Lecture 14: NumPy

Swakkhar Shatabda

B.Sc. in Data Science
Department of Computer Science and Engineering
United International University

January 17, 2024

Numerical Python

- NumPy (Numerical Python) is an open source Python library that's used in almost every field of science and engineering.
- It provides ndarray, a homogeneous n-dimensional array object, with methods to efficiently operate on it.
- NumPy can be used to perform a wide variety of mathematical operations on arrays.
- It supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices.
- To access NumPy and its functions import it in your Python code like this:

```
import numpy as np
```

List and a NumPy array

- A Python list can contain different data types within a single list, all of the elements in a NumPy array should be homogeneous.
- NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently.
- Similar other data structures: Series, DataFrames and Tensors.
- The numpy module provides various functions for creating arrays. Here we use the array function, which receives as an argument an array or other collection of elements and returns a new array containing the argument's elements.

```
n1 = np.array([2, 3, 5, 7, 11])
n2 = np.array([[1, 2, 3], [4, 5, 6]])
```

Array Attributes

- An array object provides attributes that enable you to discover information about its structure and contents.

```
import numpy as np
n1 = np.array([2, 3, 5, 7, 11])
n2 = np.array([[2.1, 3.1, 5, 7, 11],
               [0.1, 3.1, 5.5, 7, 3.1]])
print("type(n1)",type(n1))
print("n1.dtype",n1.dtype)
print("n1.ndim:",n1.ndim)
print("n1.shape:",n1.shape)
print("type(n2)",type(n2))
print("n1.dtype",n2.dtype)
print("n2.ndim:",n2.ndim)
print("n2.shape:",n2.shape)
```

Filling arrays

- NumPy provides functions zeros, ones and full for creating arrays containing 0s, 1s or a specified value, respectively.

```
import numpy as np
a1 = np.zeros(5)
a2 = np.ones((2, 4), dtype=int)
a3 = np.full((3, 5), 13)
print(a1)
print(a2)
print(a3)
```

```
[0. 0. 0. 0. 0.]
[[1 1 1 1]
 [1 1 1 1]]
[[13 13 13 13 13]
 [13 13 13 13 13]
 [13 13 13 13 13]]
```

Creating arrays from Ranges

- Use NumPy's arange function to create integer ranges—similar to using built-in function range.
- Produce evenly spaced floating-point ranges with NumPy's linspace function. The function's first two arguments specify the starting and ending values in the range, and the ending value is included in the array.

```
a1 = np.arange(5)
a2 = np.arange(5, 10)
a3 = np.arange(10, 1, -2)
a4 = np.linspace(0.0, 1.0, num=5)
```

$\frac{H - L}{n - 1}$

L H ↪ 5

```
[0 1 2 3 4]
[5 6 7 8 9]
[10  8  6  4  2]
[0.   0.25 0.5  0.75 1.  ]
```

Reshaping an array

- You also can create an array from a range of elements, then use array method reshape to transform the one-dimensional array into a multidimensional array.

```
a1 = np.arange(1, 21).reshape(4, 5)
print(a1)
```

```
[[ 1  2  3  4  5]
 [ 6  7  8  9 10]
 [11 12 13 14 15]
 [16 17 18 19 20]]
```

arange(2, 21, 2).
reshape(2,5)

Array Operators

- Element-wise operations are applied to every element.

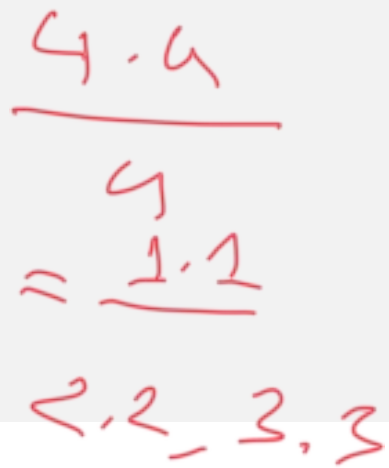
```
import numpy as np
a = np.arange(1, 6)
print(a)
print(a*2)
print(a**3)
print(a)
```

```
[1 2 3 4 5]
[ 2  4  6  8 10]
[  1   8  27  64 125]
[1 2 3 4 5]
```

Arithmetic Operations Between arrays

- You may perform arithmetic operations and augmented assignments between arrays of the same shape.

```
import numpy as np
a1 = np.arange(1, 6)
a2 = np.linspace(1.1, 5.5, 5)
print(a1)
print(a2)
print(a1*a2)
```



```
[1 2 3 4 5]
[1.1 2.2 3.3 4.4 5.5]
[ 1.1  4.4  9.9 17.6 27.5]
```

Comparing arrays

- You can compare arrays with individual values and with other arrays. Comparisons are performed element-wise.

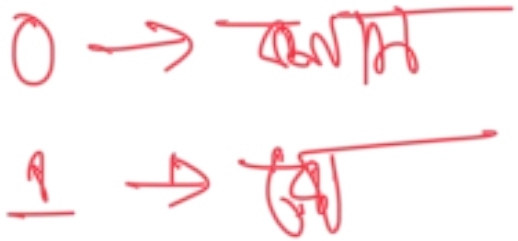
```
import numpy as np
a1 = np.arange(1, 6)
a2 = np.linspace(0, 6, 5)
print(a1)
print(a1>3)
print(a2)
print(a1>a2)
```

```
[1 2 3 4 5]
[False False False  True  True]
[0.  1.5 3.  4.5 6. ]
[ True  True False False False]
```

NumPy Calculation Methods

- An array has various methods that perform calculations using its contents. By default, these methods ignore the array's shape and use all the elements in the calculations.
- Many calculation methods can be performed on specific array dimensions, known as the array's axes.

```
import numpy as np
grades = np.array([[87, 96, 70], [100, 87, 90],
                   [94, 77, 90], [100, 81, 82]])
print(grades.sum())
print(grades.sum(axis=0))
```



Indexing and Slicing Two-Dimensional arrays

- To select an element in a two-dimensional array, specify a tuple containing the element's row and column indices in square brackets.
- To select a single row, specify only one index in square brackets.
- You can select subsets of the columns by providing a tuple specifying the row(s) and column(s) to select. Each can be a specific index, a slice or a list.
- Slices create views. The array method copy returns a new array object with a deep copy of the original array object's data.

```
import numpy as np
grades = np.array([[87, 96, 70],
                   [100, 87, 90],
                   [94, 77, 90],
                   [100, 81, 82]])
print(grades[1,[1,2]])
```

Random Numbers

- The randrange function generates an integer from the first argument value up to, but not including, the second argument value.

```
import random
print(random.randrange(11,20))
print([random.randrange(11,20) for i in range(10)])
```

Your first recursion

- Recursive function is a function that calls itself.

```
def factorial(n):
    if n==1:
        return 1
    else:
        return factorial(n-1)*n

print(factorial(5))
```