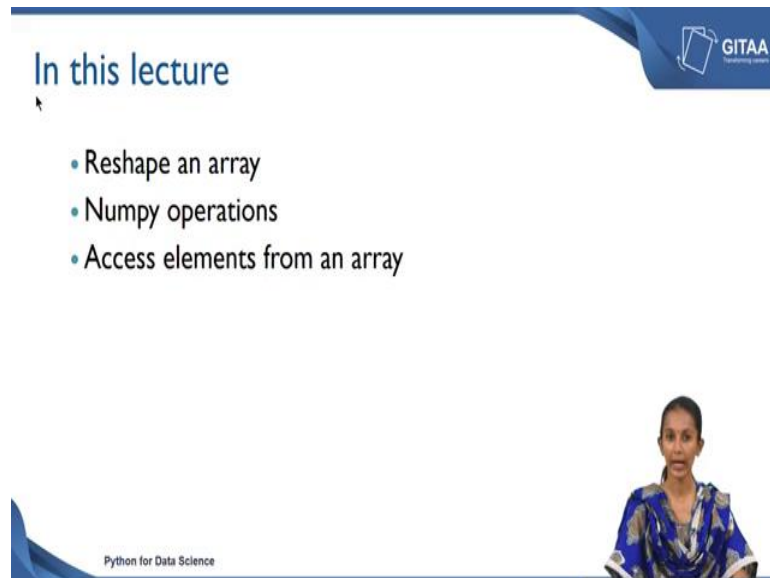


**Python for Data Science**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Madras**

**Lecture – 13**  
**Numpy Part – 2**

(Refer Slide Time: 00:16)



**In this lecture**

- Reshape an array
- Numpy operations
- Access elements from an array

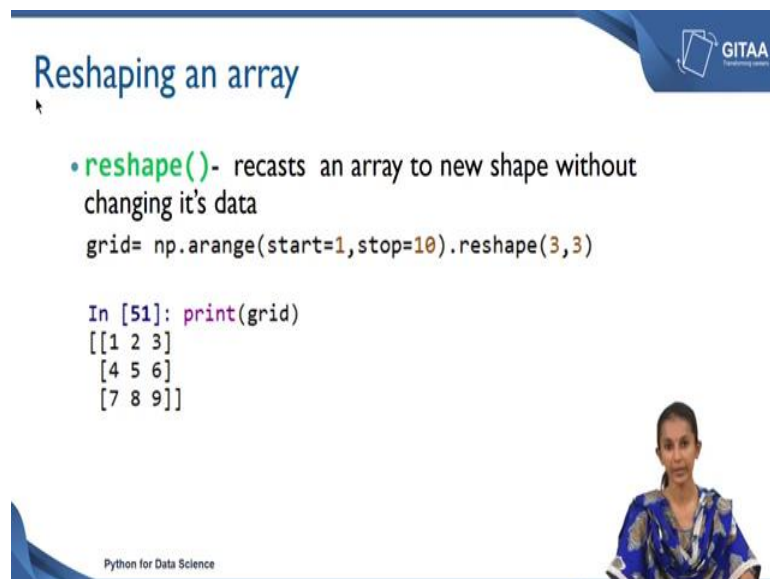
Python for Data Science

GITAA

The slide features a blue header with the text 'In this lecture' and a list of three bullet points. In the bottom right corner, there is a small video inset showing a woman in a blue and white patterned sari. The footer contains the text 'Python for Data Science' and the 'GITAA' logo.

Welcome to the lecture in this lecture we will see how to reshape an array and also we will see some of the numpy operations which are available in python. How to access elements from an array?

(Refer Slide Time: 00:29)



**Reshaping an array**

- **reshape()**- recasts an array to new shape without changing it's data

```
grid= np.arange(start=1,stop=10).reshape(3,3)
```

```
In [51]: print(grid)
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

Python for Data Science

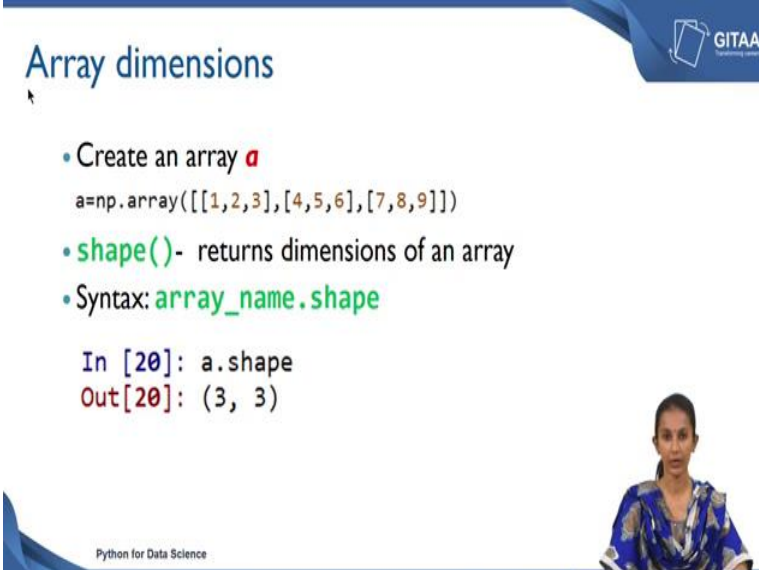
GITAA

The slide features a blue header with the text 'Reshaping an array' and a single bullet point describing the 'reshape()' function. Below the text, there are two code snippets: one for creating a grid and one for printing it. In the bottom right corner, there is a small video inset showing a woman in a blue and white patterned sari. The footer contains the text 'Python for Data Science' and the 'GITAA' logo.

So, let us get started in the previous lecture we saw how to create an array. So, using the arange function let see how to reshape an array? Reshape function is an in built function which is available in python. So, it recasts an array to the new shape it does not change this data. First we will create an array using the arange function as well as using the reshape function np.arange before calling the np we need to import numpy as np and then you can call as np.arange.

So, you have to specify the start value which is I am giving start equal to 1 and stop values equal to 10, since we wanted to create a 3 cross 3 array. So, I will use the.reshape command. So,.reshape 3 comma 3 which basically creates 3 rows and 3 columns and I am storing it in a variable call grid. So, let us print the grid so we will be getting a 3 cross 3 array which is so the first row as elements 1, 2, 3 second row as 4, 5, 6 third row consists of 7 8 and 9. So, this is also the one way of creating an array using the arange function as well as using the reshape function which is available in python.

(Refer Slide Time: 01:55)



The slide is titled "Array dimensions" and features a blue header with the GITAA logo. It contains a list of bullet points and a code block. The first bullet point is "Create an array **a**" followed by the code `a=np.array([[1,2,3],[4,5,6],[7,8,9]])`. The second bullet point is "**shape()**- returns dimensions of an array". The third bullet point is "Syntax: **array\_name.shape**". Below the code block, it shows the output: `In [20]: a.shape` and `Out[20]: (3, 3)`. In the bottom right corner, there is a small video inset of a woman in a blue patterned dress. The footer of the slide says "Python for Data Science".

- Create an array **a**  
`a=np.array([[1,2,3],[4,5,6],[7,8,9]])`
- **shape()**- returns dimensions of an array
- Syntax: **array\_name.shape**

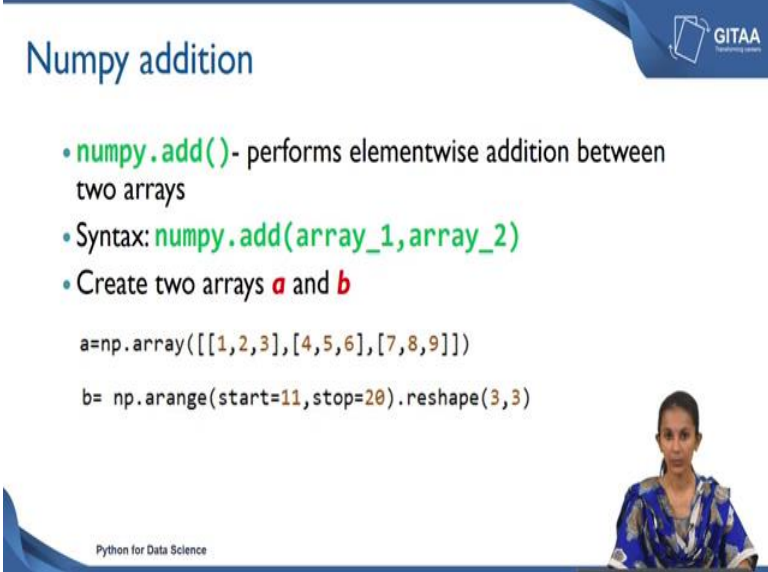
```
In [20]: a.shape
Out[20]: (3, 3)
```

Python for Data Science

Let us look at the array dimensions. So, we will create an array a so I will show you second method to create an array np.array before calling np you have to import numpy as np and then you can call np.array() inside the parenthesis you have to specify the values. If you wanted to create 3 by 3 array then you have to specify the values for the first row as well as for the second row and the third row in the first square brackets.

So, I mention the value 1, 2, 3 which corresponds to the first row; 4, 5, 6 it corresponds to the second row similarly for the third row. So, when you print a you will be getting a 3 by 3 array, if I wanted to check the dimensions of an array. So, you can use the in built command which is shape it basically returns the dimensions of an array the syntax is array name followed by the .shape a.shape it basically. Gives the dimensions it is a 3 comma 3 which consists of 3 rows and 3 columns.

(Refer Slide Time: 03:13)



## Numpy addition

- `numpy.add()` - performs elementwise addition between two arrays
- Syntax: `numpy.add(array_1, array_2)`
- Create two arrays **a** and **b**

```
a=np.array([[1,2,3],[4,5,6],[7,8,9]])  
b= np.arange(start=11,stop=20).reshape(3,3)
```

Python for Data Science

Next we will look at some of the numpy operations which is available in python. First we will start with addition. So, `numpy.add` so it is an in built command which is available in python, it performs an element wise addition between the two arrays. So, let us look at the syntax, `numpy.add()` inside the parenthesis specify the `array_1`, `array_2`.

So, let us create two array which is a and b we already created an array a, which has 3 by 3 array which consists of values from 1 to 9. Similarly we will create an another array b will use the `np.arange` command. So, I will specify the start value 11 and the stop values is 20; if you wanted to add arrays element y. So, we need a 3 by 3 array right. So, that is why we have specified `reshape 3 comma 3`.

(Refer Slide Time: 04:03)


## Numpy addition

- Print **a** and **b**

```
In [16]: print(a)
[[1 2 3]
 [4 5 6]
 [7 8 9]]

In [17]: print(b)
[[11 12 13]
 [14 15 16]
 [17 18 19]]
```

`np.add(a,b)` → `Out[23]:`  
`array([[12, 14, 16],`  
 `[18, 20, 22],`  
 `[24, 26, 28]])`



Python for Data Science


So, we will print the both the arrays printed the array a which has values from 1 to 9, similarly a printed the array b which consists of values from 11 to 19. So, we will perform the element wise addition on both these arrays. So, for that the command is `np.add a comma b`. So, `np.add` it adds the elements element wise. So, it adds the first element in the array a and the first element in the array b. So, it is  $1+11$  which is you have got 12, similarly does for all the elements.

(Refer Slide Time: 04:43)

## Numpy multiplication

- `numpy.multiply()` - performs elementwise multiplication between two arrays
- Syntax: `numpy.multiply(array_1,array_2)`
- Consider the same arrays **a** and **b**

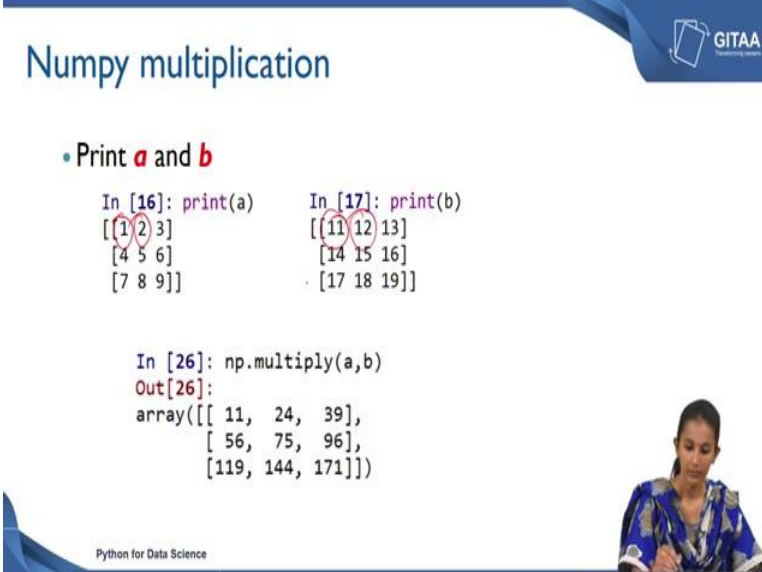
```
a=np.array([[1,2,3],[4,5,6],[7,8,9]])
b= np.arange(start=11,stop=20).reshape(3,3)
```



Python for Data Science

Next is numpy multiplication. So, numpy.multiply it performs the element wise multiplication between the two arrays. The syntax is numpy.multiply inside the parenthesis again you have to specify the array 1 and array 2. So, we will consider the same arrays which is a and b which we have already created a is equal to np.array which consist of values from 1 to 9; it is a 3 by 3 array.

(Refer Slide Time: 05:21)



## Numpy multiplication

- Print **a** and **b**

```
In [16]: print(a)      In [17]: print(b)
[[1 2 3]              [[11 12 13]
 [4 5 6]              [14 15 16]
 [7 8 9]]             [17 18 19]]
```

```
In [26]: np.multiply(a,b)
Out[26]:
array([[ 11,  24,  39],
       [ 56,  75,  96],
       [119, 144, 171]])
```


Python for Data Science


Similarly, b value starts from 11 to 19 and again it is a 3 by 3 array. So, again I am printing the both the arrays a and b will perform the element wise multiplication on these both the arrays. So, the command will be np.multiply a comma b. So, it multiplies element wise right. So, it takes the first element in the array a and first element in the array b. So, it is 1 into 11, so 11 similarly for the remaining elements so it is 2 into 12 which is 24.

(Refer Slide Time: 05:51)

## Other numpy functions

Function name	Description
<code>numpy.subtract</code>	performs element wise subtraction between two arrays
<code>numpy.divide</code>	returns an element wise division of inputs
<code>numpy.remainder</code>	Return element-wise remainder of division





Python for Data Science

So, there are also other functions which is available in python: `numpy.subtract`, you can perform the element wise subtraction between the two arrays; `numpy.divide` you can do a element wise divisions; `numpy.remainder` returns the element wise reminders of the divisions. So, you can also try these things.

(Refer Slide Time: 06:12)

## Accessing components of an array


- Components of an array can be accessed using index number


```
In [18]: print(a)
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

- Extract element with index (0,1) from **a**

```
In [22]: a[0,1]
Out[22]: 2
```





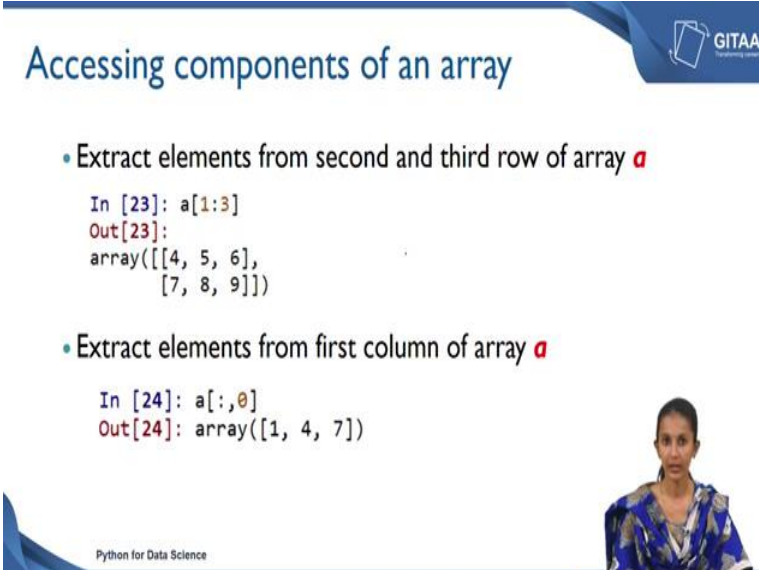
Python for Data Science

Next we will look at how to access components of an array. So, the components of an array it can be accessed using based on the index number. So, this is an array which we are created earlier. So, it is a 3 by 3 array. So, I drawn a table, I filled up with the values

from 1 to 9. So, for the first row it corresponds to the zeroth index value, for the second row it corresponds to the first index value and similarly for the third row it corresponds to the second index value. This is for the row wise index, similarly for the column wise first column it corresponds to the zeroth index, for the second column it corresponds to the first index, for a third column it corresponds to the second index.

So, in python the indexing starts from 0 to n-1. Let say if you wanted to extract element 2. So, the corresponding index will be 0 which is for the along the row wise and 1 along for the column wise. So, I wanted to extract this element 2. Let see how to do that? We have to specify the array name a and then inside the square bracket we have to specify the row index number followed by the column index number. So, 0 stands for the row index, 1 stands for the column index. So, we will be getting an output of 2.

(Refer Slide Time: 07:49)



The slide is titled "Accessing components of an array" and features the GITAA logo in the top right corner. It contains two bullet points with corresponding Python code snippets and their outputs. The first bullet point shows how to extract elements from the second and third rows of an array 'a', resulting in a 2x3 array. The second bullet point shows how to extract elements from the first column of array 'a', resulting in a 1D array. A small inset image of a woman is visible in the bottom right corner of the slide.

### Accessing components of an array

- Extract elements from second and third row of array **a**  

```
In [23]: a[1:3]  
Out[23]:  
array([[4, 5, 6],  
       [7, 8, 9]])
```
- Extract elements from first column of array **a**  

```
In [24]: a[:,0]  
Out[24]: array([1, 4, 7])
```

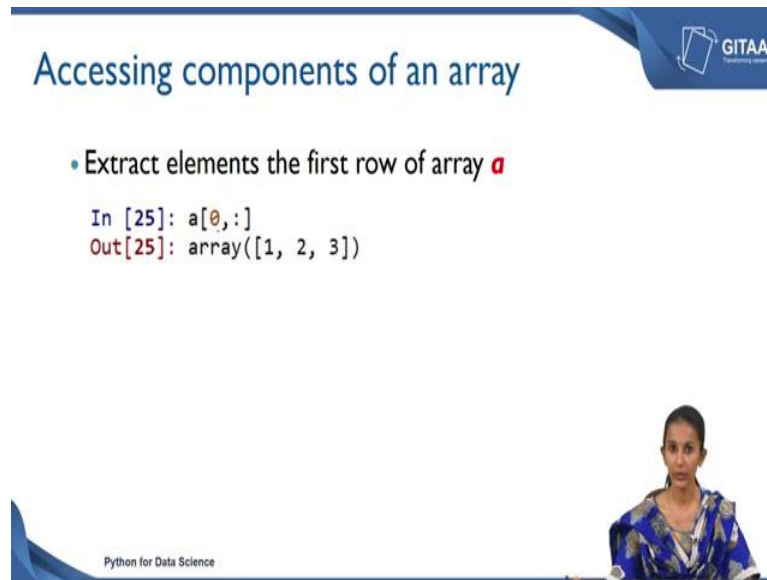
Python for Data Science

Let say if you wanted to extract elements from second and third row of an array a. So, the command is array name a followed by the index numbers. So, we wanted to extract second and third row right. So, the corresponding index will be 1 and 2. So, I am starting with the value of index number 1 colon 3 because in python indexing starts from 0 to n-1, we have specified till 3. So, it is 3-1 it returns a values for the index number 1 and 2. We have got an output of 4, 5, 6, 7, 8, 9.

So, this corresponds to the index number for 1 and 2. Let say if you wanted to extract elements from the first column of array a. So, you can specify the array name a. So,

inside the square brackets colon which means takes all the rows comma. So, I wanted to extract first column. So, the corresponding index number is 0, so I given array a colon comma 0. So, it prints the output of 1, 4 and 7.

(Refer Slide Time: 09:05)



**Accessing components of an array**

- Extract elements the first row of array **a**

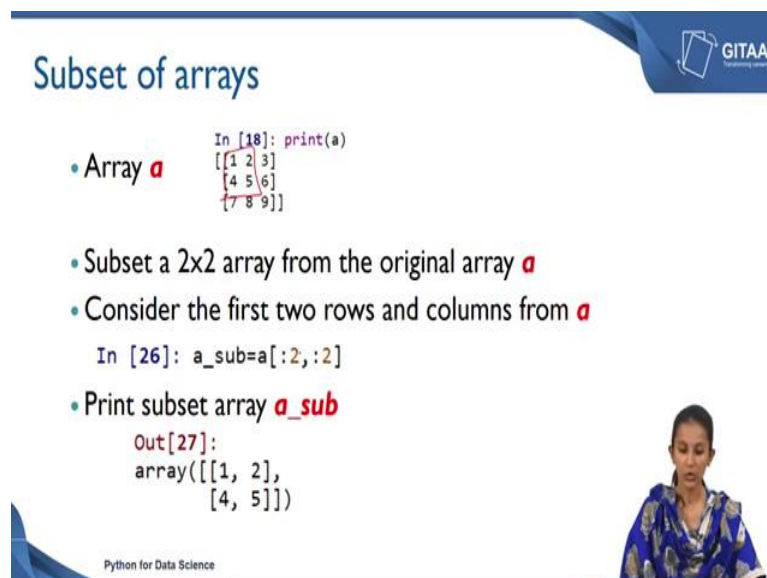
```
In [25]: a[0,:]  
Out[25]: array([1, 2, 3])
```

Python for Data Science

GITAA

If you wanted to extract elements the first row in the array a. So, you can use array name a inside the square brackets you can specify the row index. So, you wanted to extract the first row, so zeroth index comma colon. So, which means it takes the corresponding columns. So, it returns an output of 1, 2 and 3.

(Refer Slide Time: 09:31)



**Subset of arrays**

- Array **a**

```
In [18]: print(a)  
[[1 2 3]  
 [4 5 6]  
 [7 8 9]]
```

- Subset a 2x2 array from the original array **a**
- Consider the first two rows and columns from **a**

```
In [26]: a_sub=a[:2,:2]
```

- Print subset array **a\_sub**

```
Out[27]:  
array([[1, 2],  
       [4, 5]])
```

Python for Data Science

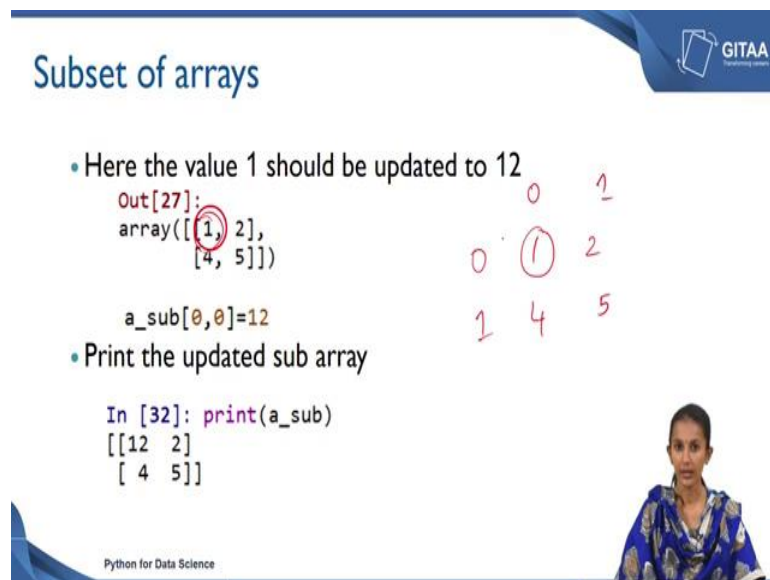
GITAA



So, till now we saw how to access the components of an array. So, if you wanted to subset arrays from the original array we can also do that let us look at how to subset the arrays. So, this is have an original array a which is the 3 by 3 array, it has values from 1 to 9. So, we will subset a 2 by 2 array from the original array. So, we will consider the first two rows from this array.

So, I will subset this 2 by 2 array which is 1, 2, 4 and 5. So, I am storing it in a variable call a underscore sub equal to a. So, I wanted to extract this first two rows and first two columns right. So, I am specifying colon 2 comma colon 2, when you print the subset array. So, we will be getting an out of 1, 2, 4 and 5.

(Refer Slide Time: 10:33)



The slide is titled "Subset of arrays" and features a GITAA logo in the top right corner. It contains the following content:

- Here the value 1 should be updated to 12

```
Out[27]:  
array([[1, 2],  
       [4, 5]])
```

Next to the code, a 2x2 array is shown with handwritten red annotations: the first row contains 0 and 1, and the second row contains 0 and 1. The value 1 in the first row, second column is circled in red.

```
a_sub[0,0]=12
```

- Print the updated sub array

```
In [32]: print(a_sub)  
[[12  2]  
 [ 4  5]]
```

At the bottom right of the slide, there is a small video inset showing a woman speaking. The text "Python for Data Science" is visible at the bottom left of the slide.

Let say if you wanted to modify the value of 1 to 12. Let see how to do that? So, the index position correspond to this sub array is it is 1, 2, 4, 5 right. So, the corresponding index position 0, 1, 0 and 1. So, for this 1 if you wanted to change the value the corresponding index position is 0 comma 0 right; a underscore sub 0 comma 0 and I am assigning to a value to 12 when we print the updated sub array. So, it will be 12, 2, 4 and 5. So, 1 will be has been modified to 12.

(Refer Slide Time: 11:18)



## Subset of arrays

- Modifying the subset will automatically update the original array as well

```
In [33]: print(a)
[[12 2 3]
 [ 4 5 6]
 [ 7 8 9]]
```

1 2 3  
4 5 6  
7 8 9

Python for Data Science



So, modifying the subset will automatically update the original array as well. The earlier we had 1, 2, 3, 4, 5, 6, 7, 8, 9. But now when you see the left hand side the value has been changed to 12. So, modifying these subset array will automatically update the original array as well.

(Refer Slide Time: 11:44)

## Modifying array using transpose ( )



- `numpy.transpose()` - permute the dimensions of array
- Syntax: `numpy.transpose(array)`

```
In [33]: print(a)
[[12 2 3]
 [ 4 5 6]
 [ 7 8 9]]
```

→

```
In [8]: np.transpose(a)
Out[8]:
array([[12, 4, 7],
       [ 2, 5, 8],
       [ 3, 6, 9]])
```

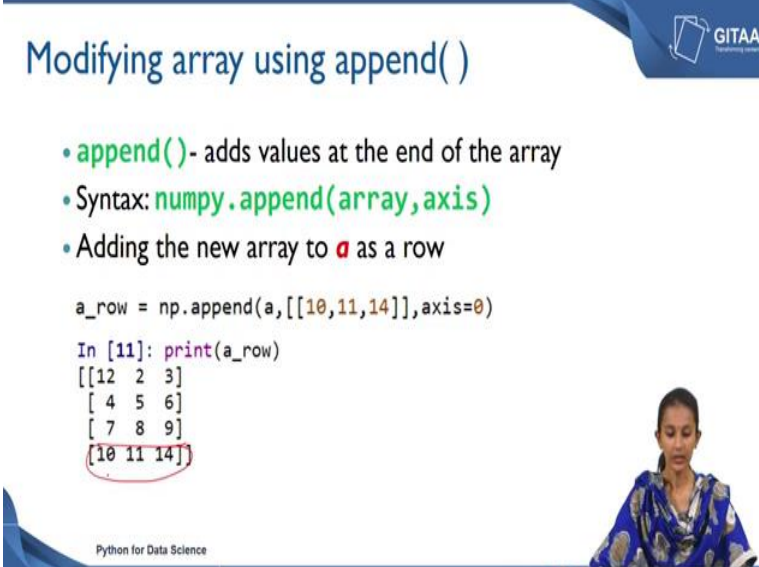
Python for Data Science



So, we can also modifying array using the transpose function. Let us say if you wanted to change the rows to columns or columns to rows you can use the transpose function which is available in python. The syntax is we have to specify the `numpy.transpose()`

inside the parenthesis we have to specify the array in a. So, this is I have an updated array, so 12, 2, 3, 4, 5, 6, 7, 8, 9. So, if you wanted to interchange rows. So, this is our rows first row, second row, third row. So, if you wanted to interchange rows to columns you can use in np.transpose of a. So, when you compare the left hand side output and the right hand side output you can visualize the change. So, 12, 2, 3 which is in the first row has been transpose to the first column.

(Refer Slide Time: 12:36)



### Modifying array using append( )

- `append()` - adds values at the end of the array
- Syntax: `numpy.append(array,axis)`
- Adding the new array to `a` as a row

```
a_row = np.append(a,[[10,11,14]],axis=0)

In [11]: print(a_row)
[[12  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 14]]
```

Python for Data Science

Let see how to add a new array to the existing array. So, you can add a new array to the existing using the append function. The syntax is numpy.append you have to specify the array name followed by the axis. So, if you wanted to add along the row wise or the column wise. So, I am adding the new row to the existing array which is a along the row wise the command is np.append.


So, this is our existing array a followed by the new values. So, I wanted to add along the row wise. So, it is already a 3 by 3 array. So, we need 3 values right. So, I am specifying the values for the rows, so which is 10 11 14 followed by the axis. So, I wanted add along the row wise and specifying axis is equal to 0. So, you can also store it in a variable a underscore row if you print a underscore row. So, we already had up to till this right. So, this is our new array which has been added along the row wise.

(Refer Slide Time: 13:58)


## Modifying array using append()

- Adding the new array to `a` as a column
- Create an array and reshape to column array

```
col=np.array([21,22,23]).reshape(3,1)
In [49]: print(col)
[[21]
 [22]
 [23]]
a_col=np.append(a,col,axis=1)
```



Python for Data Science



Let see how to add a new array to the existing array along the column wise. So, first we will create an array and then we will change the shape to a column array. So, `np.array([21,22,23]).reshape()` because our earlier a array it was a 3 by 3. So, we wanted to add along the column wise then we need to have values 3 rows and 1 column right.

So, I am creating a new array and storing it in a variable call `col` when you print call. So, the array values will be 21, 22 and 23. So, we will add this array to the existing array call `a`, so `np.append` is a function. So, this is our existing array `a` comma the new array which we have created now comma axis is equal to 1. So, we have to so we can store it in a variable call `a_col`. So, this will be an array data type, till this we had the original array.



(Refer Slide Time: 15:06)

## Modifying array using append()

- Adding the new array to `a` as a column
- Create an array and reshape to column array

```
col=np.array([21,22,23]).reshape(3,1)
In [49]: print(col)
[[21]
 [22]
 [23]]
In [14]: print(a_col)
[[12 2 3 21]
 [ 4 5 6 22]
 [ 7 8 9 23]]
```

Python for Data Science





So, now, we have created this is a new array. So, it appends array at the last our output will be 3 rows and 4 columns.

(Refer Slide Time: 15:19)

## Modifying array using insert()

- `insert()` - adds values at a given position and axis in an array
- Syntax: `numpy.insert(array,obj,values,axis)`
  - `array` - input array
  - `obj` - index position
  - `values` - array of values to be inserted
  - `axis` - axis along which values should be insert

Python for Data Science



Insert it is adds the values at the given position and we can also specify the axis. If you give axis is equal to 0 it adds along the row wise. If it is specify axis is equal to 1, it add along the column wise the syntax is `numpy.insert`, we have to specify the input array object it is a index position in which you wanted to insert and the new values and then you have to specify the axis.

(Refer Slide Time: 15:47)

## Modifying array using insert( )


- Consider array `a`

```
In [16]: a
Out[16]:
array([[12, 2, 3],
       [ 4, 5, 6],
       [ 7, 8, 9]])
```

- Insert new array along row and at the 1<sup>st</sup> index position

```
a_ins=np.insert(a,1,[13,15,16],axis=0)
In [19]: print(a_ins)
[[12 2 3]
 [13 15 16]
 [ 4 5 6]
 [ 7 8 9]]
```

Python for Data Science




So, we will consider an array `a` which is our value is 12, 2, 3, 4, 5, 6, 7, 8 and 9. So, this is the 3 by 3 array. So, we will insert a new row, along the row wise at the first index position, earlier we had 0, 1 and 2. If you wanted to add an array at the first index position we will see how to do that. So, `np.insert` the existing array which is a comma 1.

So, we need to add that the index position one and the new values. So, which is 13, 15 and 16 along the row wise. So, axis is equal to 0 when you print `a_ins`. So, earlier we had 12, 2, 3 at the zeroth index position and the first index position we had 4, 5, 6. Now at the first index position we are adding a new array 13, 15 and 16. So, now, the index position will be changing. So, it will be 0, 1, 2 and 3.

(Refer Slide Time: 16:58)

## Modifying array using delete( )

- **delete()** - removes values at a given position and axis in an array
- Syntax: **numpy.delete(array, obj, axis)**
  - **array** - input array
  - **obj** - indicate array to be removed or it's position
  - **axis** - axis along which array should be removed



Python for Data Science

So, you can also delete an array using the delete function. So, it removes the values at a given position and the axis in the array. So, the syntax is `numpy.delete()`. So, the array name input array and then index position and then the axis.


(Refer Slide Time: 17:18)

## Modifying array using delete( )

- Delete third row from the existing array **a\_ins**

```
a_del=np.delete(a_ins,2,axis=0)
```

In [19]: <code>print(a_ins)</code>	→	In [21]: <code>print(a_del)</code>
<code>[[12 2 3]</code>		<code>[[12 2 3]</code>
<code>[13 15 16]</code>		<code>[13 15 16]</code>
<code>[ 4 5 6]</code>		<code>[ 7 8 9]</code>
<code>[ 7 8 9]</code>		



Python for Data Science

Let say if you wanted to delete third row from the existing array a underscore insert. So, the command is `np.delete a underscore ins`. So, delete third row. So, the corresponding index will be 2 index number and then axis is equal to 0. So, it corresponds for the row wise when you print a underscore ins. So, when you look at the left hand side s, this was

an old array. So, we are going to delete the third row. So, the updated array is the right hand side we had remove the third row which means 4, 5, 6 will be removed from the array.

(Refer Slide Time: 17:59)



The slide features a blue header with the word "Summary" in white. To the right of the header is a logo for "GITAA" (Geometric Information Technology Applications) with a small icon of a document. Below the header is a bulleted list of five items: "Reshape an array", "Numpy operations", "Accessing components", "Subset of arrays", and "Modifying array". In the bottom right corner, there is a small inset image of a woman with dark hair, wearing a blue and white patterned top. At the bottom left of the slide, the text "Python for Data Science" is visible.

Let summarize we saw how to reshape an array we also saw some of the numpy operations. So, sum is available in python. So, it calculates the overall sum we can also specify the axis is to calculate the row wise sum and as well as that column wise sum and we also saw the addition function. So, it calculates the element wise arrays. So, multiply and there are also various function. So, you can try those functions as well.

We also saw how to access the components of an array based on the index number we saw how to subset an array. So, sub setting an array we subset at 2 by 2 array and then we modified that array. Modifying the subset array will automatically update in the original array as well we also saw how to modify an array using the insert function, using the append function and using the as well as delete function.

Thank you.