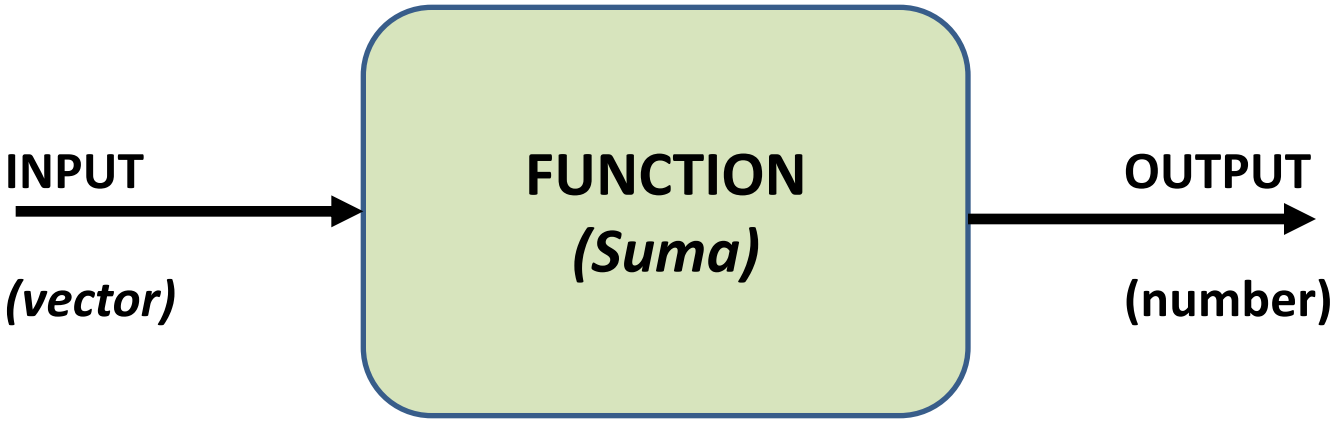
Programming with Matlab

Day 3: Functions

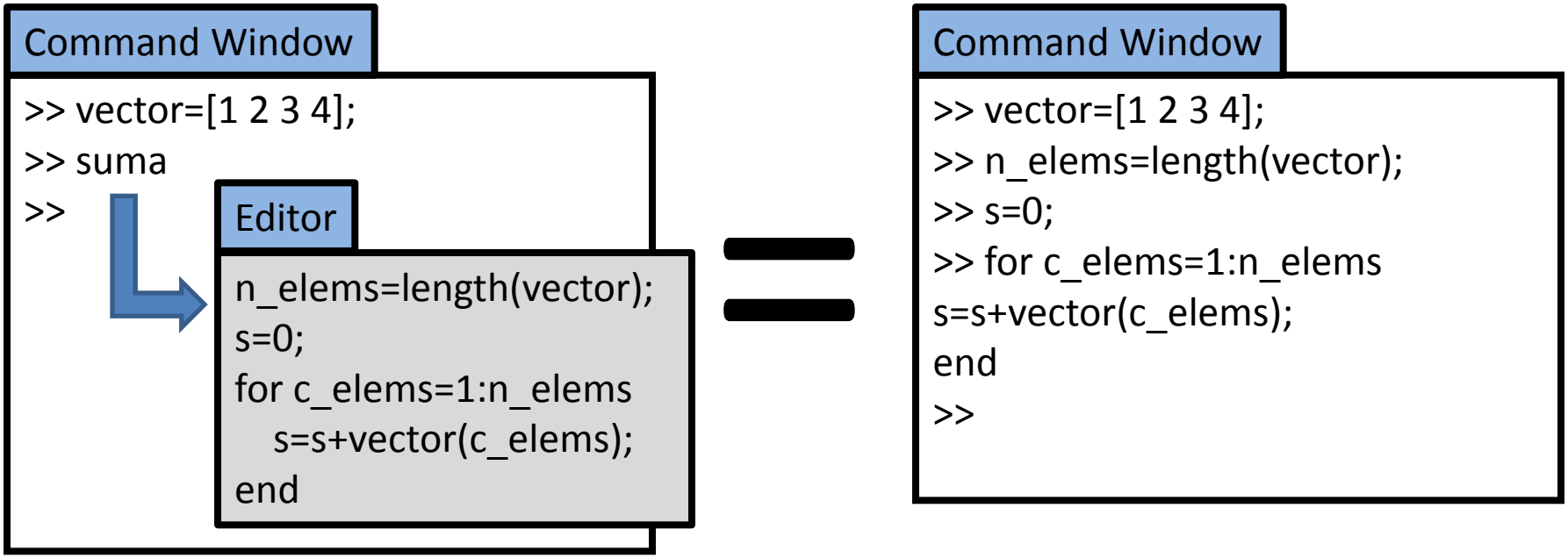
Solving problems: Functions

Function: Code that performs a specific task



Difference between functions and scripts

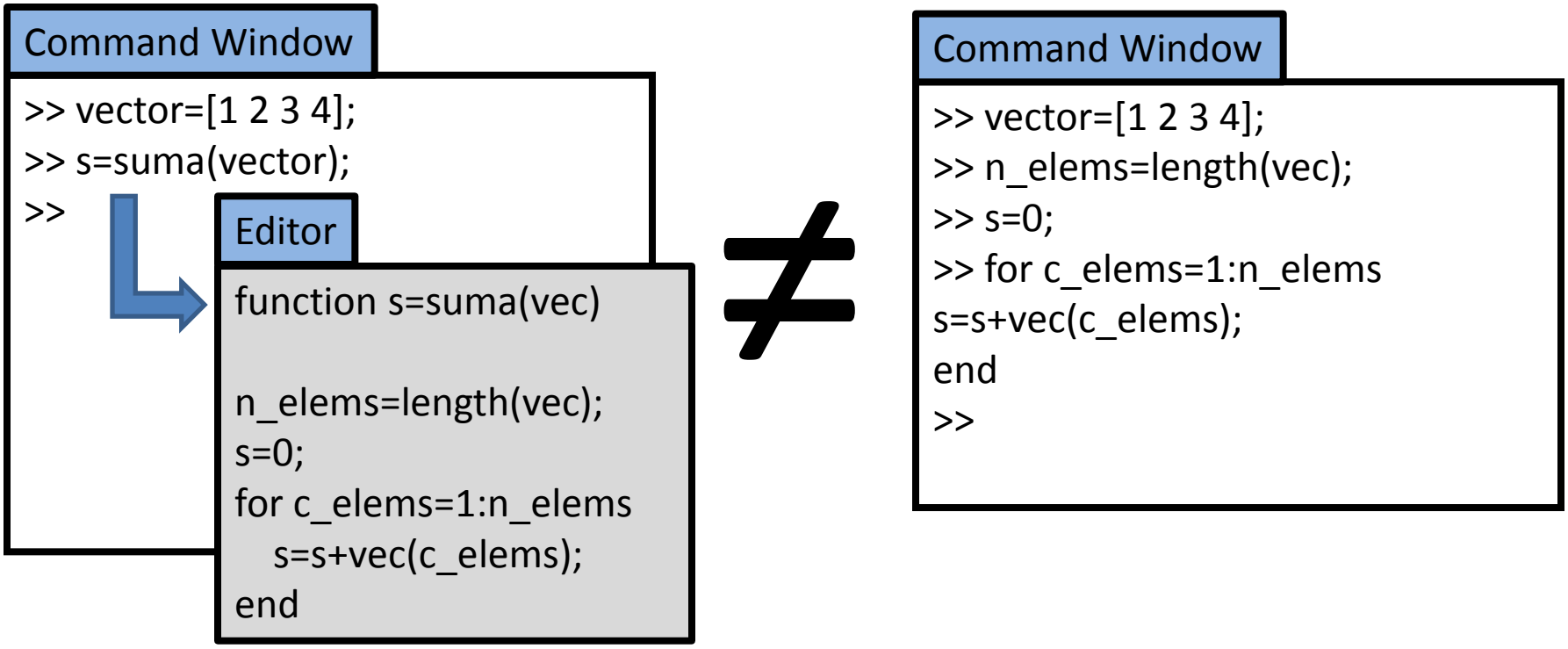
Scripts



- The script uses previously defined variables (for example, `vector`).
- All variables defined in the script (i.e. `n_elems`, `s`, `c_elems`) remain in the workspace after the script is executed.

Difference between functions and scripts

Functions



Difference between functions and scripts

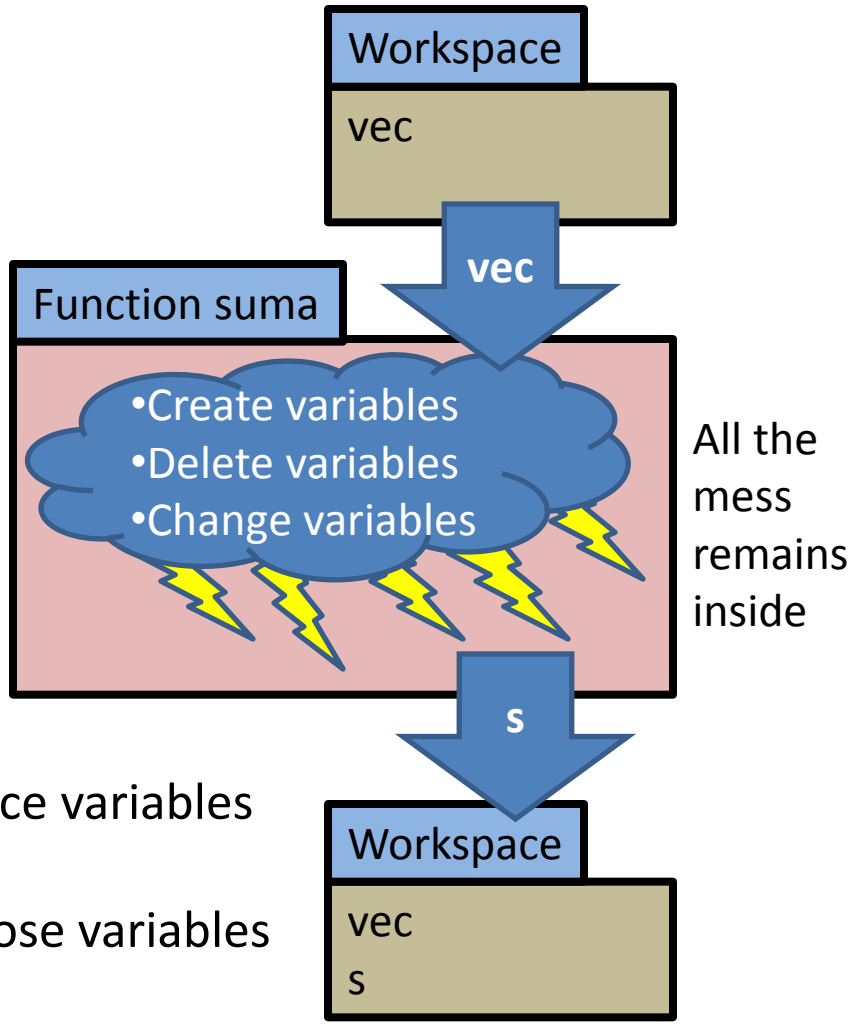
Functions

Command Window

```
>> vec=[1 2 3 4];
>> s=suma(vec);
>>
```

Editor

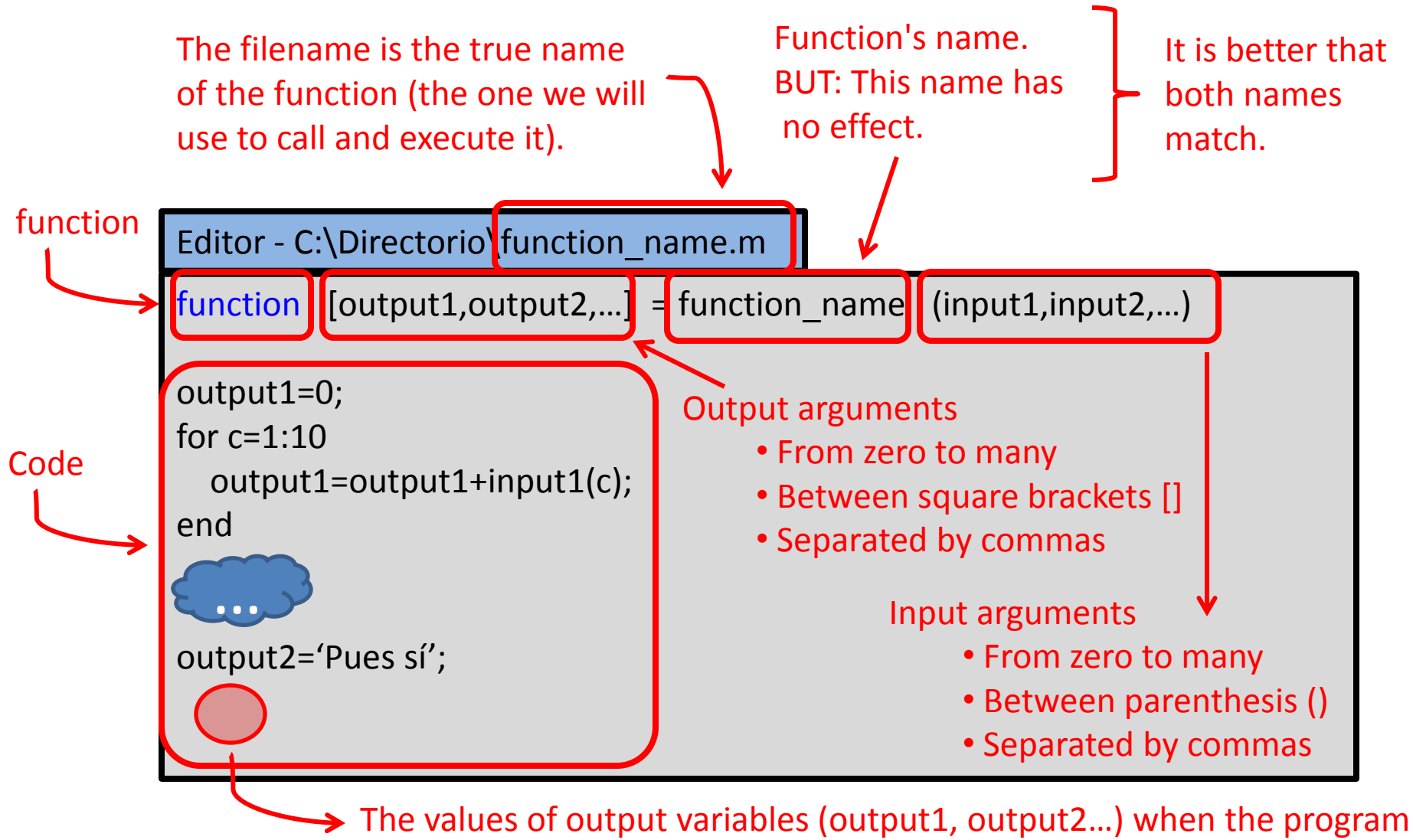
```
function s=suma(vector)

n_elems=length(vector);
s=0;
for c_elems=1:n_elems
    s=s+vector(c_elems);
end
```



All the mess remains inside

- A function can only access those workspace variables that are given to it as input arguments.
- A function only adds to the workspace those variables specified as output arguments.

Function definition



Use of functions

Editor - C:\Directorio\sumadorl.m

```
function   s  = sumadorl   (sumando1,sumando2)

s = sumando1 + sumando2;
```

Command Window

```
>> vector=[1 2 3 4];
>> suma=sumadorl(vector,10)

suma =

        11     12      13      14

>>
```

Use of functions

Editor - C:\Directorio\sumadorl.m

```
function   s  = sumadorl  (sumando1,sumando2)

s = sumando1 + sumando2;
```

Command Window

```
>> vector=[1 2 3 4];
>> suma=sumadorl (vector,10)

suma =

       11     12     13     14

>>
```

Variable names do not need to match inside and outside the function (but they may match).

Input arguments may be either variables or just numbers/vectors typed directly.

Local Variables

Editor - C:\Directorio\sumadorl.m

```
function   s  = sumadorl(sumando1,sumando2)

s = sumando1 + sumando2;
```

Local variables: Used inside a function. Each function has its own Workspace, and they never get mixed.

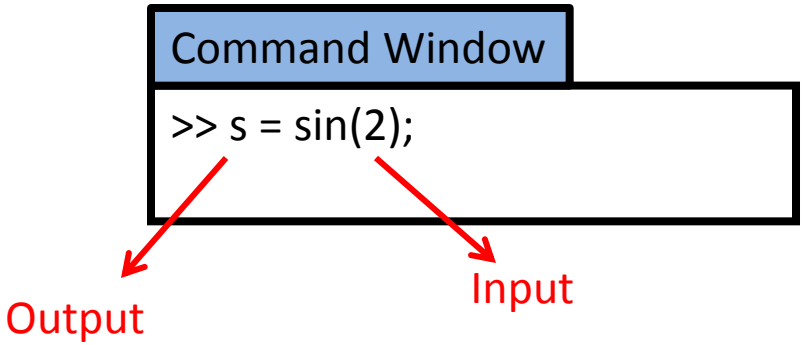
Command Window

```
>> vector=[1 2 3 4];
>> suma=sumadorl(vector,10)

suma =

      11     12     13     14

>>
```

Base Workspace: Used from Command Window.

There may be variables with the same name in different workspaces, but they are completely different variables.

Turns out, we have been using functions all the time



Logical operators AND, OR, NOT

- AND: &
- OR: |
- NOT: ~
- They may be nested.

  Examples: (a==b & c==d) | e>f

  a==b & (c==d | e>f)

  a==b & ~(c>d)

Efficiency: Preallocating variables

```
>> a(1)=5;
>> a(2)=3;
>> a(3)=7;
```

→ Reserve space for one number
→ Reserve space for another number
→ Reserve space for another number

Three space requests

```
>> a=zeros(1,3);
>> a(1)=5;
>> a(2)=3;
>> a(3)=7;
```

→ Reserve space for three numbers

Space was already reserved

One space request
MUCH FASTER

Specially important in loops

If you do not know the final length from the beginning, preallocate plenty of space and cut the vector in the end