## Create Array Using Python List

import numpy as np

# create a list named list1

list1 = [2, 4, 6, 8]

# create numpy array using list1

array1 = np.array(list1)

print(array1)

# Output: [2 4 6 8]

import numpy as np

# create numpy array using a list

array1 = np.array([2, 4, 6, 8])

print(array1)

# Output: [2 4 6 8]

## Create an Array Using np.zeros()

The np.zeros() function allows us to create an array filled with all zeros.

import numpy as np

# create an array with 4 elements filled with zeros

array1 = np.zeros(4)

print(array1)

# Output: [0. 0. 0. 0.]

**Create an Array With np.arange()**

The np.arange() function returns an array with values within a specified interval. For example,

import numpy as np

# create an array with values from 0 to 4

array1 = np.arange(5)

print("Using np.arange(5):", array1)

# create an array with values from 1 to 8 with a step of 2

array2 = np.arange(1, 9, 2)

print("Using np.arange(1, 9, 2):",array2)

**Output**

Using np.arange(5): [0 1 2 3 4]

Using np.arange(1, 9, 2): [1 3 5 7]

**Create an Array With np.random.rand()**

The np.random.rand() function is used to create an array of random numbers.

Let's see an example to create an array of **5** random numbers,

import numpy as np

# generate an array of 5 random numbers

array1 = np.random.rand(5)

print(array1)

**Output**

[0.08455648 0.56379034 0.66463204 0.97608605 0.30700052]

**Creating the array using for Loop**

x=[]

for i in range(5):

num=int(input("Enter the number"))

x.append(num)

print(x)

### **Create a 2-D NumPy Array**

Let's create a 2D NumPy array with **2** rows and **4** columns using lists.

import numpy as np

# create a 2D array with 2 rows and 4 columns

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

[5, 6, 7, 8]])

print(array1)

**Output**

[[1 2 3 4]

[5 6 7 8]]

### Create a 3-D NumPy Array

Let's say we want to create a 3-D NumPy array consisting of two **"slices"** where each slice has **3** rows and **4** columns.

import numpy as np

# create a 3D array with 2 "slices", each of 3 rows and 4 columns

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

[5, 6, 7, 8],

[9, 10, 11, 12]],

[[13, 14, 15, 16],

[17, 18, 19, 20],

[21, 22, 23, 24]]])

print(array1)

**Output**

[[[ 1 2 3 4]

[ 5 6 7 8]

[ 9 10 11 12]]

[[13 14 15 16]

[17 18 19 20]

[21 22 23 24]]]

**Create N-D Arrays using np.zeros()**

The np.zeros() function allows us to create N-D arrays filled with all zeros. For example,

import numpy as np

# create 2D array with 2 rows and 3 columns filled with zeros

array1 = np.zeros((2, 3))

print("2-D Array: ")

print(array1)

# create 3D array with dimensions 2x3x4 filled with zeros

array2 = np.zeros((2, 3, 4))

print("\n3-D Array: ")

print(array2)

**Output**

2-D Array:

[[0. 0. 0.]

[0. 0. 0.]]

3-D Array:

[[[0. 0. 0. 0.]

[0. 0. 0. 0.]

[0. 0. 0. 0.]]

[[0. 0. 0. 0.]

[0. 0. 0. 0.]

[0. 0. 0. 0.]]]

**Create N-D Array with a Specified Value**

In NumPy, we can use the np.full() function to create a multidimensional array with a specified value.

For example, to create a 2-D array with the value **5**, we can do the following:

import numpy as np

# Create a 2-D array with elements initialized to 5

numpy\_array = np.full((2, 2), 5)

print("Array:", numpy\_array)

**Output**

[[5 5]

[5 5]]

## Creating Arrays With np.random.rand()

The np.random.rand() function is used to create an array of random numbers.

Let's see an example to create an array of **5** random numbers,

import numpy as np

# create a 2D array of 2 rows and 2 columns of random numbers

array1 = np.random.rand(2, 2)

print("2-D Array: ")

print(array1)

# create a 3D array of shape (2, 2, 2) of random numbers

array2 = np.random.rand(2, 2, 2)

print("\n3-D Array: ")

print(array2)

[Run Code](https://www.programiz.com/python-programming/online-compiler)

**Output**

2-D Array:

[[0.13198621 0.54730421]

[0.36570987 0.16233836]]

3-D Array:

[[[0.15666007 0.4580507 ]

[0.84769856 0.76699589]]

[[0.45395202 0.39944328]

[0.62999479 0.39629496]]]

Here,

* np.random.rand(2, 2) - creates a 2D array of **2** rows and **2** columns of random numbers.
* np.random.rand(2, 2, 2) - creates a 3D array with 2 slices, each slice having **2** rows and **2** columns of random numbers.

## Create Empty N-D NumPy Array

To create an empty N-D NumPy array, we use the np.empty() function. For example,

import numpy as np

# create an empty 2D array with 2 rows and 2 columns

array1 = np.empty((2, 2))

print("2-D Array: ")

print(array1)

# create an empty 3D array of shape (2, 2, 2)

array2 = np.empty((2, 2, 2))

print("\n3-D Array: ")

print(array2)

[Run Code](https://www.programiz.com/python-programming/online-compiler)

**Output**

2-D Array:

[[8.86495615e-317 0.00000000e+000]

[2.21149159e-316 1.76125651e-312]]

3-D Array:

[[[1.0749539e-316 0.0000000e+000]

[0.0000000e+000 0.0000000e+000]]

[[0.0000000e+000 0.0000000e+000]

[0.0000000e+000 0.0000000e+000]]]

## Check Data Type of a NumPy Array

To check the data type of a NumPy array, we can use the dtype attribute. For example,

import numpy as np

# create an array of integers

array1 = np.array([2, 4, 6])

# check the data type of array1

print(array1.dtype)

# Output: int64

import numpy as np

# create an array of integers

int\_array = np.array([-3, -1, 0, 1])

# create an array of floating-point numbers

float\_array = np.array([0.1, 0.2, 0.3])

# create an array of complex numbers

complex\_array = np.array([1+2j, 2+3j, 3+4j])

# check the data type of int\_array

print(int\_array.dtype) # prints int64

# check the data type of float\_array

print(float\_array.dtype) # prints float64

# check the data type of complex\_array

print(complex\_array.dtype) # prints complex128

# NumPy Array Attributes

|  |  |
| --- | --- |
| Attributes | Description |
| ndim | returns number of dimension of the array |
| size | returns number of elements in the array |
| dtype | returns data type of elements in the array |
| shape | returns the size of the array in each dimension. |
| itemsize | returns the size (in bytes) of each elements in the array |
| data | returns the buffer containing actual elements of the array in memory |

mport numpy as np

# create a 2-D array

array1 = np.array([[2, 4, 6],

[1, 3, 5]])

# check the dimension of array1

print(array1.ndim)

# Output: 2

import numpy as np

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

[6, 7, 8]])

# return total number of elements in array1

print(array1.size)

# Output: 6

import numpy as np

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

[6, 7, 8]])

# return a tuple that gives size of array in each dimension

print(array1.shape)

# Output: (2,3)

import numpy as np

# create an array of integers

array1 = np.array([6, 7, 8])

# check the data type of array1

print(array1.dtype)

# Output: int64

import numpy as np

# create a default 1-D array of integers

array1 = np.array([6, 7, 8, 10, 13])

# create a 1-D array of 32-bit integers

array2 = np.array([6, 7, 8, 10, 13], dtype=np.int32)

# use of itemsize to determine size of each array element of array1 and array2

print(array1.itemsize) # prints 8

print(array2.itemsize) # prints 4

import numpy as np

array1 = np.array([6, 7, 8])

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

[6, 7, 8]])

# print memory address of array1's and array2's data

print("\nData of array1 is: ",array1.data)

print("Data of array2 is: ",array2.data)

**Output**

Data of array1 is: <memory at 0x7f746fea4a00>

Data of array2 is: <memory at 0x7f746ff6a5a0>

### Access Array Elements Using Index

import numpy as np

array1 = np.array([1, 3, 5, 7, 9])

# access numpy elements using index

print(array1[0]) # prints 1

print(array1[2]) # prints 5

print(array1[4]) # prints 9

import numpy as np

# create a numpy array

numbers = np.array([2, 4, 6, 8, 10])

# change the value of the first element

numbers[0] = 12

print("After modifying first element:",numbers) # prints [12 4 6 8 10]

# change the value of the third element

numbers[2] = 14

print("After modifying third element:",numbers) # prints [12 4 14 8 10]

import numpy as np

# create a numpy array

numbers = np.array([1, 3, 5, 7, 9])

# access the last element

print(numbers[-1]) # prints 9

# access the second-to-last element

print(numbers[-2]) # prints 7

import numpy as np

# create a numpy array

numbers = np.array([2, 3, 5, 7, 11])

# modify the last element

numbers[-1] = 13

print(numbers) # Output: [2 3 5 7 13]

# modify the second-to-last element

numbers[-2] = 17

print(numbers) # Output: [2 3 5 17 13]