FUNCTIONS

functions are like recipes for things you want to do lots of times but might have different values each time – for example, you might want to print text to your screen without writing how to convert inputs to pixels each time.

similar to writing a recipe for pizza – you might change the toppings or use a different kind of flour or sauce, but you will take the same steps with what you're given!

FUNCTIONS

It is important to note that functions and in general matlab code is sequential. So like recipes is important to follow a certain order. If you don’t follow the order it is possible that you end up burning the recipe or with buggy code.

you decide what your function accepts!

```
>> 'hello'+'world'

ans =

   223    212    222    216    211
```

unexpected but successful!

```
>> 'hi'+'world'
Arrays have incompatible sizes for this operation.

Related documentation
```

code fails!

FUNCTION NOTATION: Inside the function

```
function [result1, result2] = myfuncname(input1, input2)

% myfuncname : last updated haley keglovits 01.25.22
% accepts:
%       input1: (integer/double)
%       input2: (integer/double)
% returns:
%       result1: (float) the sum of input1 and input2
%       result2: (boolean) whether (input1)^2 is larger than input2

result1 = input1 + input2;
input1_squared = input1^2;
result2 = input1_squared > input2;

end
```

FUNCTION NOTATION: Inside the function

```
function [result1, result2] = myfuncname(input1, input2)

% myfuncname : last updated haley keglovits 01.25.22
% accepts:
%       input1: (integer/double)
%       input2: (integer/double)
% returns:
%       result1: (float) the sum of input1 and input2
%       result2: (boolean) whether (input1)^2 is larger than input2

result1 = input1 + input2;
input1_squared = input1^2;
result2 = input1_squared > input2;

end
```

body

FUNCTION NOTATION: Inside the function

```
function [result1, result2] = myfuncname(input1, input2)

% myfuncname : last updated haley keglovits 01.25.22
% accepts:
%       input1: (integer/double)
%       input2: (integer/double)
% returns:
%       result1: (float) the sum of input1 and input2
%       result2: (boolean) whether (input1)^2 is larger than input2

result1 = input1 + input2;
input1_squared = input1^2;
result2 = input1_squared > input2;

end
```

argument(s)
(aka parameters)

FUNCTION NOTATION: Inside the function

```
>> upper('hello')

ans =

    'HELLO'
```

not all functions have multiple arguments or results! what does this function do?

FUNCTION NOTATION: Inside the function

```
function [result1, result2] = myfuncname(input1, input2)

% myfuncname : last updated haley kegiovits 01.25.22
% accepts:
%       input1: (integer/double)
%       input2: (integer/double)
% returns:
%       result1: (float) the sum of input1 and input2
%       result2: (boolean) whether (input1)^2 is larger than input2

result1 = input1 + input2;
input1_squared = input1^2;
result2 = input1_squared > input2;

end
```

output(s)

FUNCTION NOTATION: Inside the function

```
function [result1, result2] = myfuncname(input1, input2)

% myfuncname : last updated haley keglovits 01.25.22
% accepts:
%       input1: (integer/double)
%       input2: (integer/double)
% returns:
%       result1: (float) the sum of input1 and input2
%       result2: (boolean) whether (input1)^2 is larger than input2

result1 = input1 + input2;
input1_squared = input1^2;
result2 = input1_squared > input2;

end
```

function definition and end

FUNCTION NOTATION: Inside the function

```
function [result1, result2] = myfuncname(input1, input2)

% myfuncname : last updated haley keglovits 01.25.22
% accepts:
%       input1: (integer/double)
%       input2: (integer/double)
% returns:
%       result1: (float) the sum of input1 and input2
%       result2: (boolean) whether (input1)^2 is larger than input2

result1 = input1 + input2;
input1_squared = input1^2;
result2 = input1_squared > input2;

end
```

comments

FUNCTION NOTATION: Inside the function

```
1   function [result1, result2] = myfuncname(input1, input2)
2
3   % myfuncname : last updated haley keglovits 01.25.22
4   % accepts:
5   %       input1: (integer/double)
6   %       input2: (integer/double)
7   % returns:
8   %       result1: (float) the sum of input1 and input2
9   %       result2: (boolean) whether (input1)^2 is larger than input2
10
11  result1 = input1 + input2;
12  input1_squared = input1^2;
13  result2 = input1_squared > input2;
14
15  end
```

line numbers

FUNCTION NOTATION: Outside the function

```
>> myfuncname(1,2)

ans =

     3
```

```
>> [r1,r2] = myfuncname(1,2)

r1 =

     3


r2 =

  logical

   0
```

```
>> r1,r2 = myfuncname(1,2)
Unrecognized function or variable 'r1'.
```

```
>> [egg,salad] = myfuncname(pizza,sauce)

egg =

     3


salad =

  logical

   0
```

SCOPE

functions have a scope that is different from the scope of your whole program!

scope means “definition of variables”

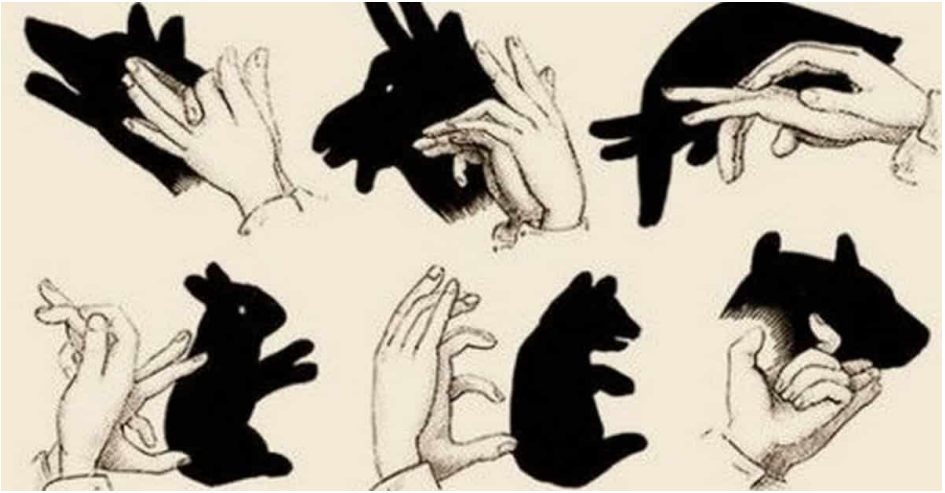
why would we want this?

example: pizza recipe

if you are making a pizza sauce and dough, and read the instruction “knead the mixture,” how could this be confusing if you are following both recipes?

why would it be more helpful to see “topping” instead of “onion”?

SCOPE



scope is like what allows us to watch a puppet show and focus on the characters and story (a fox and a bird) and not on how they are made (paper, hands, etc)

outside and inside the function work with different information!

SCOPE

```
function [result1, result2] = myfuncname(input1, input2)

% myfuncname : last updated haley keglovits 01.25.22
% accepts:
%       input1: (integer/double)
%       input2: (integer/double)
% returns:
%       result1: (float) the sum of input1 and input2
%       result2: (boolean) whether (input1)^2 is larger than input2

result1 = input1 + input2;
input1_squared = input1^2;
result2 = input1_squared > input2;

end
```

```
>> [r1,r2] = myfuncname(2,5)

r1 =

     7


r2 =

  logical

   0

>> input1_squared
Unrecognized function or variable 'input1_squared'.
```

any variables you define inside your function are in **local** scope of that function – they live and die when that function is being used

if you don't return a value, it is lost forever!

goes both ways! you need to pass information to a function for it to use it

SCOPE

```
function [result1, result2] = myfuncname(input1, input2)

% myfuncname : last updated haley keglovits 01.25.22
% accepts:
%       input1: (integer/double)
%       input2: (integer/double)
% returns:
%       result1: (float) the sum of input1 and input2
%       result2: (boolean) whether (input1)^2 is larger than input2

result1 = input1 + input2;
input1_squared = input1^2;
result2 = input1_squared > input2;

end
```

```
>> input1 = 100;
>> input2 = 5;
>> myfuncname(input2,input1)

ans =

   105

>> [r1,r2] = myfuncname(input2,input1)

r1 =

   105


r2 =

  logical

   0
```

variables get “renamed” inside your function when they are passed into specific input parameters.

in a recipe, it will just call something ‘topping’ even if you know that your topping is spinach

common mistake! don’t assume your names will hold inside the function. it only knows the **values** it is passed, not their names

why is R2 false? 100^2 is larger than 5.

SCOPE

at first, this might seem confusing. but it's very helpful to not have to know all the guts and variables in every function you use.

if you did, you couldn't name any variables what any other functions have used... it would be impossible to code!

this is also why you can pass variables or values to a function and it will work both ways

```
>> input1 = 2; input2 = 3;
>> [r1,r2] = myfuncname(input1,input2)

r1 =

     5


r2 =

  logical

   1

>> [r1,r2] = myfuncname(2,3)

r1 =

     5


r2 =

  logical

   1
```

SCOPE

area of caution: changing values of a variable

remember – inside the function does not know what outside names are

```
function [variable1] = newfunc(variable1)

% last edit haley keglovits 1.26.22
% this function adds 10 to any number passed to it.
% accepts: variable1, a number
% returns, variable1, a number.

variable1 = variable1 + 10;

end
```

```
>> variable1 = 5;
>> newfunc(variable1)

ans =

    15

>> variable1

variable1 =

     5
```

Riddle: Mystery Function

what does this function do? **a** and **b** should be **integers**
(you all know this function, you used it in assignments this week!)
try picking two integers and see what happens when you write it down / code it yourself!

```
function c = mystery(a, b)
% a function that calls itself!
% an example of recursion... we won't cover in this class
    if b == 1
        c = a;
    else
        c = a + mystery(a, b-1);
    end
end
```