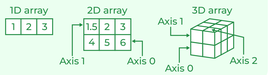
NumPy Cheat Sheet

NumPy stands for Numerical Python.

It is one of the most important foundational packages for numerical computing & data analysis in Python. Most computational packages providing scientific functionality use NumPy's array objects as the lingua franca for data exchange.

Types of Numpy Array



Creating Arrays Commands

One Dimensional Array

From Python List	np.array([1, 2, 3, 4, 5])
From Python Tuple	np.array((1, 2, 3, 4, 5))
fromiter() function	np.fromiter((a for a in range(8)), float)

Python3

- create a NumPy array from a list
```
li = [1, 2, 3, 4]
print(np.array(li))
```
- create a NumPy array from a tuple
```
tup = (5, 6, 7, 8)
print(np.array(tup))
```
- create a NumPy array using fromiter()
```
iterable = (a for a in range(8))
print(np.fromiter(iterable, float))
```

Multi-Dimensional Array

Using Python Lists	np.array([[1, 2, 3, 4],[5, 6, 7, 8], [9, 10, 11, 12]])
Using empty()	np.empty([4, 3], dtype=int)

Python3

- create a NumPy array from a list
```
list_1 = [1, 2, 3, 4]
list_2 = [5, 6, 7, 8]
list_3 = [9, 10, 11, 12]
print(np.array([list_1, list_2, list_3]))
```
- create a NumPy array using numpy.empty()
```
print(np.empty([4, 3], dtype=int))
```

Initial Placeholders

One Dimensional Array

arange()	np.arange(1, 10)
linespace()	np.linspace(1, 10, 3)
zeros()	np.zeros(5, dtype=int)
ones()	np.ones(5, dtype=int)
random.rand()	np.random.rand(5)
random.randint()	np.random.randint(5, size=10)

Python3

- create a NumPy array using numpy.arange()
```
print(np.arange(1, 10))
```
- create a NumPy array using numpy.linspace()
```
print(np.linspace(1, 10, 3))
```
- create a NumPy array using numpy.zeros()
```
print(np.zeros(5, dtype=int))
```
- ccreate a NumPy array using numpy.ones()
```
print(np.ones(5, dtype=int))
```
- create a NumPy array using numpy.random.rand()
```
print(np.random.rand(5))
```
- create a NumPy array using numpy.random.randint()
```
print(np.random.randint(5, size=10))
```

N-dimensional Numpy Arrays

zeros()	np.zeros([4, 3], dtype = np.int32)
ones()	np.ones([4, 3], dtype = np.int32)
full()	np.full([2, 2], 67, dtype = int)
eye()	np.eye(4)

Python3

- create a NumPy array using numpy.zeros()
```
print(np.arange(1, 10))
```
- create a NumPy array using numpy.ones()
```
print(np.ones([4, 3], dtype = np.int32))
```
- create a NumPy array using numpy.full()
```
print(np.full([2, 2], 67, dtype = int))
```
- create a NumPy array using numpy.eye()
```
print(np.eye(4))
```

Inspecting Properties

Size	arr.size
Length	len(arr)
Shape	arr.shape
Datatype	arr.dtype
Changing Datatype of Array	arr.astype('float64')
Converting Array to List	arr.tolist()

Saving and Loading File

Saving array on disk	np.save("file", np.arange(5))
Loading a file	np.load("file.npy")
Importing a Text File	np.loadtxt('file.txt')
Importing CSV File	np.genfromtxt('file.csv' delimiter=',')
Write Text File	np.savetxt('file.txt',arr delimiter=' ')

Data Types

Signed 64-bit integer types	np.int64
Standard double-precision floating point	np.float32
Complex numbers represented by 128 floats	np.complex
Boolean type storing TRUE & FALSE values	np.bool
python object type	np.object
Fixed-length string type	np.string_
Fixed-length unicode type	np.unicode_

Sorting Array

Sorting 1D Array	arr.sort()
Sorting along the first axis of the 2D array	np.sort(a, axis = 0)

NumPy Array Manipulation

Appending Elements to Array

One-Dimensional array

Python3
Adding the values at the end

- of a numpy array
```
print("Original Array:", arr)
```
- appending to the array
```
arr = np.append(arr, [7])
print("Array after appending:", arr)
```

Output:

```
Original Array: [[ 1.  2.  3.  4.]
 [ 5.  6.  7.  8.]
 [ 9. 10. 11. 12.]]
Array after appending: [ 1.  2.  3.  4.  5.  6.  7.  8.
9. 10. 11. 12.  7.]
```

N-Dimensional Array

Python3
Adding the values at the end

- of a numpy array
```
arr = np.arange(1, 13).reshape(2, 6)
print("Original Array")
print(arr, "\n")
```
- create another array which is
  - to be appended column-wise
```
col = np.arange(5, 11).reshape(1, 6)
arr_col = np.append(arr, col, axis=0)
print("Array after appending the values
column wise")
print(arr_col, "\n")
```
  - to be appended row wise
```
row = np.array([1, 2]).reshape(2, 1)
arr_row = np.append(arr, row, axis=1)
print("Array after appending the values row wise")
print(arr_row)
```

Output:

Original Array

```
[[ 1  2  3  4  5  6]
 [ 7  8  9 10 11 12]]
Array after appending the values column wise
[[ 1  2  3  4  5  6]
 [ 7  8  9 10 11 12]
 [ 5  6  7  8  9 10]]
Array after appending the values row wise
[[ 1  2  3  4  5  6  1]
 [ 7  8  9 10 11 12  2]]
```

Inserting Elements into the Array

One-Dimensional array

Python3
arr = np.asarray([1, 2, 3, 4])

- Python Program illustrating numpy.insert()
```
print("1D arr:", arr)
print("Shape:", arr.shape)
```
- Inserting value 9 at index 1
```
a = np.insert(arr, 1, 9)
print("\nArray after insertion:", a)
print("Shape:", a.shape)
```

Removing Elements from Numpy Array

One-Dimensional array

Python3
Python Program illustrating

- numpy.delete()
```
print("Original arr:", arr)
print("Shape : ", arr.shape)
```
- deletion from 1D array
```
object = 2
a = np.delete(arr, object)
print("\ndeleteing the value at index {} from
array:\n {}".format(object,a))
print("Shape : ", a.shape)
```

Reshaping Array

Python3

- creating a numpy array
```
array = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9,
10, 11, 12, 13, 14, 15, 16])
```
- printing array
```
print("Array: " + str(array))
```
- reshaping numpy array
  - converting it to 2-D from 1-D array
```
reshaped1 = array.reshape((4, array.size//4))
```
  - printing reshaped array
```
print("First Reshaped Array:")
print(reshaped1)
```
  - creating another reshaped array
```
reshaped2 = np.reshape(array, (2, 8))
```
  - printing reshaped array
```
print("\nSecond Reshaped Array:")
print(reshaped2)
```

Resizing an Array

Numpy arrays can be resized using the resize() function. It returns nothing but changes the original array.

Python3

- Making a random array
```
arr = np.array([1, 2, 3, 4, 5, 6])
```
- Required values 12, existing values 6
```
arr.resize(3, 4)
print(arr)
```

Flatten a Two Dimensional array

One-Dimensional array

Python3

- Two dimensional numpy array
```
list_1 = [1, 2, 3, 4]
list_2 = [5, 6, 7, 8]
arr = np.array([list_1, list_2])

print(arr.flatten())
```

Transpose

Python3

- making a 3x3 array
```
gfg = np.array([[1, 2],
                [4, 5],
                [7, 8]])
```
- before transpose
```
print(gfg, end ='\n\n')
```
- after transpose
```
print(gfg.transpose(1, 0))
```

Combining and Splitting Commands

Combining Arrays	np.concatenate((arr1, arr2), axis = 0)
Splitting array	np.split(arr, 3, 1)
Horizontal Split	np.hsplit(arr, 3)
Vertical Split	np.vsplit(a, 3)

Indexing, Slicing and Subsetting

Subsetting Numpy Array

Python3

- Index values can be negative.
```
print(arr)
print("Elements are:", arr[np.array([1, 3, -3])])
```

Slicing Numpy Array

The ":" operator means all elements till the end.

Python3
print(arr)

- a[start:stop:step]
```
print("a[-2:7:1] = ",arr[-2:7:1])
print("a[1:] = ",arr[1:])
```

Indexing Numpy Array

Numpy array indexing is of two types: Integer indexing and Boolean indexing.

Python3

- Integer Indexing
```
a = np.array([[1 ,2 ],[3 ,4 ],[5 ,6 ]])
print(a[[0 ,1 ,2 ],[0 ,0 ,1]])
```
- Boolean Indexing
```
a = np.array([10, 40, 80, 50, 100])
print(a[a>50])
```

Copying and Viewing Array

Coping to new memory space	arr.copy()
Shallow Copy	arr.view()

NumPy Array Mathematics

Arithmetic Operations

Adds elements of 2 Array	np.add(a, b)
Substracts elements of 2 Array	np.subtract(a, b)
Multiply elements of 2 Array	np.multiply(a, b)
Divide elements of 2 Array	np.divide(a, b)
Modulo elements of 2 Array	np.mod(a, b)
Remainder elements of 2 Array	np.remainder(a,b)
Power elements of 2 Array	np.power(a, b)
Exponant elements of 2 Array	np.exp(b)

Comparison

Python3

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

comparison = an_array == another_array
equal_arrays = comparison.all()

print(equal_arrays)
```

Vector Math

Python3
arr = np.array([1.5, 1.5, 2.5, 3.5, 4.5, 10.1])

- numpy.delete()
```
print("Original arr:", arr)
print("Shape : ", arr.shape)
```
- applying sqrt() method
```
print("Square-root:", np.sqrt(arr))
```
- applying log() method
```
print("Log Value: ", np.log(arr))
```
- applying absolute() method
```
print("Absolute Value:", np.absolute(arr))
```
- applying sin() method
```
print("Sine values:", np.sin(arr))
```
- applying ceil() method
```
print("Ceil values:", np.ceil(arr))
```
- applying floor() method
```
print("Floor Values:", np.floor(arr))
```
- applying round_() method
```
print ("Rounded values:", np.round_(arr))
```

Statistic

Python3

- numpy.delete()
```
print("Original arr:", arr)
print("Shape : ", arr.shape)
```
- 1D array
```
arr = [20, 2, 7, 1, 34]
```
- mean
```
print("Mean of arr:", np.mean(arr))
```
- median
```
print("Median of arr:", np.median(arr))
```
- sum
```
print("Sum of arr(uint8):",
np.sum(arr, dtype = np.uint8))
print("Sum of arr(float32):",
np.sum(arr, dtype = np.float32))
```
- min and max
```
print("maximum element:", np.max(arr))
print("minimum element:", np.min(arr))
```
- var
```
print("var of arr:", np.var(arr))
print("var of arr(float32):",
np.var(arr, dtype = np.float32))
```
- standard deviation
```
print("std of arr:", np.std(arr))
print ("More precision with float32",
np.std(arr, dtype = np.float32))
```

corrcoef

Python3
create numpy 1d-array

- create numpy 1d-array
```
array1 = np.array([0, 1, 2])
array2 = np.array([3, 4, 5])
```
- pearson product-moment correlation - coefficients of the arrays
```
rslt = np.corrcoef(array1, array2)

print(rslt)
```

Output:

```
[[1. 1.]
 [1. 1.]]
```

GeeksforGeeks