```
import numpy as np
```

Q1 - Write a NumPy program to get the NumPy version and show the NumPy build configuration.

```
print(np.__version__)
```

```
1.26.4
```

Q2 - Write a NumPy program to get help with the add function.

```
print(np.info(np.add))
```

```
add(x1, x2, /, out=None, *, where=True, casting='same_kind',
order='K', dtype=None, subok=True[, signature, extobj])

Add arguments element-wise.

Parameters
----------
x1, x2 : array_like
    The arrays to be added.
    If ``x1.shape != x2.shape``, they must be broadcastable to a
common
    shape (which becomes the shape of the output).
out : ndarray, None, or tuple of ndarray and None, optional
    A location into which the result is stored. If provided, it must
have
    a shape that the inputs broadcast to. If not provided or None,
    a freshly-allocated array is returned. A tuple (possible only as a
    keyword argument) must have length equal to the number of outputs.
where : array_like, optional
    This condition is broadcast over the input. At locations where the
    condition is True, the `out` array will be set to the ufunc
result.
    Elsewhere, the `out` array will retain its original value.
    Note that if an uninitialized `out` array is created via the
default
    ``out=None``, locations within it where the condition is False
will
    remain uninitialized.
**kwargs
    For other keyword-only arguments, see the
    :ref:`ufunc docs <ufuncs.kwargs>`.

Returns
-------
add : ndarray or scalar
    The sum of `x1` and `x2`, element-wise.
```

```
    This is a scalar if both `x1` and `x2` are scalars.

Notes
-----
Equivalent to `x1` + `x2` in terms of array broadcasting.

Examples
--------
>>> np.add(1.0, 4.0)
5.0
>>> x1 = np.arange(9.0).reshape((3, 3))
>>> x2 = np.arange(3.0)
>>> np.add(x1, x2)
array([[  0.,   2.,   4.],
       [  3.,   5.,   7.],
       [  6.,   8.,  10.]])

The ``+`` operator can be used as a shorthand for ``np.add`` on
ndarrays.

>>> x1 = np.arange(9.0).reshape((3, 3))
>>> x2 = np.arange(3.0)
>>> x1 + x2
array([[ 0.,  2.,  4.],
       [ 3.,  5.,  7.],
       [ 6.,  8., 10.]])
None
```

Q3 - Write a NumPy program to test whether none of the elements of a given array are zero.

```
arr = np.array([0, 0, 0, 0])

result = not np.any(arr)

print("None of the elements are non-zero:", result)

None of the elements are non-zero: True
```

Q4 - Write a NumPy program to test if any of the elements of a given array are non-zero.

```
x = np.array([0,0,5])
print(np.any(x))

True
```

Q5 - Write a NumPy program to test a given array element-wise for finiteness (not infinity or not a number).

```
x = np.array([1, 0, np.nan, np.inf])
print(np.isfinite(x))  # Returns True only for finite numbers
```

```
[ True  True False False]
```

Q6 - Write a NumPy program to test elements-wise for positive or negative infinity.

```
x = np.array([1, 0, np.nan, np.inf])
print(np.isnan(x))  # Returns True where value is NaN
```

```
[False False  True False]
```

Q7 - Write a NumPy program to test element-wise for NaN of a given array.

```
x = np.array([1, 0, np.nan, np.inf, -np.inf])
print(np.isinf(x))  # Returns True where value is ±infinity
```

```
[False False False  True  True]
```

Q8 - Write a NumPy program to test element-wise for complex numbers, real numbers in a given array. Also test if a given number is of a scalar type or not.

```
arr = np.array([0, 0, 1, 0])
result = np.any(arr)

print("Any non-zero value?:", result)
```

```
Any non-zero value?: True
```

Q9 - Write a NumPy program to test whether two arrays are element-wise equal within a tolerance.

```
x = np.array([1, 0, np.inf, -np.inf])
print(np.isneginf(x))  # Returns True only for -infinity
```

```
[False False False  True]
```

Q10 - Write a NumPy program to create an element-wise comparison (greater, greater_equal, less and less_equal) of two given arrays.

```
x = np.array([1.0, 2.0, 3.0000001])
y = np.array([1.0, 2.0, 3.0000002])
print(np.allclose(x, y))  # Returns True if they are close within
small diff
```

```
True
```

Q11 - Write a NumPy program to create an element-wise comparison (equal, equal within a tolerance) of two given arrays.

```
x = np.array([1, 2, 3])
y = np.array([1, 2, 3])
print(np.array_equal(x, y))  # True if completely equal
```

```
True
```

Q12 - 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.

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

# Check size of each element in bytes
element_size = arr.itemsize  # returns size of one element in bytes

# Total memory used by array = number of elements * size of each
element
total_memory = arr.nbytes  # or use len(arr) * arr.itemsize

print("Array:", arr)
print("Each element size (bytes):", element_size)
print("Total memory used (bytes):", total_memory)
```

```
Array: [  1   7  13 105]
Each element size (bytes): 4
Total memory used (bytes): 16
```

Q13 - Write a NumPy program to create an array of 10 zeros, 10 ones, and 10 fives.

```
arr = np.zeros(10)
arr[5] = 11
print("Updated array:", arr)
```

```
Updated array: [ 0.  0.  0.  0.  0. 11.  0.  0.  0.  0.]
```

Q14 - Write a NumPy program to create an array of integers from 30 to 70.

```
even_array = np.arange(30, 71, 2)

print("Even numbers from 30 to 70:", even_array)
```

```
Even numbers from 30 to 70: [30 32 34 36 38 40 42 44 46 48 50 52 54 56
58 60 62 64 66 68 70]
```

Q15 - Write a NumPy program to create an array of all even integers from 30 to 70.

```
matrix = np.arange(2, 11).reshape(3, 3)

print("3x3 Matrix from 2 to 10:\n", matrix)
```

```
3x3 Matrix from 2 to 10:
 [[ 2  3  4]
 [ 5  6  7]
 [ 8  9 10]]
```

Q16 - Write a NumPy program to create a 3x3 identity matrix.

```
im = np.identity(3)
print(im)
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

Q17 - Write a NumPy program to generate a random number between 0 and 1.

```
rn = np.random.rand()
print(rn)
```

```
0.352263079730463
```

Q18 - Write a NumPy program to generate an array of 15 random numbers from a standard normal distribution.

```
rn = np.random.uniform(5,50)
print(rn)
```

```
47.71792648342832
```

Q19 - Write a NumPy program to create a vector with values ranging from 15 to 55 and print all values except the first and last.

```
matrix = np.diag([1, 2, 3, 4], k=-1)
print("5x5 Matrix with values just below the diagonal:\n", matrix)
```

```
5x5 Matrix with values just below the diagonal:
 [[0 0 0 0 0]
 [1 0 0 0 0]
 [0 2 0 0 0]
 [0 0 3 0 0]
 [0 0 0 4 0]]
```

Q20 - Write a NumPy program to create a 3x4 array and iterate over it.

```
ww = np.random.rand(3,3,3)
print(ww)
```

```
[[[0.83196314 0.53206721 0.159326  ]
  [0.42632636 0.38619222 0.79652441]
  [0.13862533 0.00910606 0.56226244]]

 [[0.56264919 0.08063038 0.79366437]
  [0.7224712  0.97822564 0.75740137]
  [0.60793259 0.17083647 0.11784266]]

 [[0.54834089 0.46097483 0.45817302]
  [0.01129686 0.21409055 0.49143162]
  [0.17936881 0.32304462 0.16630053]]]
```

Q21 - Write a NumPy program to create a vector of length 10 with values evenly distributed between 5 and 50.

```
x = np.linspace(5,50,10)
x
```

```
array([ 5., 10., 15., 20., 25., 30., 35., 40., 45., 50.])
```

Q22 - Write a NumPy program to create a vector with values from 0 to 20 and change the sign of the numbers in the range from 9 to 15.

```
x = np.arange(21)                          # □ Create vector from 0 to 20
x[(x >= 9) & (x <= 15)] *= -1              # □ Change sign of elements
from 9 to 15
print(x)
```

```
[  0   1   2   3   4   5   6   7   8  -9 -10 -11 -12 -13 -14 -15  16
17
  18  19  20]
```

Q23 - Write a NumPy program to create a vector of length 5 filled with arbitrary integers from 0 to 10.

```
x = np.random.randint(0,11,5)
x
```

```
array([4, 4, 2, 7, 0])
```

Q24 - Write a NumPy program to multiply the values of two given vectors.

```
x = np.array([1,2,3])
y = np.array([4,5,6])

print(np.multiply(x,y))
```

```
[ 4 10 18]
```

Q25 - Write a NumPy program to create a 3x4 matrix filled with values from 10 to 21.

```
x = np.arange(10,22).reshape(3,4)
x
```

```
array([[10, 11, 12, 13],
       [14, 15, 16, 17],
       [18, 19, 20, 21]])
```

Q26 - Write a NumPy program to find the number of rows and columns in a given matrix.

```
x = np.arange(12).reshape(3,4)
x.shape
```

```
(3, 4)
```

```
x.shape[1]
```

```
4
```

Q27 - Write a NumPy program to create a 3x3 identity matrix, i.e. the diagonal elements are 1, the rest are 0.

```
np.identity(3)
```

```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

Q28 - Write a NumPy program to create a 10x10 matrix, in which the elements on the borders will be equal to 1, and inside 0.

```
x = np.ones((10, 10))
x[1:-1, 1:-1] = 0
print(x)
```

```
[[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]]
```

Q29 - Write a NumPy program to create a 5x5 zero matrix with elements on the main diagonal equal to 1, 2, 3, 4, 5.

```
np.fill_diagonal(x, [1, 2, 3, 4, 5])   # □ Fill diagonal with 1 to 5
print(x)
```

```
  Cell In[30], line 1
    x = np.zeroes(5,5))
                      ^
SyntaxError: unmatched ')'
```

Q30 - Write a NumPy program to create a 4x4 matrix in which 0 and 1 are staggered, with zeros on the main diagonal.

Q31 - Write a NumPy program to create a 3x3x3 array filled with arbitrary values.

Q32 - Write a NumPy program to compute the sum of all elements, the sum of each column and the sum of each row in a given array.

Q33 - Write a NumPy program to compute the inner product of two given vectors.

Q34 - Write a NumPy program to add a vector to each row of a given matrix.

Q35 - Write a NumPy program to save a given array to a binary file.

Q36 - Write a NumPy program to save a given array to a binary file.

Q37 - Write a NumPy program to save a given array to a text file and load it.

Q38 - Write a NumPy program to convert a given array into bytes, and load it as an array.

Q39 - Write a NumPy program to convert a given list into an array, then again convert it into a list. Check initial list and final list are equal or not.

Q40 - Write a NumPy program to compute the x and y coordinates for points on a sine curve and plot the points using matplotlib.

Q41 - Write a NumPy program to convert numpy dtypes to native Python types.

Q42 - Write a NumPy program to add elements to a matrix. If an element in the matrix is 0, we will not add the element below this element.

Q43 - Write a NumPy program to find missing data in a given array.

Q44 - Write a NumPy program to check whether two arrays are equal (element wise) or not.

Q45 - Write a NumPy program to create a one-dimensional array of single, two and three-digit numbers.

Q46 - Write a NumPy program to create a two-dimensional array of a specified format.

Q47 - Write a NumPy program to create a one-dimensional array of forty pseudo-randomly generated values. Select random numbers from a uniform distribution between 0 and 1.

Q48 - Write a NumPy program to create a two-dimensional array with shape (8,5) of random numbers. Select random numbers from a normal distribution (200,7).

Q49 - Write a NumPy program to generate a uniform, non-uniform random sample from a given 1-D array with and without replacement.

Q50 - Write a NumPy program to create a 4x4 array with random values. Create an array from the said array by swapping first and last rows.

Q51 - Write a NumPy program to create a new array of given shape (5,6) and type, filled with zeros.

Q52 - Write a NumPy program to sort a given array by row and column in ascending order.

Q53 - Write a NumPy program to extract all numbers from a given array less and greater than a specified number.

Q54 - Write a NumPy program to replace all numbers in a given array equal, less and greater than a given number.

Q55 - Write a NumPy program to create an array of equal shape and data type for a given array.

Q56 - Write a NumPy program to create a three-dimensional array with the shape (3,5,4) and set it to a variable.

Q57 - Write a NumPy program to create a 4x4 array. Create an array from said array by swapping first and last, second and third columns.

Q58 - Write a NumPy program to swap rows and columns of a given array in reverse order.

Q59 - Write a NumPy program to multiply two given arrays of the same size element-by-element.

```
```