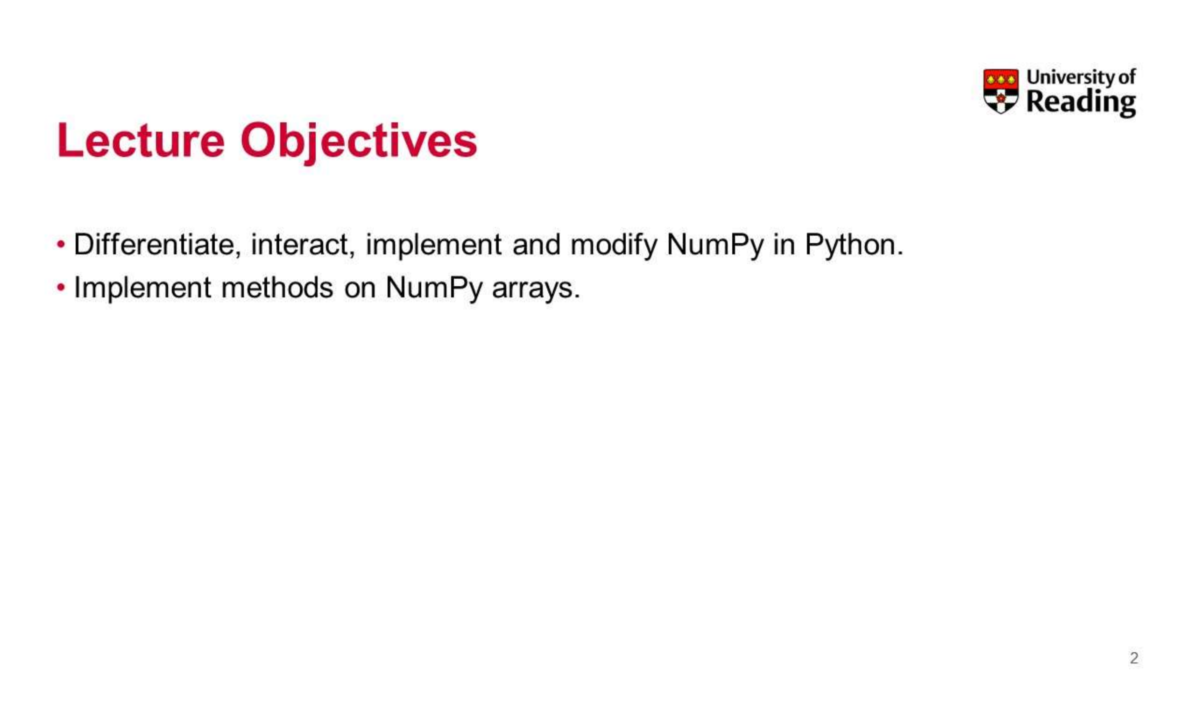
Department of Computer Science

CSMAD21 – Applied Data Science with Python

NumPy

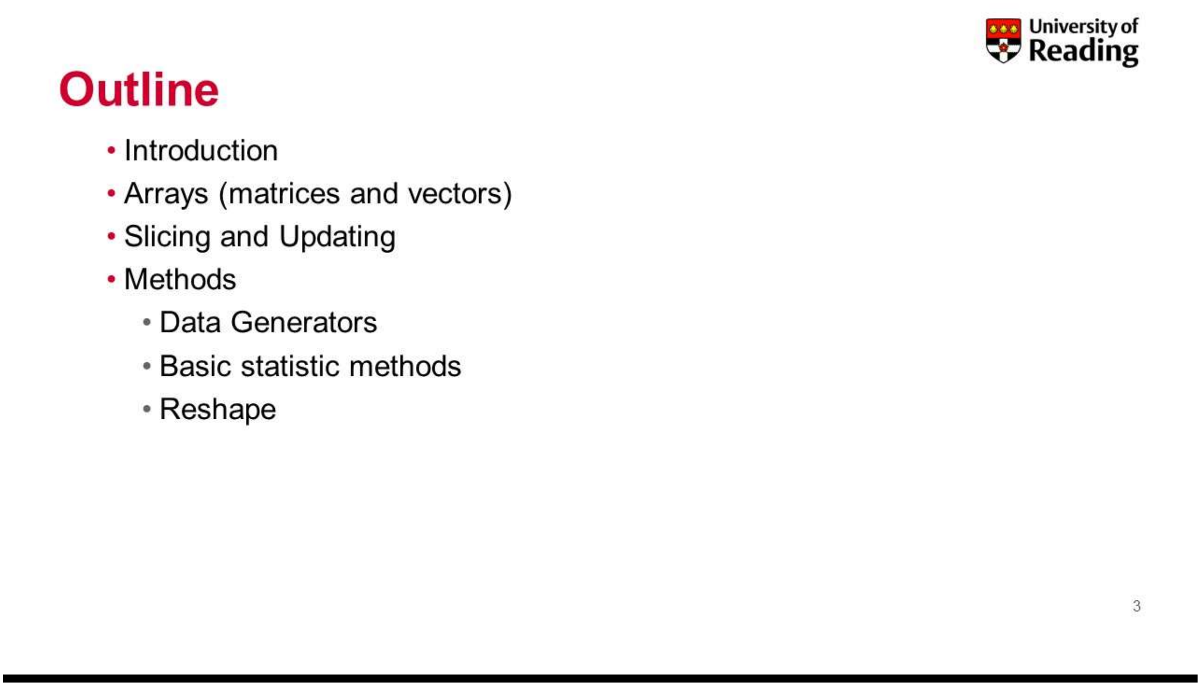
1



Lecture Objectives

- Differentiate, interact, implement and modify NumPy in Python.
- Implement methods on NumPy arrays.

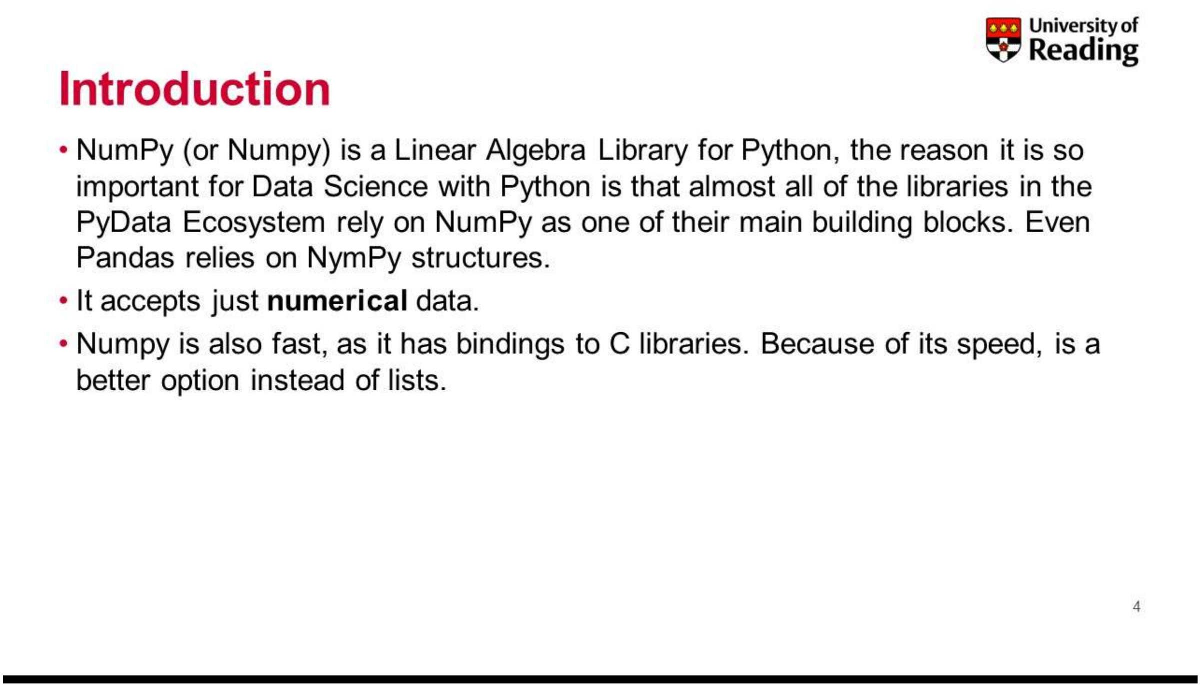
2



Outline

- Introduction
- Arrays (matrices and vectors)
- Slicing and Updating
- Methods
  - Data Generators
  - Basic statistic methods
  - Reshape

3



Introduction

- NumPy (or Numpy) is a Linear Algebra Library for Python, the reason it is so important for Data Science with Python is that almost all of the libraries in the PyData Ecosystem rely on NumPy as one of their main building blocks. Even Pandas relies on NymPy structures.
- It accepts just **numerical** data.
- Numpy is also fast, as it has bindings to C libraries. Because of its speed, is a better option instead of lists.

4

In [ ]:

```python
import numpy as np
```

Defining NumPy Arrays

Arrays can be created directly from lists by the command **np.array(list)**

In [ ]:

```python
lst1 = [1,2,3,4,5]
lst1
```

In [ ]:

```python
array1 = np.array(lst1)
array1
```

In [ ]:

```python
##A list of list
```

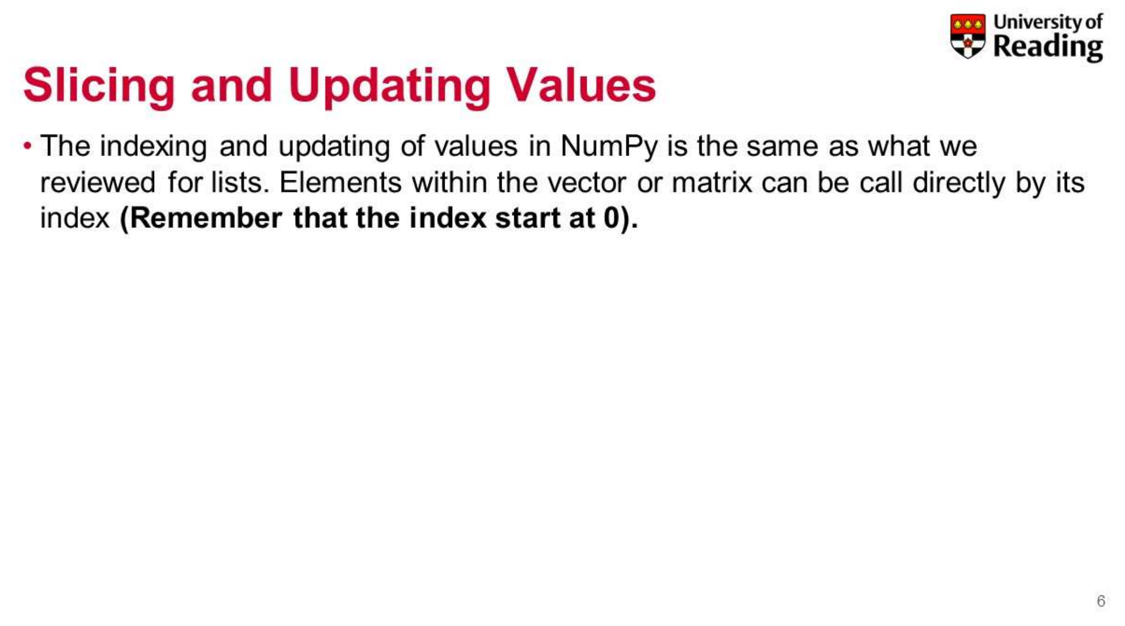
```
array2 = np.array([[1,2,3],[4,5,6],[7,8,9]])
array2
```

In [ ]:

```
array1.shape
```

In [ ]:

```
array2.shape
```



University of Reading

Slicing and Updating Values

- The indexing and updating of values in NumPy is the same as what we reviewed for lists. Elements within the vector or matrix can be call directly by its index **(Remember that the index start at 0).**

6

Slicing - Vectors

In [ ]:

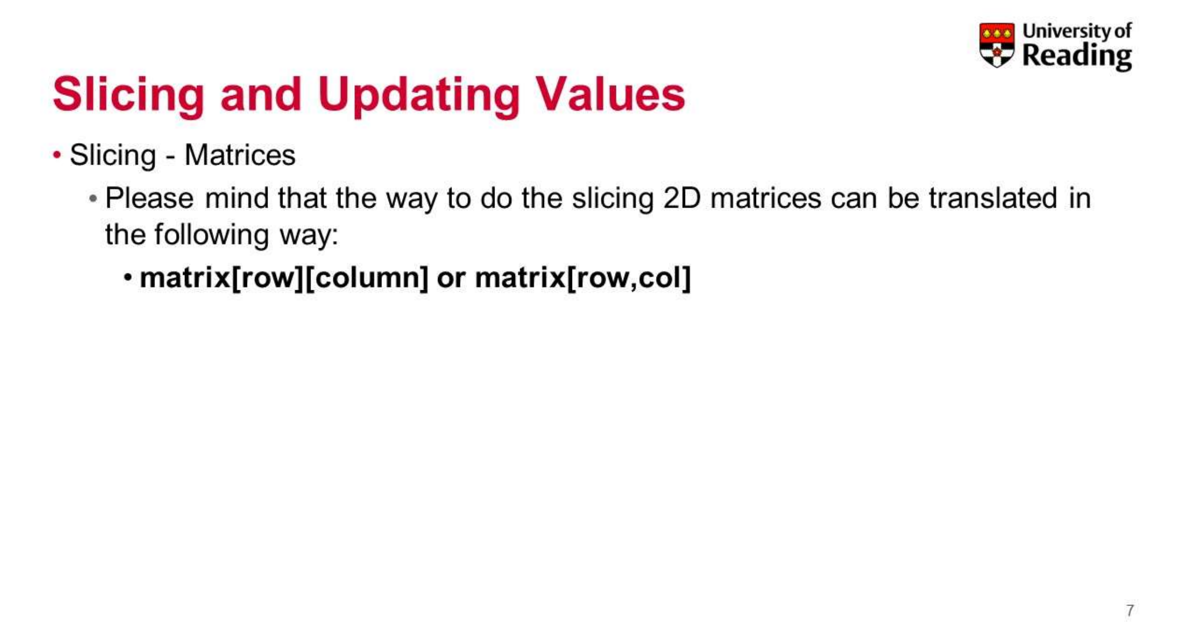
```
array1 = np.array([1,2,3,4,5])
array1
```

In [ ]:

```
array1[3]
```

In [ ]:

```
array1[0:3]
```



University of Reading

Slicing and Updating Values

- Slicing - Matrices
  - Please mind that the way to do the slicing 2D matrices can be translated in the following way:
    - **matrix[row][column] or matrix[row,col]**

7

In [ ]:

```
array2 = np.array([[1,2,3],[4,5,6],[7,8,9]])
array2
```

In [ ]:

```
#Lets see the shape of our array
array2.shape
```

In [ ]:

```
##Looking at the values of a particular row
array2[0]
```

In [ ]:

```
##Looking at the value of a particular row and column
array2[1,0]
```

In [ ]:

```
##Looking at the values of a particular column
array2[:,2]
```

Updating - Vectors

In [ ]:

```
array1 = np.array([1,2,3,4,5])
array1
```

In [ ]:

```
array1[0] = 50
array1
```

In [ ]:

```
array1[1:4] = 0
```

In [ ]:

```
array1
```

Updating Matrices

In [ ]:

```
array2 = np.array([[1,2,3],[4,5,6],[7,8,9]])
array2
```

In [ ]:

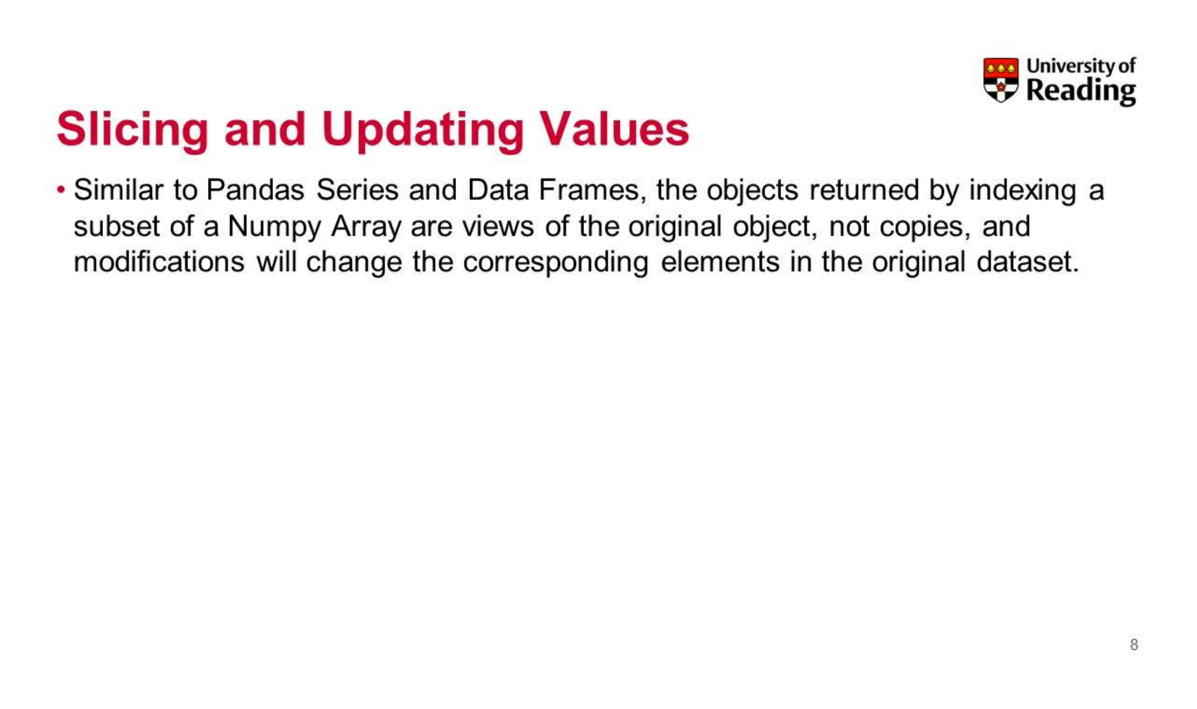
```
##Rows
array2[0] = 500
array2
```

In [ ]:

```
##Columns
array2[:,2] = 200
array2
```

In [ ]:

```
##Particular value
array2[1,0] = 100
array2
```



Slicing and Updating Values

- Similar to Pandas Series and Data Frames, the objects returned by indexing a subset of a Numpy Array are views of the original object, not copies, and modifications will change the corresponding elements in the original dataset.

8

In [ ]:

```
##Original array
array2 = np.array([[1,2,3],[4,5,6],[7,8,9]])
array2
```

In [ ]:

```
##Slice of the original array
array_tmp = array2[0:2,:]
array_tmp
```

In [ ]:

```
##We update the values of the temporal array
array_tmp[:,:] = 200
array_tmp
```

In [ ]:

```
##Because the temp array is a memory reference, the value of the
#original array is also updated
array2
```

- To avoid unwanted changes in the original arrays, we need to specify that a copy of the array is needed.

In [ ]:

```
##Original array
array2 = np.array([[1,2,3],[4,5,6],[7,8,9]])
array2
```

In [ ]:

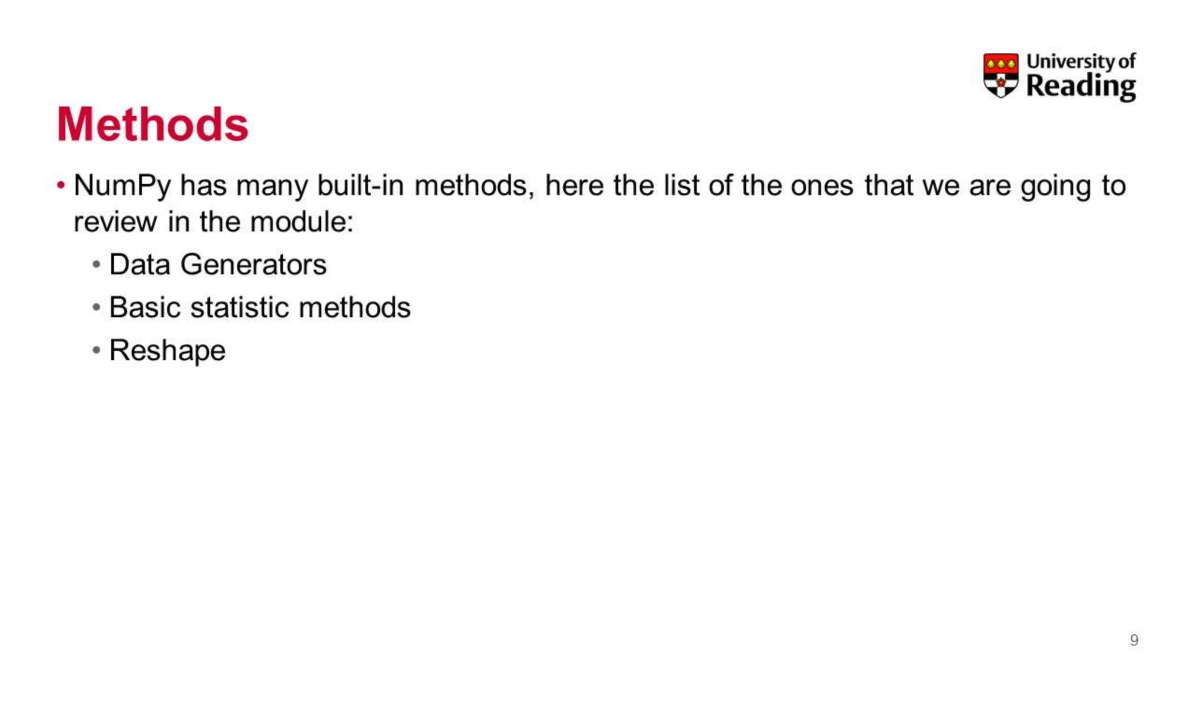
```
##Copy a slice of the original array
array_tmp = array2[0:2,:].copy()
array_tmp
```

In [ ]:

```
##We update the values of the temporal array
array_tmp[:,:] = 200
array_tmp
```

In [ ]:

```
##We validate that the original array is not affected.
array2
```

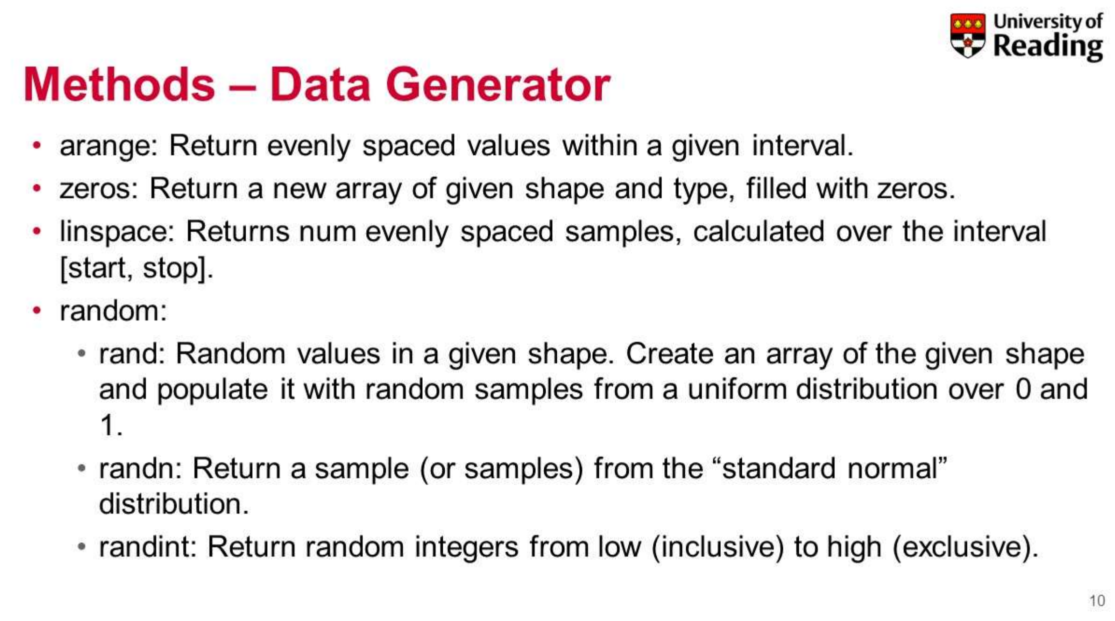


University of Reading

Methods

- NumPy has many built-in methods, here the list of the ones that we are going to review in the module:
  - Data Generators
  - Basic statistic methods
  - Reshape

9



Methods – Data Generator

- arange: Return evenly spaced values within a given interval.
- zeros: Return a new array of given shape and type, filled with zeros.
- linspace: Returns num evenly spaced samples, calculated over the interval [start, stop].
- random:
  - rand: Random values in a given shape. Create an array of the given shape and populate it with random samples from a uniform distribution over 0 and 1.
  - randn: Return a sample (or samples) from the “standard normal” distribution.
  - randint: Return random integers from low (inclusive) to high (exclusive).

10

Data Generators

arange

In [ ]:

```
array1 = np.arange(10,20)
array1
```

In [ ]:

```
array1 = np.arange(10,20,2.5)
array1
```

zeros

In [ ]:

```
##vector
array1 = np.zeros([5,5])
array1
```

linspace

In [ ]:

```
array1 = np.linspace(10,20,10)
array1
```

random.rand

In [ ]:

```
##Vector
array1 = np.random.rand(10)
array1
```

In [ ]:

```
##Matrix
array1 = np.random.rand(3,3)
array1
```

random.randint

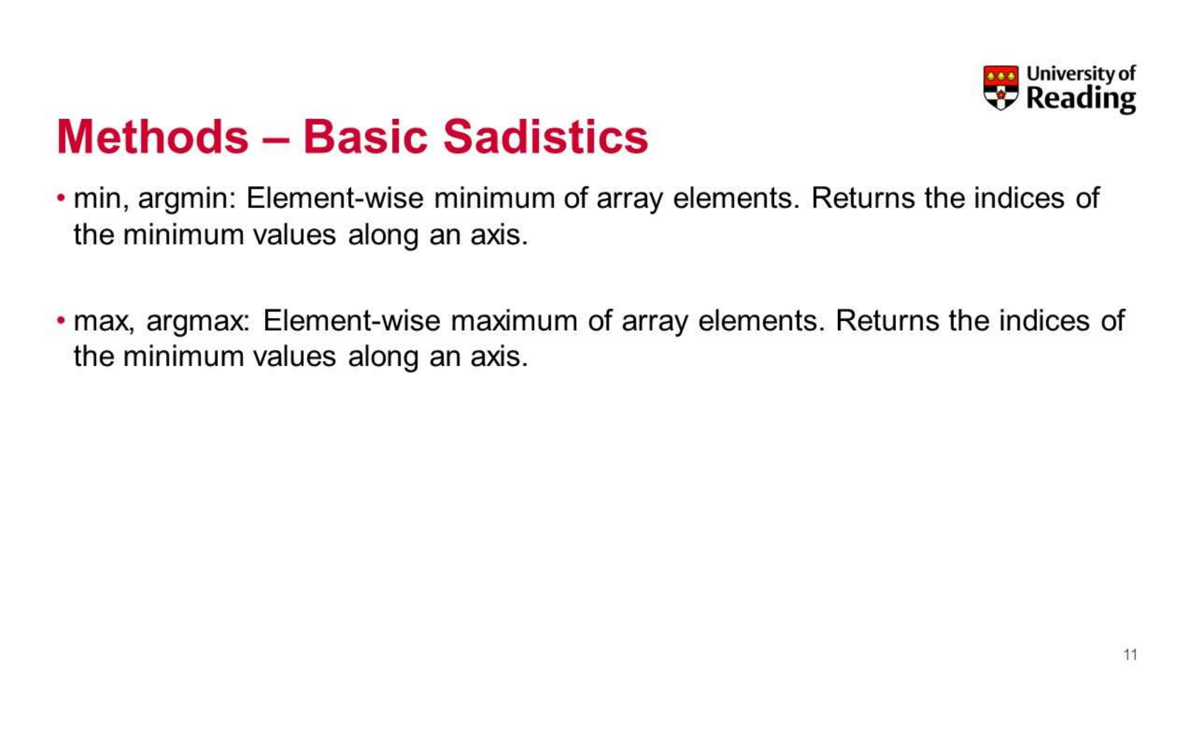
In [ ]:
```
array1 = np.random.randint(1,50)
array1
```

In [ ]:
```
array1 = np.random.randint(1,50,15)
array1
```

random.randn

In [ ]:
```
array1 = np.random.randn(20)
array1
```

In [ ]:
```
array1 = np.random.randn(5,5)
array1
```



University of Reading

Methods – Basic Sadistics

- min, argmin: Element-wise minimum of array elements. Returns the indices of the minimum values along an axis.
- max, argmax: Element-wise maximum of array elements. Returns the indices of the minimum values along an axis.

11

Basic statistic methods

min, max, argmin, argmax

In [ ]:
```
array1 = np.random.randint(1,50,15)
array1
```

In [ ]:
```
print('Min:', array1.min(), ', Max:', array1.max())
```

In [ ]:
```
print('Min Array Index Possition:', array1.argmin(), ', Max Array Index Possition:',
```

Reshape

In [ ]:
```
array1 = np.random.randint(1,50,15)
```

```
array1
```

In [ ]:

```
array1.shape
```

In [ ]:

```
array1.reshape(5,3)
```

In [ ]:

```

```

Data Selection

In [87]:

```
array2 = np.array([[1,2,3],[4,5,6],[7,8,9]])
array2
```

Out[87]:

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [88]:

```
array2>2
```

Out[88]:

```
array([[False, False,  True],
       [ True,  True,  True],
       [ True,  True,  True]])
```

In [89]:

```
array2[array2>2]
```

Out[89]:

```
array([3, 4, 5, 6, 7, 8, 9])
```

In [90]:

```
array2.mean()
```

Out[90]:

```
5.0
```

In [91]:

```
array2[array2 > array2.mean()]
```

Out[91]:

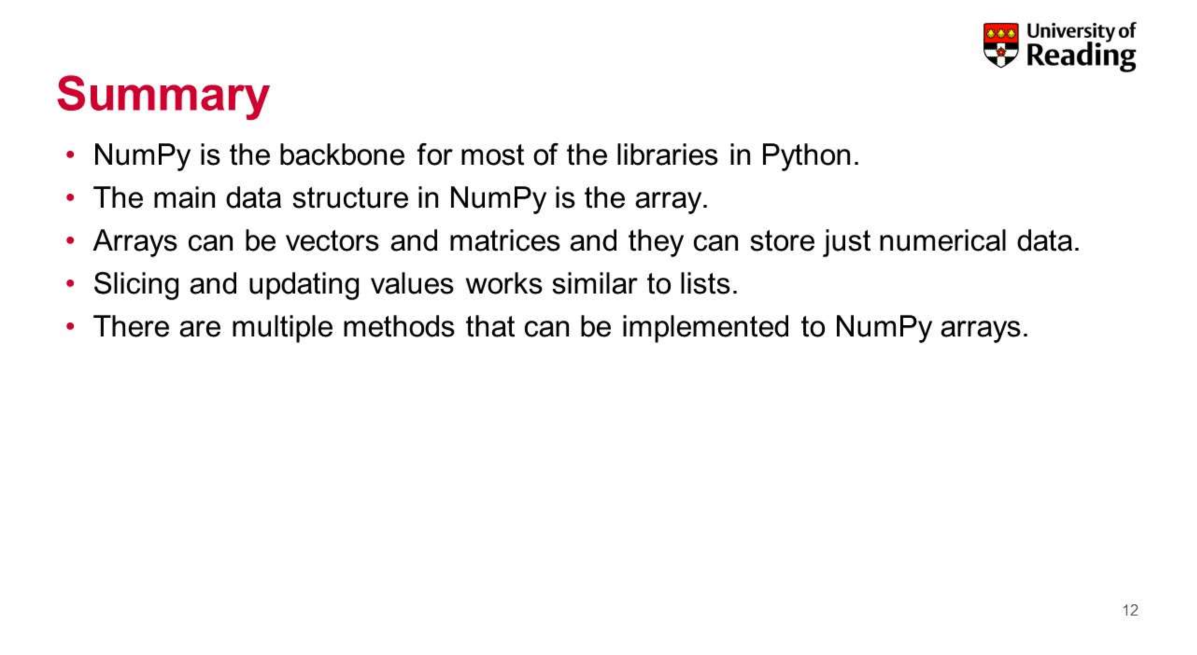
```
array([6, 7, 8, 9])
```

In [92]:

```
##values in column 0 above the mean of the array
array2[:,0][array2[:,0] > array2.mean()]
```

Out[92]:

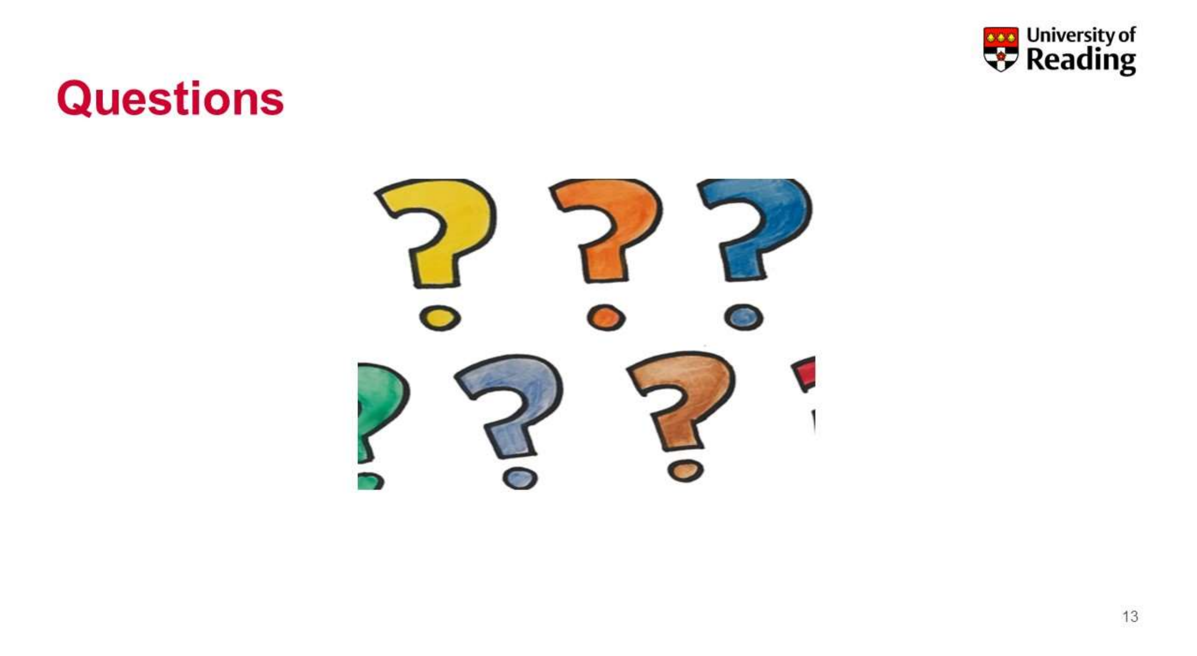
```
array([7])
```



Summary

- NumPy is the backbone for most of the libraries in Python.
- The main data structure in NumPy is the array.
- Arrays can be vectors and matrices and they can store just numerical data.
- Slicing and updating values works similar to lists.
- There are multiple methods that can be implemented to NumPy arrays.

12



Questions

13

References:

https://numpy.org/doc/stable/user/quickstart.html#basic-operations

```
In [ ]:
```