# NumPy Basic: Exercises, Practice, Solution

**1.**Write a NumPy program to get the numpy version and show numpy build configuration.

import numpy as np

print(np.\_\_version\_\_)

print(np.show\_config())

**2.** Write a NumPy program to  get help on the add function.

import numpy as np

print(np.info(np.add))

**3.** Write a NumPy program to test whether none of the elements of a given array is zero.

import numpy as np

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

print("Original array:")

print(x)

print("Test if none of the elements of the said array is zero:")

print(np.all(x))

x = np.array([0, 1, 2, 3])

print("Original array:")

print(x)

print("Test if none of the elements of the said array is zero:")

print(np.all(x))

**Note:** [numpy.any()](https://kite.com/python/docs/numpy.any) returns True if at least one element in a NumPy array evaluates to True while [numpy.all()](https://kite.com/python/docs/numpy.all) returns True only if all elements in a NumPy array evaluate to True.

**4.** Write a NumPy program to test whether any of the elements of a given array is non-zero.

import numpy as np

x = np.array([1, 0, 0, 0])

print("Original array:")

print(x)

print("Test whether any of the elements of a given array is non-zero:")

print(np.any(x))

x = np.array([0, 0, 0, 0])

print("Original array:")

print(x)

print("Test whether any of the elements of a given array is non-zero:")

print(np.any(x))

**5.** Write a NumPy program to test a given array element-wise for finiteness (not infinity or not a Number).

**Note:**

**numpy.Inf**

IEEE 754 floating point representation of (positive) infinity.

**numpy.NAN**

IEEE 754 floating point representation of Not a Number (NaN).

**numpy.NINF**

IEEE 754 floating point representation of negative infinity.

import numpy as np

a = np.array([1, 0, np.nan, np.inf])

print("Original array")

print(a)

print("Test a given array element-wise for finiteness :")

print(np.isfinite(a))

**6.** Write a NumPy program to test element-wise for positive or negative infinity.

import numpy as np

a = np.array([1, 0, np.nan, np.inf])

print("Original array")

print(a)

print("Test element-wise for positive or negative infinity:")

print(np.isinf(a))

**7.** Write a NumPy program to test element-wise for NaN of a given array.

import numpy as np

a = np.array([1, 0, np.nan, np.inf])

print("Original array")

print(a)

print("Test element-wise for NaN:")

print(np.isnan(a))

**8.** Write a NumPy program to test element-wise for complex number, real number of a given array. Also test whether a given number is a scalar type or not.

**Note**

**numpy.isscalar(*element*)**[**[source]**](https://github.com/numpy/numpy/blob/master/numpy/core/numeric.py#L1865-L1943)

Returns True if the type of *element* is a scalar type.

**Parameters**

**element*any***

Input argument, can be of any type and shape.

**Returns**

**val*bool***

True if *element* is a scalar type, False if it is not.

import numpy as np

a = np.array([1+1j, 1+0j, 4.5, 3, 2, 2j])

print("Original array")

print(a)

print("Checking for complex number:")

print(np.iscomplex(a))

print("Checking for real number:")

print(np.isreal(a))

print("Checking for scalar type:")

print(np.isscalar(3.1))

print(np.isscalar([3.1]))

**9.** Write a NumPy program to create an element-wise comparison (greater, greater\_equal, less and less\_equal) of two given arrays.

import numpy as np

x = np.array([3, 5])

y = np.array([2, 5])

print("Original numbers:")

print(x)

print(y)

print("Comparison - greater")

print(np.greater(x, y))

print("Comparison - greater\_equal")

print(np.greater\_equal(x, y))

print("Comparison - less")

print(np.less(x, y))

print("Comparison - less\_equal")

print(np.less\_equal(x, y))

**10.** Write a NumPy program to create an element-wise comparison (equal, equal within a tolerance) of two given arrays.

import numpy as np

x = np.array([72, 79, 85, 90, 150, -135, 120, -10, 60, 100])

y = np.array([72, 79, 85, 90, 150, -135, 120, -10, 60, 100.000001])

print("Original numbers:")

print(x)

print(y)

print("Comparison - equal:")

print(np.equal(x, y))

print("Comparison - equal within a tolerance:")

print(np.allclose(x, y))

**ma.allclose(*a*, *b*, *masked\_equal=True*, *rtol=1e-05*, *atol=1e-08*)**

Returns True if two arrays are element-wise equal within a tolerance.

This function is equivalent to **[allclose](https://numpy.org/devdocs/reference/generated/numpy.allclose.html" \l "numpy.allclose" \o "numpy.allclose)** except that masked values are treated as equal (default) or unequal, depending on the **[masked\_equal](https://numpy.org/devdocs/reference/generated/numpy.ma.masked_equal.html" \l "numpy.ma.masked_equal" \o "numpy.ma.masked_equal)** argument.

**11.** Write a NumPy program to create an array with the values 1, 7, 13, 105 and determine the size of the memory occupied by the array.

import numpy as np

X = np.array([1, 7, 13, 105])

print("Original array:")

print(X)

print("Size of the memory occupied by the said array:")

print("%d bytes" % (X.size \* X.itemsize))

**12.** Write a NumPy program to create an array of 10 zeros,10 ones, 10 fives.

import numpy as np

array=np.zeros(10)

print("An array of 10 zeros:")

print(array)

array=np.ones(10)

print("An array of 10 ones:")

print(array)

array=np.ones(10)\*5

print("An array of 10 fives:")

print(array)

**13.** Write a NumPy program to create an array of the integers from 30 to70.

import numpy as np

array=np.arange(30,71)

print("Array of the integers from 30 to70")

print(array)

**14.** Write a NumPy program to create an array of all the even integers from 30 to 70.

import numpy as np

array=np.arange(30,71,2)

print("Array of all the even integers from 30 to 70")

print(array)