```
Init signature: np.dtype(self, /, *args, **kwargs)
Docstring:
dtype(dtype, align=False, copy=False)

Create a data type object.

A numpy array is homogeneous, and contains elements described by a
dtype object. A dtype object can be constructed from different
combinations of fundamental numeric types.

Parameters
----------
dtype
    Object to be converted to a data type object.
align : bool, optional
    Add padding to the fields to match what a C compiler would output
    for a similar C-struct. Can be ``True`` only if `obj` is a dictionary
    or a comma-separated string. If a struct dtype is being created,
    this also sets a sticky alignment flag ``isalignedstruct``.
copy : bool, optional
    Make a new copy of the data-type object. If ``False``, the result
    may just be a reference to a built-in data-type object.

See also
--------
result_type

Examples
--------
Using array-scalar type:

>>> np.dtype(np.int16)
dtype('int16')

Structured type, one field name 'f1', containing int16:

>>> np.dtype([('f1', np.int16)])
dtype([('f1', '<i2')])

Structured type, one field named 'f1', in itself containing a structured
type with one field:

>>> np.dtype([('f1', [('f1', np.int16)])])
dtype([('f1', [('f1', '<i2')])])

Structured type, two fields: the first field contains an unsigned int, the
second an int32:

>>> np.dtype([('f1', np.uint64), ('f2', np.int32)])
dtype([('f1', '<u8'), ('f2', '<i4')])

Using array-protocol type strings:
```

```
>>> np.dtype([('a','f8'),('b','S10')])
dtype([('a', '<f8'), ('b', 'S10')])

Using comma-separated field formats.  The shape is (2,3):

>>> np.dtype("i4, (2,3)f8")
dtype([('f0', '<i4'), ('f1', '<f8', (2, 3))])

Using tuples.  ``int`` is a fixed type, 3 the field's shape.  ``void``
is a flexible type, here of size 10:

>>> np.dtype([('hello',(np.int64,3)),('world',np.void,10)])
dtype([('hello', '<i8', (3,)), ('world', 'V10')])

Subdivide ``int16`` into 2 ``int8``'s, called x and y.  0 and 1 are
the offsets in bytes:

>>> np.dtype((np.int16, {'x':(np.int8,0), 'y':(np.int8,1)}))
dtype((numpy.int16, [('x', 'i1'), ('y', 'i1')]))

Using dictionaries.  Two fields named 'gender' and 'age':

>>> np.dtype({'names':['gender','age'], 'formats':['S1',np.uint8]})
dtype([('gender', 'S1'), ('age', 'u1')])

Offsets in bytes, here 0 and 25:

>>> np.dtype({'surname':('S25',0),'age':(np.uint8,25)})
dtype([('surname', 'S25'), ('age', 'u1')])
File:           c:\users\unknown\anaconda\lib\site-packages\numpy\__init__.py
Type:           _DTypeMeta
Subclasses:     dtype[bool_], dtype[int8], dtype[uint8], dtype[int16], dtype[uint16], dtype[intc], dt
ype[uintc], dtype[int32], dtype[uint32], dtype[int64], ...
```