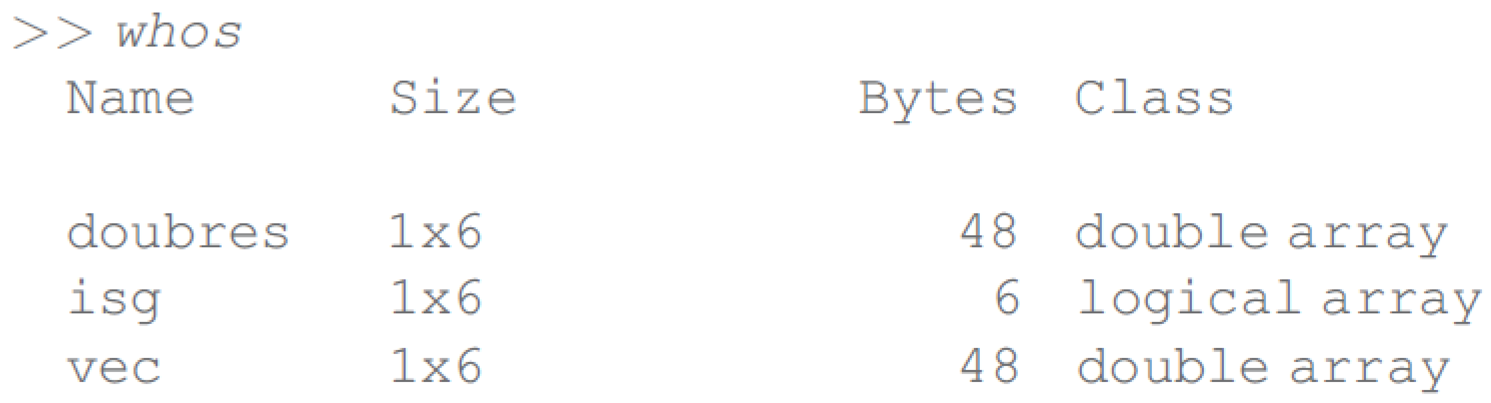
Logical Indexing

by AhmetSacan

Logical Vectors

```
>> vec = [ 1 5 8 9 2 7 ];
>> isg = vec > 4


>> doubleres = isg +5
```



```
>> whos
  Name          Size                    Bytes  Class

  doubres       1x6                        48  double array
  isg           1x6                         6  logical array
  vec           1x6                        48  double array
```

Logical Indexing

- A logical vector/matrix can be used to index (select) elements of a vector/matrix.
  - The logical vector/matrix must have the same size as the vector/matrix being indexed.

```
>> vec = [ 1 5 8 9 2 7 ];
>> isg = vec > 4


>> vec (isg)


>> vec (isg) = vec(isg) + 1
```

Exercise

- a=[5 8 1 7 9 2]
- Create another vector b that contains the even elements of a.
- Change odd elements of a with 99.
- a=[5 8 1 7 9 2]
- Change the odd elements of a, with +1 more than each.
- a=[5 8 1 7 9 2]
- Remove all elements of a that are divisible by 3.
- Repeat steps above for a=[5 8 1; 7 9 2]

Logical Indexing of Matrices

- m=randi(100,3,5)
- m([1 3], [2 4])
- m([5 8 10])
- m([true false true], [false true false true false])
- m([true false true], [1 5])
- m([false true false false true , true false true false false true false true false])

Beware of 0/1.

```
>> a = [ 5 2 1 7 9 6 ];
>> a ( [1 0 1 0 1 0] )
??? Subscript indices must either be real positive
integers or logicals.

>> a( logical( [1 0 1 0 1 0] ) )

>> a( [true false true false true false] )
```

find

- find(x) function returns the indices of non-zero elements of x.

```
>> find ( [ false true false true ] )


>> find ( [ 3 0 2 5 0  ] )



>> vec = [ 1 5 8 9 2 7 ];
>> find ( vec > 5 )
```

Equivalence of logical indexing & indexing with find()

- find() can be used to convert a logical index to a numeric index
- Assuming I is a logical vector/matrix, the following expressions are equivalent:
  - m (I)
  - m (find(I))

Exercise: Zero-crossing

- Find the indices of the elements of vector v, that are followed by a number with an opposite sign. Assume that v does not contain zero. Hints: sign, diff, find.

v = [5 -1 -2 -1 3 8 -2]

Logical Operators: Element-wise and, or, not

```
>> a = [ 3 0 5  1 ];
>> b = [ 0 0 -5 0 ];
>> a & b



>> a | b



>> ~a
```

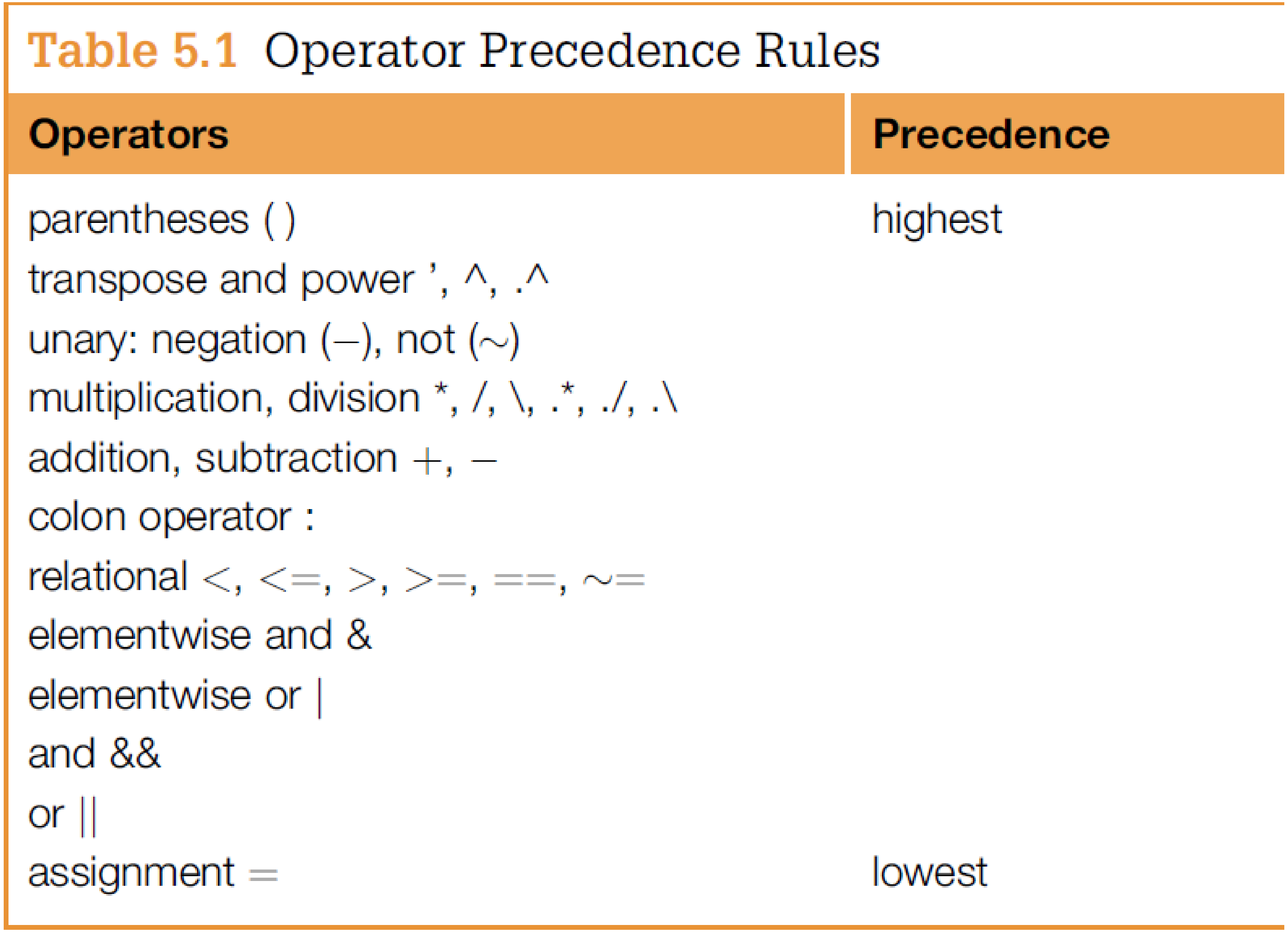


Table 5.1 Operator Precedence Rules

Operators	Precedence
parentheses ( )	highest
transpose and power ', ^, .^	
unary: negation (−), not (~)	
multiplication, division *, /, \, .*, ./, .\	
addition, subtraction +, −	
colon operator :	
relational <, <=, >, >=, ==, ~=	
elementwise and &	
elementwise or |	
and &&	
or ||	
assignment =	lowest

Programming Concept

- When you want to operate on elements of a matrix that satisfy a condition, first construct an Index for elements that satisfy the condition.

Exercise

- Let *A*=magic(4)
- Replace with 0, all elements of *A* that are odd **and** whose log() is greater than 2.

Exercise

- Given a three-column matrix A, store in B all rows of A where the sum of the first two numbers is less than the third number.
- e.g.,

```
A = [
   35   91   23
   35    7   77
   33   22    3
   93   87   40
   29   67   21
   61   79   67
   17   25   92
   85   56    1]
```

Your expression should produce:

```
B = [35    7   77
     17   25   92]
```

Exercise

- Given a three-column matrix A, store in B all rows of A where the sum of the first two numbers is **greater than or equal to** the third number.
- e.g.,

```
A = [
   35   91   23
   35    7   77
   33   22    3
   93   87   40
   29   67   21
   61   79   67
   17   25   92
   85   56    1]
```

Your expression should produce:

B = [35 91 23; 33 22 3; 93 87 40; 29 67 21; 61 79 67; 85 56 1]

Logical Functions: all(), any()

```
>> vec1 = [1 3 1 1 2];
>> all ( vec1 )
>> any ( vec1 )

>> vec2 = [ 1 1 0 1]
>> all ( vec2 )
>> any ( vec2 )
```

any(), all() with matrices

- If a matrix is given as input, these functions apply to each column.

```
>> A=[0 0 3 2; 4 0 0 5; 6 0 5 8]
     0    0    3    2
     4    0    0    5
     6    0    5    8
>> any( A )
1×4 logical array
   1  0  1  1
```

- If you want to apply to each column, provide the second "dimension" argument.

```
>> any(A, 2)
  3×1 logical array
   1
   1
   1
```

- **any(A,1)** is same as **any(A)**

meshgrid

- meshgrid creates a mesh (all combinations) of coordinates

```
>> [x, y] = meshgrid([1 2 3], [5,6])
x =
    1    2    3
    1    2    3
y =
    5    5    5
    6    6    6
```

meshgrid example

- Find all integers x, y less than or equal to N (as example, use N=4) where:
  - $x^2 + y = 3 * x$

meshgrid example

- Make a 3D plot of the following function:

$$f(x, y) = x * \sin(x + 2 * y)$$

- Use x=0:.1:5, y=0:.1:5
- Use plot3 and surf functions.