

# MATLAB

Class 1: Basics part 1

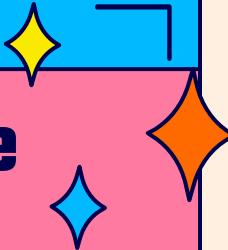


# Goals of this MATLAB course

Taking you from **MATDRAB** to **MATFAB**



- ❖ By the end of the course, you should be comfortable reading MATLAB code, writing basic scripts, plotting data, and performing basic analyses



# The philosophy of a programming language



What language do computers speak?  
Is it like Portuguese? French?

How are computer languages  
different from English languages?

What is an algorithm?



# MATLAB introduction

Proprietary programming language

Interpreted language

User friendly

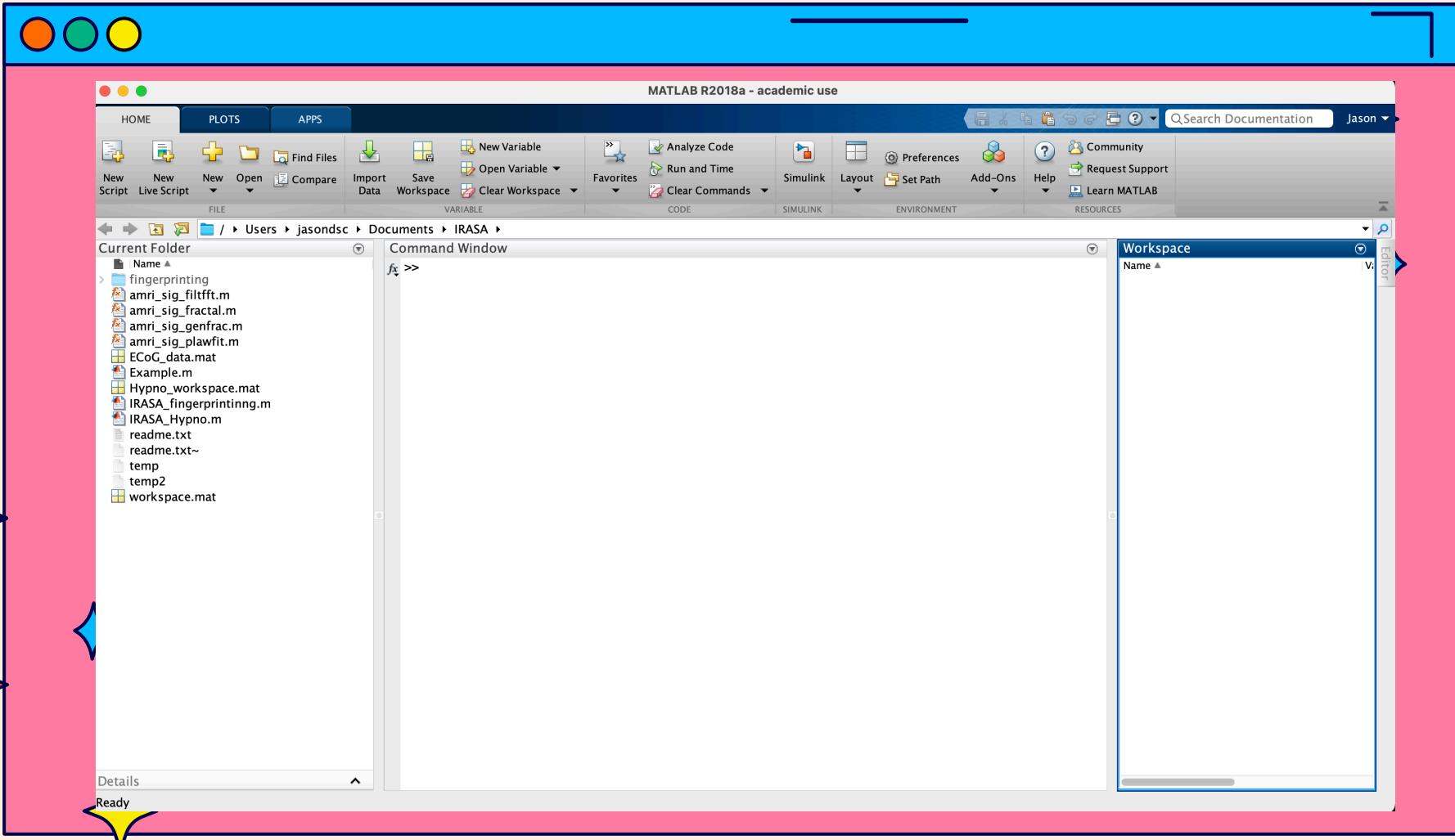
Many toolboxes and addons

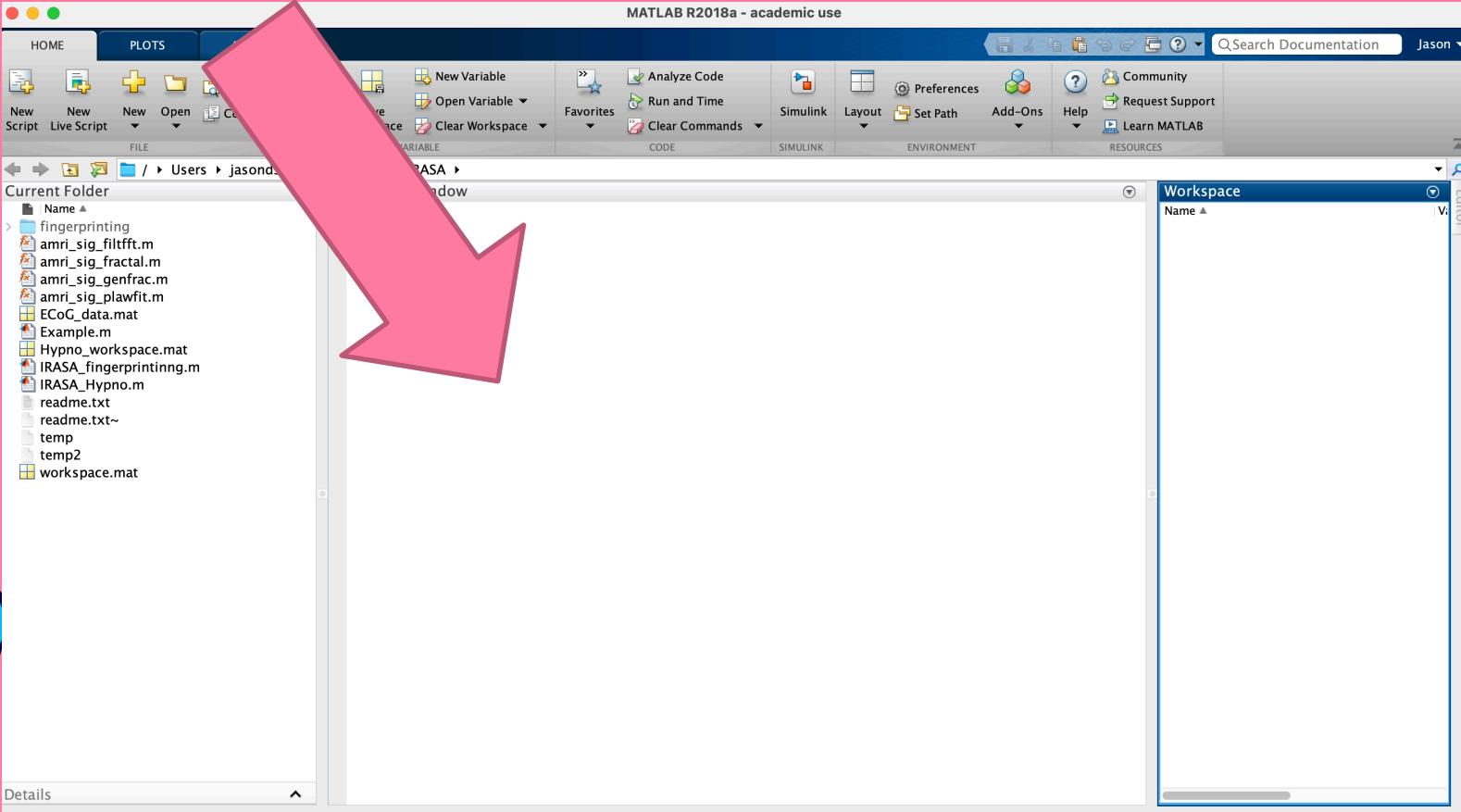
Designed for matrix operations, signal processing,  
equation solving

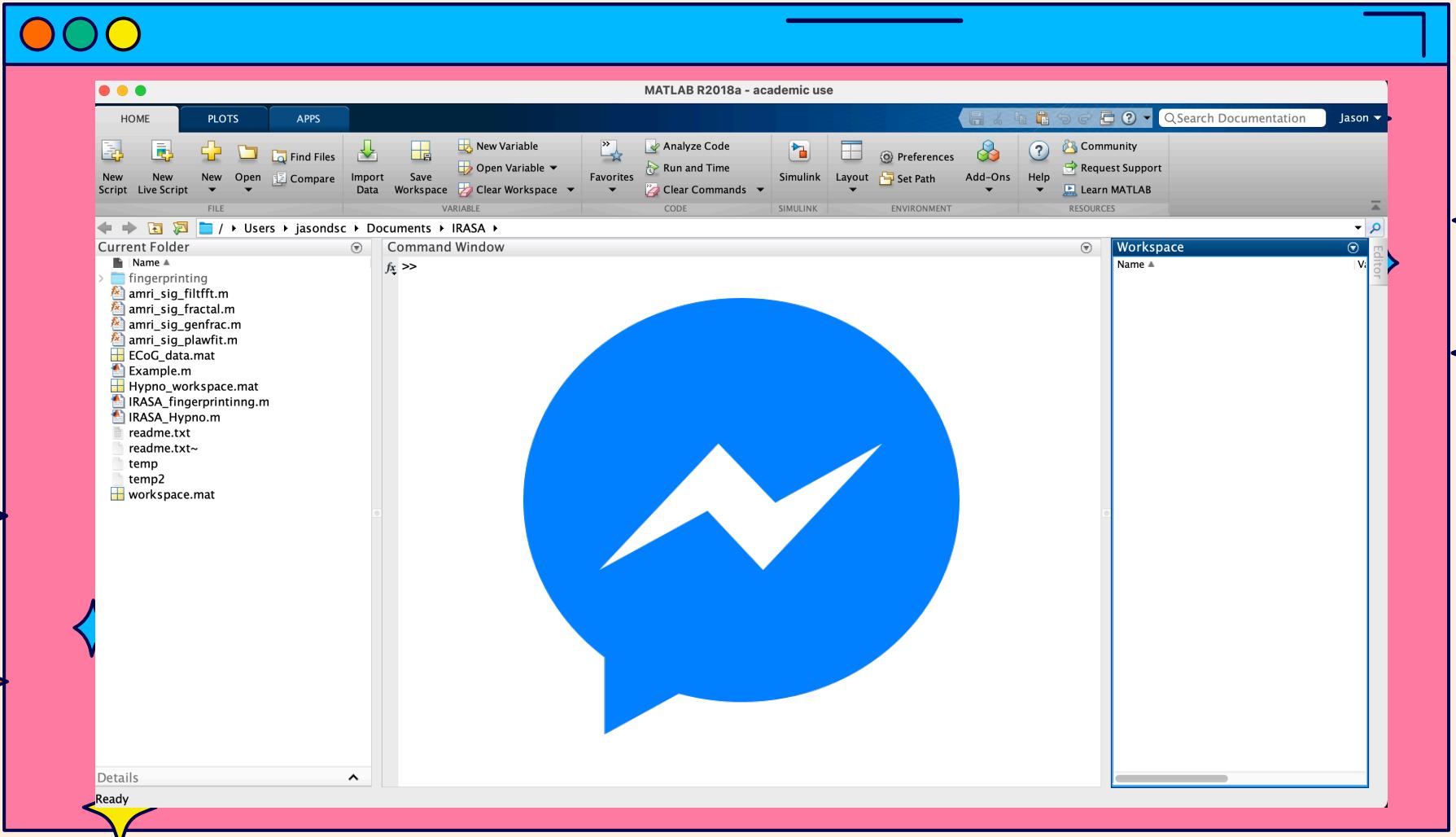
# MATLAB Environment

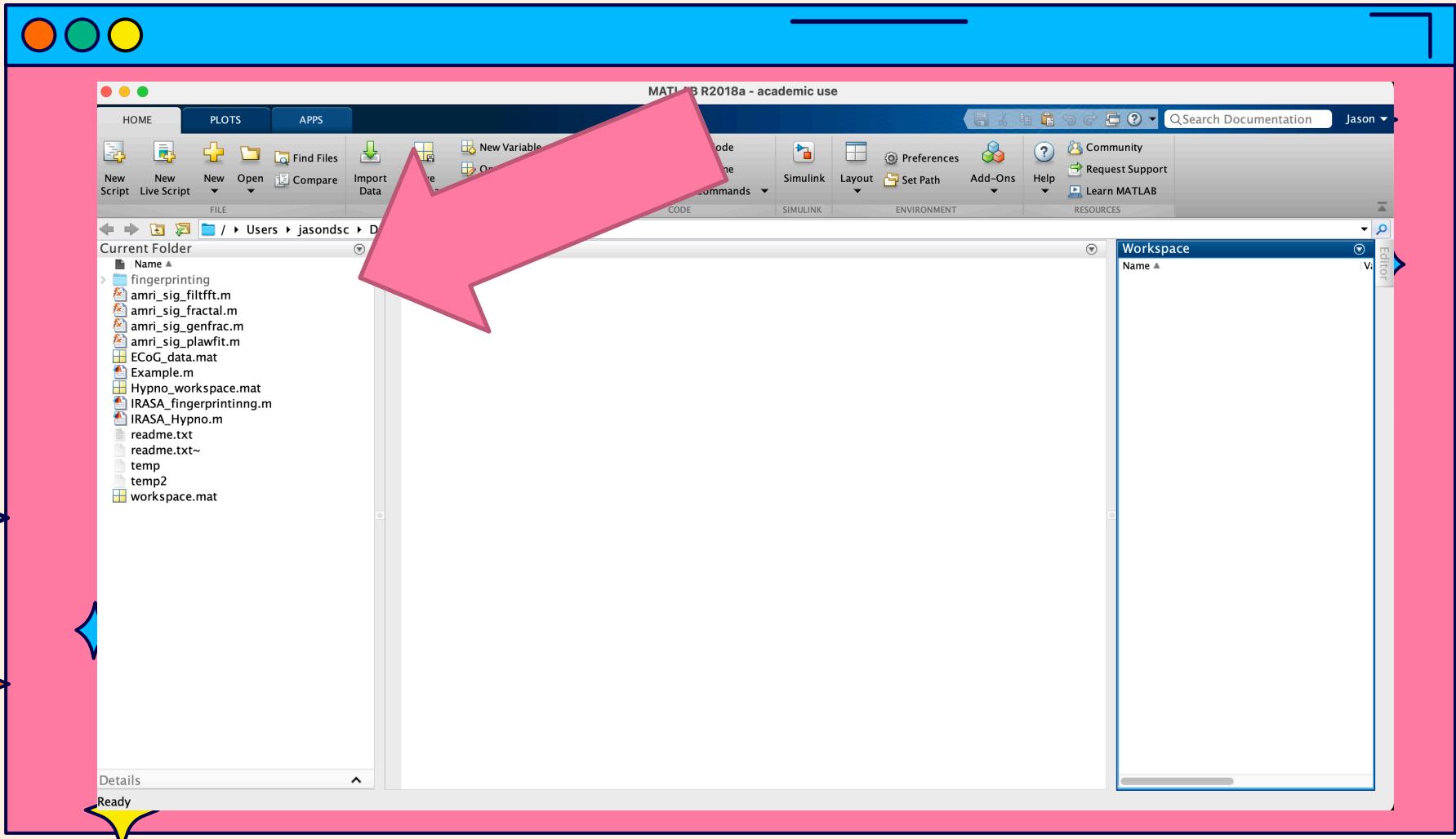


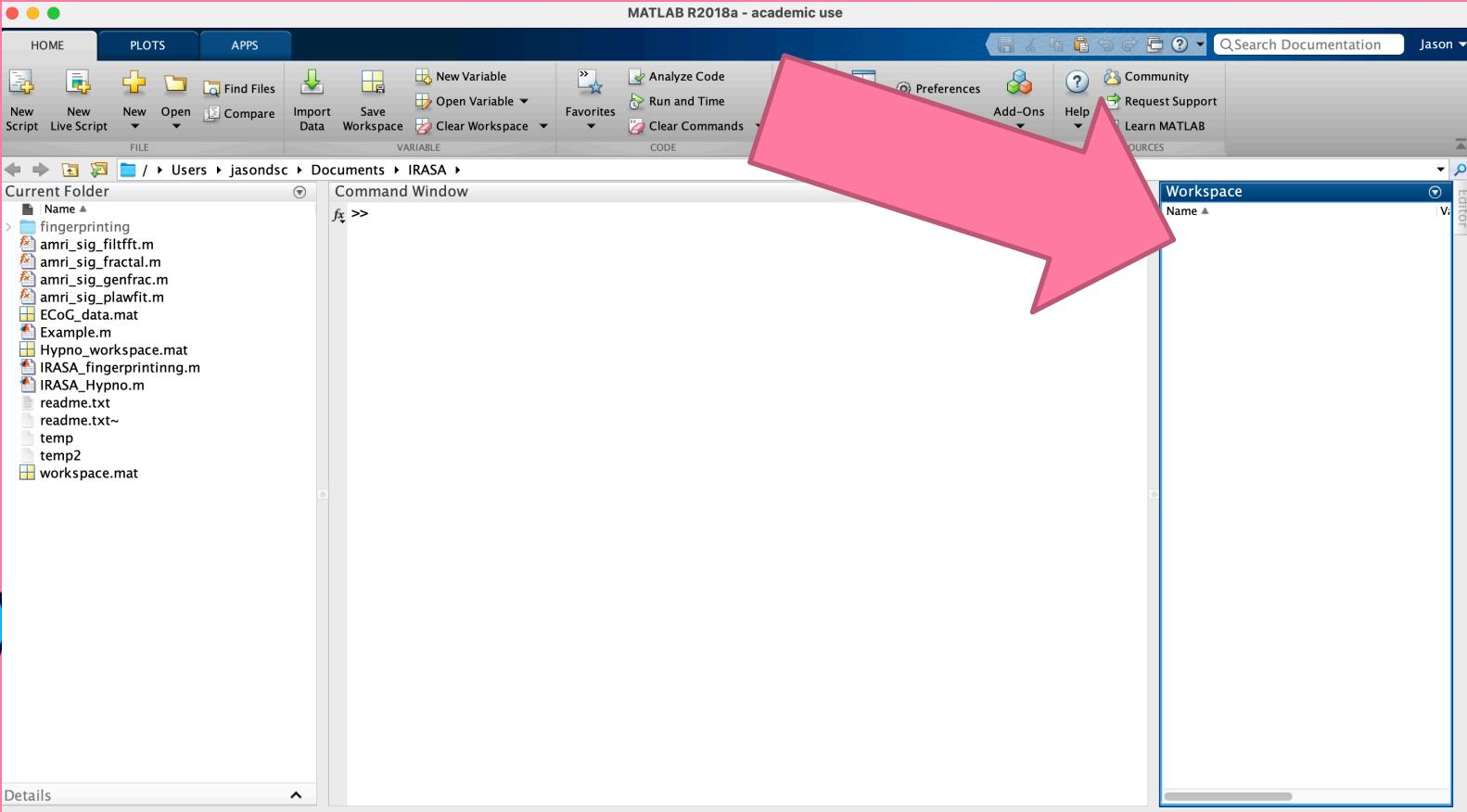
- ◆ Command Window
- ◆ Current Folder
- ◆ Workspace













# Basic MATLAB Commands

Commands to clean your environment **clear**, **clc** and  
**close**

To find what files are in a directory **dir()**

To find what directory you are in or change it:  
**pwd()** and **cd()**

To open and write .mat files **load()** and **save()**



# Basic MATLAB Commands

To check if a variable or file exists: **exist()**

To list what MATLAB .m and .mat files are in your folder: **what()**

To find out which version of a function you are using is: **which()**



# Functions

Functions are called by using (), their outputs and inputs can vary

Example calls of functions:

`max(a);`

`figure or figure()`

`[maxA, location] = max(A);`

`[~, name, ext] = fileparts( helpfile );`

`coeff = pca(X(:,3:15), 'Rows', 'pairwise');`



I don't know her.



# Help Function

When you are unclear on what a function does, takes as inputs, or outputs you can always ask for **HELP**

The function `help` (followed by a function name) returns a description of that function. For details read more about each function on [MATLAB's website](#)



# Help Example

**>> help fileparts**

fileparts Filename parts.

[FILEPATH, NAME, EXT] = fileparts(FILE) returns the path, file name, and file name extension for the specified FILE. The FILE input is the name of a file or folder, and can include a path and file name extension. The function interprets all characters following the right-most path delimiter as a file name plus extension.

If the FILE input consists of a folder name only, be sure that the right-most character is a path delimiter (/ or \). Otherwise, fileparts parses the trailing portion of FILE as the name of a file and returns it in NAME instead of in FILEPATH.



# What is a variable

A variable is assigning a location in memory some information you want to keep for later

**variable\_a = 514,398,6644**

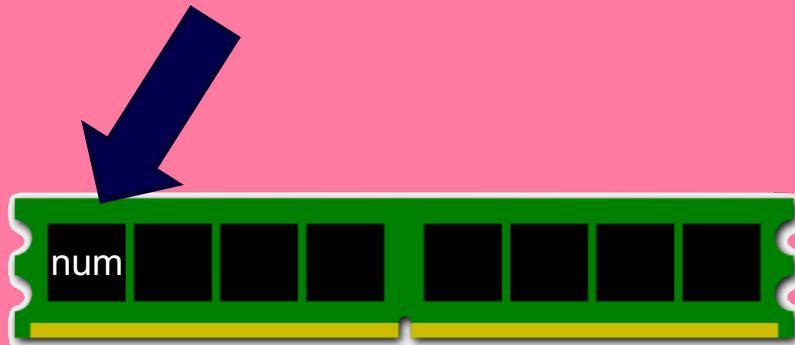




# What is a variable

A variable is assigning a location in memory some information you want to keep for later

**num = 514,398,6644**



# Data Types

**Booleans**

---

Variables  
that are  
either  
TRUE or  
FALSE

**Char**

---

Variables  
that  
represent  
words or  
text

**Integers**

---

Variables  
that  
represent  
numbers



**YOU CAN'T SIT WITH US**

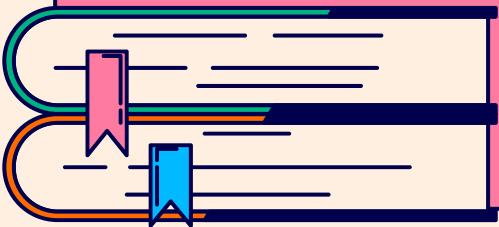


# Booleans

Variable that can only take two states: **TRUE** or **FALSE**

Used in accessing information, loops, and conditional statements

They are the computer's equivalent to a question





# Boolean Operators

How can we ask the computer a question:

Is equal to ==

Is greater than >

Is less than <

Is NOT equal to ~=

The OR operator ||

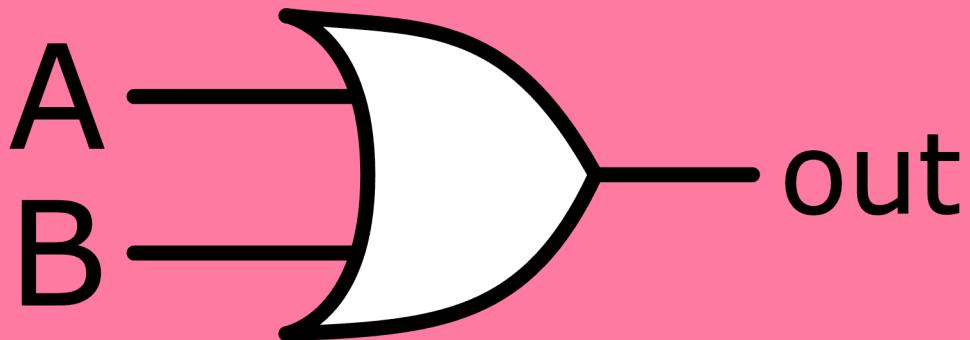
The AND operator &&





# Boolean Operators

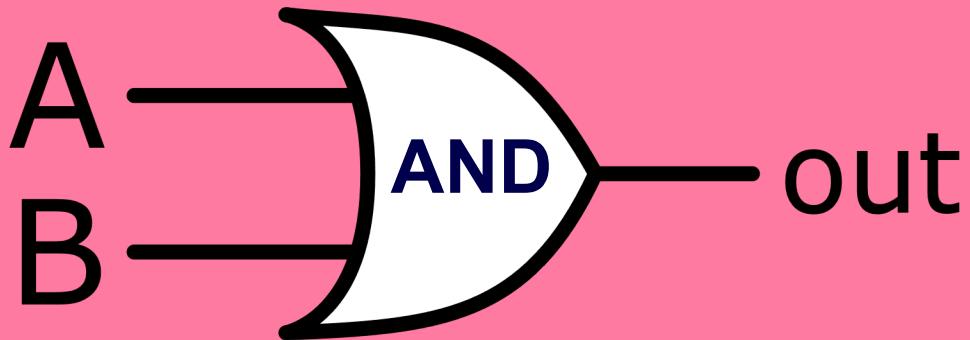
Two very special operators: **AND** and **OR**





# Boolean Operators

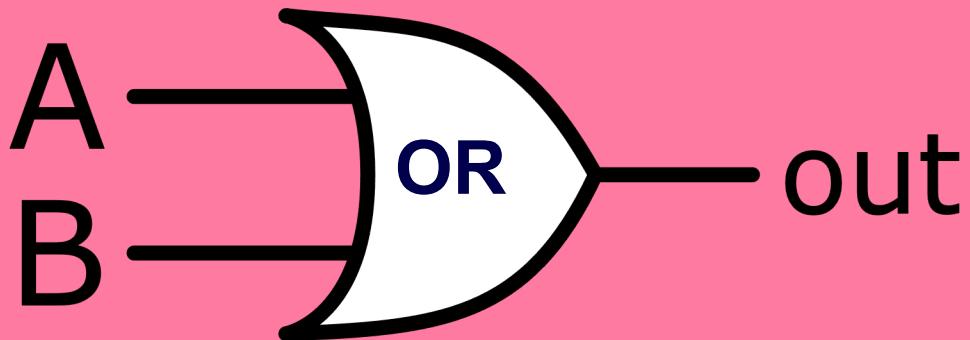
Two very special operators: **AND** and **OR**





# Boolean Operators

Two very special operators: **AND** and **OR**





# Integers

Variable type to hold numeric information:

◆ **int(8-64)** – real integer value

◆ **double** – decimal number

◆ **unit8/16/32** – stores info in 1, 2 etc bytes

◆ **complex numbers** – numbers with real and  
imaginary values



# Char and Strings

Holds information concerning words/ text

**char** – single or array of alpha-numeric symbols ''

**string** – array of text ""

For most purposes it does not matter which you use



# Char and Strings differences

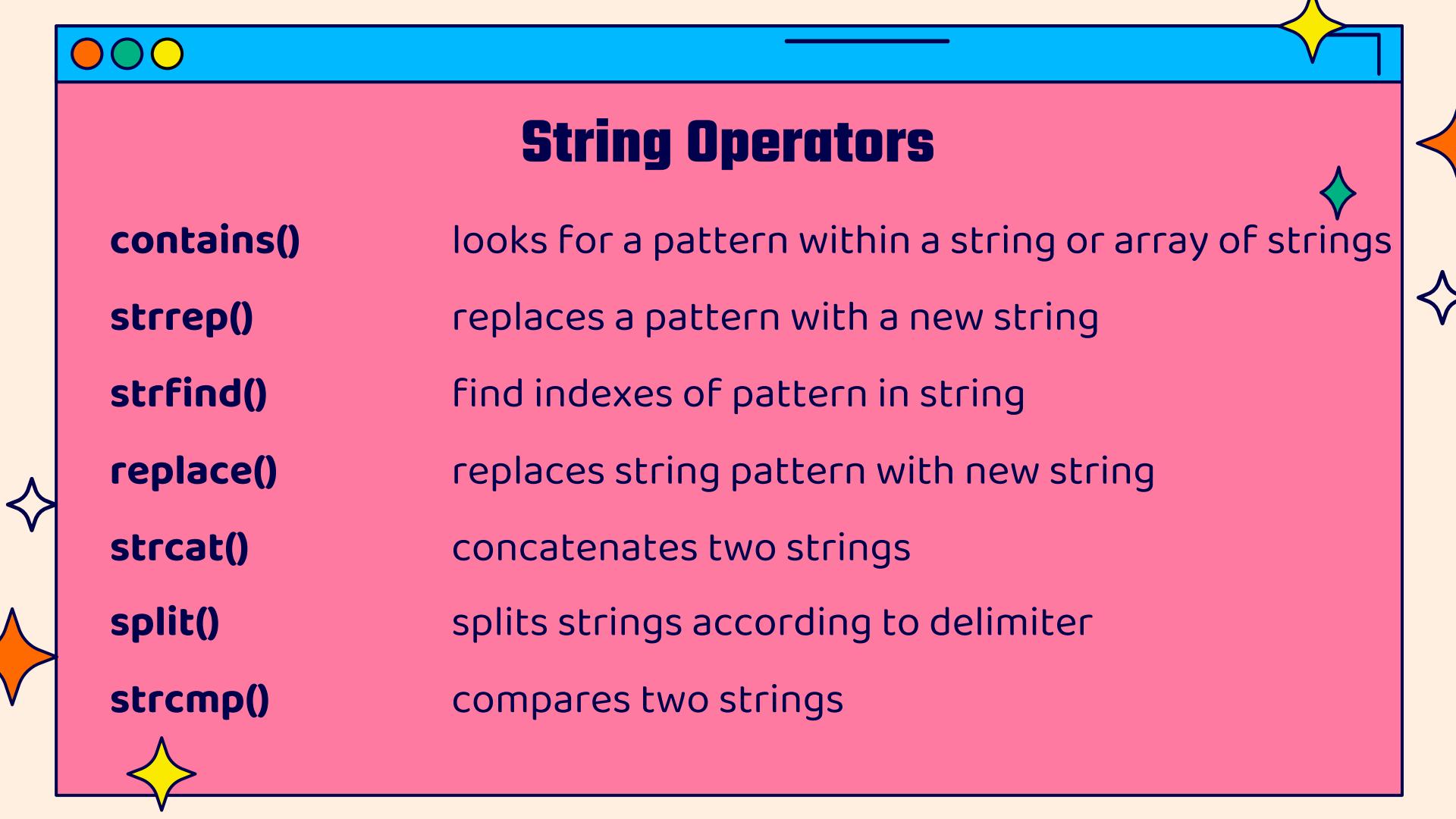
```
secret_message = ['a'; 'ab'; 'abc']
```

```
secret_message = ["a"; "ab"; "abc"]
```

```
word= 'abc'; word(2)
```

```
word= "abc"; word{1}(2)
```

\*\* these differences have to do with how char and  
strings are stored in memory



# String Operators

**contains()**

looks for a pattern within a string or array of strings

**strrep()**

replaces a pattern with a new string

**strfind()**

find indexes of pattern in string

**replace()**

replaces string pattern with new string

**strcat()**

concatenates two strings

**split()**

splits strings according to delimiter

**strcmp()**

compares two strings

# Data Structures

## Arrays

One dimensional list of variables of the **SAME** type

## Matrices

Two-dimensional matrix of variables of the **SAME** type

## Multi-dimensional Matrix

Like a matrix but with 3+ dimensions



# Matrix Operations

Defining arrays and matrices:

**ALL VARIABLES MUST BE OF THE SAME TYPE**

```
a= [1 2 3; 4 5 6; 7 8 9]
```

```
words= ['abc'; 'def'; 'ghi']
```

```
z = zeros(5,1)
```

```
n=ones(10,5);
```



# Matrix Operations

Basic matrix operations:

`n + 10` (adds 10 to each value in the matrix)

`a *2` (multiply each element of matrix by 2)

`a'` (transpose of a matrix)

`a.*a` (element wise multiplication)

`n*z` (matrix multiplication resulting in 10 by 1 matrix)



# Matrix Operations

Basic matrix operations:

It is important to note that operators that have a ''  
ahead preform ELEMENTWISE operations so

$a.*b \sim= a^*b$





# Matrix Operations

Array and Matrix Indexing

```
a=[1 2 3; 4 5 6; 7 8 9];
```

Get the first element:  $a(1,1)$

Get second element of first row:  $a(1,2)$

Get first row:  $a(1, :)$

Get first column:  $a(:, 1)$

Get first three elements of the second row:  $a(2, 1:3)$

Note: you can index by multiples e.g. 1:2:100



# Matrix Operations

Matrix manipulation:

**zeros**: create a matrix of all zeros

**ones**: create a matrix of all ones

**eye**: create identify matrix

**size**: returns matrix dimensions

**eig**: evaluates eigenvalues and eigenvectors

**det**: returns determinant of matrix

**inv**: returns inverse of matrix



# I/O introduction

As we have seen we can load .mat data files with **load()**

For other types of data we can use MATLAB's import tools

Simple text files can be read with **csvread()** or **tdfread()** these  
only accept numeric values. Refer to **readtable()** for strings or  
mixed type .csv files

\*\* note that MATLAB after 2019 recommends **readmatrix()**



# RESOURCES

- ◆ <https://www.mathworks.com/>
  - ◆ <https://stackoverflow.com>
  - ◆ Mastering MATLAB
  - ◆ Analyzing Neural Time Series Data
- 