```
Docstring:
array(object, dtype=None, *, copy=True, order='K', subok=False, ndmin=0,
      like=None)

Create an array.

Parameters
----------
object : array_like
    An array, any object exposing the array interface, an object whose
    __array__ method returns an array, or any (nested) sequence.
dtype : data-type, optional
    The desired data-type for the array.  If not given, then the type will
    be determined as the minimum type required to hold the objects in the
    sequence.
copy : bool, optional
    If true (default), then the object is copied.  Otherwise, a copy will
    only be made if __array__ returns a copy, if obj is a nested sequence,
    or if a copy is needed to satisfy any of the other requirements
    (`dtype`, `order`, etc.).
order : {'K', 'A', 'C', 'F'}, optional
    Specify the memory layout of the array. If object is not an array, the
    newly created array will be in C order (row major) unless 'F' is
    specified, in which case it will be in Fortran order (column major).
    If object is an array the following holds.

    ===== ========= ===================================================
    order  no copy                     copy=True
    ===== ========= ===================================================
    'K'   unchanged F & C order preserved, otherwise most similar order
    'A'   unchanged F order if input is F and not C, otherwise C order
    'C'   C order   C order
    'F'   F order   F order
    ===== ========= ===================================================

    When ``copy=False`` and a copy is made for other reasons, the result is
    the same as if ``copy=True``, with some exceptions for 'A', see the
    Notes section. The default order is 'K'.
subok : bool, optional
    If True, then sub-classes will be passed-through, otherwise
    the returned array will be forced to be a base-class array (default).
ndmin : int, optional
    Specifies the minimum number of dimensions that the resulting
    array should have.  Ones will be pre-pended to the shape as
    needed to meet this requirement.
like : array_like
    Reference object to allow the creation of arrays which are not
    NumPy arrays. If an array-like passed in as ``like`` supports
    the ``__array_function__`` protocol, the result will be defined
    by it. In this case, it ensures the creation of an array object
    compatible with that passed in via this argument.
    .. versionadded:: 1.20.0

Returns
-------
out : ndarray
    An array object satisfying the specified requirements.

See Also
--------
empty_like : Return an empty array with shape and type of input.
ones_like : Return an array of ones with shape and type of input.
zeros_like : Return an array of zeros with shape and type of input.
full_like : Return a new array with shape of input filled with value.
empty : Return a new uninitialized array.
ones : Return a new array setting values to one.
zeros : Return a new array setting values to zero.
full : Return a new array of given shape filled with value.


Notes
-----
When order is 'A' and `object` is an array in neither 'C' nor 'F' order,
and a copy is forced by a change in dtype, then the order of the result is
not necessarily 'C' as expected. This is likely a bug.

Examples
--------
>>> np.array([1, 2, 3])
array([1, 2, 3])

Upcasting:

>>> np.array([1, 2, 3.0])
array([ 1.,  2.,  3.])

More than one dimension:

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

Minimum dimensions 2:

>>> np.array([1, 2, 3], ndmin=2)
array([[1, 2, 3]])

Type provided:

>>> np.array([1, 2, 3], dtype=complex)
array([ 1.+0.j,  2.+0.j,  3.+0.j])

Data-type consisting of more than one element:
```

```
>>> x = np.array([(1,2),(3,4)],dtype=[('a','<i4'),('b','<i4')])
>>> x['a']
array([1, 3])

Creating an array from sub-classes:

>>> np.array(np.mat('1 2; 3 4'))
array([[1, 2],
       [3, 4]])

>>> np.array(np.mat('1 2; 3 4'), subok=True)
matrix([[1, 2],
        [3, 4]])
Type:      builtin_function_or_method
```