In [1]:
```
#Write a NumPy program to get the numpy version and show numpy build configuratio
import numpy as np
print(np.__version__)
print(np.show_config())
```

```
1.16.5
mkl_info:
    libraries = ['mkl_rt']
    library_dirs = ['C:/Users/MADHU/Anaconda3\\Library\\lib']
    define_macros = [('SCIPY_MKL_H', None), ('HAVE_CBLAS', None)]
    include_dirs = ['C:\\Program Files (x86)\\IntelSWTools\\compilers_and_libra
ries_2019.0.117\\windows\\mkl', 'C:\\Program Files (x86)\\IntelSWTools\\compile
rs_and_libraries_2019.0.117\\windows\\mkl\\include', 'C:\\Program Files (x86)
\\IntelSWTools\\compilers_and_libraries_2019.0.117\\windows\\mkl\\lib', 'C:/Use
rs/MADHU/Anaconda3\\Library\\include']
blas_mkl_info:
    libraries = ['mkl_rt']
    library_dirs = ['C:/Users/MADHU/Anaconda3\\Library\\lib']
    define_macros = [('SCIPY_MKL_H', None), ('HAVE_CBLAS', None)]
    include_dirs = ['C:\\Program Files (x86)\\IntelSWTools\\compilers_and_libra
ries_2019.0.117\\windows\\mkl', 'C:\\Program Files (x86)\\IntelSWTools\\compile
rs_and_libraries_2019.0.117\\windows\\mkl\\include', 'C:\\Program Files (x86)
\\IntelSWTools\\compilers_and_libraries_2019.0.117\\windows\\mkl\\lib', 'C:/Use
rs/MADHU/Anaconda3\\Library\\include']
blas_opt_info:
    libraries = ['mkl_rt']
    library_dirs = ['C:/Users/MADHU/Anaconda3\\Library\\lib']
    define_macros = [('SCIPY_MKL_H', None), ('HAVE_CBLAS', None)]
    include_dirs = ['C:\\Program Files (x86)\\IntelSWTools\\compilers_and_libra
ries_2019.0.117\\windows\\mkl', 'C:\\Program Files (x86)\\IntelSWTools\\compile
rs_and_libraries_2019.0.117\\windows\\mkl\\include', 'C:\\Program Files (x86)
\\IntelSWTools\\compilers_and_libraries_2019.0.117\\windows\\mkl\\lib', 'C:/Use
rs/MADHU/Anaconda3\\Library\\include']
lapack_mkl_info:
    libraries = ['mkl_rt']
    library_dirs = ['C:/Users/MADHU/Anaconda3\\Library\\lib']
    define_macros = [('SCIPY_MKL_H', None), ('HAVE_CBLAS', None)]
    include_dirs = ['C:\\Program Files (x86)\\IntelSWTools\\compilers_and_libra
ries_2019.0.117\\windows\\mkl', 'C:\\Program Files (x86)\\IntelSWTools\\compile
rs_and_libraries_2019.0.117\\windows\\mkl\\include', 'C:\\Program Files (x86)
\\IntelSWTools\\compilers_and_libraries_2019.0.117\\windows\\mkl\\lib', 'C:/Use
rs/MADHU/Anaconda3\\Library\\include']
lapack_opt_info:
    libraries = ['mkl_rt']
    library_dirs = ['C:/Users/MADHU/Anaconda3\\Library\\lib']
    define_macros = [('SCIPY_MKL_H', None), ('HAVE_CBLAS', None)]
    include_dirs = ['C:\\Program Files (x86)\\IntelSWTools\\compilers_and_libra
ries_2019.0.117\\windows\\mkl', 'C:\\Program Files (x86)\\IntelSWTools\\compile
rs_and_libraries_2019.0.117\\windows\\mkl\\include', 'C:\\Program Files (x86)
\\IntelSWTools\\compilers_and_libraries_2019.0.117\\windows\\mkl\\lib', 'C:/Use
rs/MADHU/Anaconda3\\Library\\include']
None
```

In [2]:
```
#Write a NumPy program to  get help on the add function.
import numpy as np
np.info(np.add)
```

```
add(x1, x2, /, out=None, *, where=True, casting='same_kind', order='K', dtype=N
one, 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 may be the shape of one or
    the other).
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
    Values of True indicate to calculate the ufunc at that position, values
    of False indicate to leave the value in the output alone.
**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.]])
```

In [4]:
```
#Write a NumPy program to test whether none of the elements of a given array is z
import numpy as np
x=np.array([1,2,3,4])
np.all(x)
```

Out[4]: True

In [5]:
```
x=np.array([0,1,2,3,4])
np.all(x)
```

Out[5]: False

In [6]:
```
#Write a NumPy program to test whether any of the elements of a given array is no
y=np.array([1,2,3,4,5])
np.any(y)
```

Out[6]: True

In [7]:
```
y=np.array([0,0,0])
np.any(y)
```

Out[7]: False

In [10]:
```
#Write a NumPy program to test a given array element-wise for finiteness (not inf
y=np.array([1,2,3,4])
np.isfinite(y)
```

Out[10]: array([True, True, True, True])

In [11]:
```
# Write a NumPy program to test element-wise for positive or negative infinity
np.isposinf(y)
```

Out[11]: array([False, False, False, False])

In [12]:
```
np.isinf(y)
```

Out[12]: array([False, False, False, False])

In [13]:
```
np.isinf(np.inf)
```

Out[13]: True

In [14]:
```
np.isinf(np.NINF)
```

Out[14]: True

In [15]:
```
#Write a NumPy program to test element-wise for NaN of a given array.
np.isnan(y)
```

Out[15]: array([False, False, False, False])

In [16]:
```
np.isnan(0)
```

Out[16]: False

In [17]:
```
np.isnan(np.nan)
```

Out[17]: True

In [21]:
```
#Write a NumPy program to test element-wise for complex number, real number of a
x=np.array([32,1+1j,7,9.8])
np.iscomplex(x)
```

Out[21]:
```
array([False,  True, False, False])
```

In [22]:
```
np.isreal(x)
```

Out[22]:
```
array([ True, False,  True,  True])
```

In [24]:
```
np.isscalar(3.3)
```

Out[24]:
```
True
```

In [25]:
```
np.isscalar(1)
```

Out[25]:
```
True
```

In [ ]:
```
#Write a NumPy program to test whether two arrays are element-wise equal within a
```