Logical expressions can be combined using four logical devices,

```
% BEGIN TABLE
Keyword (Scalar) & Function & Bitwise & True if . . . \\ \hline
and & logical_and & Both & True \\ \hline
or  & logical_or & Either or Both True \\ \hline
not & logical_not & ~ & Not True \\ \hline
& logical_xor & ^ & One True and One False \\ \hline

% END OF TABLE
```

There are three versions of all operators except XOR. The keyword version (e.g. and) can only be used with scalars and so it not useful when working with NumPy. Both the function and bitwise operators can be used with NumPy arrays, although care is requires when using the bitwise operators.

Bitwise operators

Bitwise operators have high priority – higher than logical comparisons – and so parentheses are requires around comparisons. For example, $(x > 1)\&(x < 5)$ is a valid statement, while $x > 1\&x < 5$, which is evaluated as $(x > (1\&x)) < 5$, produces an error.

```
>>> x = arange(2.0,4)
>>> y = x >= 0
>>> z = x < 2
>>> logical_and(y, z)
array([False, False, True, True, False, False], dtype=bool)
>>> y & z
array([False, False, True, True, False, False], dtype=bool)
>>> (x > 0) & (x < 2)
array([False, False, True, True, False, False], dtype=bool)
```

0.1 Multiple tests: all and any

The commands all and any take logical input and are self-descriptive. all returns True if all logical elements in an array are 1.

- If all is called without any additional arguments on an array, it returns True if all elements of the array are logical true and 0 otherwise.
- any returns logical(True) if any element of an array is True.

Both all and any can be also be used along a specific dimension using a second argument or the keyword argument axis to indicate the axis of operation (0 is column-wise and 1 is row-wise).

When used column- or row-wise, the output is an array with one less dimension than the input, where each element of the output contains the truth value of the operation on a column or row.

```
>>> x = array([[1,2][3,4]])
>>> y = x <= 2
>>> y
array([[ True, True],
[False, False]], dtype=bool)
>>> any(y)
True
>>> any(y,0)
array([[ True, True]], dtype=bool)
>>> any(y,1)
array([[ True],
[False]], dtype=bool)
```

is* A number of special purpose logical tests are provided to determine if an array has special characteristics. Some operate element-by-element and produce an array of the same dimension as the input while other produce only scalars. These functions all begin with is.

```
Operator True if . . . Method of operation
isnan 1 if nan element-by-element
isinf 1 if inf element-by-element
isfinite 1 if not inf and not nan element-by-element
isposfin,isnegfin 1 for positive or negative inf element-by-element
isreal 1 if not complex valued element-by-element
iscomplex 1 if complex valued element-by-element
isreal 1 if real valued element-by-element
is_string_like 1 if argument is a string scalar
is_numlike 1 if is a numeric type scalar
isscalar 1 if scalar scalar
isvector 1 if input is a vector scalar
```