Courses @ GeeksforGeeks

Internships @ GeeksforGeeks

Coding Practice

How to write an Interview Experience?

Must Do Coding Questions Company-wise

Must Do Coding Questions Topic-wise

Basic

Easy

Medium

Hard

Expert

Step by Step Preparation

Company Preparation

Top Topics

Company Specific Practice

Software Design Patterns

Placements Preparation Course

Python Numpy

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python.
Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data.

DBMS

Computer Networks

Engineering Mathematics

Design Patterns

Common Interview Puzzles

Web Technology

G-Facts

Computer Graphics

Image Processing

Project Ideas

ENJOY THE BIG BARGAIN OF GEARBEST.
UP TO 50% OFF
on Gearbest orders
+
₹ 250 CASHBACK*
when you pay with PayPal for $ 20 or more
13th May - 30th June 2019
Shop Now

Arrays in Numpy

Array in Numpy is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In Numpy, number of dimensions of the array is called rank of the array.A tuple of integers giving the size of the array along each dimension is known as shape of the array. An array class in Numpy is called as **ndarray**. Elements in Numpy arrays are accessed by using square brackets and can be initialized by using nested Python Lists.

Creating a Numpy Array
Arrays in Numpy can be created by multiple ways, with various number of Ranks, defining the size of the Array. Arrays can also be created with the use of various data types such as lists, tuples, etc. The type of the resultant array is deduced from the type of the elements in the sequences.
Note: Type of array can be explicitly defined while creating the array.

```
# Python program for
# Creation of Arrays
import numpy as np

# Creating a rank 1 Array
arr = np.array([1, 2, 3])
print("Array with Rank 1: \n",arr)

# Creating a rank 2 Array
arr = np.array([[1, 2, 3],
                [4, 5, 6]])
print("Array with Rank 2: \n", arr)

# Creating an array from tuple
arr = np.array((1, 3, 2))
print("\nArray created using "
      "passed tuple:\n", arr)
```

Run on IDE

Output:

```
Array with Rank 1:
 [1 2 3]
Array with Rank 2:
 [[1 2 3]
 [4 5 6]]

Array created using passed tuple:
 [1 3 2]
```

Accessing the array Index

In a numpy array, indexing or accessing the array index can be done in multiple ways. To print a range of an array, slicing is done. Slicing of an array is defining a range in a new array which is used to print a range of elements from the original array. Since, sliced array holds a range of elements of the original array, modifying content with the help of sliced array modifies the original array content.

```
# Python program to demonstrate
# indexing in numpy array
import numpy as np

# Initial Array
arr = np.array([[-1, 2, 0, 4],
                [4, -0.5, 6, 0],
                [2.6, 0, 7, 8],
                [3, -7, 4, 2.0]])
print("Initial Array: ")
print(arr)

# Printing a range of Array
# with the use of slicing method
sliced_arr = arr[:2, ::2]
print ("Array with first 2 rows and"
    " alternate columns(0 and 2):\n", sliced_arr)

# Printing elements at
# specific Indices
Index_arr = arr[[1, 1, 0, 3],
                [3, 2, 1, 0]]
print ("\nElements at indices (1, 3), "
    "(1, 2), (0, 1), (3, 0):\n", Index_arr)
```

Run on IDE

Output:

```
Initial Array:
[[-1.   2.   0.   4. ]
 [ 4.  -0.5  6.   0. ]
 [ 2.6  0.   7.   8. ]
 [ 3.  -7.   4.   2. ]]
Array with first 2 rows and alternate columns(0 and 2):
 [[-1.  0.]
 [ 4.  6.]]

Elements at indices (1, 3), (1, 2), (0, 1), (3, 0):
 [ 0. 54.  2.  3.]
```

Basic Array Operations

In numpy, arrays allow a wide range of operations which can be performed on a particular array or a combination of Arrays. These operation include some basic Mathematical operation as well as Unary and Binary operations.

```
# Python program to demonstrate
# basic operations on single array
import numpy as np

# Defining Array 1
a = np.array([[1, 2],
              [3, 4]])

# Defining Array 2
b = np.array([[4, 3],
              [2, 1]])

# Adding 1 to every element
print ("Adding 1 to every element:", a + 1)

# Subtracting 2 from each element
print ("\nSubtracting 2 from each element:", b - 2)
```

```
# sum of array elements
# Performing Unary operations
print ("\nSum of all array "
       "elements: ", a.sum())

# Adding two arrays
# Performing Binary operations
print ("\nArray sum:\n", a + b)
```

Run on IDE

Output:

```
Adding 1 to every element:
 [[2 3]
 [4 5]]

Subtracting 2 from each element:
 [[ 2  1]
 [ 0 -1]]

Sum of all array elements:  10

Array sum:
 [[5 5]
 [5 5]]
```

More on Numpy Arrays

- Basic Array Operations in Numpy
- Advanced Array Operations in Numpy
- Basic Slicing and Advanced Indexing in NumPy Python

Data Types in Numpy

Every Numpy array is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. Every ndarray has an associated data type (dtype) object. This data type object (dtype) provides information about the layout of the array. The values of an ndarray are stored in a buffer which can be thought of as a contiguous block of memory bytes which can be interpreted by the dtype object. Numpy provides a large set of numeric datatypes that can be used to construct arrays. At the time of Array creation, Numpy tries to guess a datatype, but functions that construct arrays usually also include an optional argument to explicitly specify the datatype.

Constructing a Datatype Object

In Numpy, datatypes of Arrays need not to be defined unless a specific datatype is required. Numpy tries to guess the datatype for Arrays which are not predefined in the constructor function.

```
# Python Program to create
# a data type object
import numpy as np

# Integer datatype
# guessed by Numpy
x = np.array([1, 2])
print("Integer Datatype: ")
print(x.dtype)

# Float datatype
# guessed by Numpy
x = np.array([1.0, 2.0])
print("\nFloat Datatype: ")
print(x.dtype)

# Forced Datatype
```

```
x = np.array([1, 2], dtype = np.int64)
print("\nForcing a Datatype: ")
print(x.dtype)
```

Run on IDE

Output:

```
Integer Datatype:
int64

Float Datatype:
float64

Forcing a Datatype:
int64
```

Math Operations on DataType array

In Numpy arrays, basic mathematical operations are performed element-wise on the array. These operations are applied both as operator overloads and as functions. Many useful functions are provided in Numpy for performing computations on Arrays such as **sum**: for addition of Array elements, **T**: for Transpose of elements, etc.

```
# Python Program to create
# a data type object
import numpy as np

# First Array
arr1 = np.array([[4, 7], [2, 6]],
                 dtype = np.float64)

# Second Array
arr2 = np.array([[3, 6], [2, 8]],
                 dtype = np.float64)

# Addition of two Arrays
Sum = np.add(arr1, arr2)
print("Addition of Two Arrays: ")
print(Sum)

# Addition of all Array elements
# using predefined sum method
Sum1 = np.sum(arr1)
print("\nAddition of Array elements: ")
print(Sum1)

# Square root of Array
Sqrt = np.sqrt(arr1)
print("\nSquare root of Array1 elements: ")
print(Sqrt)

# Transpose of Array
# using In-built function 'T'
Trans_arr = arr1.T
print("\nTranspose of Array: ")
print(Trans_arr)
```

Run on IDE

Output:

```
Addition of Two Arrays:
```

```
[[ 7. 13.]
 [ 4. 14.]]

Addition of Array elements:
19.0

Square root of Array1 elements:
[[2.         2.64575131]
 [1.41421356 2.44948974]]

Transpose of Array:
[[4. 2.]
 [7. 6.]]
```

More on Numpy Data Type

- Data type Object (dtype) in NumPy

Methods in Numpy

all()	arange()	dot()
any()	place()	vdot()
take()	extract()	trunc()
put()	compress()	divide()
apply_along_axis()	rot90()	floor_divide()
apply_over_axes()	tile()	true_divide()
argmin()	reshape()	random.rand()
argmax()	ravel()	random.randn()
nanargmin()	isinf()	ndarray.flat()
nanargmax()	isrealobj()	expm1()
amax()	isscalar()	bincount()
amin()	isneginf()	rint()
insert()	isposinf()	equal()
delete()	iscomplex()	not_equal()
append()	isnan()	less()
around()	iscomplexobj()	less_equal()
flip()	isreal()	greater()
fliplr()	isfinite()	greater_equal()
flipud()	isfortran()	prod()

triu()	exp()	square()
tril()	exp2()	cbrt()
tri()	fix()	logical_or()
empty()	hypot()	logical_and()
empty_like()	absolute()	logical_not()
zeros()	ceil()	logical_xor()
zeros_like()	floor()	array_equal()
ones()	degrees()	array_equiv()
ones_like()	radians()	sin()
full_like()	npv()	cos()
diag()	fv()	tan()
diagflat()	pv()	sinh()
diag_indices()	power()	cosh()
asmatrix()	float_power()	tanh()
bmat()	log()	arcsin()
eye()	log1()	arccos()
roll()	log2()	arctan()
identity()	log10()	arctan2()

Recent articles on Numpy

Programs on Numpy

- Python | Check whether a list is empty or not
- Python | Get unique values from a list
- Python | Multiply all numbers in the list (3 different ways)
- Transpose a matrix in Single line in Python
- Multiplication of two Matrices in Single line using Numpy in Python
- Python program to print checkerboard pattern of nxn using numpy
- Graph Plotting in Python | Set 1, Set 2, Set 3

Useful Numpy Articles

- Matrix manipulation in Python
- Basic Slicing and Advanced Indexing in NumPy Python
- Differences between Flatten() and Ravel()
- rand vs normal in Numpy.random in Python

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

1 Comment **GeeksforGeeks** 1 **Login**

Recommend Tweet Share Sort by Newest

Join the discussion…

LOG IN WITH OR SIGN UP WITH DISQUS ?

Name

Ram • 16 days ago

There's is a typo:
Elements at indices (1, 3), (1, 2), (0, 1), (3, 0): ***should be***
[0. 6. 2. 3.]

• Reply • Share ›

Subscribe Add Disqus to your siteAdd DisqusAdd

Disqus' Privacy PolicyPrivacy PolicyPrivacy

GeeksforGeeks
A computer science portal for geeks

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org

COMPANY	LEARN	PRACTICE	CONTRIBUTE
About Us	Algorithms	Courses	Write an Article
Careers	Data Structures	Company-wise	Write Interview Experience
Privacy Policy	Languages	Topic-wise	Internships
Contact Us	CS Subjects	How to begin?	Videos
	Video Tutorials		