Numpy - Single dimensional arrays

Creating an array from a list	x = np.array(['a', 'b', '9', '8']
Array Data Types	Consist of integers, floating-point numbers, or strings. Data type must be consistent. Numpy's record array gives mixed DTypes
Find the length on an array	np.len(x)
Accessing elements from an array	x[index_num] *x[1]*
Assign elements from index	*x[1]=c*
Slicing x[start:end])	print(x[1:2]) *['b', '9']*
Slicing x[start:end: step])	print(x[1:3:2]) *['b', '8']*
Modify a new version of an array without changing the original	y = x.copy()
Negative slices, single value x[-Distance from end]	x[-2] *b*
Negative slices, reversal x[start:end: -step])	x[3:0:-2]) *8, b*
Adding to an array	x.append('7')

Numpy - Single dimensional arrays (cont)

Saving to binary file	np.save(open('data.npy', 'wb'), data)

Useful Numpy Functions

Array Creation:	arange, array, copy, empty, empty_like, eye, fromfile, fromfunction, identity, linspace, logspace, mgrid, ogrid, ones, ones_like, r_, zeros, zeros_like
Conversions	ndarray.astype, atleast_1d, atleast_2d, atleast_3d, mat
Manipulations:	array_split, column_stack, concatenate, diagonal, dsplit, dstack, hsplit, hstack, ndarray.item, newaxis, ravel, repeat, reshape, resize, squeeze, swapaxes, take, transpose, vsplit, vstack
Questions:	all, any, nonzero, where
Ordering:	argmax, argmin, argsort, max, min, ptp, searchsorted, sort
Operations:	choose, compress, cumprod, cumsum, inner, ndarray.fill, imag, prod, put, putmask, real, sum
Basic Statistics:	cov, mean, std, var

Useful Numpy Functions (cont)

Basic Linear Algebra:	cross, dot, outer, linalg.svd, vdot

Combining Arrays

np.vtack((a1, a2) =	np.concatenate((a1, a2, axis = 0)
np.hstack(a1,a2) =	np.concatenate((a1, a2, axis = 1)
vsplit	splits along the vertical axis
hsplit	splits along the horizontal axis

Universal Functions (ufuncts)

Implement vectorization (operations applied to whole arrays instead of individual elements) in NumPy which is way faster than iterating over elements.	Also provide broadcasting (when smaller array is cast across the larger array so that they have compatible shapes)
x = [1, 2, 3, 4] y = [4, 5, 6, 7] z = np.add(x, y)	[5 7 9 11]
Check if a function is a ufunc:	print(type(np.add))
arr1 = np.array([10, 20, 30, 40, 50, 60])	arr2 = np.array([20, 21, 22, 23, 24, 25])
np.subtract(arr1, arr2)	newarr = np.multiply(arr1, arr2)

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

Universal Functions (ufuncts) (cont)

newarr = np.divide(arr1, arr2)	newarr = np.power(arr1, arr2)
newarr = np.remainder(arr1, arr2)	newarr = np.absolute(arr)

Iterating

arr = np.array([[1, 2, 3], [4, 5, 6]])	
for x in arr: print(x)	[1, 2, 3], [4, 5, 6]
for x in arr: for y in x: print(y)	123456
for x in np.nditer(arr):	123456
for x in np.nditer(arr, flags=['buffered'], op_dtypes=['S']): print(x)	b '1', b'2', b'3'
for x in np.nditer(arr[:, ::2]): print(x)	1, 3, 5, 7
Enumeration means mentioning sequence number of somethings one by one.	
for idx, x in np.ndenumerate(arr): print(idx, x)	
for idx, x in np.ndenumerate(arr): print(idx, x) (	0,) 1 (1,) 2 (2,) 3

Iterating (cont)

for idx, x in np.ndenumerate(arr): print(idx, x)	Result Size: 1425 x 1251 import numpy as np arr = np.array([[1, 2, 3, 4], [5, 6, 7, 8]]) for idx, x in np.ndenumerate(arr): print(idx, x) (0, 0) 1 (0, 1) 2 (0, 2) 3 (0, 3) 4 (1, 0) 5 (1, 1) 6 (1, 2) 7 (1, 3) 8

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com