NumPy Basics

25.09.2019

Python Notes

Intro

- Numpy is the most basic and a powerful package for working with data in python.
- NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
- At the core, **numpy provides excellent n-dimensional arrays**.
- Well, there are very significant advantages of using numpy arrays overs lists.

Lists and Arrays

- **Similarities:**
  - Both are used for storing data
  - Both are mutable
  - Both can be indexed and iterated through
  - Both can be sliced
- **Differences:**
  - Arrays support vectorised operations, while lists don’t.

- Let’s suppose you want to add the number 2 to every item in the list. That was not possible with a list. But you can do that on a ndarray.
  - `list1 + 2 # error`
  - `array1 + 2 #  adds 2 to each element`
- A numpy array must have all items to be of the same data type, unlike lists.
- Once an array is created, you cannot change its size. You will have to create a new array or overwrite the existing one.
- An equivalent numpy array occupies much less space than a python list of lists.

Create a NumPy Array

- One of the most common ways is to create one from a list or a list like an object by passing it to the `np.array` function.

`Example: Create a 1d array from a list`

```
import numpy as np

list1 = [0,1,2,3,4]
arr1 = np.array(list1)
```

`Example: Create a 2d array from a list of lists`

```
list2 = [[0,1,2],
         [3,4,5],
         [6,7,8]]


arr2 = np.array(list2)

# Get the array. It will be a 3x3 matrix.
arr2      array([[0,1,2], [3,4,5], [6,7,8]])
```

`Example: Create zeros and ones matrixes`

```
zerosMatrix = np.zeros((3,4))

onesMatrix =  np.ones((3,4))
```

`Example: Create matrixes via Arange and Linspace`

```
vect1 = np.arange(10,50,5)      array([10, 15, 20, 25, 30, 35, 40, 45])

vect2 = np.linspace(0,10,5)       array([ 0. ,  2.5,  5. ,  7.5, 10. ])
```

Change Array Elements

- Assign a value to the element at ith row and jth column by: matrixName[i , j] = value.
- Note that i, j = 0,1,2,3...

Example: Assign a value to the element at ith row and jth column

```
zerosMatrix = np.zeros((3,4))

zerosMatrix[0,0] = 5
```

Array Size, Stored Datatype, Type

Example:

```
arr1 = np.array([0,1,2,3,4])

# Get the array, its size, its stored data type, its type
arr1               array([0, 1, 2, 3, 4])
arr1.size          5
arr1.dtype        'int32'
type(arr1)         numpy.ndarray
```

Shape for Row and Column Vectors

Example:

```
columnVector1 = np.array([1,2,3,4,5])
columnVector1.shape   (5, )

columnVector2 = np.array([[1],[2],[3],[4],[5]])
columnVector2.shape   (5,1)
```

```
rowVector1 = np.array([[1,2,3,4,5]])
rowVector1.shape      (1,5)


columnVector1[0]       returns 0th row: 1
columnVector1[0,0]     ERROR: There is no column


columnVector2[0]       returns 0th row: array([1])
columnVector2[0,0]     returns 0th row, 0th column: 1


rowVector1[0]          returns 0th row: array([1,2,3,4,5])
rowVector1[0,0]        returns 0th row, 0th column: 1
```

Array Datatypes

- You may also specify the datatype by setting the **`dtype`** argument.
- Some of the most commonly used numpy dtypes are: `'float'`, `'int'`, `'bool'`, `'str'` and `'object'`.
- To control the memory allocations you may choose to use one of 'float32', 'float64', 'int8', 'int16' or 'int32'.

Example: Create a float 2d array

```
arr2_f = np.array(list2, dtype='float')
```

- You can also convert it to a different datatype using the **`astype`** method.

Example: Convert float array to a int array

```
arr2_f.astype('int')
```

- If you are uncertain about what datatype your array will hold or if you want to hold characters and numbers in the same array, you can set the `dtype` as `'object'`.

Example: Create an object array to hold numbers as well as strings

```
arr3_obj = np.array([1,'A','333',45,'Dance'], dtype ='object')
```

- You can always convert an array back to a python list using `tolist()`.

Example: Convert an array back to a list

```
arr3_obj.tolist()
```

Extract Specific Items from Array

- You can extract specific portions on an array using indexing starting with 0.

Example:

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


#Extract the first 2 rows and columns
arr[0:2,0:2] # result will be a 2x2 array
arr[:2,:2]   # alternative way
```

Example: Reaching the last row or column

```
arr[-1,:] # last row
arr[:,-1] # last column
```

Reversing Rows and Columns

Example: Reverse row and column positions

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

arr[::-1,] # last row becomes the first and so on...
arr[::-1,::-1] # reverse both row and column positions
```

Shape Manipulation

- Flatten and Ravel : Ravel method changes the parent array!
- Reshape and Resize: Resize method changes the parent array!

Example: Flatten and Ravel - Reshape and Resize

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

flattenArray = arrray.flatten()   array([1, 2, 3, 4, 5, 6, 7, 8, 9])
flattenArray = array.ravel()       array([1, 2, 3, 4, 5, 6, 7, 8, 9])

reshapedArray = flattenArray.reshape(3,3)
resizedArray = flattenArray.reshape(3,3)
```

Basic Numeric Operations

Example: Elementwise Operations

```
a = np.array([[1,2,3]])
b = np.array([[4,5,6]])

a+b  # elementwise addition
a-b  # elementwise subtraction
a*b  # elementwise multiplication
a**2 # a.^2

np.sqrt(a)   # a.^(1/2)
np.square(a) # a.^2
np.sin(a)    # gets sinus for each element
a<2          # checks and returns true or false for each element
```

Example: Matrix Operations

```
m1 = np.array([[1,2,3],
              [4,5,6]]) #2x3

m2 = np.array([[1,2,3],
               [4,5,6],
               [7,8,9]]) #3x3

m1.dot(m2)  # matrix product
m1.T        # matrix transpose
```

Example: Creating a random 5x10 matrix. Elements have values b/w 0-1

```
randomMatrix = np.random.random((5,10))

#sum all elements
randomMatrix.sum()

#sum columns. Result is a 1x10 vector. Each element represents the sum of related columns
randomMatrix.sum(axis=0)

#sum rows. Result is a 5x1 vector. Each element represents the sum of related rows
randomMatrix.sum(axis=1)


#Get mean, max and min values
randomMatrix.mean()
randomMatrix.max()
randomMatrix.min()

#Row wise and Column wise min
np.amin(randomMatrix,axis=0)
np.amin(randomMatrix,axis=1)
```

Stacking Arrays

Example:

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

array2 = np.array([[-1,-2],
                   [-3,-4]])


#vertical stack
array3 = np.vstack((array1,array2))

#horizontal stack
array4 = np.hstack((array1,array2))
```

Convert and Copy Array

Example:

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


#Convert arrays to lists
array.tolist()

#Shallow Copy. Both array2 and array point the same object.
array2 = array


#Deep Copy. array and array3 are seperate objects.
array3 = array.copy()
```

More blogs