes. It is very useful for working with linear algebra, Fourier transforms, and also random number generations.

So, it has several methods to generate random numbers, and of course, in case you have used MatLab, so, Numpy along with SciPy has very nice features and a lot of features that MatLab has and of course, it is very very useful in data analysis. So, among data scientists, it is very very popular. So, these are some of the features of Numpy we will see some of it.

So, we have already seen how to generate a list 1-dimensional list or 2-dimensional list in python, but there were several drawbacks or limitations to this list. So, the question is why not use the Python list? Why do we want to use Numpy array?

(Refer Slide Time: 09:19)



- Useful for linear algebra, Fourier transform, and random number capabilities
- Numpy along with SciPy is thought of as an alternative to MatLab
- very useful for data analysis

Python List vs Numpy

- consumes less memory.
- faster
- versetile

```
**'range' vs 'arange'**
```

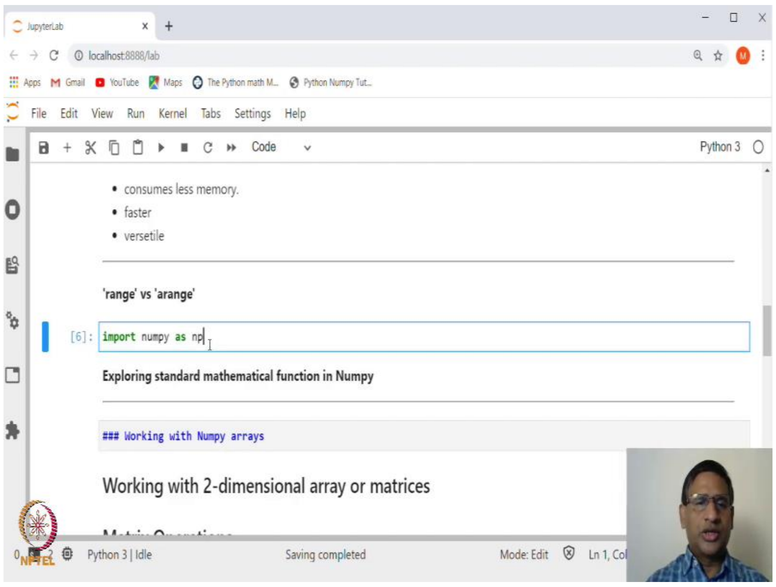
Exploring standard mathematical function in Numpy

```
### Working with Numpy arrays
```

So, there are few advantages: one, the Numpy list consumes less memory and as a result it is fast and at the same time it is quite versatile. So, let me explain what do I mean by this.

So, let us look at a function "range" in Python which we have already used. But when we used range, the range works only for a sequence of numbers that are integers and you can have integer increments. If I want decimal increment it does not work, we have already seen that, whereas, in Numpy, there is a function called "arange" where you can not only create a sequence of integers, but you can also create a sequence with increment as a decimal number.

(Refer Slide Time: 10:32)



- consumes less memory.
- faster
- versetile

'range' vs 'arange'

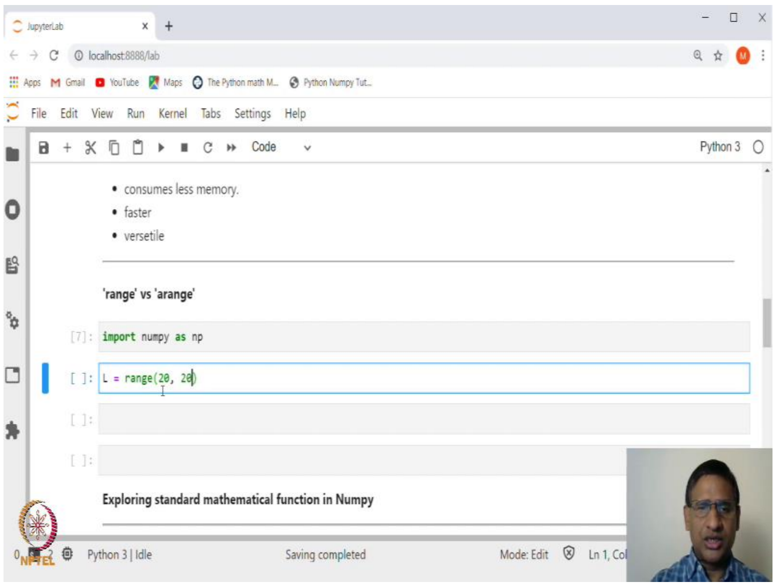
```
[6]: import numpy as np
```

Exploring standard mathematical function in Numpy

```
### Working with Numpy arrays
```

Working with 2-dimensional array or matrices

So, let us import Numpy and give it a name np. So, we will say import Numpy as np. (Refer Slide Time: 10:44)



- consumes less memory.
- faster
- versetile

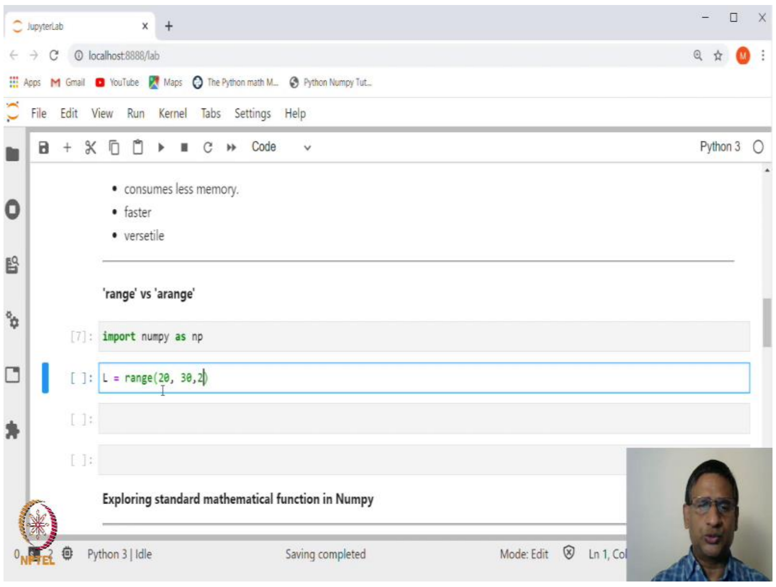
**'range' vs 'arange'**

```
[7]: import numpy as np
[ ]: L = range(20, 20)
[ ]:
[ ]:
```

**Exploring standard mathematical function in Numpy**

Next, let us first generate a list using the range function. So, if I say, let me say L is equal to range and let us say the range is from, let us say 10 to 20.

(Refer Slide Time: 11:04)



- consumes less memory.
- faster
- versetile

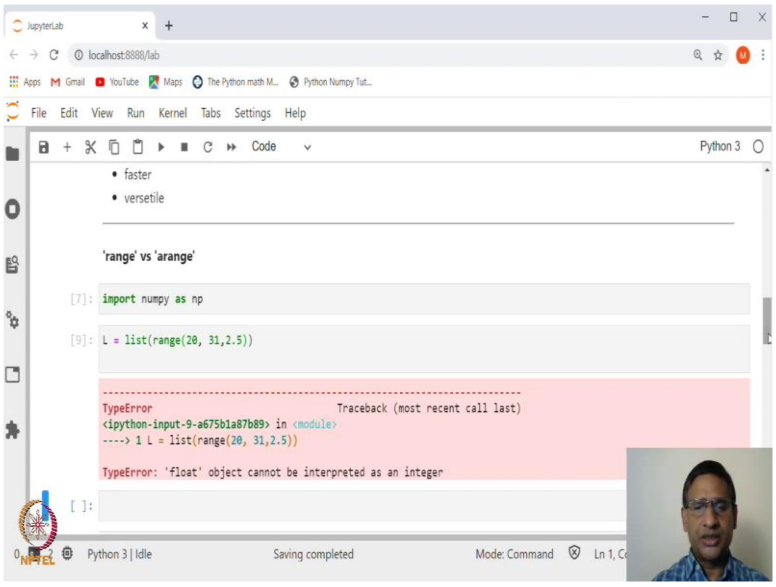
**'range' vs 'arange'**

```
[7]: import numpy as np
[ ]: L = range(20, 30,2)
[ ]:
[ ]:
```

**Exploring standard mathematical function in Numpy**

Or let 10 to 30 and with increment let us say 2. So, let me go up to 30.

(Refer Slide Time: 11:09)

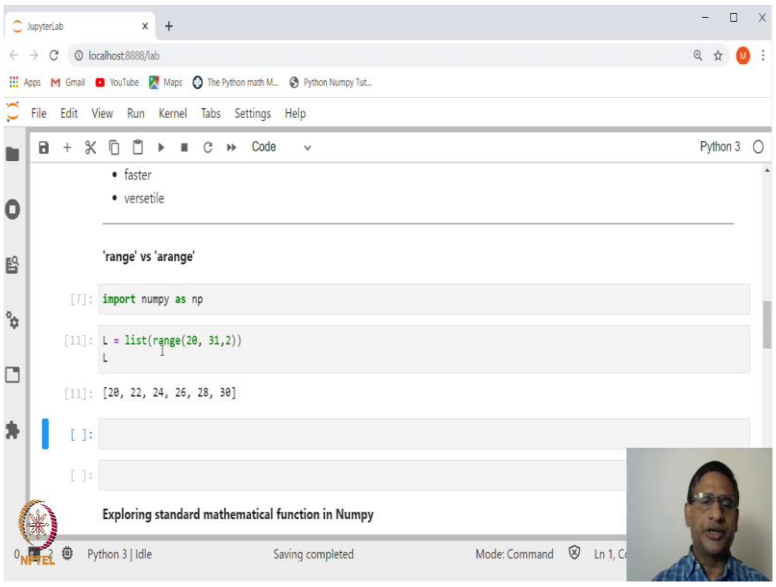


- faster
- versetile

'range' vs 'arange'

```
[7]: import numpy as np

[9]: L = list(range(20, 31,2.5))

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-9-a675b1a87b89> in <module>
----> 1 L = list(range(20, 31,2.5))

TypeError: 'float' object cannot be interpreted as an integer

[ ]:
```

So, I will say range 20 to 30. So, this will create a list, now you can let me put it also list here so that you will see it as a list. This is a list. And we have already seen that instead of increment 2 if I say 2.5 then it will give me an error because it does not accept noninteger increment. (Refer Slide Time: 11:34)



- faster
- versetile

'range' vs 'arange'

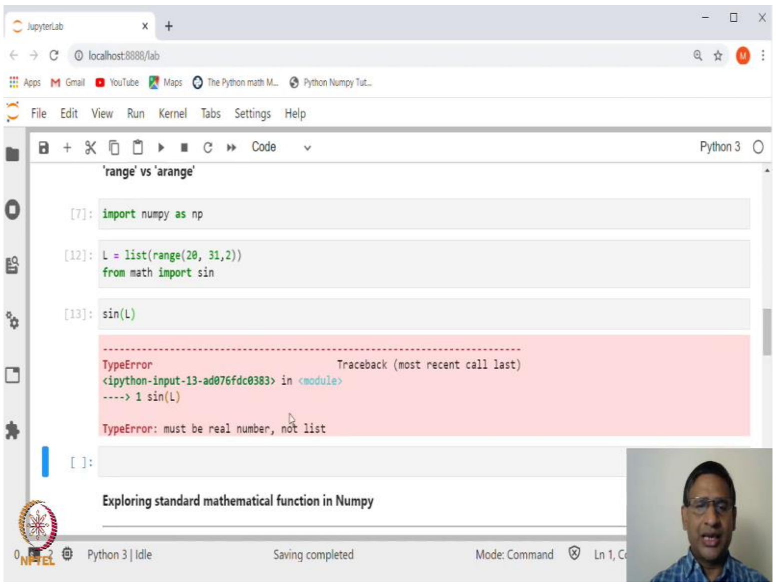
```
[7]: import numpy as np

[11]: L = list(range(20, 31,2))
      L

[11]: [20, 22, 24, 26, 28, 30]

[ ]:

[ ]:
```

Exploring standard mathematical function in Numpy

So, let us have a list. So, what is L? L is 20, 22, 24  up to 30.

Now, suppose we want to, let us say, apply some function to each member of this list. So, I want a list, let us say, something like sine of L. So, sine of 20, sine of 22, and so on, and let us see whether we are able to do this or not. We have already seen several methods that can be applied on L. For example, you can find the length, you can access any entry, you can add two lists and you can append, you can sort all these things, we have seen, right?

(Refer Slide Time: 12:19)



'range' vs 'arange'

```
[7]: import numpy as np

[12]: L = list(range(20, 31,2))
      from math import sin

[13]: sin(L)

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-13-ad076fdc0383> in <module>
----> 1 sin(L)

TypeError: must be real number, not list

[ ]:
```

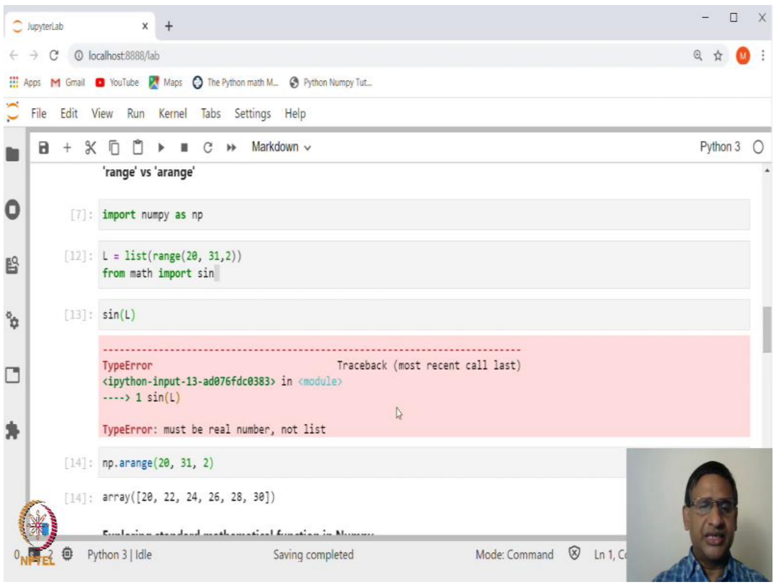
Exploring standard mathematical function in Numpy

So, what do we want to do? So, let us say if I say import, or let me say, from math import let us say sine function right, and let us say, if I say sin of sin(L), then it does not work. It gives me an error and it says that must be a real number, not a list.

So, the sine function actually can be thought of as a scalar function. So, it takes only one argument at a time. You cannot have a list. So that is one drawback of creating a list whereas, Numpy also has all these scientific functions, mathematical functions like trigonometric function, exponential function, logarithmic and all these things, etc., and there it will work we will see in a minute.

So, that is the somewhat disadvantage of using a list from the standard python library. Now, let us create a list using “arange”.

(Refer Slide Time: 13:38)



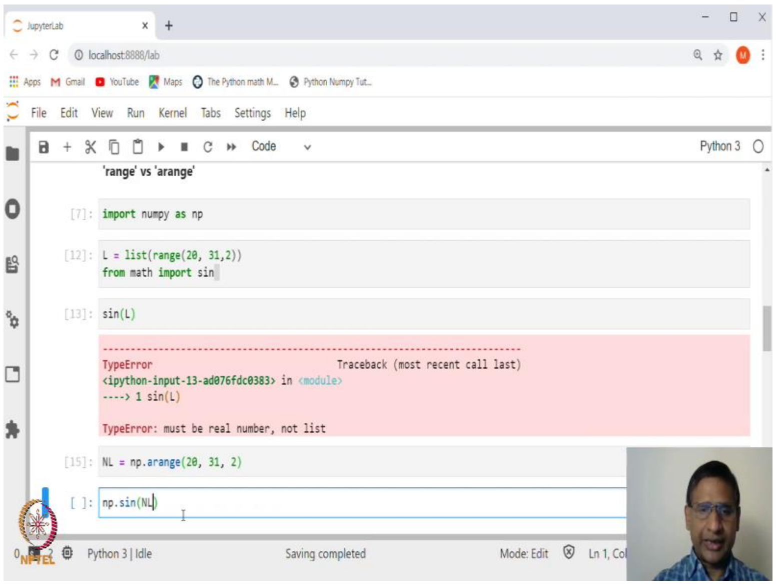
'range' vs 'arange'

```
[7]: import numpy as np

[12]: L = list(range(20, 31,2))
      from math import sin

[13]: sin(L)

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-13-ad076fdc0383> in <module>
----> 1 sin(L)

TypeError: must be real number, not list

[14]: np.arange(20, 31, 2)

[14]: array([20, 22, 24, 26, 28, 30])
```

So, we can say np.arange, and then if I again simply create 20 to 31 with the increment 2, again it will create the same list, but this time it is not a simple list; it is an array?

So, let us see, now let me give this name as it is an array from Numpy.

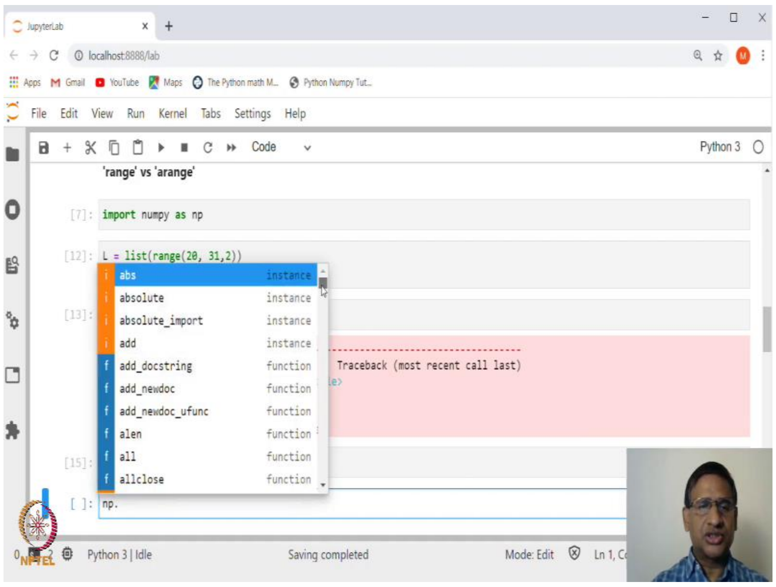
(Refer Slide Time: 14:06)



'range' vs 'arange'

```
[7]: import numpy as np

[12]: L = list(range(20, 31,2))
      from math import sin

[13]: sin(L)

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-13-ad076fdc0383> in <module>
----> 1 sin(L)

TypeError: must be real number, not list

[15]: NL = np.arange(20, 31, 2)

[ ]: np.sin(NL)
```

So, let me call NL is equal to this and let us see, we want to apply the sine function on this whether it works. So, first, we will import the sine function from the Numpy library. (Refer Slide Time: 14:25)



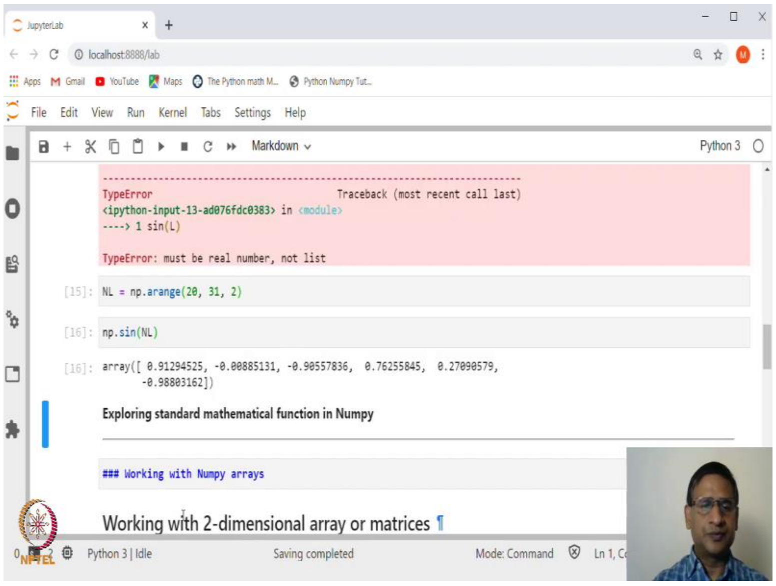
'range' vs 'arange'

```
[7]: import numpy as np

[12]: L = list(range(20, 31,2))

[13]:
      ---------------------------------
      Traceback (most recent call last)

[15]:

[ ]: np.
```

```
abs                 instance
absolute            instance
absolute_import     instance
add                 instance
add_docstring       function
add_newdoc          function
add_newdoc_ufunc    function
alen                function
all                 function
allclose            function
```

So, when you say np and press dot and tab, then you will see several functions inside this, including sin, cosine, exponential, logarithm, etc. I mean you can go through this list and try to explore some of it, we will make use of a few of them, but there are just too many. I cannot explain each and every function from this list. So, let us call sin and apply this to NL?

(Refer Slide Time: 14:58)



```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-13-ad076fdc0383> in <module>
----> 1 sin(L)

TypeError: must be real number, not list

[15]: NL = np.arange(20, 31, 2)

[16]: np.sin(NL)

[16]: array([ 0.91294525, -0.00885131, -0.90557836,  0.76255845,  0.27090579,
             -0.98803162])
```

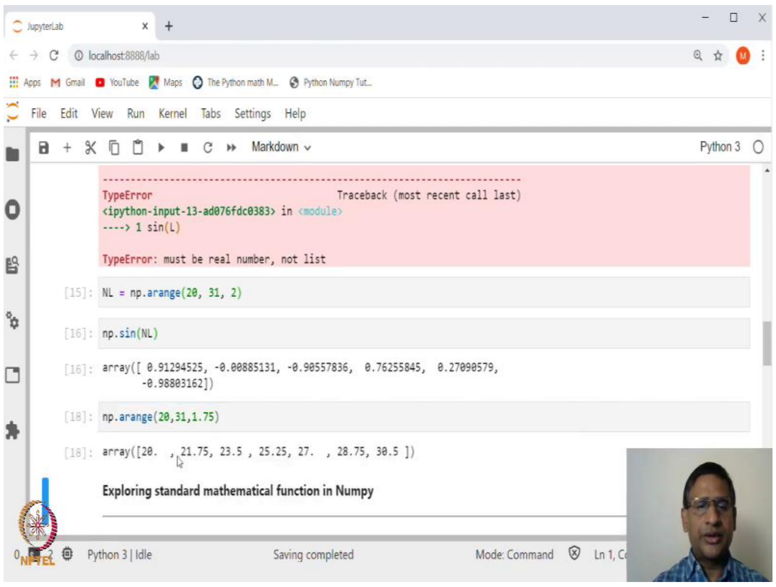
**Exploring standard mathematical function in Numpy**

```
### Working with Numpy arrays
```

Working with 2-dimensional array or matrices ¶

So, you can see here, now we are able to apply sine function to this list NL, which is an array of numbers from 20 to 30  with the common difference 2. So, this is sine of 20, this is sine of 22, this is sine of 24 and so on, of course, we know that this is sine of 20 radians? So, you can apply actually any function to a Numpy array?

(Refer Slide Time: 15:39)



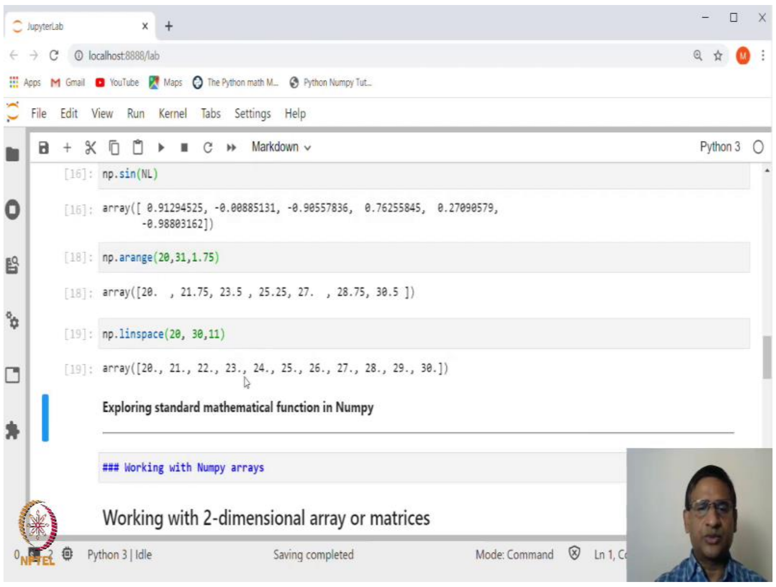
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-13-ad076fdc0383> in <module>
----> 1 sin(L)

TypeError: must be real number, not list
```

```
[15]: NL = np.arange(20, 31, 2)

[16]: np.sin(NL)

[16]: array([ 0.91294525, -0.00885131, -0.90557836,  0.76255845,  0.27090579,
             -0.98803162])

[18]: np.arange(20,31,1.75)

[18]: array([20.  , 21.75, 23.5 , 25.25, 27.  , 28.75, 30.5 ])
```

**Exploring standard mathematical function in Numpy**

Let us generate another, let us also look at if I want to generate np.arange and again 20 to 31 and let us say this time common difference is 1.75.  np, right? So, again you can see here this, you can generate a list with increment as a noninteger or floating-point number? There is another function which is also useful inside this Numpy, and that is called linspace.

(Refer Slide Time: 16:18)



```
[16]: np.sin(NL)

[16]: array([ 0.91294525, -0.00885131, -0.90557836,  0.76255845,  0.27090579,
             -0.98803162])

[18]: np.arange(20,31,1.75)

[18]: array([20.  , 21.75, 23.5 , 25.25, 27.  , 28.75, 30.5 ])

[19]: np.linspace(20, 30,11)

[19]: array([20., 21., 22., 23., 24., 25., 26., 27., 28., 29., 30.])
```

Exploring standard mathematical function in Numpy

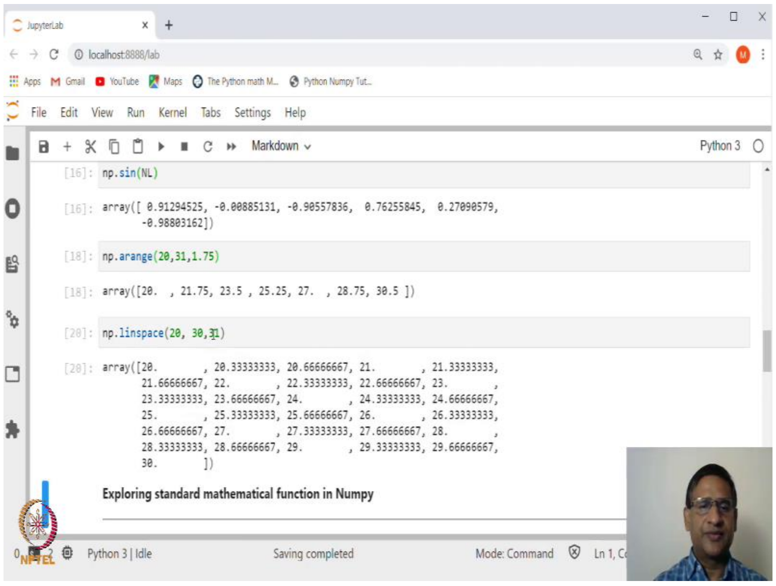
```
### Working with Numpy arrays
```

Working with 2-dimensional array or matrices

So, if I say np.linspace, this is very similar to the function linspace function in Matlab. So, what you need to do is suppose again, I give 20 and 30 and suppose this interval 20 to 30, we want to divide into 10 equal parts right? So, when you divide into 10 equal parts then there will be 11 points inside this interval 20 to 30.

So, if I write here 11 this means what will it do? It will insert 11 points including the endpoints the two points will be the endpoint 20 and 30 and the remaining 9 points, they will all be equally placed? So, if I do this what will you get? You should get 20, 21 up to 30.

(Refer Slide Time: 17:17)

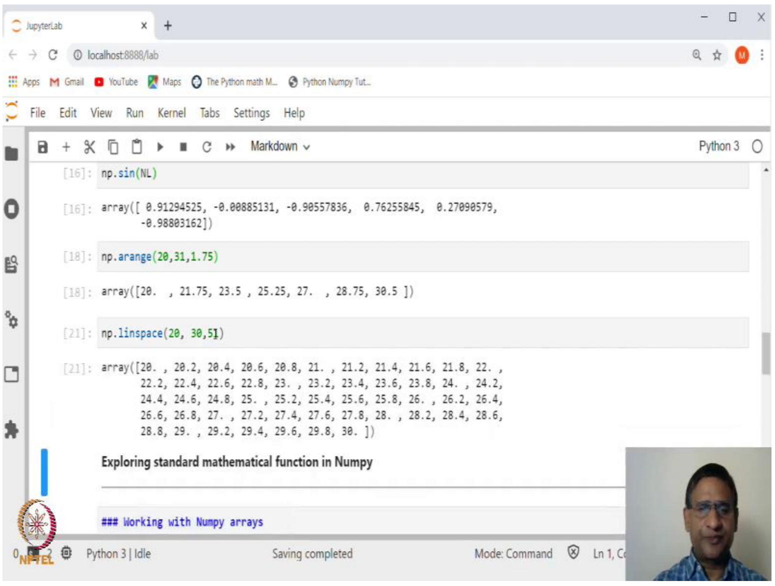


```
[16]: np.sin(NL)

[16]: array([ 0.91294525, -0.00885131, -0.90557836,  0.76255845,  0.27090579,
             -0.98803162])

[18]: np.arange(20,31,1.75)

[18]: array([20.  , 21.75, 23.5 , 25.25, 27.  , 28.75, 30.5 ])

[20]: np.linspace(20, 30,31)

[20]: array([20.        , 20.33333333, 20.66666667, 21.        , 21.33333333,
             21.66666667, 22.        , 22.33333333, 22.66666667, 23.        ,
             23.33333333, 23.66666667, 24.        , 24.33333333, 24.66666667,
             25.        , 25.33333333, 25.66666667, 26.        , 26.33333333,
             26.66666667, 27.        , 27.33333333, 27.66666667, 28.        ,
             28.33333333, 28.66666667, 29.        , 29.33333333, 29.66666667,
             30.        ])
```

Exploring standard mathematical function in Numpy

And that is what you have instead of 11 if I say give me 30 points, 31 points, then you have 20, 20.333, 20.666, 21, and so on.

(Refer Slide Time: 17:29)


JupyterLab
localhost:8888/lab
Apps Gmail YouTube Maps The Python math M... Python Numpy Tut...
File Edit View Run Kernel Tabs Settings Help
Markdown
Python 3
Python 3 | Idle
Saving completed
Mode: Command
Ln 1, C

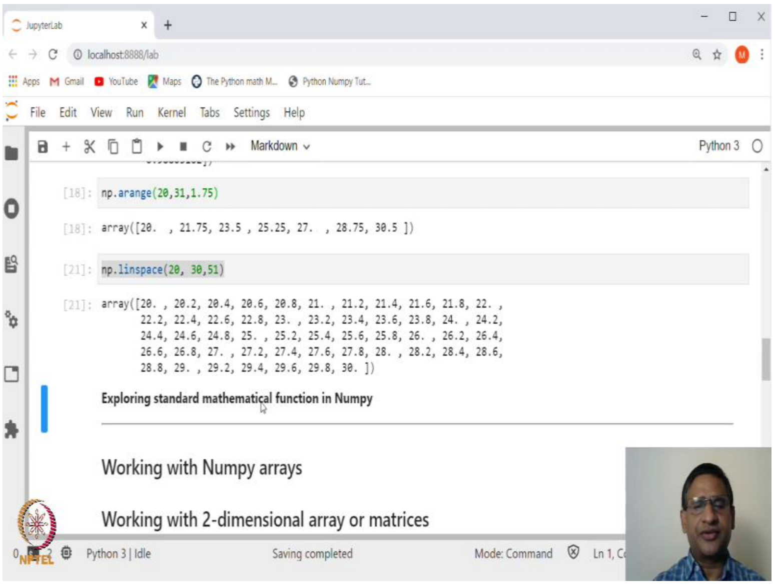
```
[16]: np.sin(NL)

[16]: array([ 0.91294525, -0.00885131, -0.90557836,  0.76255845,  0.27090579,
             -0.98803162])

[18]: np.arange(20,31,1.75)

[18]: array([20.  , 21.75, 23.5 , 25.25, 27.  , 28.75, 30.5 ])

[21]: np.linspace(20, 30,51)

[21]: array([20. , 20.2, 20.4, 20.6, 20.8, 21. , 21.2, 21.4, 21.6, 21.8, 22. ,
             22.2, 22.4, 22.6, 22.8, 23. , 23.2, 23.4, 23.6, 23.8, 24. , 24.2,
             24.4, 24.6, 24.8, 25. , 25.2, 25.4, 25.6, 25.8, 26. , 26.2, 26.4,
             26.6, 26.8, 27. , 27.2, 27.4, 27.6, 27.8, 28. , 28.2, 28.4, 28.6,
             28.8, 29. , 29.2, 29.4, 29.6, 29.8, 30. ])
```

**Exploring standard mathematical function in Numpy**

```
### Working with Numpy arrays
```

Instead of, let us say I want 50 points. Then I can say 51. You will get this. So, this is again a very useful function to generate a list of numbers or a list of points in the interval, which are equally placed? So, this, so you can see what is the difference between the arange in Numpy and range in python. arange function has more facilities than the range function?

(Refer Slide Time: 18:12)


JupyterLab
localhost:8888/lab
Apps Gmail YouTube Maps The Python math M... Python Numpy Tut...
File Edit View Run Kernel Tabs Settings Help
Markdown
Python 3
Python 3 | Idle
Saving completed
Mode: Command
Ln 1, C

```
[18]: np.arange(20,31,1.75)

[18]: array([20.  , 21.75, 23.5 , 25.25, 27.  , 28.75, 30.5 ])

[21]: np.linspace(20, 30,51)

[21]: array([20. , 20.2, 20.4, 20.6, 20.8, 21. , 21.2, 21.4, 21.6, 21.8, 22. ,
             22.2, 22.4, 22.6, 22.8, 23. , 23.2, 23.4, 23.6, 23.8, 24. , 24.2,
             24.4, 24.6, 24.8, 25. , 25.2, 25.4, 25.6, 25.8, 26. , 26.2, 26.4,
             26.6, 26.8, 27. , 27.2, 27.4, 27.6, 27.8, 28. , 28.2, 28.4, 28.6,
             28.8, 29. , 29.2, 29.4, 29.6, 29.8, 30. ])
```

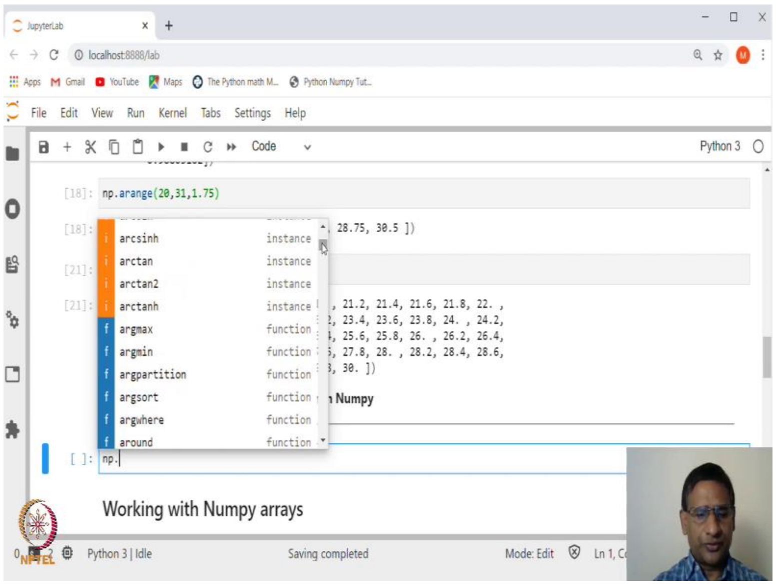
**Exploring standard mathematical function in Numpy**

Working with Numpy arrays

Working with 2-dimensional array or matrices

So,  let us also look at some of the other mathematical functions inside Numpy library. . So, let me insert one input cell here.

(Refer Slide Time: 18:21)



```
[18]: np.arange(20,31,1.75)

[18]:                                         , 28.75, 30.5 ])

[21]:

[21]:                                         , 21.2, 21.4, 21.6, 21.8, 22. ,
                                             2, 23.4, 23.6, 23.8, 24. , 24.2,
                                             4, 25.6, 25.8, 26. , 26.2, 26.4,
                                             5, 27.8, 28. , 28.2, 28.4, 28.6,
                                             3, 30. ])
```

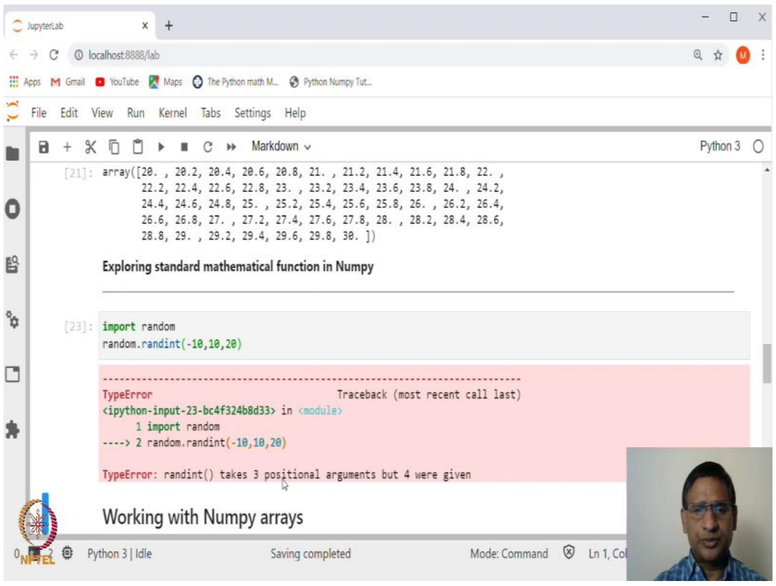
```
arcsinh        instance
arctan         instance
arctan2        instance
arctanh        instance
argmax         function
argmin         function
argpartition   function
argsort        function
argwhere       function
around         function
```

```
[ ]: np.
```

Working with Numpy arrays

So,  if I say np dot and press tab you will get a whole lot of options you can go through all these things. So, for example, let us create a list. Let us create some list of random numbers? So, there is a module inside Numpy called random.

(Refer Slide Time: 18:49)



```
[21]: array([20. , 20.2, 20.4, 20.6, 20.8, 21. , 21.2, 21.4, 21.6, 21.8, 22. ,
             22.2, 22.4, 22.6, 22.8, 23. , 23.2, 23.4, 23.6, 23.8, 24. , 24.2,
             24.4, 24.6, 24.8, 25. , 25.2, 25.4, 25.6, 25.8, 26. , 26.2, 26.4,
             26.6, 26.8, 27. , 27.2, 27.4, 27.6, 27.8, 28. , 28.2, 28.4, 28.6,
             28.8, 29. , 29.2, 29.4, 29.6, 29.8, 30. ])
```

**Exploring standard mathematical function in Numpy**

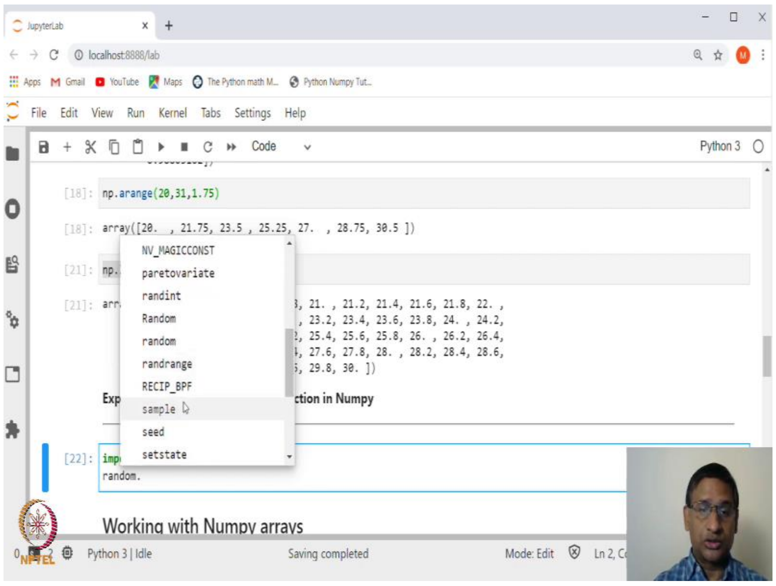
```
[23]: import random
      random.randint(-10,10,20)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-23-bc4f324b8d33> in <module>
      1 import random
----> 2 random.randint(-10,10,20)

TypeError: randint() takes 3 positional arguments but 4 were given
```

Working with Numpy arrays

So, let me import - import random, random, and inside this random if I look at random, random dot, and press tab.

(Refer Slide Time: 19:02)



```
[18]: np.arange(20,31,1.75)

[18]: array([20.  , 21.75, 23.5 , 25.25, 27.  , 28.75, 30.5 ])

[21]: np.

[21]: arr   3, 21. , 21.2, 21.4, 21.6, 21.8, 22. ,
            , 23.2, 23.4, 23.6, 23.8, 24. , 24.2,
            2, 25.4, 25.6, 25.8, 26. , 26.2, 26.4,
            4, 27.6, 27.8, 28. , 28.2, 28.4, 28.6,
            5, 29.8, 30. ])

[22]: imp
      random.
```

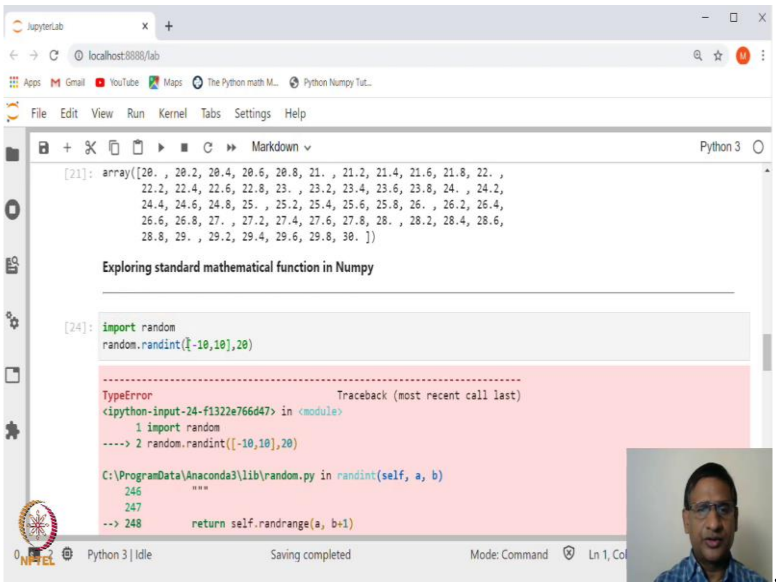
Again you will see different functions inside and you have one “randint” which will generate a random integer, you have “random” which will generate a random number, you have other options.

You also have “sample” we have seen how to generate a sample, and many other things. You can generate random numbers based on particular distributions?

So, let us, so, let us generate a random int randint, and let us say some random int between let us say, -10 to 10, and how many do we want?

Let us say we want 20 of them. So, it takes 3 positions argument, but 4 are given randint. So, possibly let us say this is, ok…

(Refer Slide Time: 20:11)

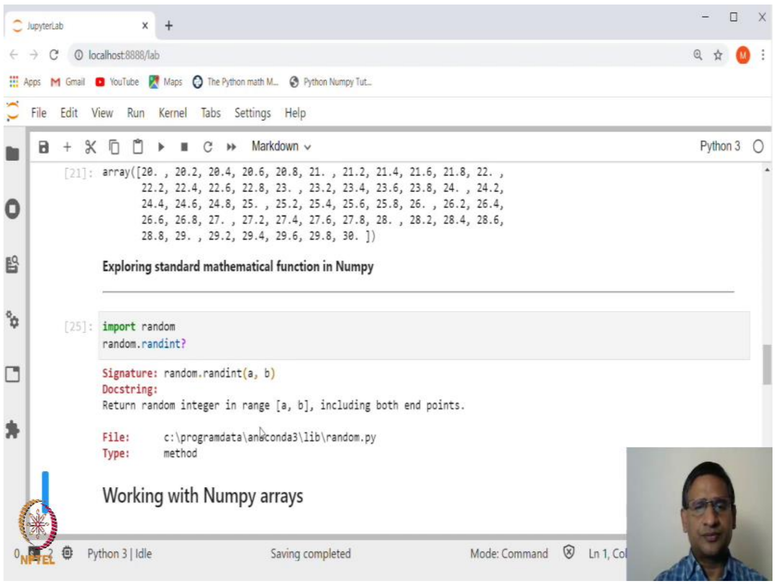


```
[21]: array([20. , 20.2, 20.4, 20.6, 20.8, 21. , 21.2, 21.4, 21.6, 21.8, 22. ,
       22.2, 22.4, 22.6, 22.8, 23. , 23.2, 23.4, 23.6, 23.8, 24. , 24.2,
       24.4, 24.6, 24.8, 25. , 25.2, 25.4, 25.6, 25.8, 26. , 26.2, 26.4,
       26.6, 26.8, 27. , 27.2, 27.4, 27.6, 27.8, 28. , 28.2, 28.4, 28.6,
       28.8, 29. , 29.2, 29.4, 29.6, 29.8, 30. ])
```

**Exploring standard mathematical function in Numpy**

```
[24]: import random
      random.randint([-10,10],20)

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-24-f1322e766d47> in <module>
      1 import random
----> 2 random.randint([-10,10],20)

C:\ProgramData\Anaconda3\lib\random.py in randint(self, a, b)
    246         """
    247
--> 248         return self.randrange(a, b+1)
```

(Refer Slide Time: 20:19)



```
[21]: array([20. , 20.2, 20.4, 20.6, 20.8, 21. , 21.2, 21.4, 21.6, 21.8, 22. ,
       22.2, 22.4, 22.6, 22.8, 23. , 23.2, 23.4, 23.6, 23.8, 24. , 24.2,
       24.4, 24.6, 24.8, 25. , 25.2, 25.4, 25.6, 25.8, 26. , 26.2, 26.4,
       26.6, 26.8, 27. , 27.2, 27.4, 27.6, 27.8, 28. , 28.2, 28.4, 28.6,
       28.8, 29. , 29.2, 29.4, 29.6, 29.8, 30. ])
```

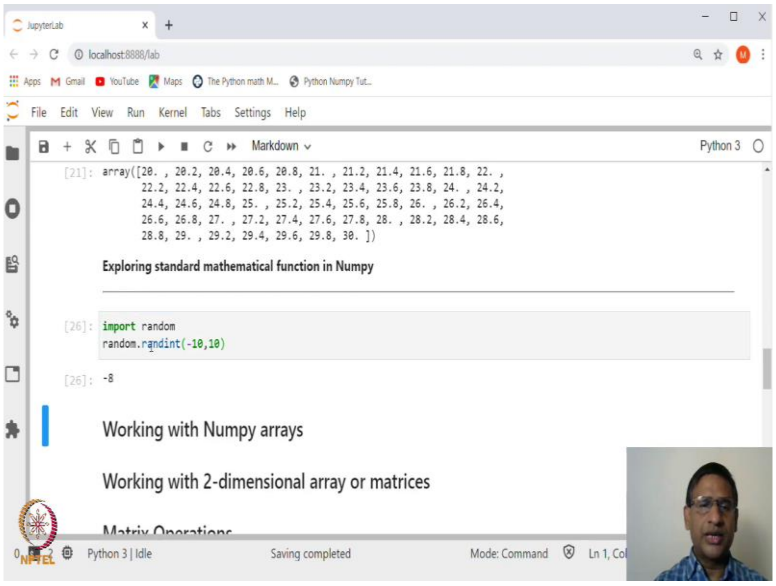
**Exploring standard mathematical function in Numpy**

```
[25]: import random
      random.randint?

Signature: random.randint(a, b)
Docstring:
Return random integer in range [a, b], including both end points.

File:      c:\programdata\anaconda3\lib\random.py
Type:      method
```

Working with Numpy arrays

Let me, let us look at what is the way to make use of this. So, we will look at question mark and then press enter so that will give me. So, this is, this generates only one random integer at a time.

(Refer Slide Time: 20:29)



```
[21]: array([20. , 20.2, 20.4, 20.6, 20.8, 21. , 21.2, 21.4, 21.6, 21.8, 22. ,
             22.2, 22.4, 22.6, 22.8, 23. , 23.2, 23.4, 23.6, 23.8, 24. , 24.2,
             24.4, 24.6, 24.8, 25. , 25.2, 25.4, 25.6, 25.8, 26. , 26.2, 26.4,
             26.6, 26.8, 27. , 27.2, 27.4, 27.6, 27.8, 28. , 28.2, 28.4, 28.6,
             28.8, 29. , 29.2, 29.4, 29.6, 29.8, 30. ])
```

**Exploring standard mathematical function in Numpy**

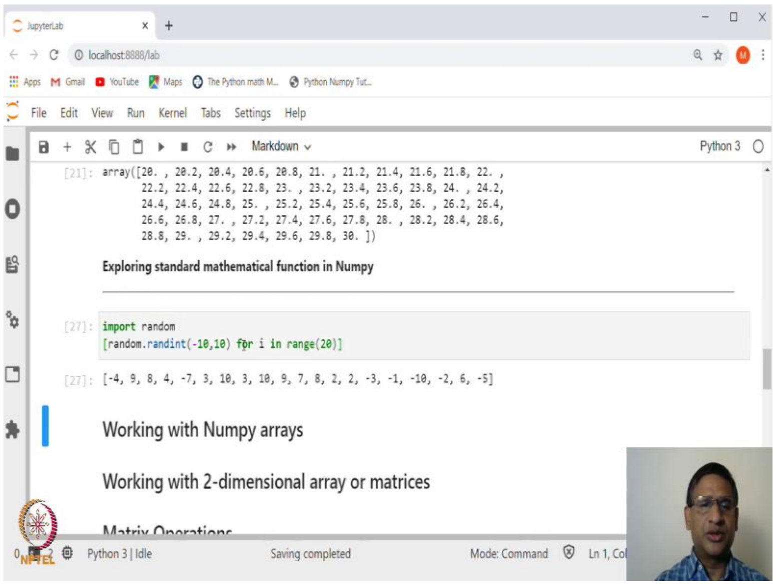
```
[26]: import random
      random.randint(-10,10)

[26]: -8
```

Working with Numpy arrays

Working with 2-dimensional array or matrices

Matrix Operations

(Refer Slide Time: 20:35)



```
[21]: array([20. , 20.2, 20.4, 20.6, 20.8, 21. , 21.2, 21.4, 21.6, 21.8, 22. ,
             22.2, 22.4, 22.6, 22.8, 23. , 23.2, 23.4, 23.6, 23.8, 24. , 24.2,
             24.4, 24.6, 24.8, 25. , 25.2, 25.4, 25.6, 25.8, 26. , 26.2, 26.4,
             26.6, 26.8, 27. , 27.2, 27.4, 27.6, 27.8, 28. , 28.2, 28.4, 28.6,
             28.8, 29. , 29.2, 29.4, 29.6, 29.8, 30. ])
```

**Exploring standard mathematical function in Numpy**

```
[27]: import random
      [random.randint(-10,10) for i in range(20)]

[27]: [-4, 9, 8, 4, -7, 3, 10, 3, 10, 9, 7, 8, 2, 2, -3, -1, -10, -2, 6, -5]
```

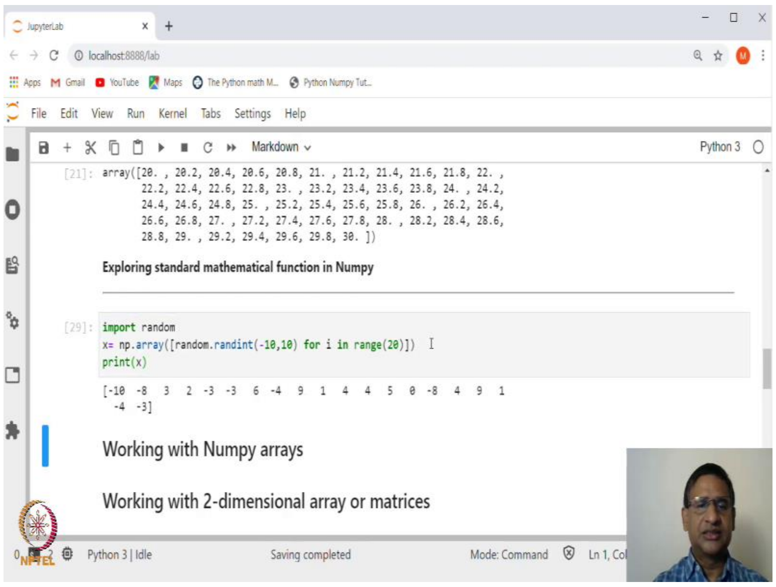
Working with Numpy arrays

Working with 2-dimensional array or matrices

Matrix Operations

So, we will simply say -10 to 10, this is one, and let us make a list of this list. How many we want? For i in range 20. This will be 20 random integers and create a, create a, for example, an array of this.

(Refer Slide Time: 20:55)



```
[21]: array([20. , 20.2, 20.4, 20.6, 20.8, 21. , 21.2, 21.4, 21.6, 21.8, 22. ,
             22.2, 22.4, 22.6, 22.8, 23. , 23.2, 23.4, 23.6, 23.8, 24. , 24.2,
             24.4, 24.6, 24.8, 25. , 25.2, 25.4, 25.6, 25.8, 26. , 26.2, 26.4,
             26.6, 26.8, 27. , 27.2, 27.4, 27.6, 27.8, 28. , 28.2, 28.4, 28.6,
             28.8, 29. , 29.2, 29.4, 29.6, 29.8, 30. ])
```

**Exploring standard mathematical function in Numpy**

```
[29]: import random
      x= np.array([random.randint(-10,10) for i in range(20)])
      print(x)

[-10  -8   3   2  -3  -3   6  -4   9   1   4   4   5   0  -8   4   9   1
  -4  -3]
```

Working with Numpy arrays

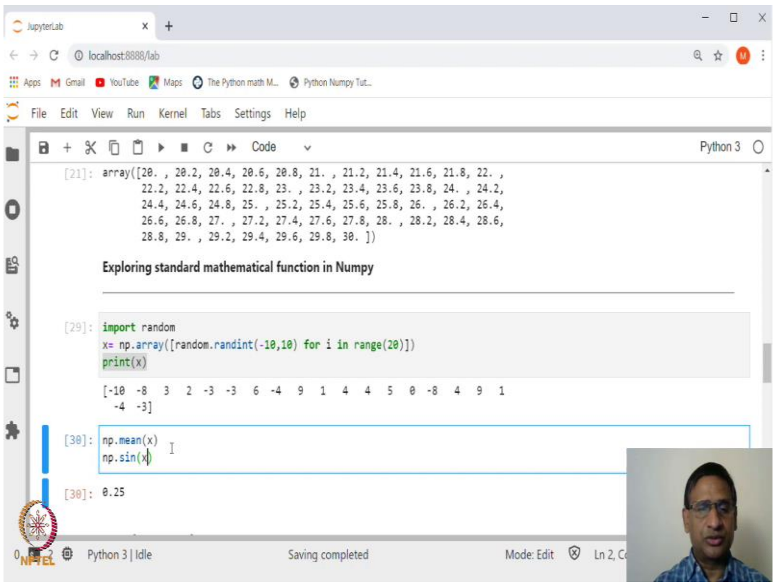
Working with 2-dimensional array or matrices

So, we will say np dot array; np dot array.

So, inside Numpy, there is a function called array which can generate array one-dimensional array, it can generate multidimensional array. So, we will see how to generate 2-dimensional array that is a matrix, or 1-dimensional array can be thought of also as a vector. So, this,let me store this into, let me store this in x. So, this is a. So, let us see what is x. print(x) that is the 1-dimensional array?

Now, let us say we want to apply any scientific function to this including functions which we define on our own that is user-defined function or any function that is inside Numpy. So, Numpy functions can be operated on any array? So, that is the one difference between mathematical functions inside math module and functions inside Numpy module.

(Refer Slide Time: 22:09)



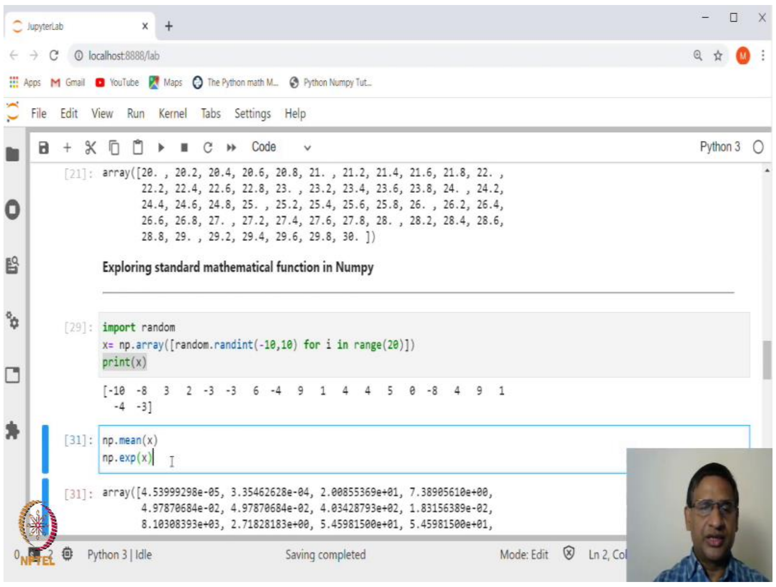
```
[21]: array([20. , 20.2, 20.4, 20.6, 20.8, 21. , 21.2, 21.4, 21.6, 21.8, 22. ,
             22.2, 22.4, 22.6, 22.8, 23. , 23.2, 23.4, 23.6, 23.8, 24. , 24.2,
             24.4, 24.6, 24.8, 25. , 25.2, 25.4, 25.6, 25.8, 26. , 26.2, 26.4,
             26.6, 26.8, 27. , 27.2, 27.4, 27.6, 27.8, 28. , 28.2, 28.4, 28.6,
             28.8, 29. , 29.2, 29.4, 29.6, 29.8, 30. ])
```

**Exploring standard mathematical function in Numpy**

```
[29]: import random
      x= np.array([random.randint(-10,10) for i in range(20)])
      print(x)

      [-10  -8   3   2  -3  -3   6  -4   9   1   4   4   5   0  -8   4   9   1
        -4  -3]

[30]: np.mean(x)
      np.sin(x)

[30]: 0.25
```

So, if I say, for example, x   n n np dot, let us say mean, right? So, the mean of x, it will give me what is the mean.

(Refer Slide Time: 22:32)



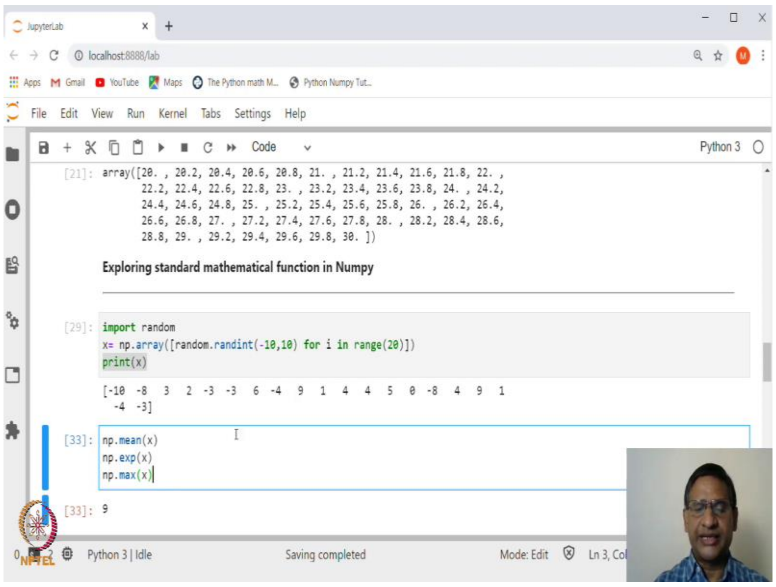
```
[21]: array([20. , 20.2, 20.4, 20.6, 20.8, 21. , 21.2, 21.4, 21.6, 21.8, 22. ,
             22.2, 22.4, 22.6, 22.8, 23. , 23.2, 23.4, 23.6, 23.8, 24. , 24.2,
             24.4, 24.6, 24.8, 25. , 25.2, 25.4, 25.6, 25.8, 26. , 26.2, 26.4,
             26.6, 26.8, 27. , 27.2, 27.4, 27.6, 27.8, 28. , 28.2, 28.4, 28.6,
             28.8, 29. , 29.2, 29.4, 29.6, 29.8, 30. ])
```

**Exploring standard mathematical function in Numpy**

```
[29]: import random
      x= np.array([random.randint(-10,10) for i in range(20)])
      print(x)

      [-10  -8   3   2  -3  -3   6  -4   9   1   4   4   5   0  -8   4   9   1
        -4  -3]

[31]: np.mean(x)
      np.exp(x)

[31]: array([4.53999298e-05, 3.35462628e-04, 2.00855369e+01, 7.38905610e+00,
             4.97870684e-02, 4.97870684e-02, 4.03428793e+02, 1.83156389e-02,
             8.10308393e+03, 2.71828183e+00, 5.45981500e+01, 5.45981500e+01,
```

If I say np dot sine of x it will give me the sine of each of this function, instead of sine I can say exponential of this, it will give me the exponential.

(Refer Slide Time: 22:37)



```
[21]: array([20. , 20.2, 20.4, 20.6, 20.8, 21. , 21.2, 21.4, 21.6, 21.8, 22. ,
             22.2, 22.4, 22.6, 22.8, 23. , 23.2, 23.4, 23.6, 23.8, 24. , 24.2,
             24.4, 24.6, 24.8, 25. , 25.2, 25.4, 25.6, 25.8, 26. , 26.2, 26.4,
             26.6, 26.8, 27. , 27.2, 27.4, 27.6, 27.8, 28. , 28.2, 28.4, 28.6,
             28.8, 29. , 29.2, 29.4, 29.6, 29.8, 30. ])
```

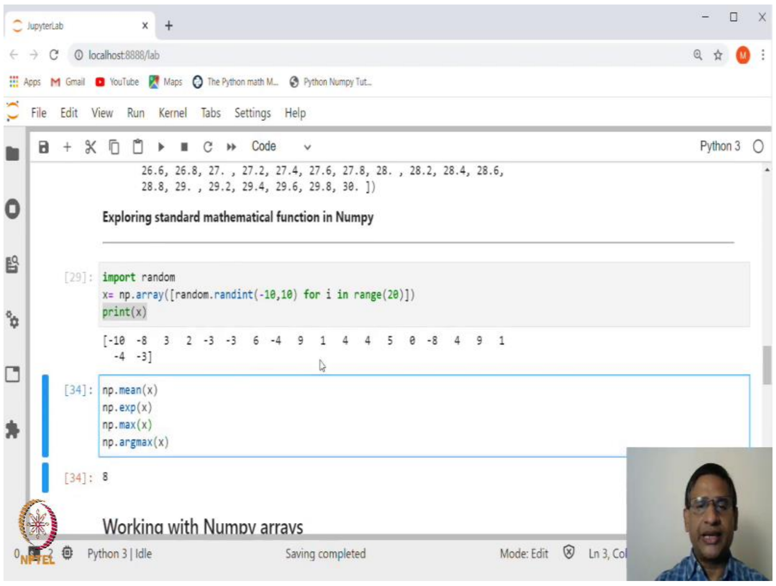
**Exploring standard mathematical function in Numpy**

```
[29]: import random
      x= np.array([random.randint(-10,10) for i in range(20)])
      print(x)

      [-10  -8   3   2  -3  -3   6  -4   9   1   4   4   5   0  -8   4   9   1
        -4  -3]

[33]: np.mean(x)
      np.exp(x)
      np.max(x)

[33]: 9
```

I can say np dot and then let us use, make use of some other functions, for example, I can say max.

So, it will find, sorry np dot max of x it will find maximum of the list that is what the maximum is 9 here.

(Refer Slide Time: 23:02)



```
             26.6, 26.8, 27. , 27.2, 27.4, 27.6, 27.8, 28. , 28.2, 28.4, 28.6,
             28.8, 29. , 29.2, 29.4, 29.6, 29.8, 30. ])
```

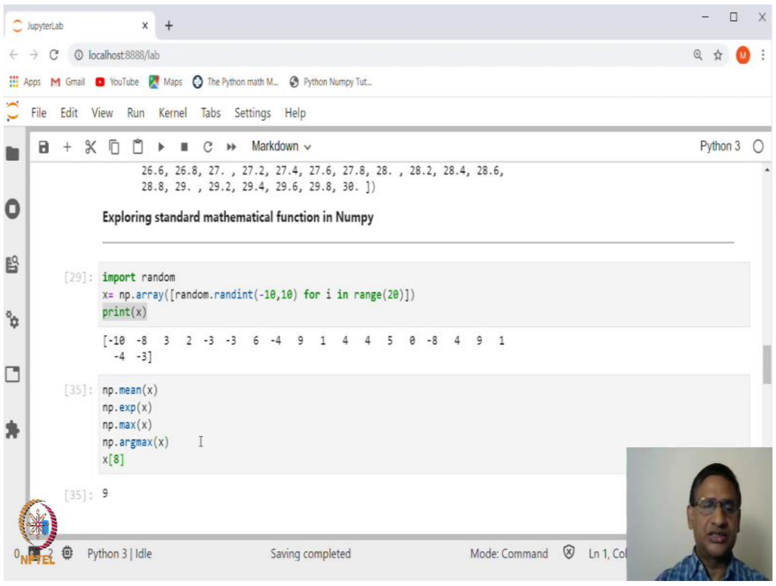
**Exploring standard mathematical function in Numpy**

```
[29]: import random
      x= np.array([random.randint(-10,10) for i in range(20)])
      print(x)

      [-10  -8   3   2  -3  -3   6  -4   9   1   4   4   5   0  -8   4   9   1
        -4  -3]

[34]: np.mean(x)
      np.exp(x)
      np.max(x)
      np.argmax(x)

[34]: 8
```

Working with Numpy arrays

And you can simply say, you can also say np dot argmax of x. What will it find? It will find the argument of the maximum value. So, in this case, 9, 9 let us execute this it says 8; that means, the x of 8 is 9 you can check here x of 8 is 9 correct.

(Refer Slide Time: 23:23)



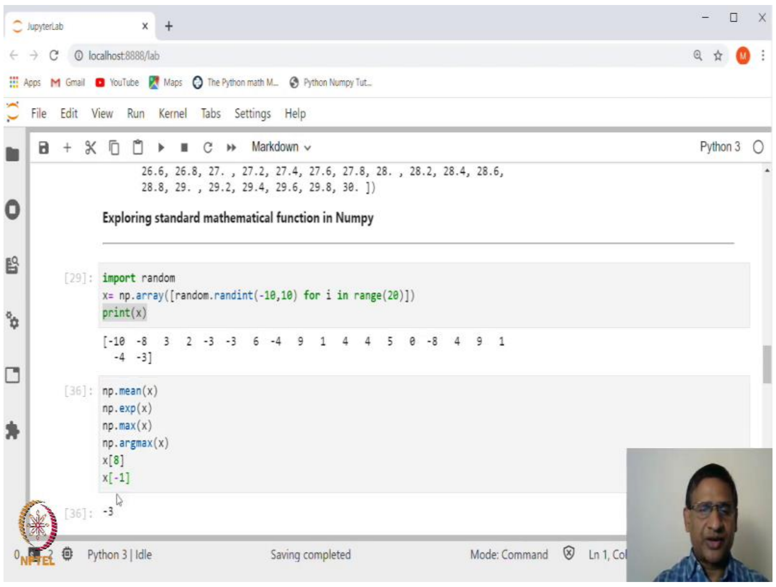
```
26.6, 26.8, 27. , 27.2, 27.4, 27.6, 27.8, 28. , 28.2, 28.4, 28.6,
28.8, 29. , 29.2, 29.4, 29.6, 29.8, 30. ])
```

**Exploring standard mathematical function in Numpy**

```
[29]: import random
      x= np.array([random.randint(-10,10) for i in range(20)])
      print(x)

      [-10  -8   3   2  -3  -3   6  -4   9   1   4   4   5   0  -8   4   9   1
        -4  -3]

[35]: np.mean(x)
      np.exp(x)
      np.max(x)
      np.argmax(x)
      x[8]

[35]: 9
```

So, it gives you the index for which that particular index is the maximum. Similarly, you can do argmin and things like that? So, actually, if you want to take any slice of this list it is exactly similar to how we took slice of any list in, in Python. So, that is exactly the same.

(Refer Slide Time: 23:48)



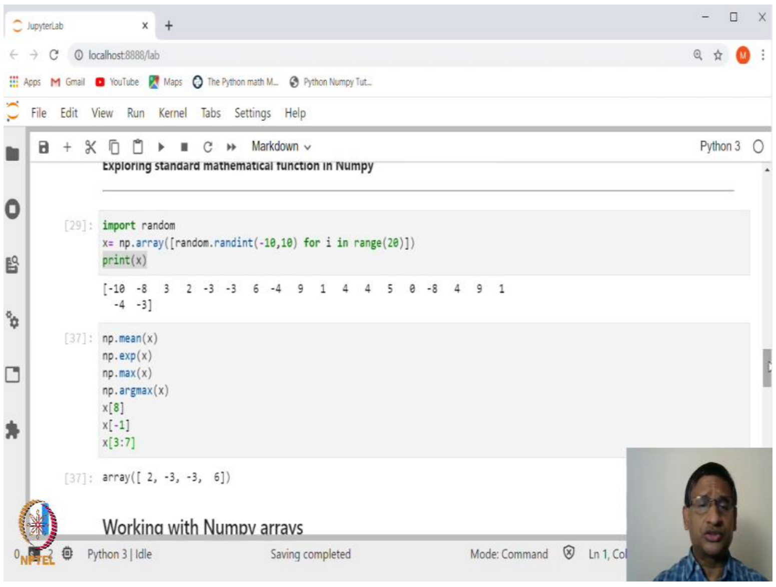
```
26.6, 26.8, 27. , 27.2, 27.4, 27.6, 27.8, 28. , 28.2, 28.4, 28.6,
28.8, 29. , 29.2, 29.4, 29.6, 29.8, 30. ])
```

**Exploring standard mathematical function in Numpy**

```
[29]: import random
      x= np.array([random.randint(-10,10) for i in range(20)])
      print(x)

      [-10  -8   3   2  -3  -3   6  -4   9   1   4   4   5   0  -8   4   9   1
        -4  -3]

[36]: np.mean(x)
      np.exp(x)
      np.max(x)
      np.argmax(x)
      x[8]
      x[-1]

[36]: -3
```

I can also see if I want to look at what is the last number I can say x of minus 1.

(Refer Slide Time: 23:55)



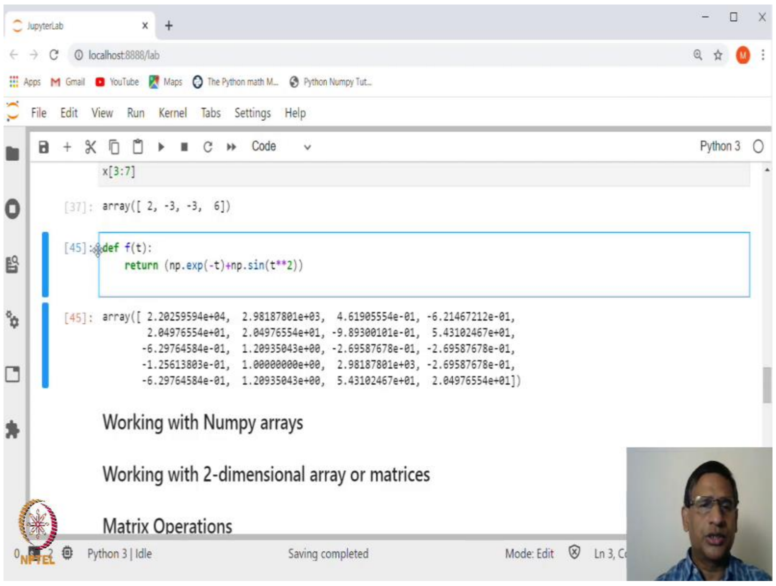
**Exploring standard mathematical function in Numpy**

```
[29]: import random
      x= np.array([random.randint(-10,10) for i in range(20)])
      print(x)

      [-10  -8   3   2  -3  -3   6  -4   9   1   4   4   5   0  -8   4   9   1
        -4  -3]

[37]: np.mean(x)
      np.exp(x)
      np.max(x)
      np.argmax(x)
      x[8]
      x[-1]
      x[3:7]

[37]: array([ 2, -3, -3,  6])
```

Working with Numpy arrays

If I want to find, let us say x of 3 to 7 then it will give me x 3, x 4, x 5, x 6 that is all you have and of course, it will create as an array. So, you can apply any function to this list right?

(Refer Slide Time: 24:14)



```
      x[3:7]

[37]: array([ 2, -3, -3,  6])

[45]: def f(t):
          return (np.exp(-t)+np.sin(t**2))

[45]: array([ 2.20259594e+04,  2.98187801e+03,  4.61905554e-01, -6.21467212e-01,
              2.04976554e+01,  2.04976554e+01, -9.89300101e-01,  5.43102467e+01,
             -6.29764584e-01,  1.20935043e+00, -2.69587678e-01, -2.69587678e-01,
             -1.25613803e-01,  1.00000000e+00,  2.98187801e+03, -2.69587678e-01,
             -6.29764584e-01,  1.20935043e+00,  5.43102467e+01,  2.04976554e+01])
```

Working with Numpy arrays

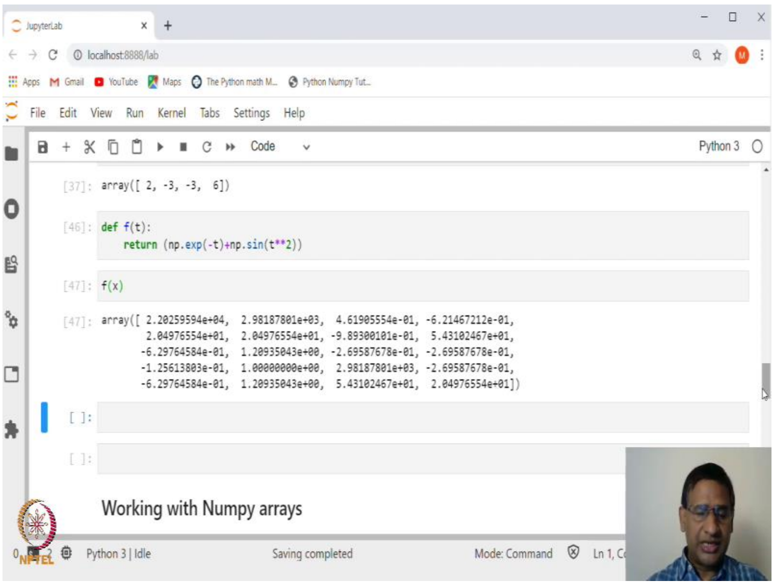
Working with 2-dimensional array or matrices

Matrix Operations

So, we can create our own function let us say f(t) is equal to exponential(t) + sin(t^2). So, of course, we want to apply this on a list x then we have to use this exponential and sin function from the Numpy library. So, that is why this returns np dot exponential minus t np plus np.sin(t^2).

Now let us run this and let us call this function.

(Refer Slide Time: 24:46)



```
[37]: array([ 2, -3, -3,  6])

[46]: def f(t):
          return (np.exp(-t)+np.sin(t**2))

[47]: f(x)

[47]: array([ 2.20259594e+04,  2.98187801e+03,  4.61905554e-01, -6.21467212e-01,
              2.04976554e+01,  2.04976554e+01, -9.89300101e-01,  5.43102467e+01,
             -6.29764584e-01,  1.20935043e+00, -2.69587678e-01, -2.69587678e-01,
             -1.25613803e-01,  1.00000000e+00,  2.98187801e+03, -2.69587678e-01,
             -6.29764584e-01,  1.20935043e+00,  5.43102467e+01,  2.04976554e+01])

[ ]:

[ ]:
```

Working with Numpy arrays

So, if I say f(x) you will see the f of each of the entries inside this list ok. So, you can apply any function to a list that you have created ok? So, next time we will look at how to deal with 2-dimensional array even 3-dimensional array and we will look at how to work with matrices.

So, there is another library inside Numpy which is called linalg module and inside linalg you have basic functions to deal with linear algebra methods ok; so, I will see you in the next lecture.

Thank you very much.