ENGR 101 – Chapter 4

Logical Operations and Indexing

Laura Alford, James Juett, Rick Niciejewski 9/11/2020

Recall: Binary Arithmetic Operations

- Essentially, doing math with two operands.

	Operator	Example	Result
Addition	+	2 + 3	5
Subtraction	-	5 - 3	2
Multiplication	.*	5 .* 3	15
Exponentiation	.^	2 .^ 3	8
Division	./	11 ./ 4	2.75
Modulo (remainder)		mod(11,4)	3

Asking Questions

- Consider the following expression:

```
x < 5
```

- In math, this would mean that x must be less than 5.
- In programming, this is a **question**.
  - "Check the current value of x… Is it less than 5?"
  - The result is called a **truth value**, and is either true or false.
  - True and false are often encoded as `1` and `0`.

false	true
`0`	`1`

Relational Operations

- These operations check for **equality** or perform **comparisons**.

	Operator	Example	Result
Equality	==	2 == 3	0
Inequality	~=	2 ~= 3	1
Less Than	<	5 < 5	0
Less Than or Equal	<=	5 <= 5	1
Greater Than	>	3 > 4	0
Greater Than or Equal	>=	4 >= 5	1

Relational Operations on Matrices

a

3	4	2
6	5	1

b

2	1	4
1	3	7

c

1

d

1	7
2	5

In MATLAB, anything that you can do with scalars, you can do with arrays (i.e. vectors and matrices).

```
a < b
```

3	4	2
6	5	1

a

<

2	1	4
1	3	7

b

ans =

0	0	1
0	0	1

Relational Operations on Matrices

a

3	4	2
6	5	1

b

2	1	4
1	3	7

c

1

d

1	7
2	5

d == c

1	7	==	1	
2	5			
d			c	

ans =

1	0
0	0

Relational Operations on Matrices

a

3	4	2
6	5	1

b

2	1	4
1	3	7

c

1

d

1	7
2	5

```
b <= a
```

2	1	4
1	3	7

b

<=

3	4	2
6	5	1

a

ans =

1	1	0
1	1	0

Relational Operations on Matrices

a

3	4	2
6	5	1

b

2	1	4
1	3	7

c

1

d

1	7
2	5

$b > (a + 2)$

2	1	4
1	3	7

b

$>$

3	4	2
6	5	1

a

$+$ 2

ans =

0	0	0
0	0	1

Relational Operations on Matrices

a

3	4	2
6	5	1

b

2	1	4
1	3	7

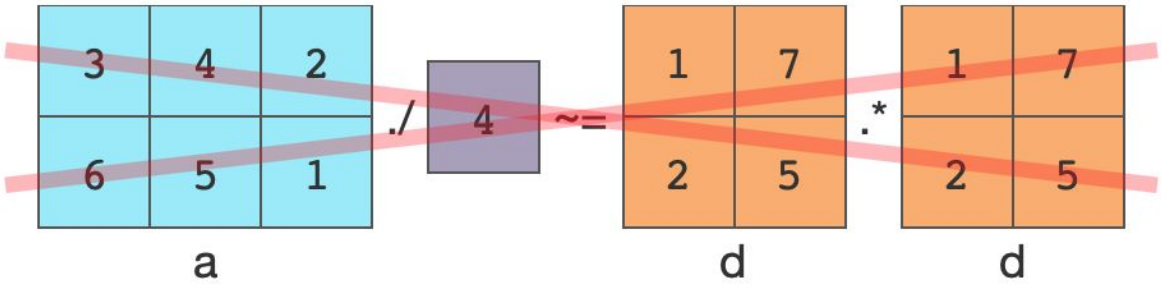
c

1

d

1	7
2	5

```
a ./ 4 ~= d .* d
```



Mismatched dimensions for operator undefined. LHS is a 2x3 and RHS is a 2x2.

Counting Elements that Match

- You can use the `sum` function, combined with a logical operation, to count the number of elements that match certain criteria.

- Example: How many elements are greater than 2?

1	2	4
3	3	2

`a`

0	0	1
1	1	0

`a > 2`

3

`sum(sum(a > 2))`

Note: The logical 0s and 1s are implicitly converted to numeric 0s and 1s when a logical matrix is passed into a function like `sum`.

Logical Arrays

- The result of a relational or logical operation is a **logical array**.
- The 1s and 0s in a logical array are a completely different type of data than regular numbers.
- In many contexts, an expression will behave differently with logical vs. regular numbers.
  - Indexing is a common example.

```
>> x = [1,2;3,4]

x =

     1     2
     3     4

>> which = x > 2

which =

  2×2 logical array

   0   0
   1   1
```

Quick Note - Hardcoding Logical Arrays

- If you want to hardcode a logical matrix, you need to wrap it up in the `logical` function (otherwise it's just regular numeric `1`s and `0`s).

```
[1,0;0,1]
```

1	0
0	1

```
logical([1,0;0,1])
```

1	0
0	1

Recall: Row/Column Indexing

- Specify separate row/column indices.
  - Rows first, then columns, separated by a comma.
  - We can use either a **single number**, a **regular array**, or a **:**.

The : is useful for selecting full rows and/or columns.

x

4	2	6	3
7	2	4	3
4	6	8	1

x(2,[4 1])

4	2	6	3
7	2	4	**3**
4	6	8	1

x

ans = | 3 | 7 |

x(3,:)

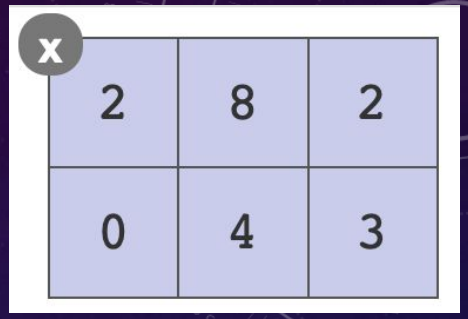
4	2	6	3
7	2	4	3
4	**6**	**8**	**1**

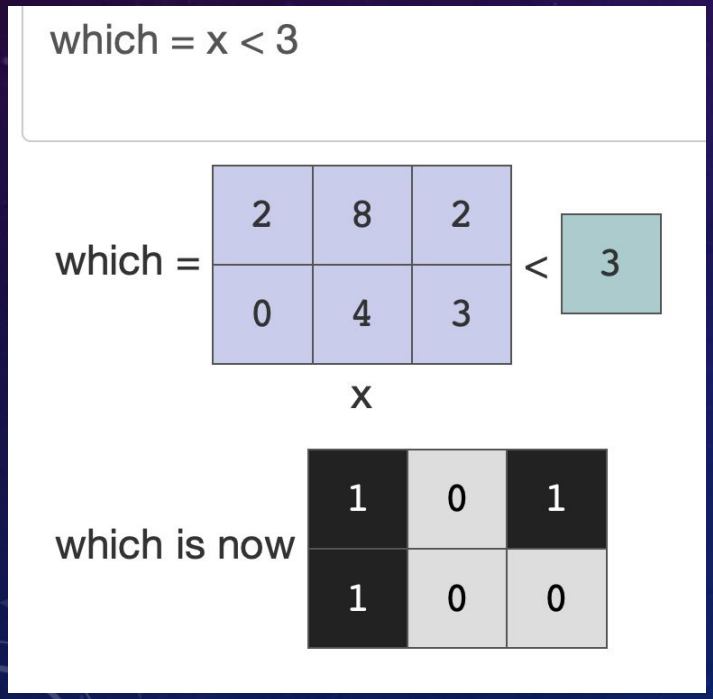
x

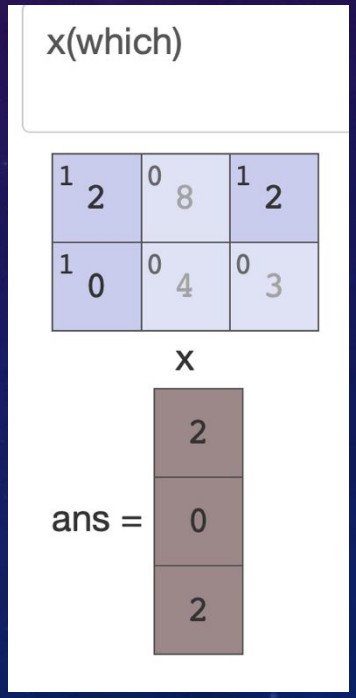
ans = | 4 | 6 | 8 | 1 |

Logical Indexing

- We can also use a **logical array** to index.
- In this case, we select all the elements from the source matrix that correspond to positions with a 1 in the logical matrix.

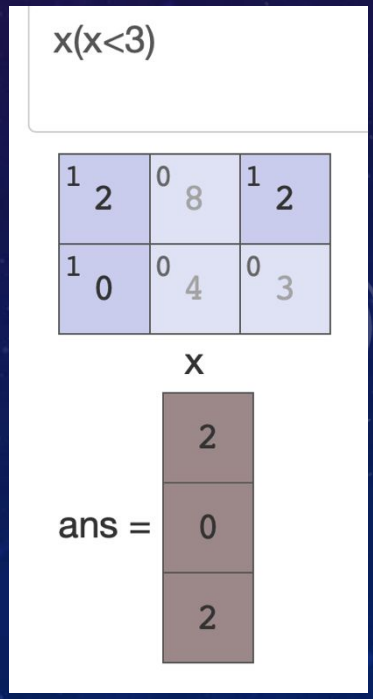






You can also combine these two steps into one statement

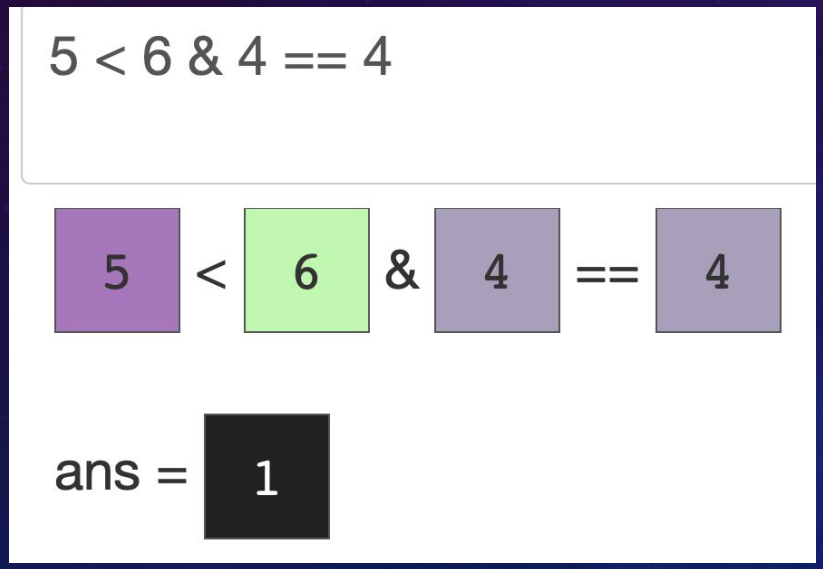
Either way is just fine!

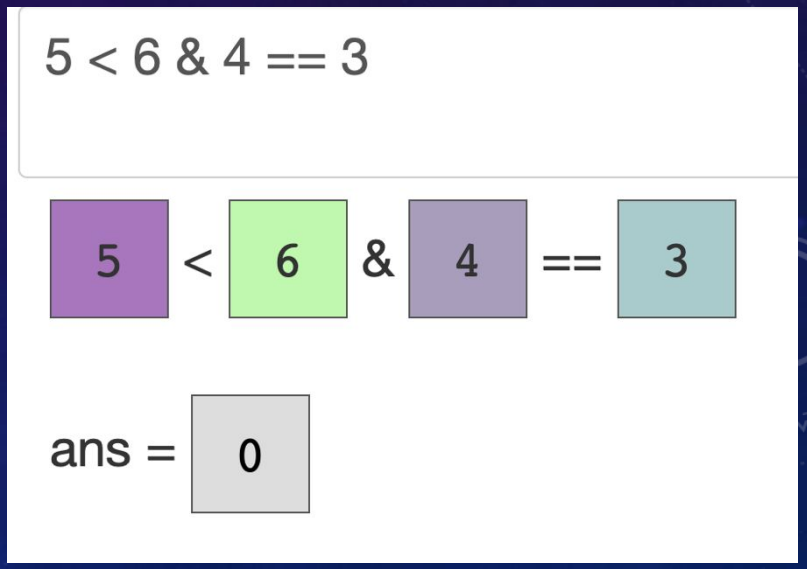


*In any reasonable case, the logical matrix used for indexing should be the same size as the source matrix.

Logical Operations

- We can use **logical operators** to combine two truth values according to the rules of **formal logic**.
- For example, the **&** operator implements **logical and**.
  - If BOTH operands are true, the whole result is true. Otherwise false.





Logical Operations

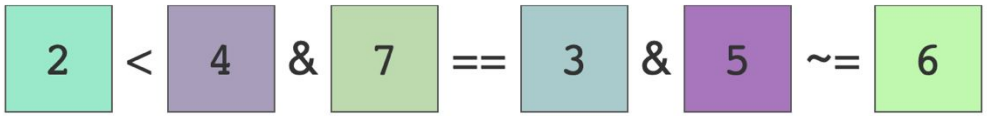
You can “chain” together multiple logical operations…

```
2 < 4 & 7 > 3 & 5 ~= 6
```


```
ans = 1
```

… but if ANY ONE of the logical operations evaluates to be false, the whole thing is false!

```
2 < 4 & 7 == 3 & 5 ~= 6
```

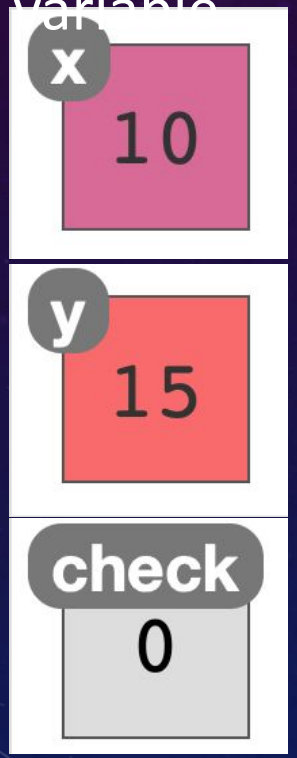



```
ans = 0
```

bit.ly/2B4Z5aM

Logical Operations

You can assign the result of a logical operation to a variable.



x = 10

y = 15

check = x >= y & y > 5

check = 10 >= 15 & 15 > 5

x y y

check is now 0

Inclusive vs. Exclusive Or

- There are two kinds of "or"…

- Inclusive Or
  - One or the other, or both.
  - "Were you in sports or music in high school?"

or

- Exclusive Or
  - One or the other, but NOT both.
  - "Would you like the soup or salad?"

xor

Logical Operations

- Essentially, combining two **truth values** in a particular way.
- Most operations have both a symbolic operator and a function.

	Operator	Example	Result
Logical And	&	`2 < 3 & 5 > 6`	`0`
Logical Or	|	`2 < 3 | 5 > 6`	`1`
Exclusive Or		`xor(0,1)`	`1`
Not	~	`~(1 == 2)`	`1`

The precedence of ~ is higher than ==, so we need parentheses here.

Supplemental Exercise: Logical Operator Truth Table

- A formal way to define the logical operators is with a truth table that shows the result for all combinations of operands.
- Fill in the missing pieces of the table…

A	B	~A	~B	A & B	A | B	xor(A, B)
0	0			0	0	
0	1	1				
1	0					
1	1		0			0

Remember that for logical values, 0 = false and 1 = true

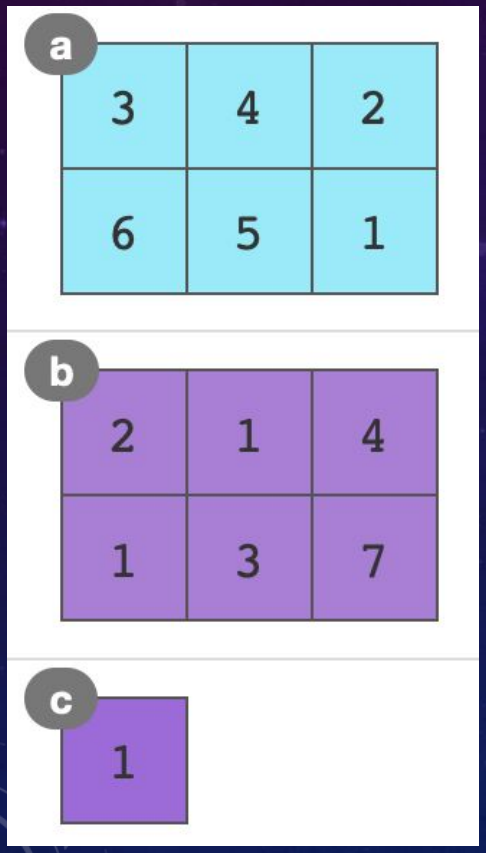
Solution: Logical Operator Truth Table

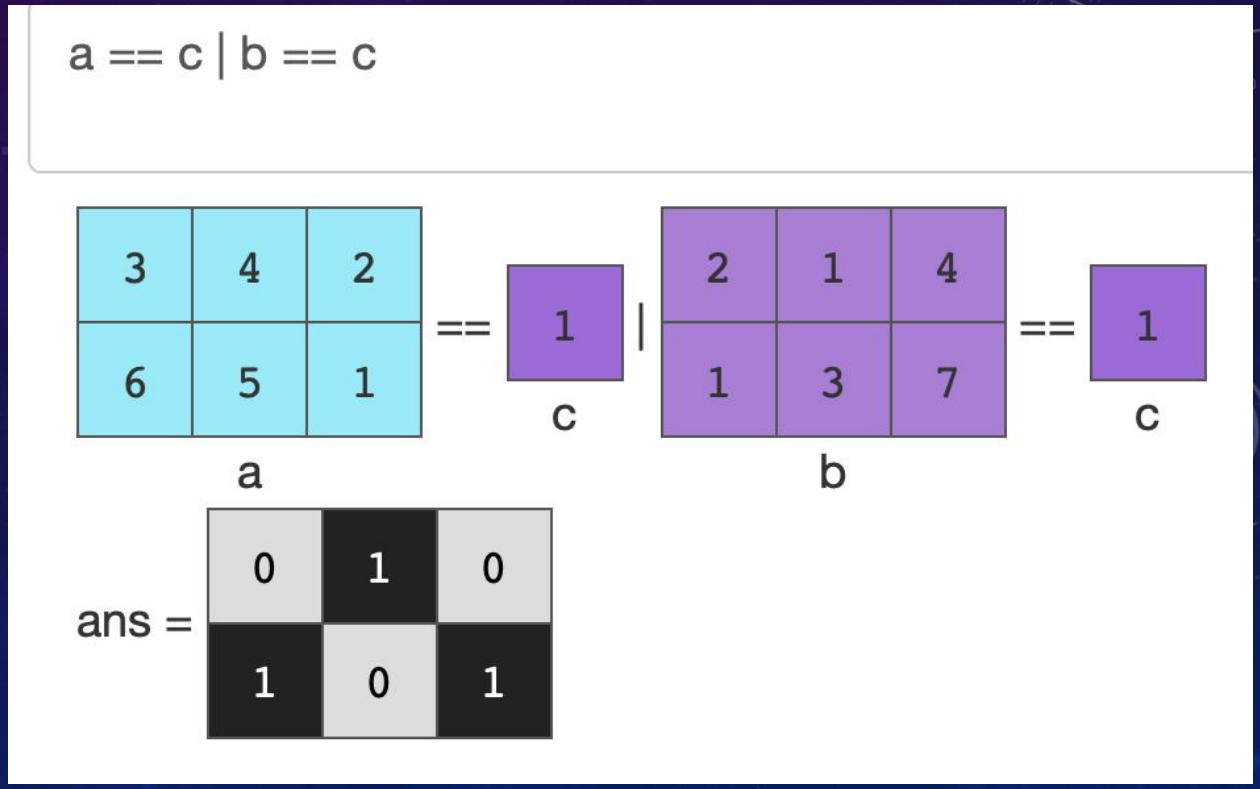
- A formal way to define the logical operators is with a truth table that shows the result for all combinations of operands.
- Fill in the missing pieces of the table…

A	B	~A	~B	A & B	A | B	xor(A, B)
0	0	1	1	0	0	0
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	1	1	0

Logical Operations with Matrices

- Of course, we can perform these same operations with matrices as well as scalars.





Logical Operations with Matrices

a

3	4	2
6	5	1

b

2	1	4
1	3	7

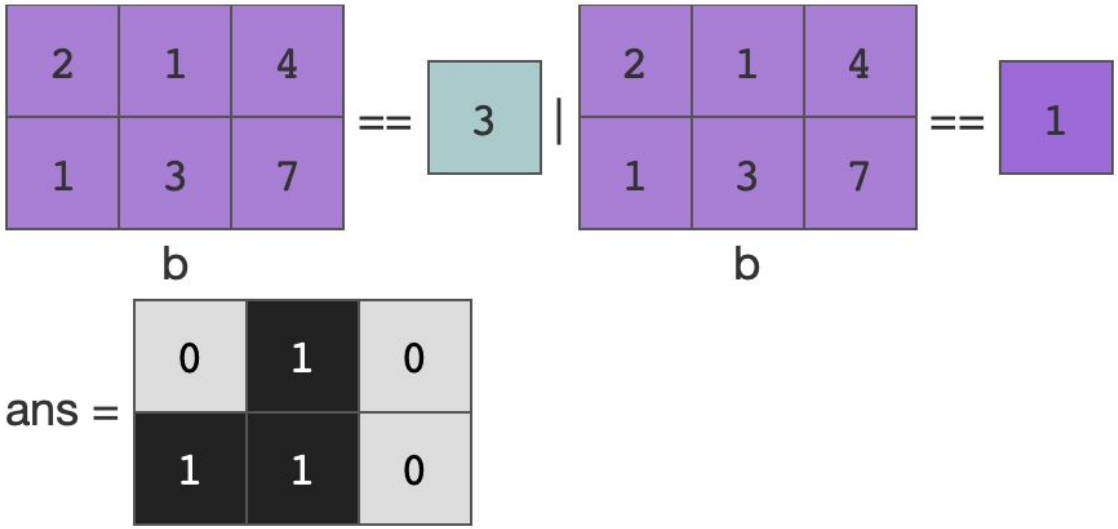
c

1

d

1	7
2	5

b == 3 | b == 1



Logical Operations with Matrices

a

3	4	2
6	5	1

b

2	1	4
1	3	7

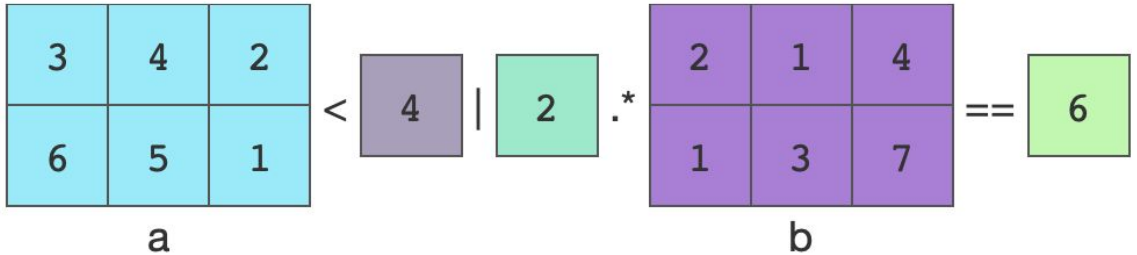
c

1

d

1	7
2	5

```
a < 4 | 2 .* b == 6
```



ans =

1	0	1
0	1	1

Logical Operations with Matrices

a

3	4	2
6	5	1

b

2	1	4
1	3	7

c

1

d

1	7
2	5

d + 3 == 10 & c == 1

1	7	+	3	==	10	&	1	==	1
2	5						c		

d

ans =

0	1
0	0

Logical Operations with Matrices

a

3	4	2
6	5	1

b

2	1	4
1	3	7

c

1

d

1	7
2	5

```
a(:,2) == b(:,3) & a(:,1 ) < b(:,1 )
```

3	4	2
6	5	1

a

==

2	1	4
1	3	7

b

&

3	4	2
6	5	1

a

<

2	1	4
1	3	7

b

ans =

0
0

Logical Operations with Matrices

a

3	4	2
6	5	1

b

2	1	4
1	3	7

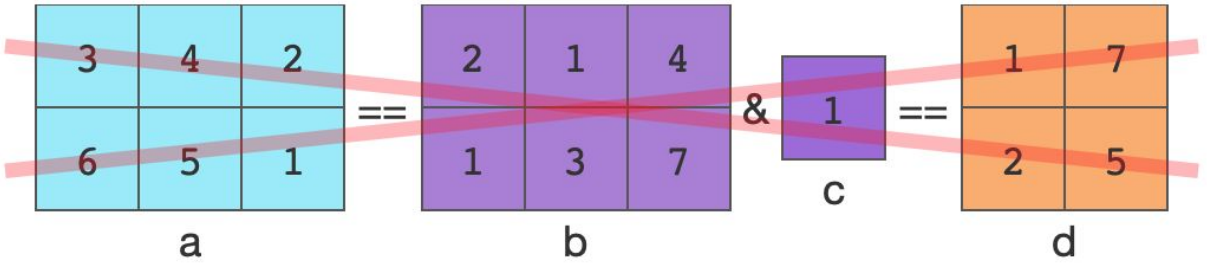
c

1

d

1	7
2	5

a == b & c == d



Mismatched dimensions for operator undefined. LHS is a 2x3 and RHS is a 2x2.

Careful!

10

x

- Let's say we wanted to check whether a number is in a particular interval. It's tempting to try something like this:

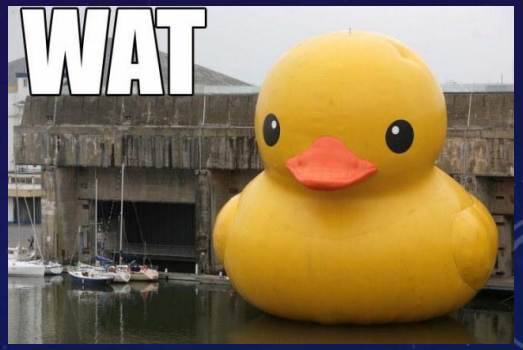
```
1 < x < 5
```

MATLAB claims this is true.

- MATLAB sees this as:

```
(1 < x) < 5
(1 < 10) < 5
(1) < 5
1
```

`1 < 10` is true, so it evaluates to 1.



- Instead, use & to specify the lower and upper bounds separately:

```
1 < x & x < 5
```

Operator Precedence

Higher (happens first)

Lower (happens last)

Operation Type	Operators					
Parentheses	( )					
Transpose / Power	.'	'	.^	^		
Unary	+	-	~			
Multiplicative	.*	*	./	/		
Additive	+	-				
Range Construction	:					
Relational	<	<=	>	>=	==	~=
Logical AND	&					
Logical OR	|					
Assignment	=					

Supplemental Exercise: Logical Operations

- Given these variables:

- Find the result of the following five expressions:

```
a ~= b
```

```
b - a == c + 3
```

```
a > c
```

```
a(2,:) == 3 .* ones(1,3)
```

```
a + 1 > 3 & b > 2
```

Remember, Lobster doesn't support MATLAB functions (yet….)

Solution: Logical Operations

- Given these variables:

1	2	4
3	3	2

a

0	2	9
3	3	6

b

2

c

- Find the result of the following expressions:

`a ~= b`

1	0	1
0	0	1

`b - a == c + 3`

0	0	1
0	0	0

`a > c`

0	0	1
1	1	0

`a(2,:) == 3 .* ones(1,3)`

1	1	0

`a + 1 > 3 & b > 2`

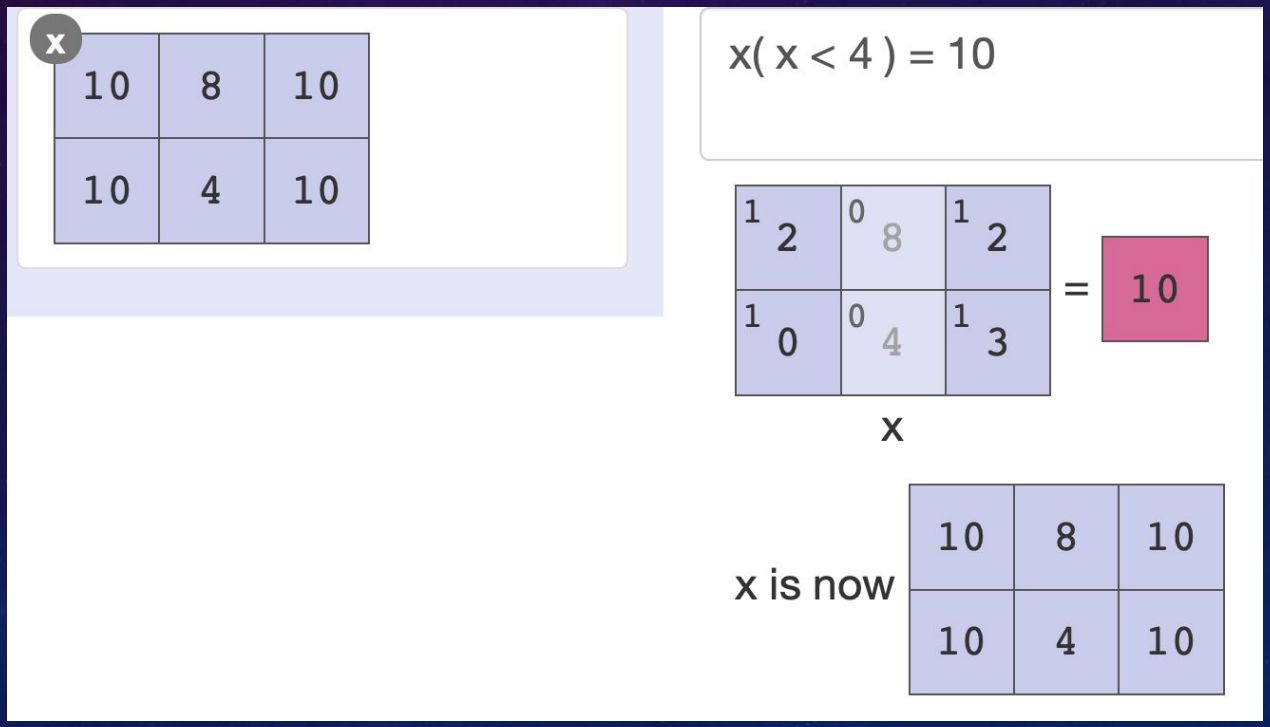
0	0	1
1	1	0

Break Time

We'll start again in 5 minutes.

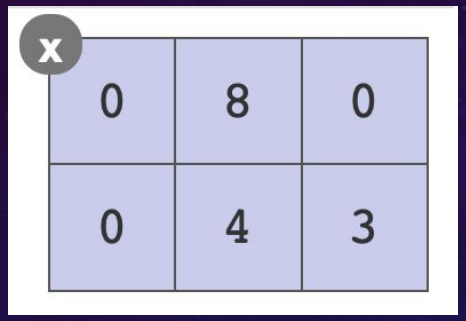
Writing Into a Logically Indexed Matrix

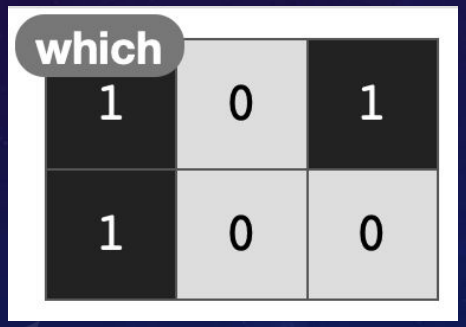
- A logical indexing expression can be assigned into.
- The RHS must either be a scalar or have the same number of elements as there are 1s in the logical index matrix.

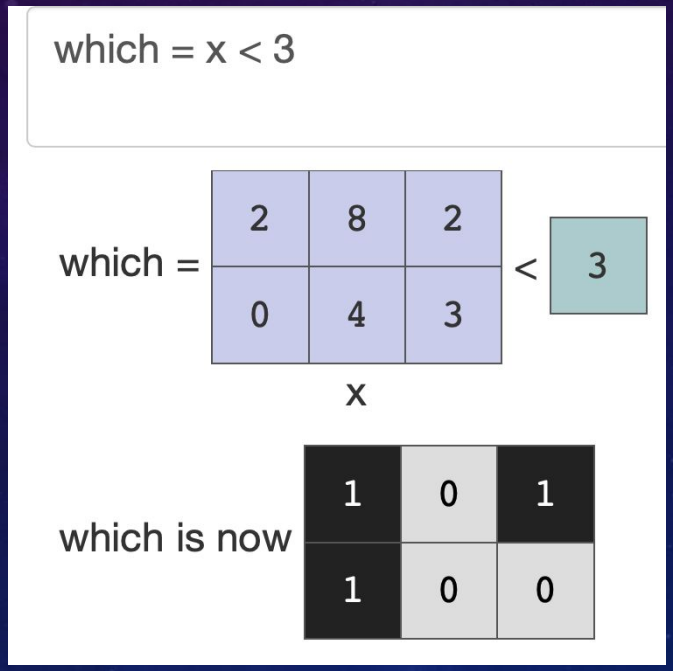


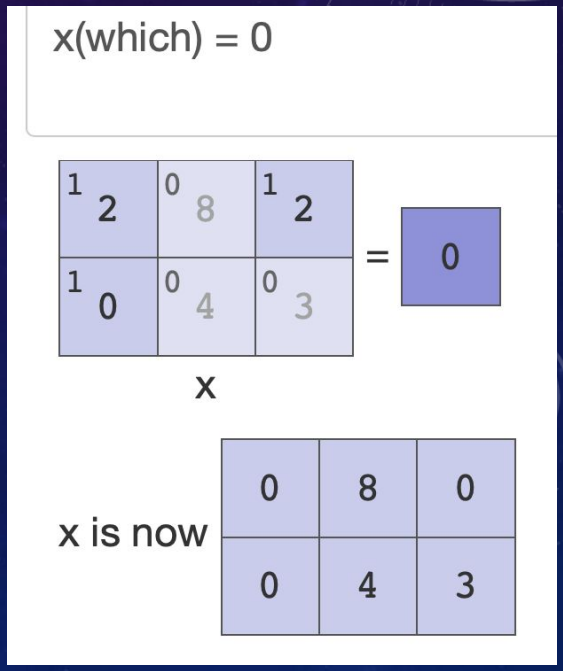
Writing Into a Logically Indexed Matrix

”set elements less than 3 to be zero”









Writing Into a Logically Indexed Matrix

”set all elements except values 2 and 4 to be -1”

x

2	−1	2
−1	4	−1

x(x ~= 2 & x ~= 4) = -1

0 2	1 8	0 2
1 0	0 4	1 3

= −1

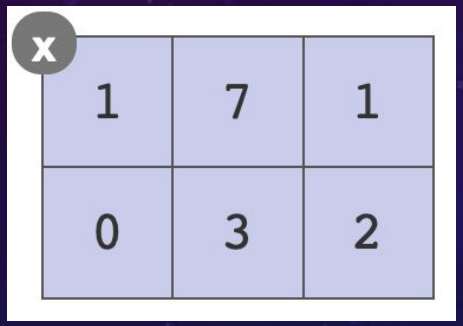
x

x is now

2	−1	2
−1	4	−1

Writing Into a Logically Indexed Matrix

”subtract one from all elements, but don't go below zero”



$x(x > 0) = x(x > 0) - 1$

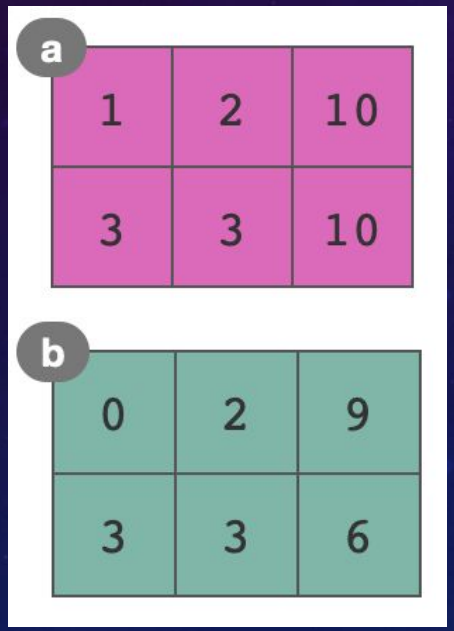
$$\begin{bmatrix} 12 & 18 & 12 \\ 00 & 14 & 13 \end{bmatrix}_{x} = \begin{bmatrix} 12 & 18 & 12 \\ 00 & 14 & 13 \end{bmatrix}_{x} - 1$$

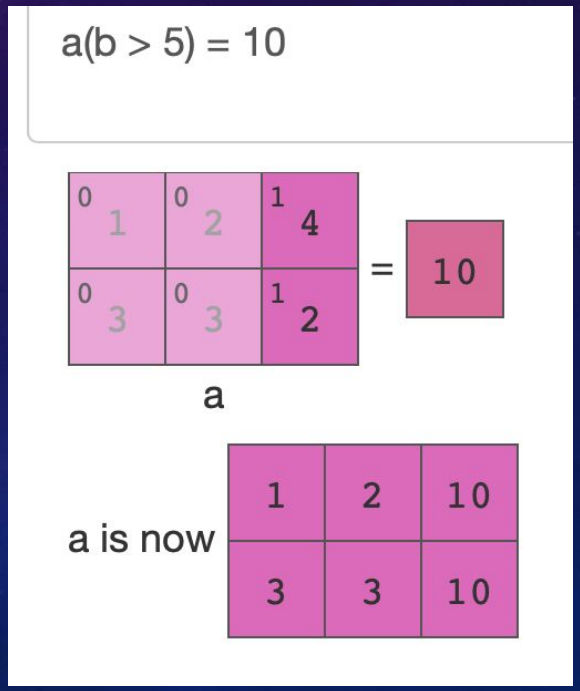
x is now

1	7	1
0	3	2

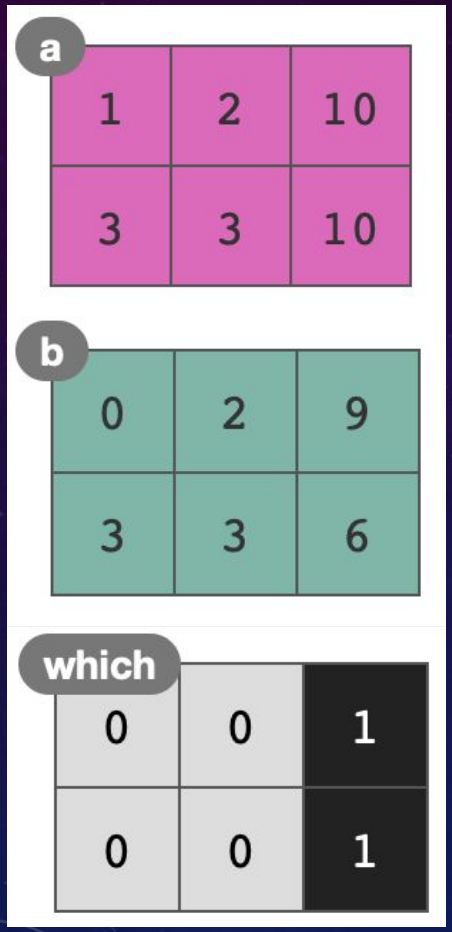
Logical Indexing with Parallel Matrices

- You can use a logical matrix derived from a different matrix than the source into which you are indexing.
- The two "parallel" matrices should be the same size.

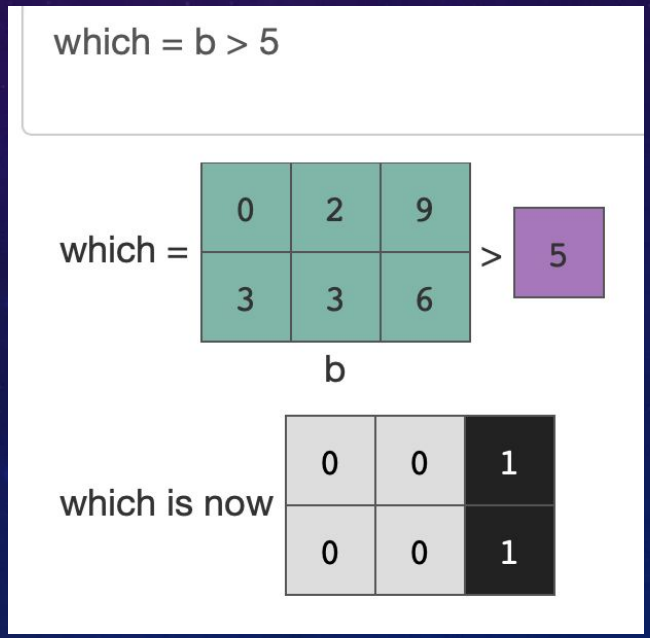


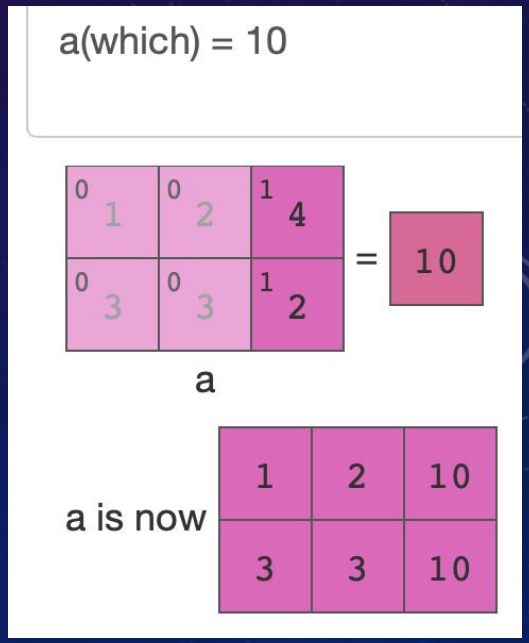


Logical Indexing with Parallel Matrices



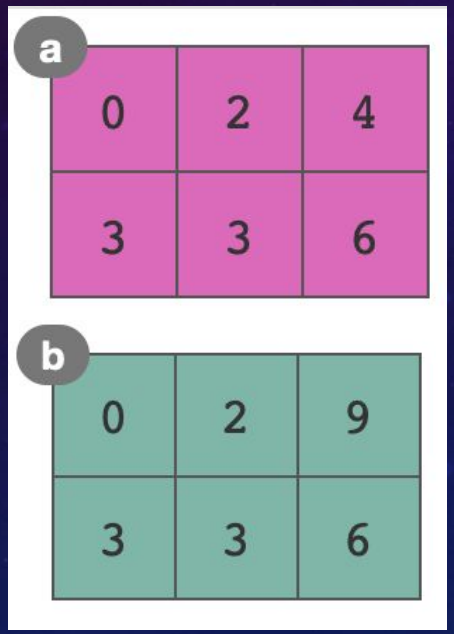
As before, you can separate this process into two steps (either way is fine!)

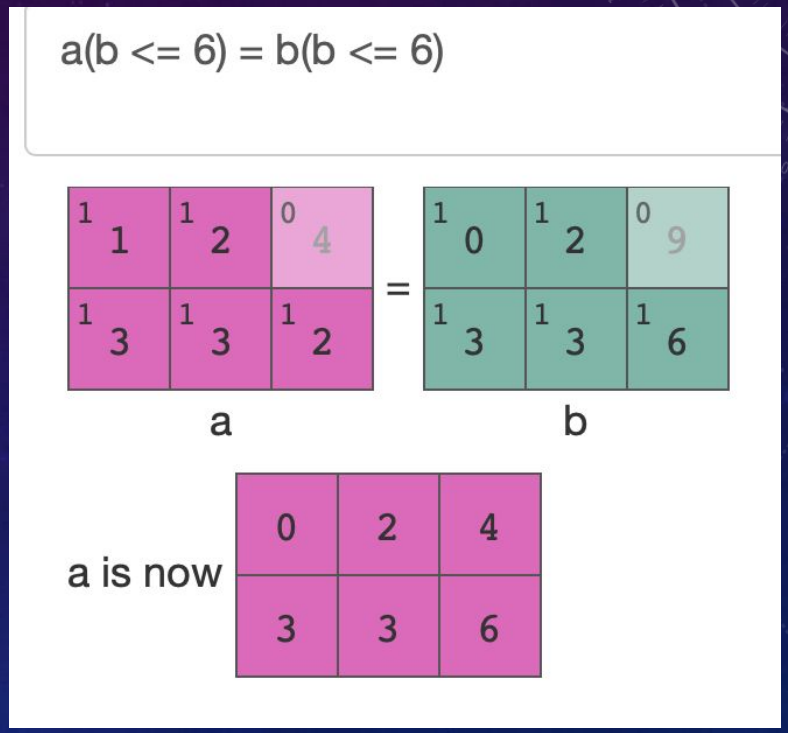




Logical Indexing with Parallel Matrices

Find the elements in the ‘b’ matrix that are less than or equal to 6, and put those elements in the corresponding locations in the ‘a’ matrix





Combining Logical Indexing with Row/Column Indexing

Example:
select only the columns that have at least one negative number

x

1	5	-9	8
2	2	10	7
-4	3	-2	0
6	5	4	4

```
loc = x < 0
```

0	0	1	0
0	0	0	0
1	0	1	0
0	0	0	0

```
howMany = sum(loc)
```

1	0	2	0

```
x(:, howMany >= 1)
```

1	-9
2	10
-4	-2
6	4

Algorithm:

1. Locate negative #s
2. Count how many negative #s are in each column
3. Keep only the columns that have more than one negative #