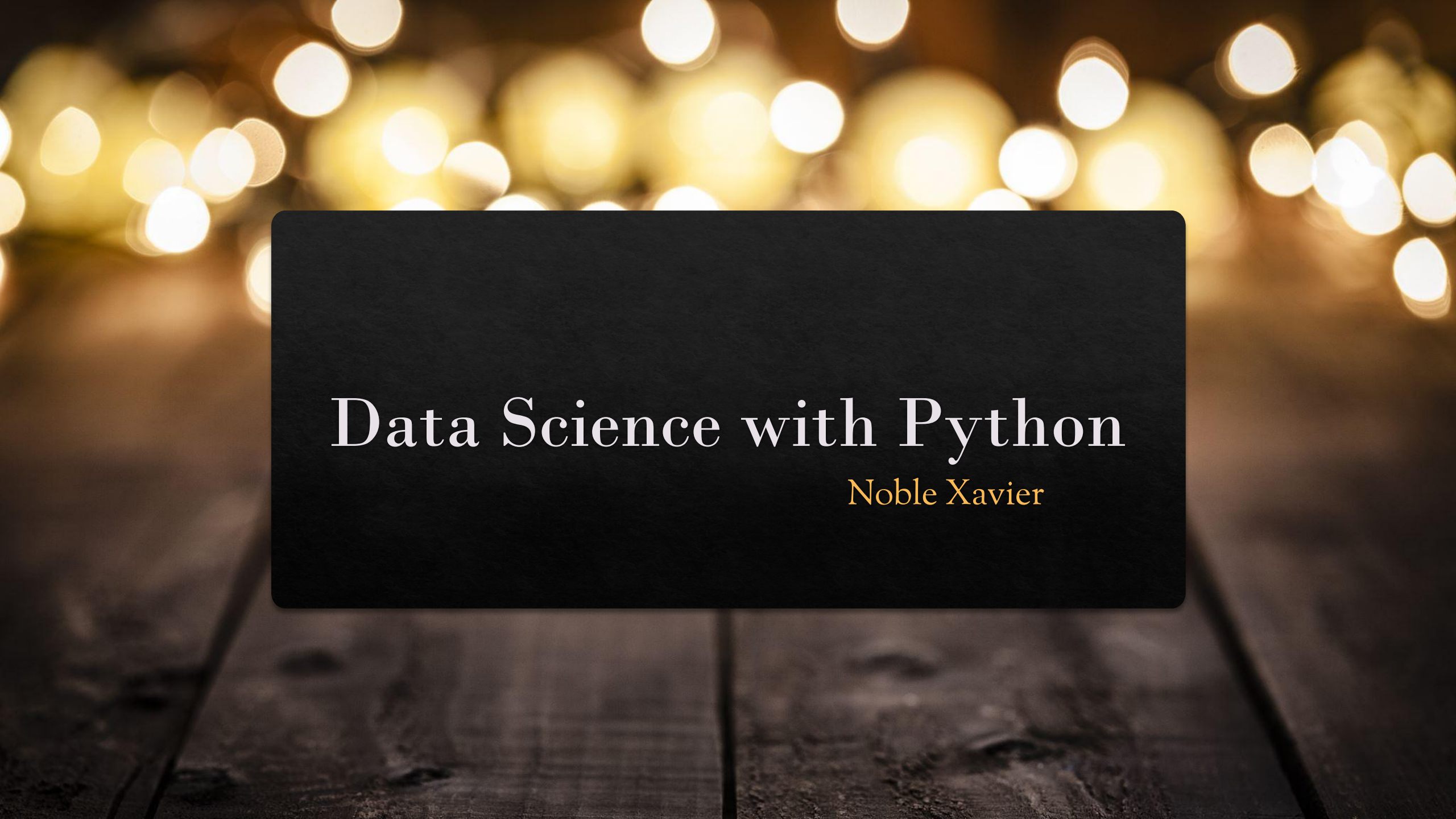
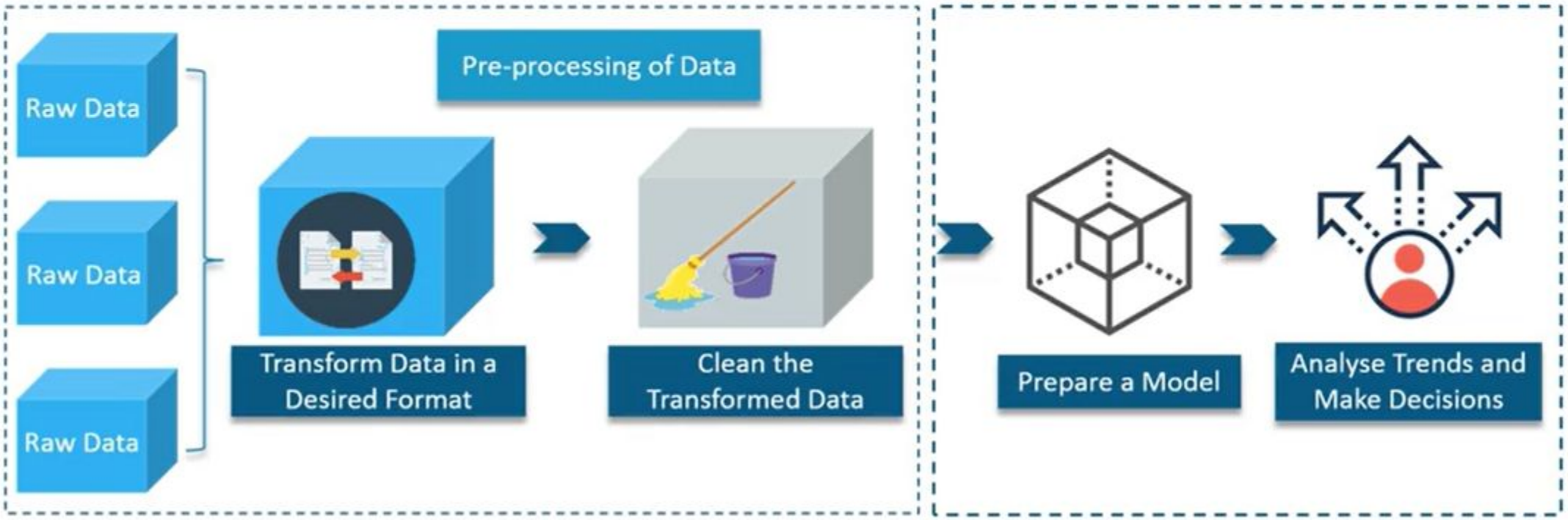

Data Science with Python
Noble Xavier

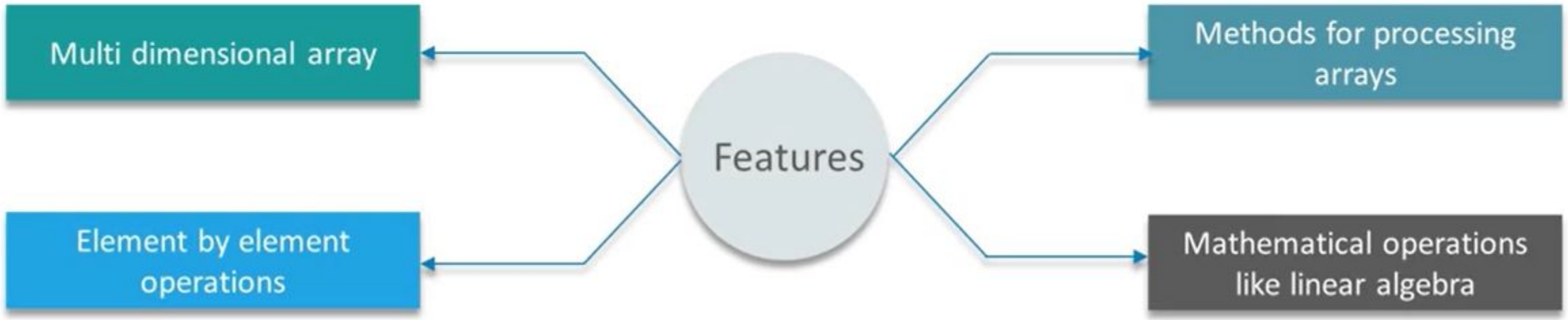
What is Data Analysis?

*Data Analysis is a process of inspecting, cleansing, transforming, and modeling **data** with the goal of discovering useful information, suggesting conclusions, and supporting decision-making.*

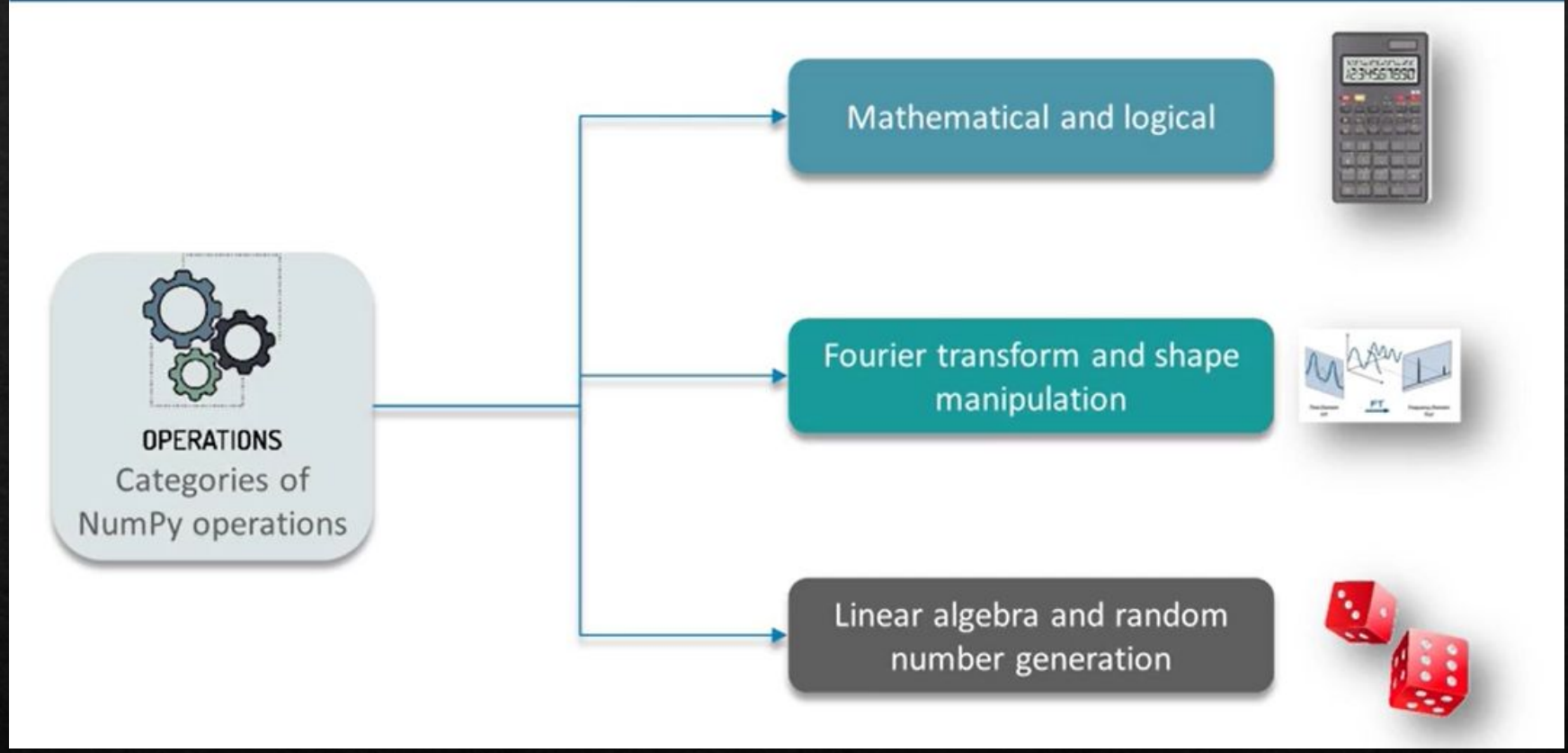


What is NumPy?

NumPy is a package for scientific computing

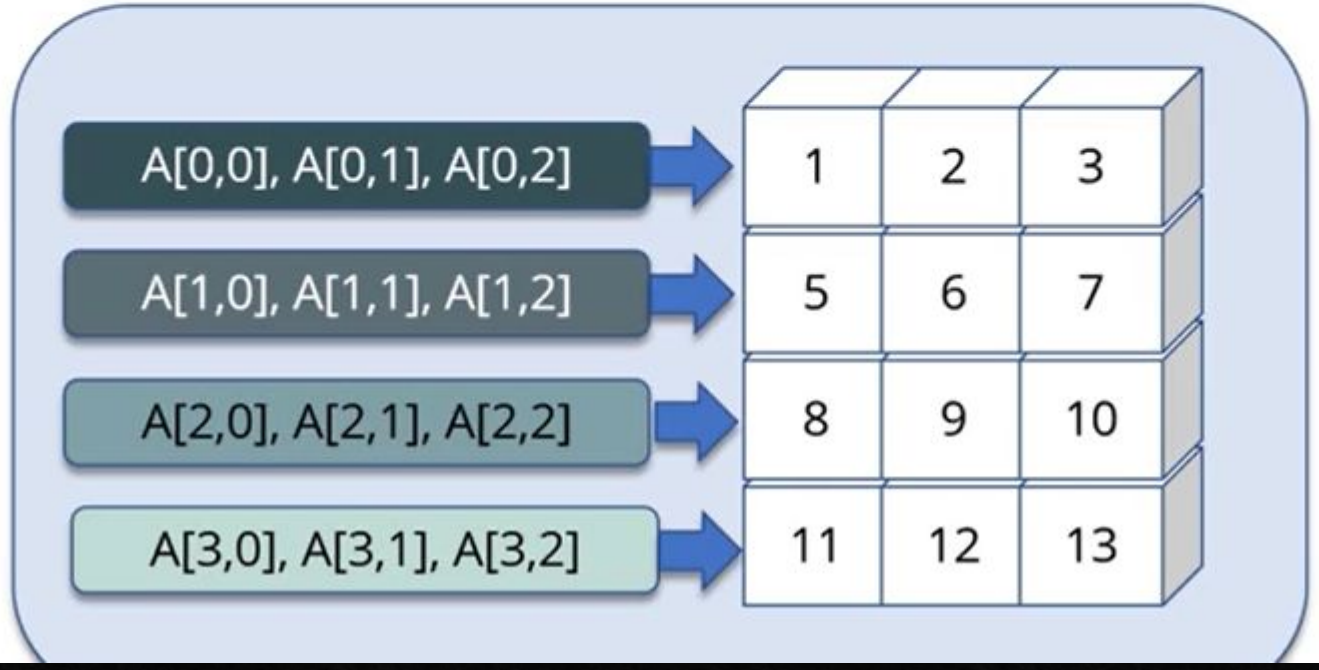


Operations in NumPy



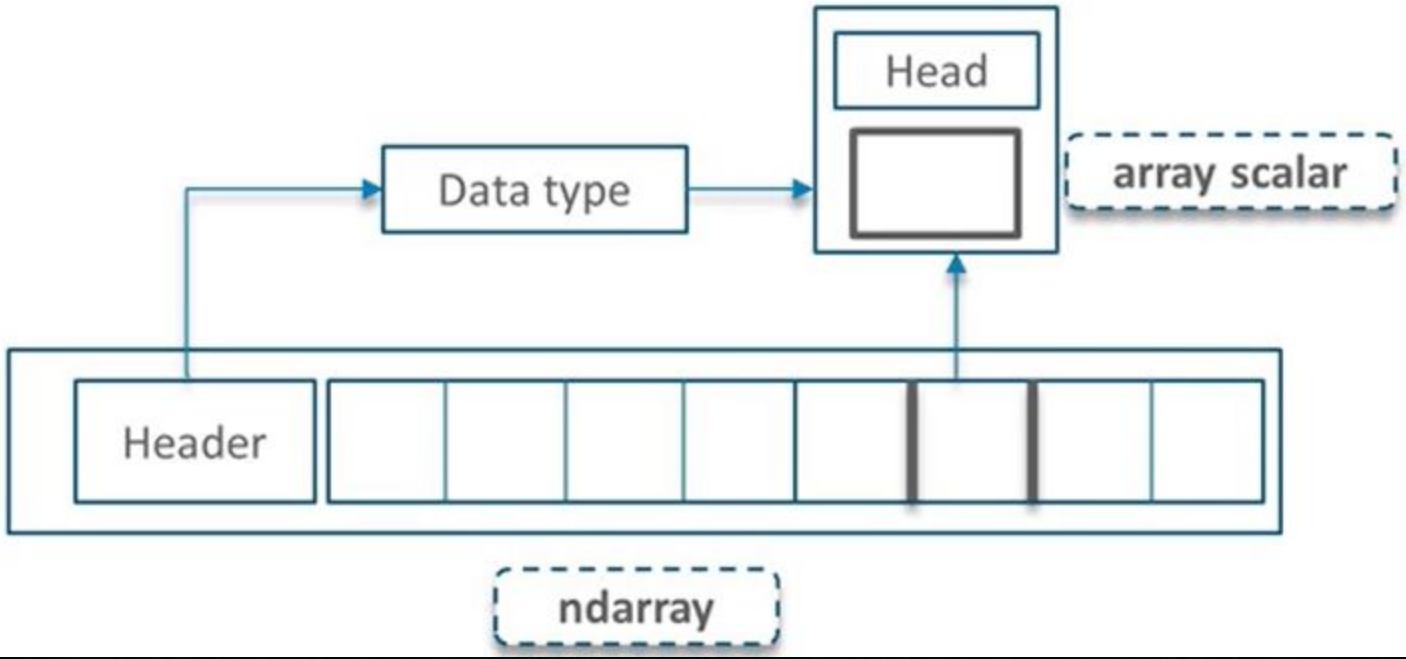
Ndarray – NumPy Array

The ndarray is a multi-dimensional array object consisting of two parts -- the actual data, some metadata which describes the stored data. They are indexed just like sequences are in Python, starting from 0



Ndarray – NumPy Array

- Each element in ndarray is an object of data-type object (called **dtype**)
- An item extracted from **ndarray**, is represented by a Python object of an array scalar type



DATA TYPES

Data Types	Description
bool_	Boolean (True or False) stored as a byte
int_	Default integer type (same as C long; normally either int64 or int32)
intc	Identical to C int (normally int32 or int64)
intp	Integer used for indexing (same as C ssize_t; normally either int32 or
int8	Byte (-128 to 127)
int16	Integer (-32768 to 32767)
int32	Integer (-2147483648 to 2147483647)
int64	Integer (-9223372036854775808 to 9223372036854775807)
uint8	Unsigned integer (0 to 255)
uint16	Unsigned integer (0 to 65535)
uint32	Unsigned integer (0 to 4294967295)
uint64	Unsigned integer (0 to 18446744073709551615)
float_	Shorthand for float64
float16	Half precision float: sign bit, 5 bits exponent, 10 bits mantissa
float32	Single precision float: sign bit, 8 bits exponent, 23 bits mantissa
float64	Double precision float: sign bit, 11 bits exponent, 52 bits mantissa
complex_	Shorthand for complex128
complex64	Complex number, represented by two 32-bit floats (real and imaginary
complex128	Complex number, represented by two 64-bit floats (real and imaginary

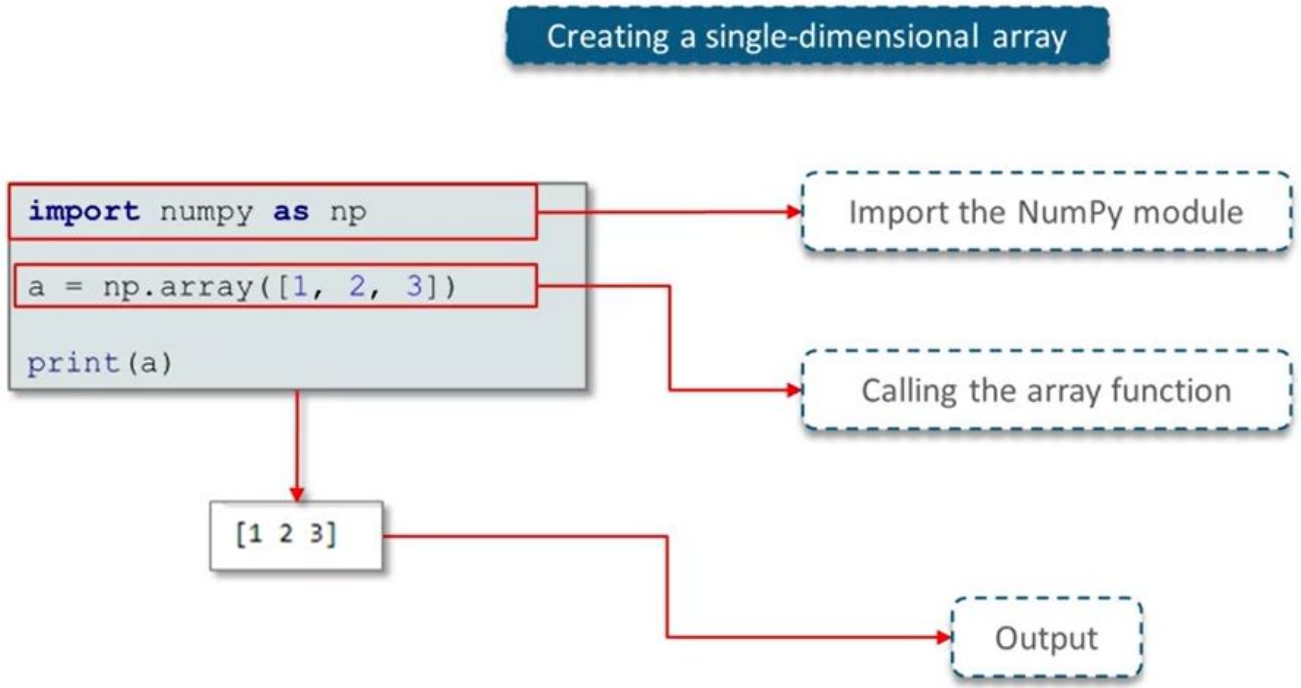
DATA TYPES

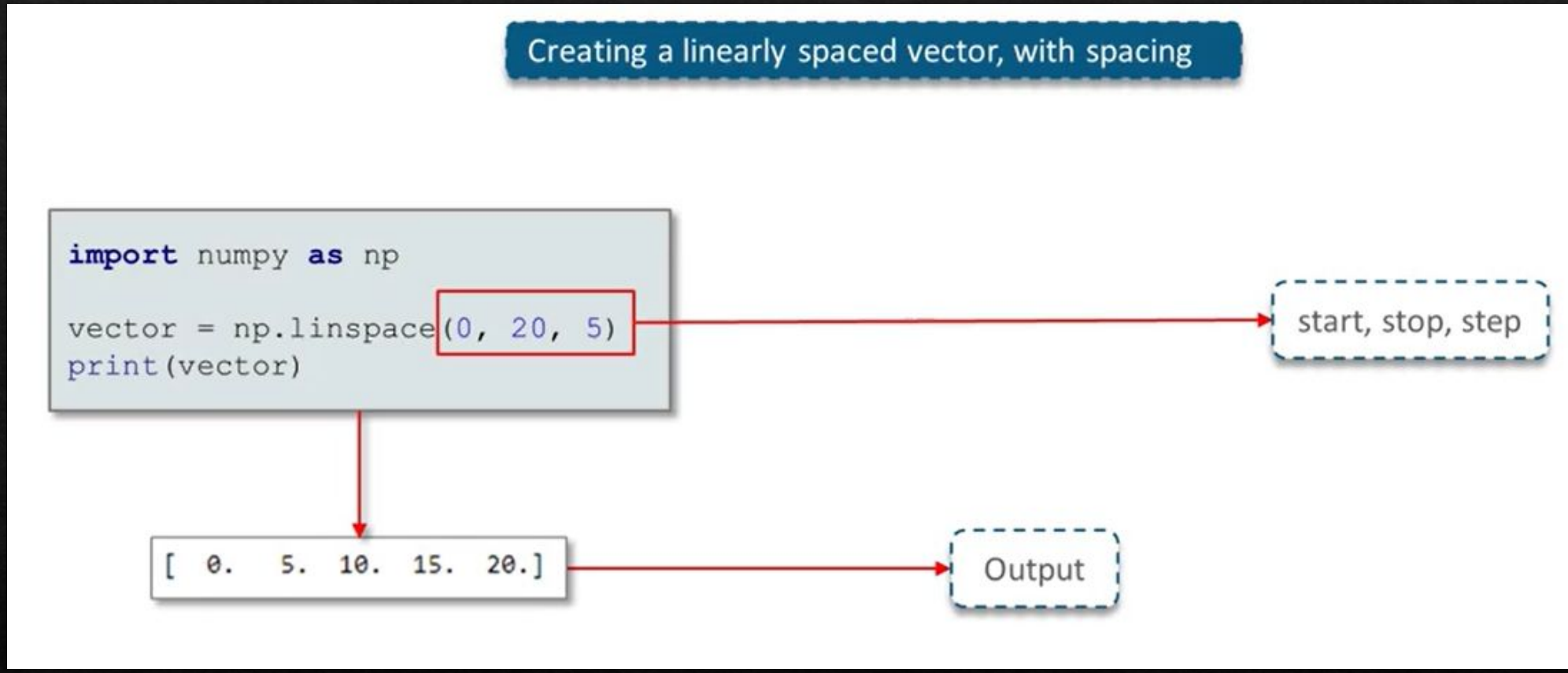
NumPy dtype	Numpy Cython type	C Cython type identifier
np.bool_	None	None
np.int_	cnp.int_t	long
np.intc	None	int
np.intp	cnp.intp_t	ssize_t
np.int8	cnp.int8_t	signed char
np.int16	cnp.int16_t	signed short
np.int32	cnp.int32_t	signed int
np.int64	cnp.int64_t	signed long long
np.uint8	cnp.uint8_t	unsigned char
np.uint16	cnp.uint16_t	unsigned short
np.uint32	cnp.uint32_t	unsigned int
np.uint64	cnp.uint64_t	unsigned long
np.float_	cnp.float64_t	double
np.float32	cnp.float32_t	float
np.float64	cnp.float64_t	double
np.complex_	cnp.complex128_t	double complex
np.complex64	cnp.complex64_t	float complex
np.complex128	cnp.complex128_t	double complex

Cython is a programming language that aims to be a superset of the Python programming language, designed to give C-like performance with code that is written mostly in Python with optional additional C-inspired syntax. Python is an interpreted programming language. Hence, Python programmers need interpreters to convert Python code into machine code. Whereas Cython is a compiled programming language.

The Cython programs can be executed directly by the CPU of the underlying computer without using any interpreter.

Creating a NumPy Array – Single-Dimensional Array




Creating a linearly spaced vector, with spacing
import numpy as np
vector = np.linspace(0, 20, 5)
print(vector)
start, stop, step
[0. 5. 10. 15. 20.]
Output

Converting a linear array of 8 elements into a 2x2x2 3D array

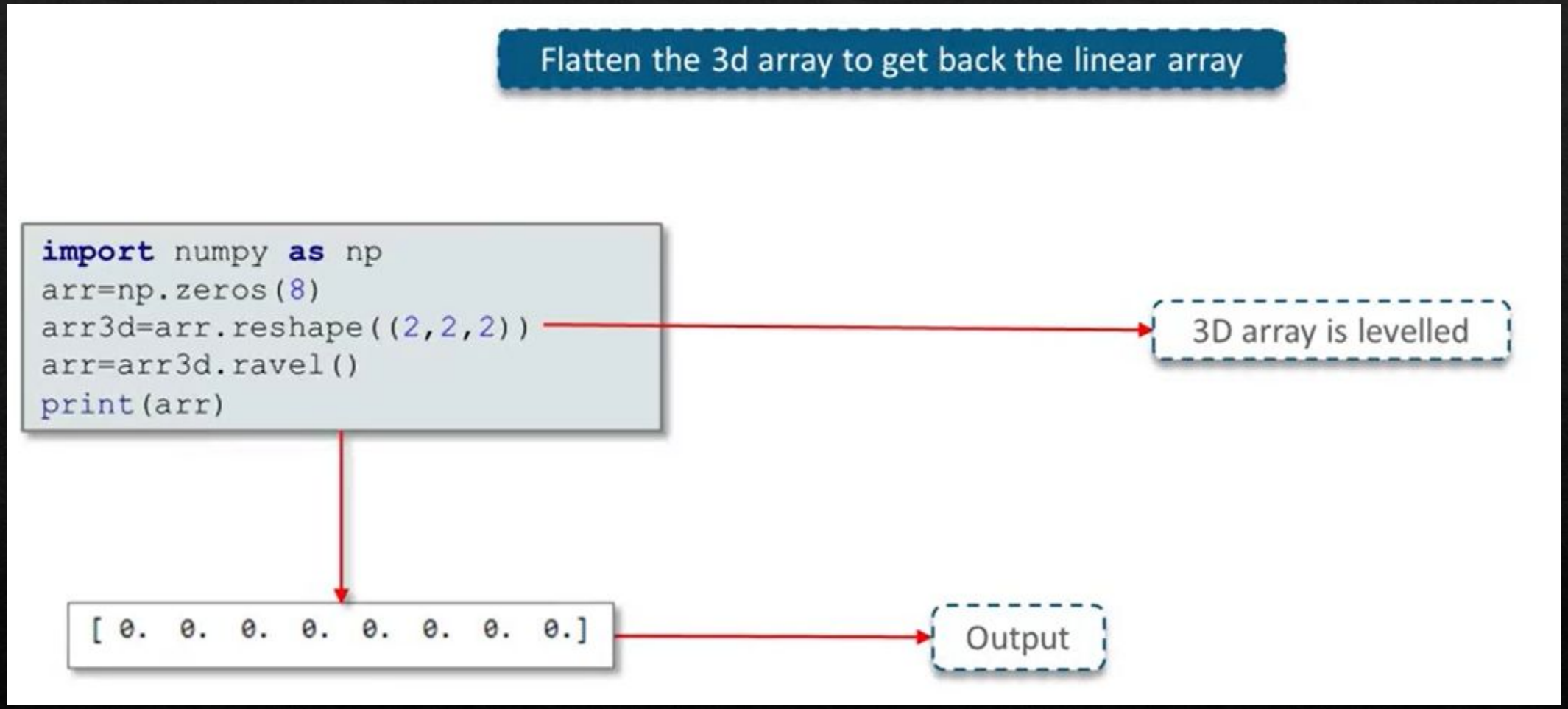
```
import numpy as np
arr=np.zeros(8)
arr3d=arr.reshape((2,2,2))
print(arr3d)
```

Flat array of eight zeroes

Restructured array

```
[[[ 0.  0.]
  [ 0.  0.]]

 [[ 0.  0.]
  [ 0.  0.]]]
```

Output


Flatten the 3d array to get back the linear array
import numpy as np
arr=np.zeros(8)
arr3d=arr.reshape((2,2,2))
arr=arr3d.ravel()
print(arr)
3D array is levelled
[0. 0. 0. 0. 0. 0. 0. 0.]
Output

Slicing items beginning with a specified index

```
import numpy as np
arr=np.arange(20)
print(arr[2:])
```

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19]

all elements starting from the index 2

Slicing items until a specified index

```
import numpy as np
arr=np.arange(20)
print(arr[:15])
```

[0 1 2 3 4 5 6 7 8 9 10 11 12 13 14]

all elements with index lesser than 15

Indexing and Slicing (Contd…)

Python's concept of lists slicing is extended to NumPy
The slice object is constructed by providing start, stop, and step parameters to slice()

```
import numpy as np
arr=np.arange(20)
arr_slice=slice(1,10,2)
element=arr[6]
print(arr[arr_slice])
```

Start, stop & step

```
[1 3 5 7 9]
```

Output

Indexing and Slicing (Contd...)

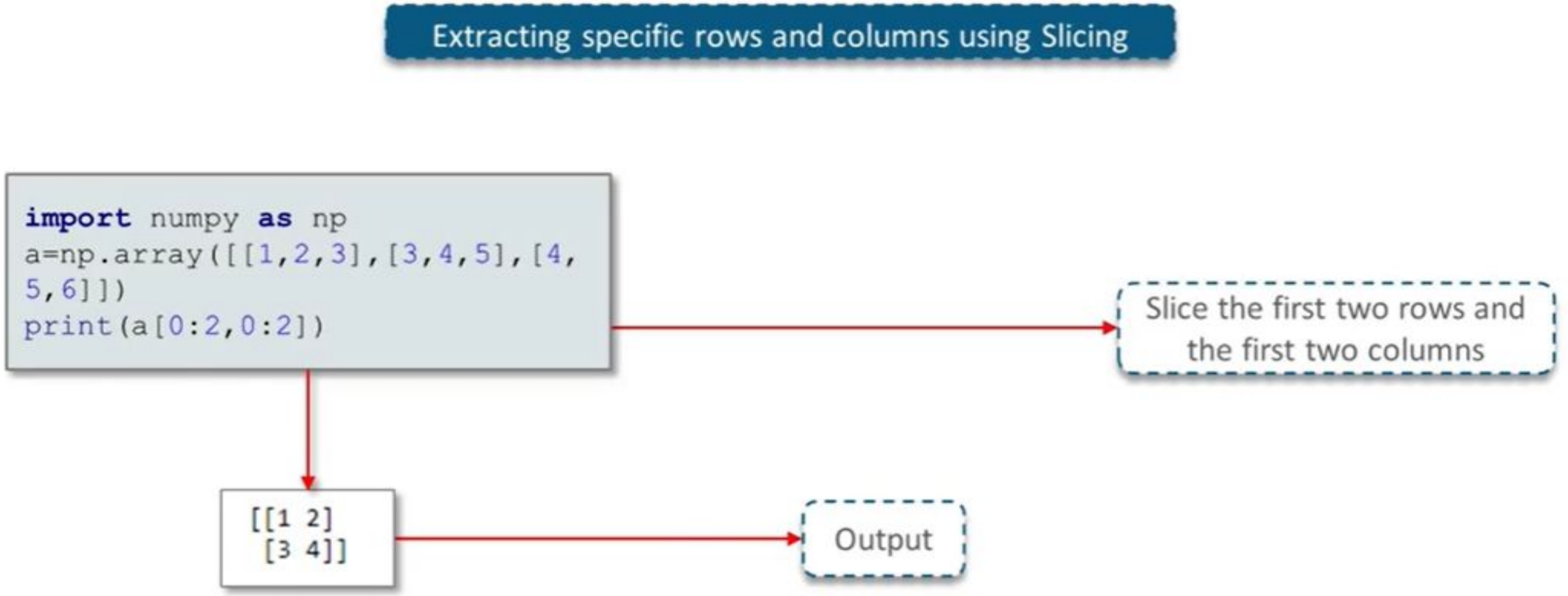
Slicing items beginning with a specified index

```
import numpy as np
arr=np.arange(20)
print(arr[2:])
```

Slicing items until a specified index

```
import numpy as np
arr=np.arange(20)
print(arr[:15])
```

Indexing and Slicing Multi-dimensional Arrays



NumPy Array Attributes

```
import numpy as np
a=np.array([[1,2,3],[3,4,5],[4,
5,6]])
print(a.shape)
print(a.ndim)
print(a.itemsize)
```

```
(3, 3)
2
8
```

Returns a tuple consisting of array dimensions

Attribute returns the number of array dimensions

Returns the length of each element of array in bytes

NumPy Array Creation Routines

- **numpy.empty** – creates an uninitialized array of specified shape and dtype

  It uses the following constructor – *numpy.empty(shape, dtype = float)*

  The constructor parameters are as follows – *shape, dtype*

```
import numpy as np
x = np.empty([3,2], dtype = int)
print(x)
```

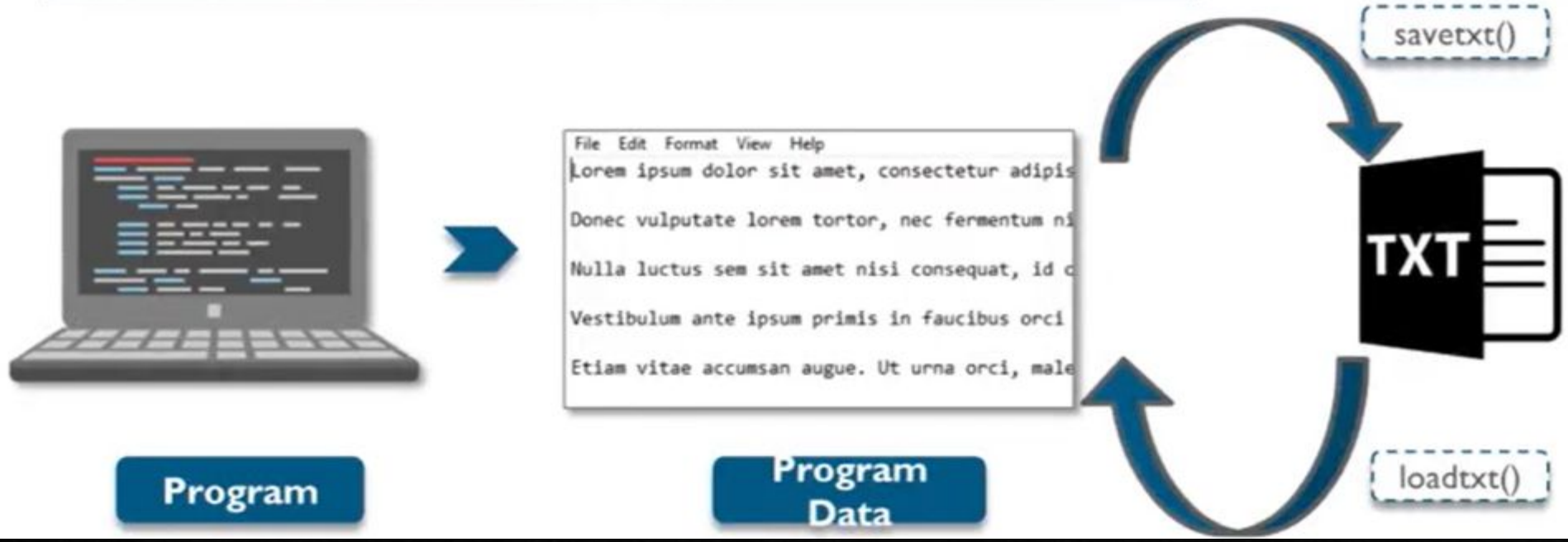
```
t_code
C:\Users\Maheshwar\AppData\Local\Programs\Python\Python36-32\py
[[0 0]
 [0 0]
 [0 0]]

Process finished with exit code 0
```

Output

Reading and Writing from Text Files

- NumPy provides the option of importing data from files directly into ndarray using the **loadtxt** function
- The **savetxt** function can be used to write data from an array into a text file

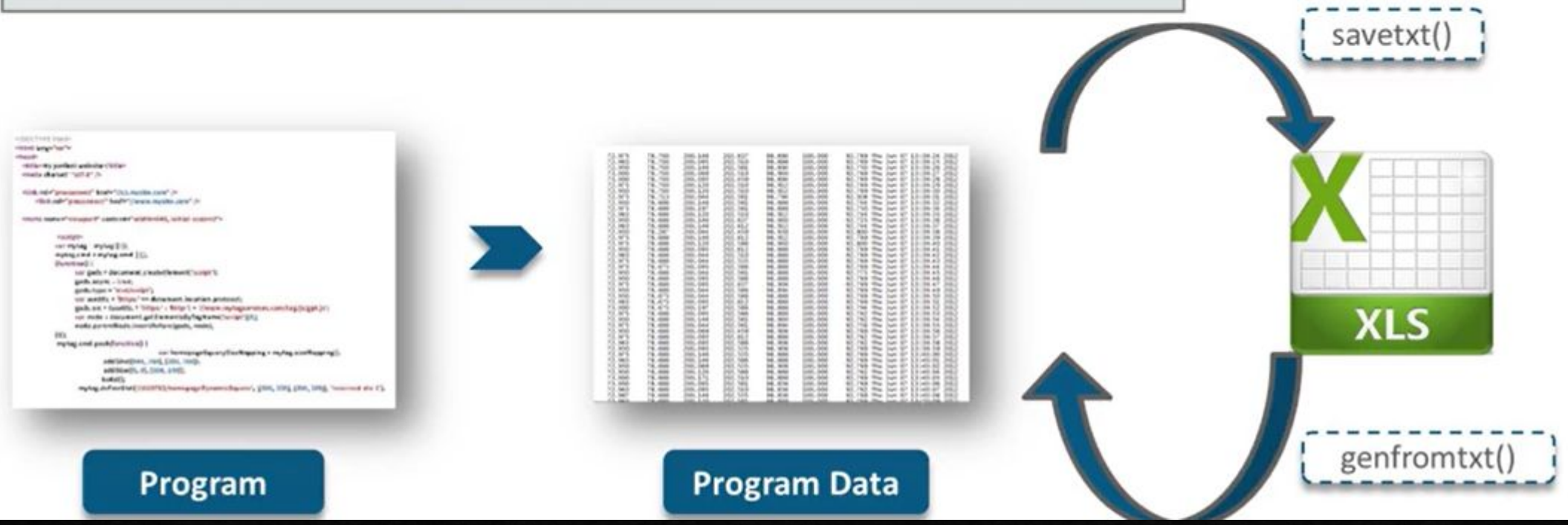
```
arr = np.loadtxt('filex.txt')
np.savetxt('newfilex.txt', arr)
```



Reading and Writing from CSV Files

- NumPy arrays can be dumped into CSV files using the **savetxt** function and the comma delimiter
- The **genfromtxt** function can be used to read data from a CSV file into a NumPy array

```
arr = genfromtxt('my_file.csv', delimiter=',')
np.savetxt('newfilex.csv', arr, delimiter = ",")
```



Numpy Summary

NumPy Basics

Operator	Description
np.array([1,2,3])	1d array
np.array([(1,2,3),(4,5,6)])	2d array
np.arange(start,stop,step)	range array

Placeholders

Operator	Description
np.linspace(0,2,9)	Add evenly spaced values btw interval to array of length
np.zeros((1,2))	Create and array filled with zeros
np.ones((1,2))	Creates an array filled with ones
np.random.random((5,5))	Creates random array
np.empty((2,2))	Creates an empty array

Numpy Summary

Array

Syntax	Description
array.shape	Dimensions (Rows,Columns)
len(array)	Length of Array
array.ndim	Number of Array Dimensions
array.dtype	Data Type
array.astype(type)	Converts to Data Type
type(array)	Type of Array

Copying/Sorting

Operators	Description
np.copy(array)	Creates copy of array
other = array.copy()	Creates deep copy of array
array.sort()	Sorts an array
array.sort(axis=0)	Sorts axis of array

Numpy Summary

Array Sort

```
arr = np.array([3, 2, 0, 1])
print(np.sort(arr))


arr = np.array(['banana', 'cherry', 'apple'])
print(np.sort(arr))
```

Sort 2 D Array

```
arr = np.array([[3, 2, 4], [5, 0, 1]])
print(np.sort(arr))
```

Sort Array with Axis

```
arr = np.array([[3, 2, 4], [5, 0, 1]])
print (arr)
print (np.sort(arr, axis = 0))
print (np.sort(arr, axis = 1))
```

Numpy Summary

The numpy.delete() function returns a new array with the deletion of sub-arrays along with the mentioned axis.

Array Delete 1 D array

```
import numpy as np
arr= np.arange(10,19)
print (arr)

# Delete based on Index
arr= np.delete (arr, 2)
print (arr)

arr= np.arange(10,19)
arr= np.delete (arr, [2,5])
print (arr)

arr= np.arange(10,19)
arr= np.delete (arr, [2,5,8])
print (arr)
```

Array Delete 2 D

Create 3 X 4 array

```
arr= np.arange(10,22)
arr2 = arr.reshape(3,4)
arr2
```

Delete 1st Row from array (0 for row 1 for column)

```
a= np.delete (arr2,1,0)
a
```

Delete 1st Column

```
a= np.delete (arr2,1,1)
a
```

Numpy Summary

Array Manipulation

Adding or Removing Elements

Operator	Description
np.append(a,b)	Append items to array
np.insert(array, 1, 2, axis)	Insert items into array at axis 0 or 1
np.resize((2,4))	Resize array to shape(2,4)
np.delete(array,1,axis)	Deletes items from array

Combining Arrays

Operator	Description
np.concatenate((a,b),axis=0)	Concatenates 2 arrays, adds to end
np.vstack((a,b))	Stack array row-wise
np.hstack((a,b))	Stack array column wise

Numpy Summary

Splitting Arrays

Operator	Description
numpy.split()	Split an array into multiple sub-arrays.
np.array_split(array, 3)	Split an array in sub-arrays of (nearly) identical size
numpy.hsplit(array, 3)	Split the array horizontally at 3rd index

More

Operator	Description
other = ndarray.flatten()	Flattens a 2d array to 1d
array = np.transpose(other) array.T	Transpose array
inverse = np.linalg.inv(matrix)	Inverse of a given matrix

Numpy Summary

Mathematics

Operations

Operator	Description
np.add(x,y) x + y	Addition
np.substract(x,y) x - y	Subtraction
np.divide(x,y) x / y	Division
np.multiply(x,y) x @ y	Multiplication
np.sqrt(x)	Square Root
np.sin(x)	Element-wise sine
np.cos(x)	Element-wise cosine
np.log(x)	Element-wise natural log
np.dot(x,y)	Dot product

Comparison

Operator	Description
==	Equal
!=	Not equal
<	Smaller than
>	Greater than
<=	Smaller than or equal
>=	Greater than or equal
np.array_equal(x,y)	Array-wise comparison

Basic Statistics

Operator	Description
np.mean(array)	Mean
np.median(array)	Median
array.corrcoef()	Correlation Coefficient
np.std(array)	Standard Deviation

Numpy Summary

Operator	Description
array.sum()	Array-wise sum
array.min()	Array-wise minimum value
array.max(axis=0)	Maximum value of specified axis
array.cumsum(axis=0)	Cumulative sum of specified axis

Slicing and Subsetting

Operator	Description
array[i]	1d array at index i
array[i,j]	2d array at index[i][j]
array[i<4]	Boolean Indexing, see Tricks
array[0:3]	Select items of index 0, 1 and 2
array[0:2,1]	Select items of rows 0 and 1 at column 1
array[:1]	Select items of row 0 (equals array[0:1, :])
array[1:2, :]	Select items of row 1
[comment]: <> (	array[1,...]
array[: :-1]	Reverses array