Day 4

Very VBA

The Data Boot Camp | June 25, 2018

Intro to Programming Logic

Ooh… Coding! (Sort Of)

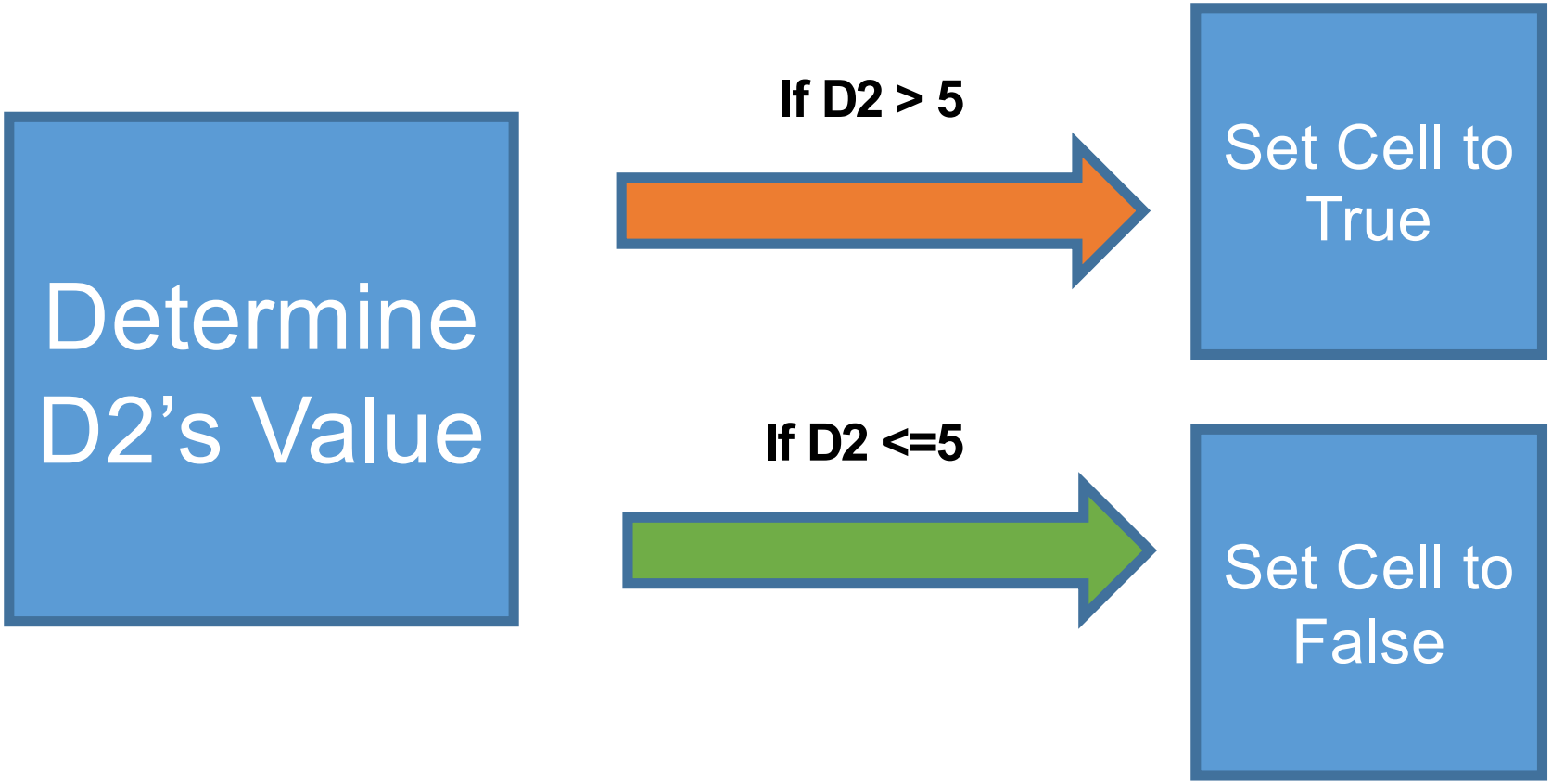
Function

=SUM(1, 2, 3)

Arguments

In a way, Excel has introduced you to a sort of proto-programming. Throughout your time writing scripts you will rely on **functions** (methods) that do *something* to or with **arguments**.

Conditionals: If This… Then That



=IF(D2>5,TRUE,FALSE)

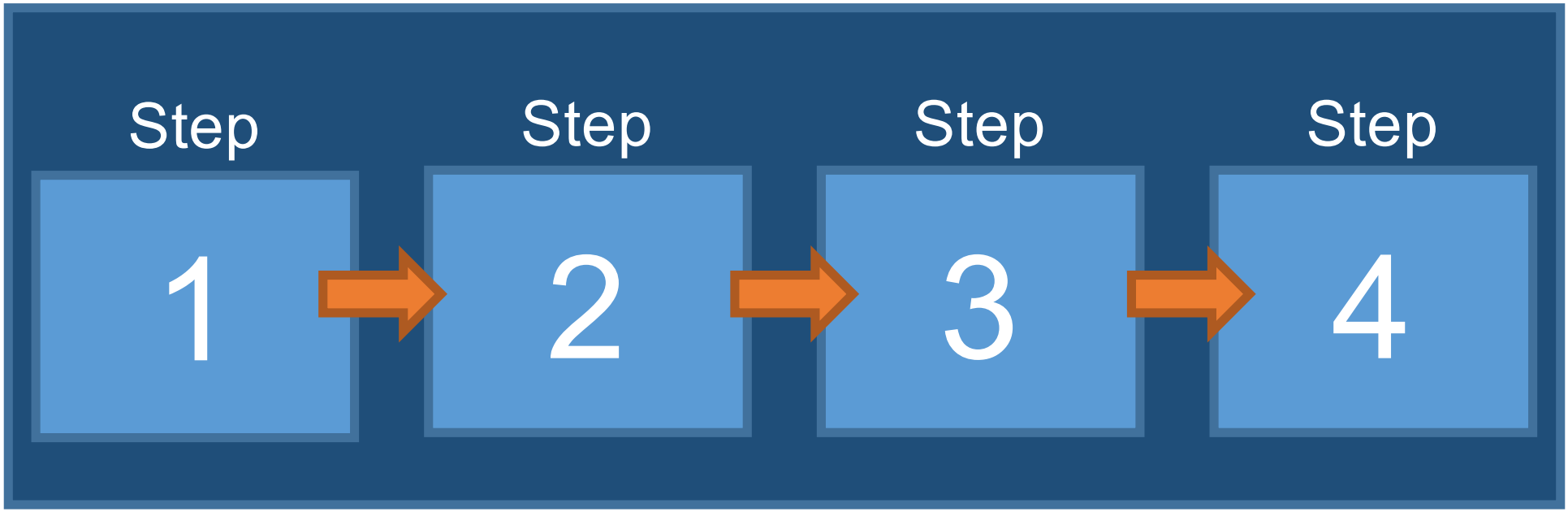
How a Computer Thinks (Procedurally)

Complex Real-World Problem

Every problem in software development begins with a complex and abstract real-world need.

How a Computer Thinks (Procedurally)

Complex Real-World Problem



In order for a computer to handle things, this "real-world" problem needs to be broken into a set of procedural steps.

How Code is Written (Procedurally)

Code (Python)

```python
# STEP 1
# ----------------------------------

vegetables = ["carrots", "broccoli"]
protein = "chicken"

# STEP 2
# ----------------------------------

chop(vegetables)

# STEP 3
# ----------------------------------

season(protein)

# STEP 4
# ----------------------------------

stirfry(vegetables)

# STEP 5
# ----------------------------------

stirfry(protein)
```

Steps

1

2

3

4

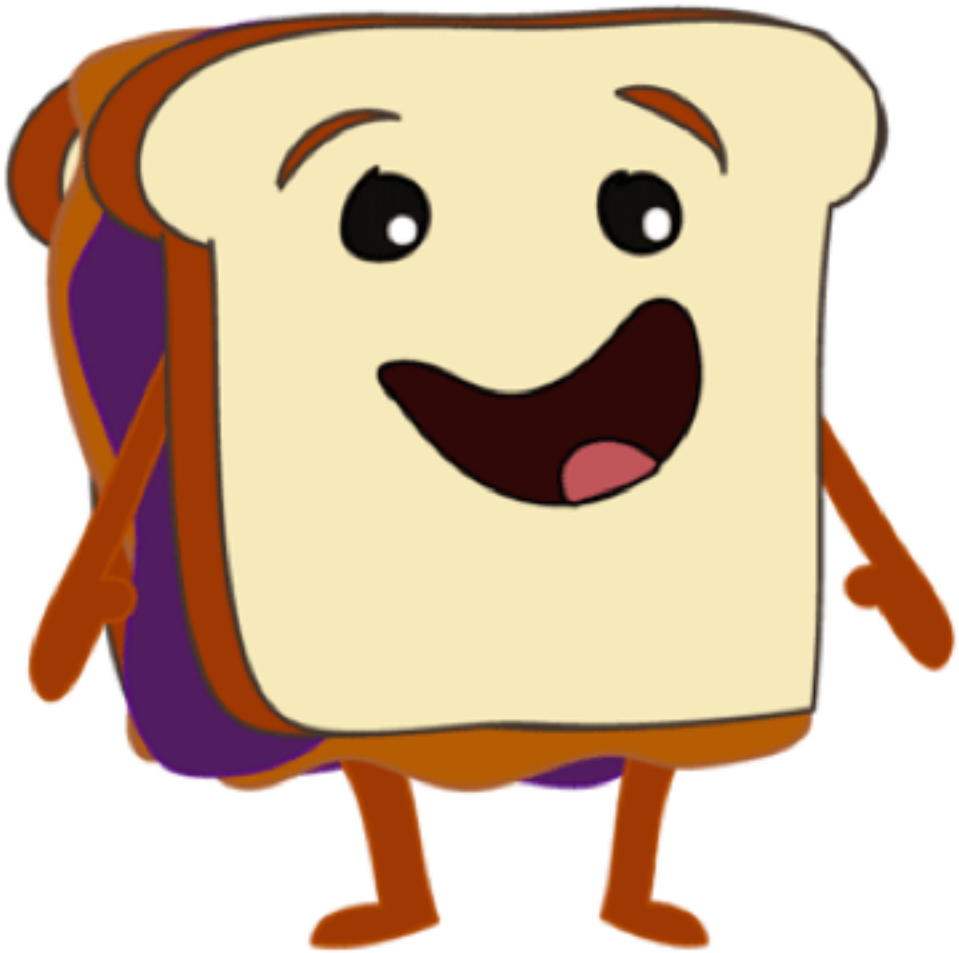
5

Fundamental Building Blocks

Common structures in nearly all languages:

1. **Variables / Arrays**
2. **Conditionals**
3. **Iterations**
4. **Functions**

To Make A Sandwich…

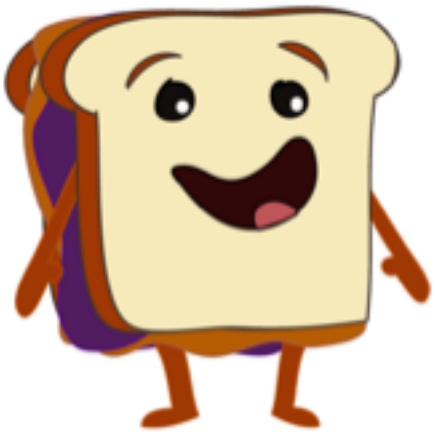


To Make a Sandwich…

Logical Procedure

1. Get Bread, Peanut Butter, and Jelly
2. Lay out bread
3. Open Peanut Butter and Jelly
4. Get knife
5. Use knife to spread peanut butter
6. Use knife to spread jelly
7. Combine bread to create sandwich

Oh! I can't wait to be eaten!



Variables / Arrays

Variables: The Nouns of Code

- **Variables** are effectively the items in a procedure.
- They can be physical things (like an ingredient) or abstractions (like a counter).
- In VBA, items can be declared as variables by using the dim followed by the type. They can then be assigned a value.

```
dim ing1 as String
dim ing2 as String
dim budget as Double
```

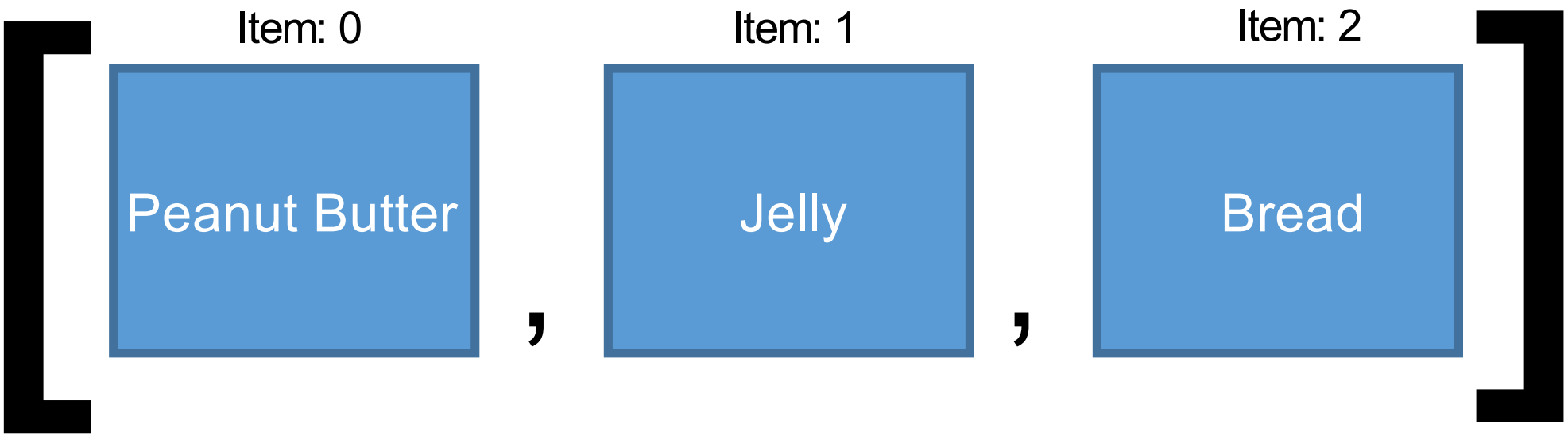
Variable Declaration

```
ing1 = "Peanut Butter"
ing2 = "Jelly"
budget = 5.00
```

Variable Assignment

Arrays: A Collection of Items

- Arrays are effectively **groups** of related items. It presents another way to store and reference like pieces of information.



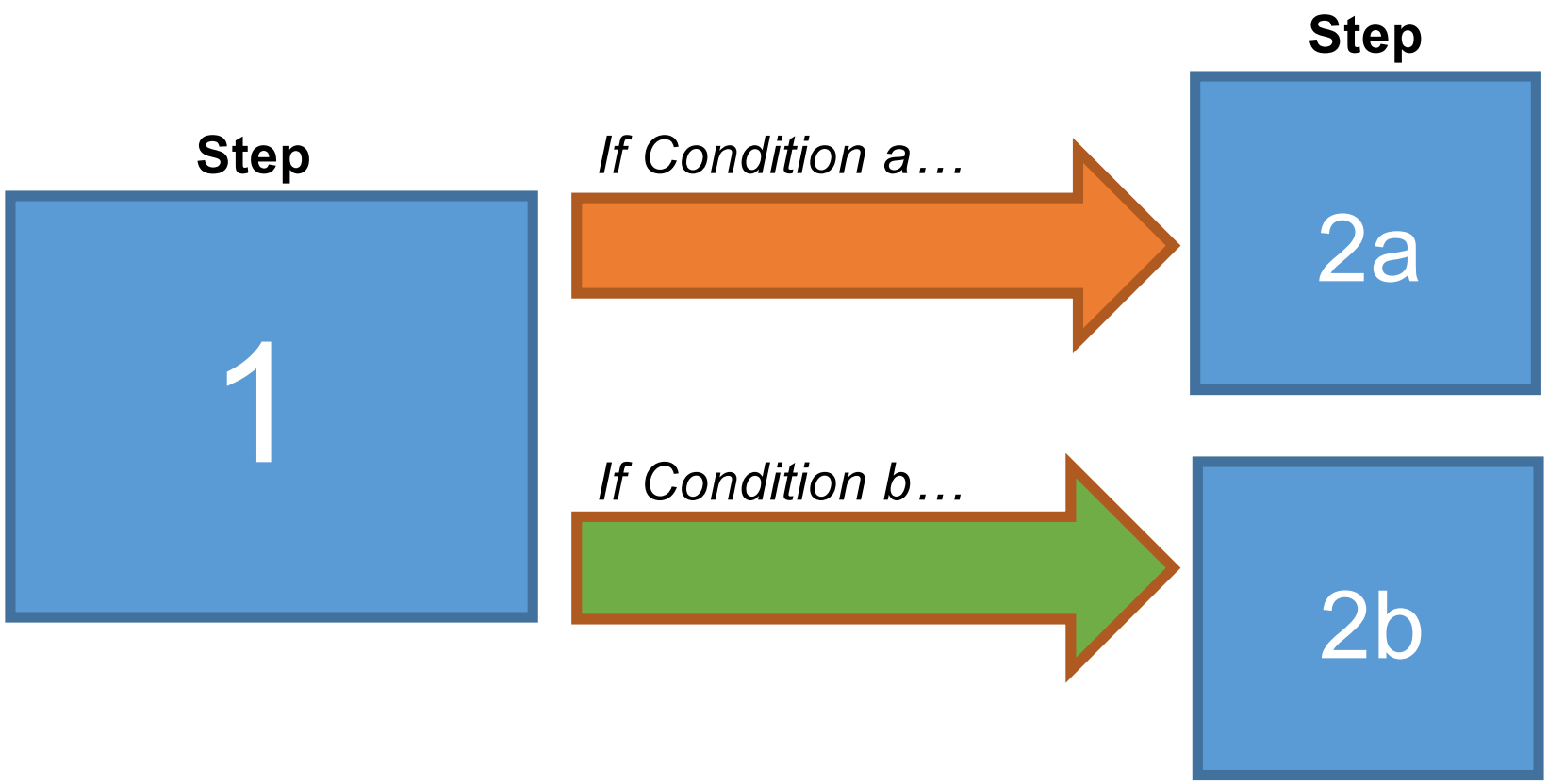

```
dim ingredients(0 to 2) as String

ingredients(0) = "Peanut Butter"
ingredients(1) = "Jelly"
ingredients(2) = "Bread"
```

Conditionals

Conditionals: If This… Then That.

- **Conditionals** present a way to control the flow of logic based on certain conditions being met.

- In most languages, we'll be using if / else code for this purpose.

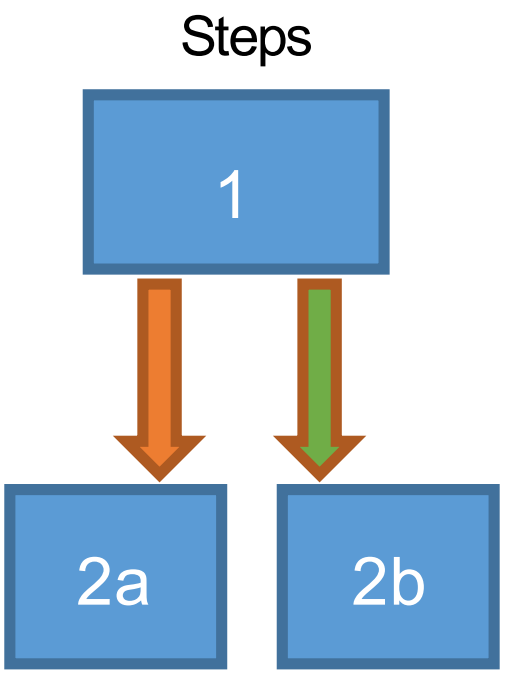


Conditionals: If This… Then That.

- In VBA, conditionals are simply declared using the keywords If, Then, Elseif, Else, and End if.

- Through VBA we can create far more sophisticated conditional logic than through Excel formulas alone.

```
If (pbThickness > 1.0) Then
  stopSpreading()

Else
  spreadMore()

end if
```

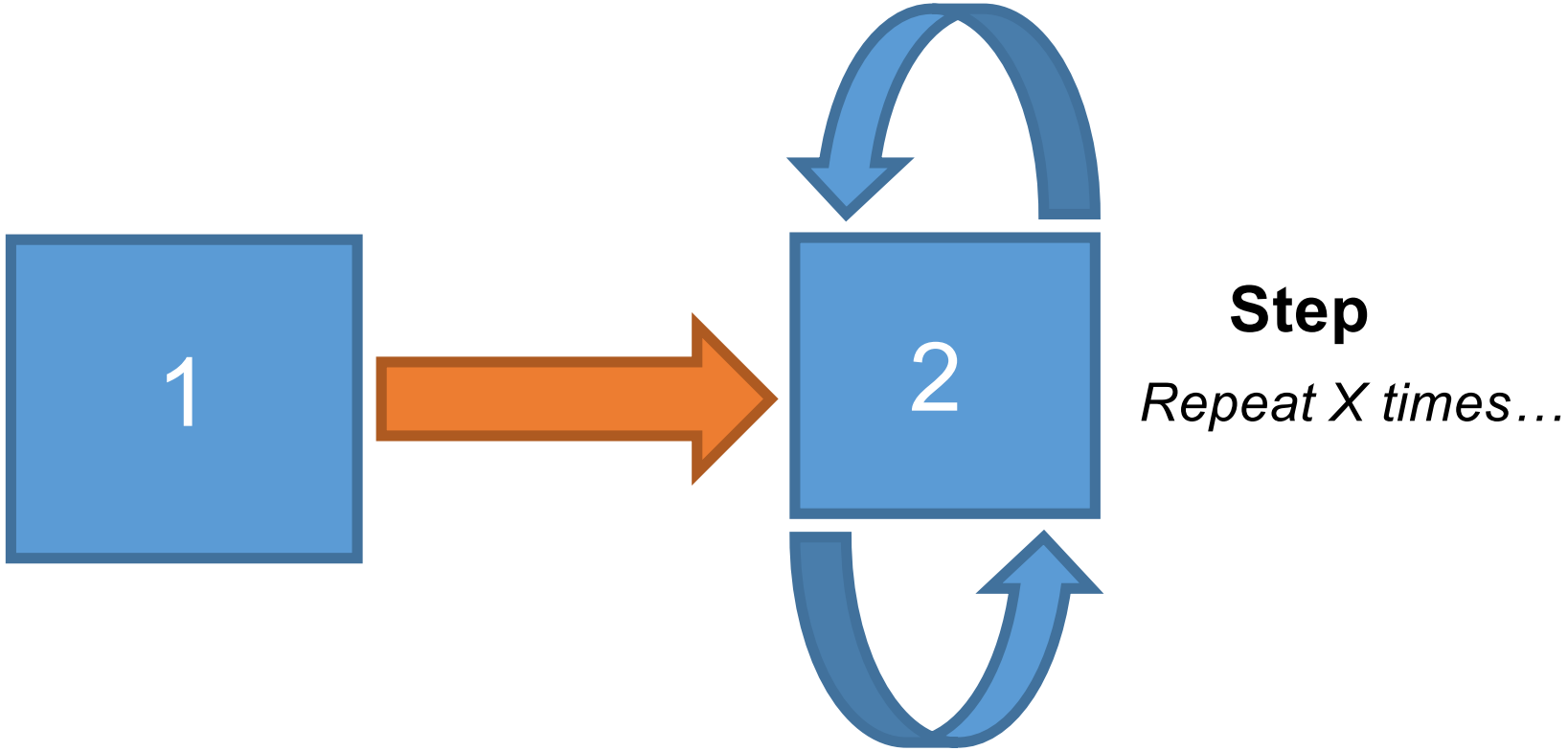


Iteration (Looping)

Iteration: Round and Round We Go!

- **Iteration** is the concept of using loops to perform a group of tasks repeatedly for a number of times.

- In almost all languages, we'll be using for-loops and while loops.



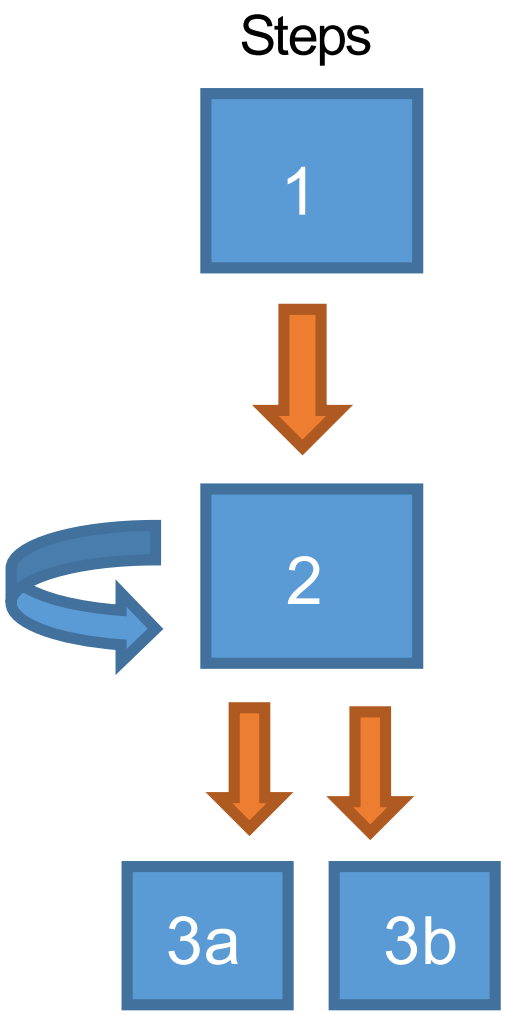
Iteration: Round and Round We Go!

- *This code will make more sense later… but basically it's the VBA way of* ***repeating the same block multiple times****.*

```
' Repeat the same step until i becomes 20
For i = 0 to 20

  ' Each time spread more
  spreadMore()

' Add one to the value of i each time
Next i
```

Build the Program!

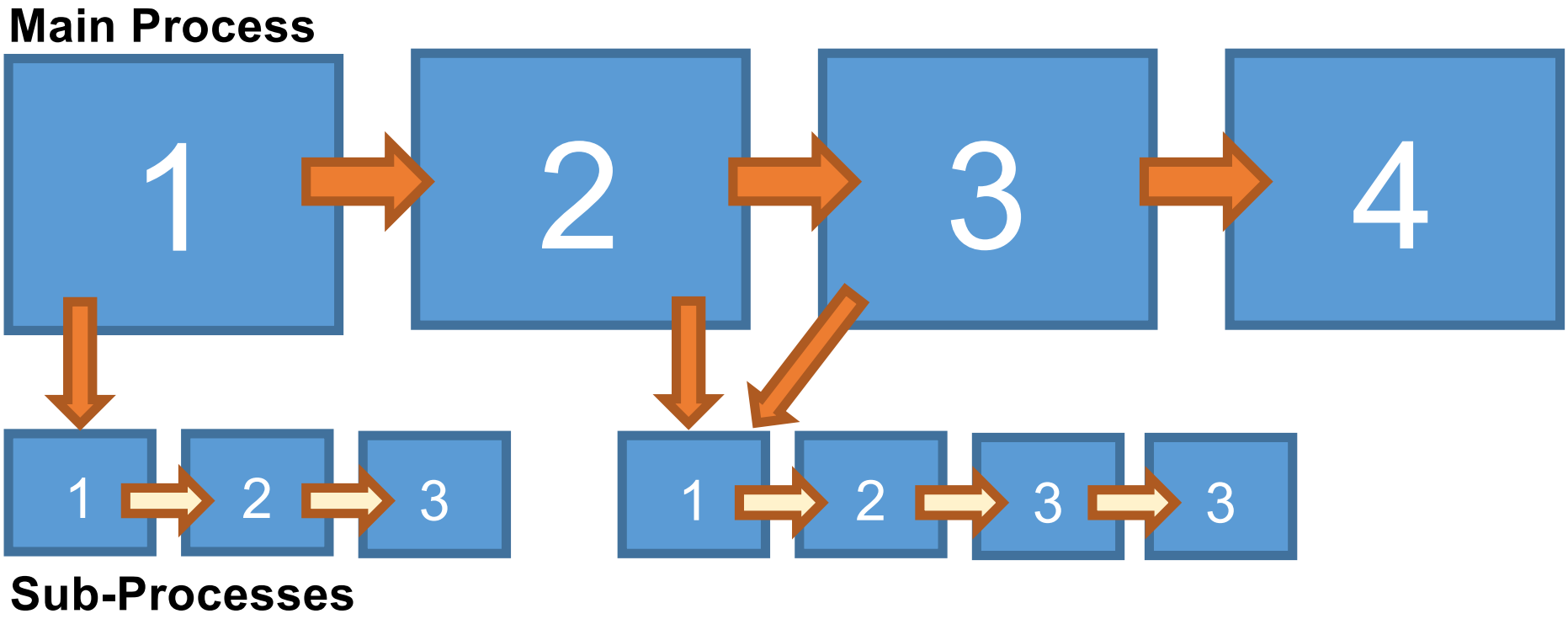
```
// Get Ingredients
dim ing1, ing2, ing3 as String
ing1 = "Peanut Butter"
ing2 = "Jelly"
ing3 = "Bread"

// Repeat the spreading process a max of 5 times
for i = 0 to 5

    // Each time, check that you haven't spread too much.
    if pbThickness >= 1.0 then

        // If you have spread too much, stop spreading.
        stopSpreading()

    // Otherwise...
    else:

        // Keep spreading.
        spreadMore()
    end if

next i
```



Functions

Functions: For When One Block *Can't* Do it All

- **<u>Functions</u>** are, in essence, a sort of "sub-processes". They allow us to create pre-made, re-usable blocks of code which can be called on demand.



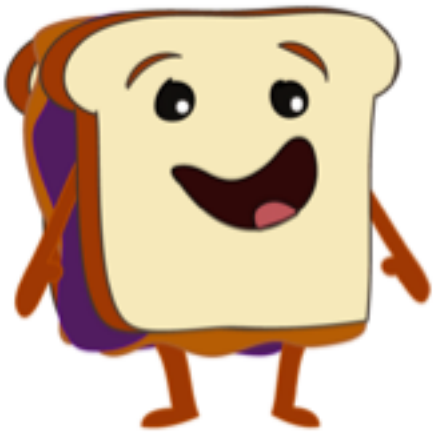
Putting It All Together…

To Make a Sandwich…

Logical Procedure

1. Get Bread, Peanut Butter, and Jelly
2. Lay out bread
3. Open Peanut Butter and Jelly
4. Get knife
5. Use knife to spread peanut butter
6. Use knife to spread jelly
7. Combine bread to create sandwich

Oh! I can't wait to be eaten!

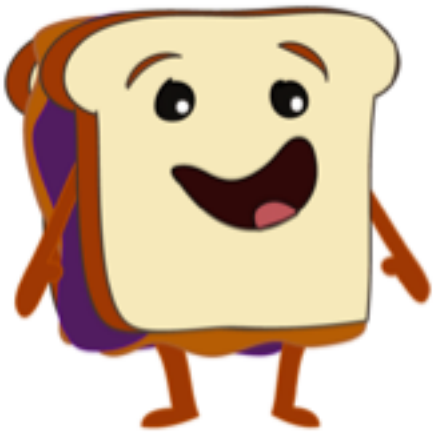


To Make a Sandwich…

Logical Procedure

1. Get Bread, Peanut Butter, and Jelly
2. Lay out bread
3. Open Peanut Butter and Jelly
4. Get knife
5. Use knife to spread peanut butter
6. Use knife to spread jelly
7. Combine bread to create sandwich

Oh! I can't wait to be eaten!



To Make a Sandwich (Full Logic)…

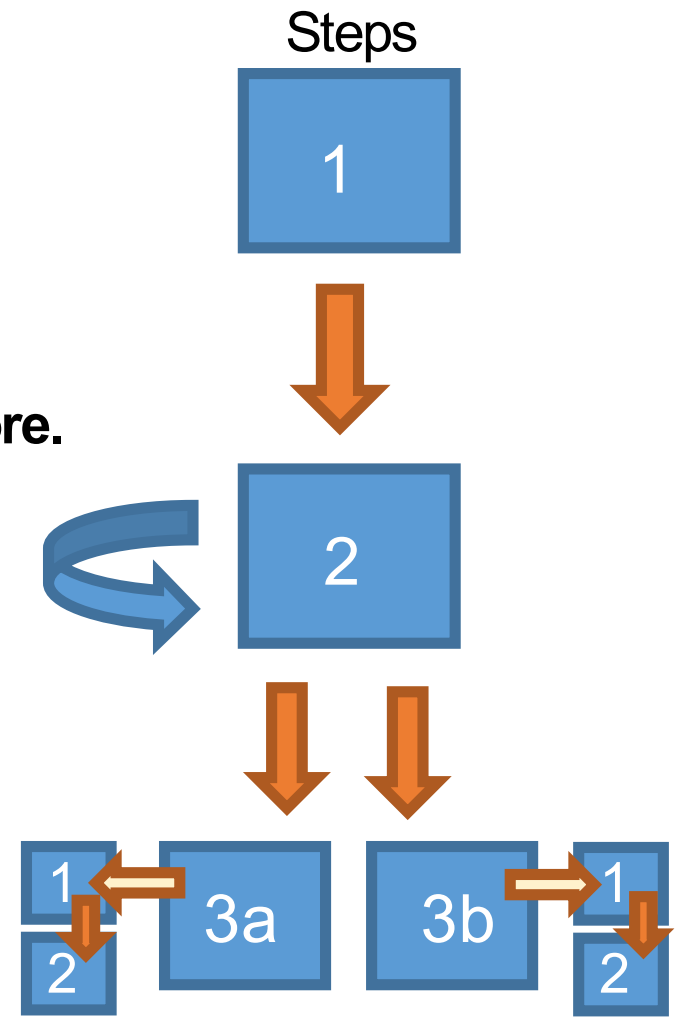
1. Get Items

2. Repeatedly "spread the Peanut Butter"

3. Check if thickness condition met.

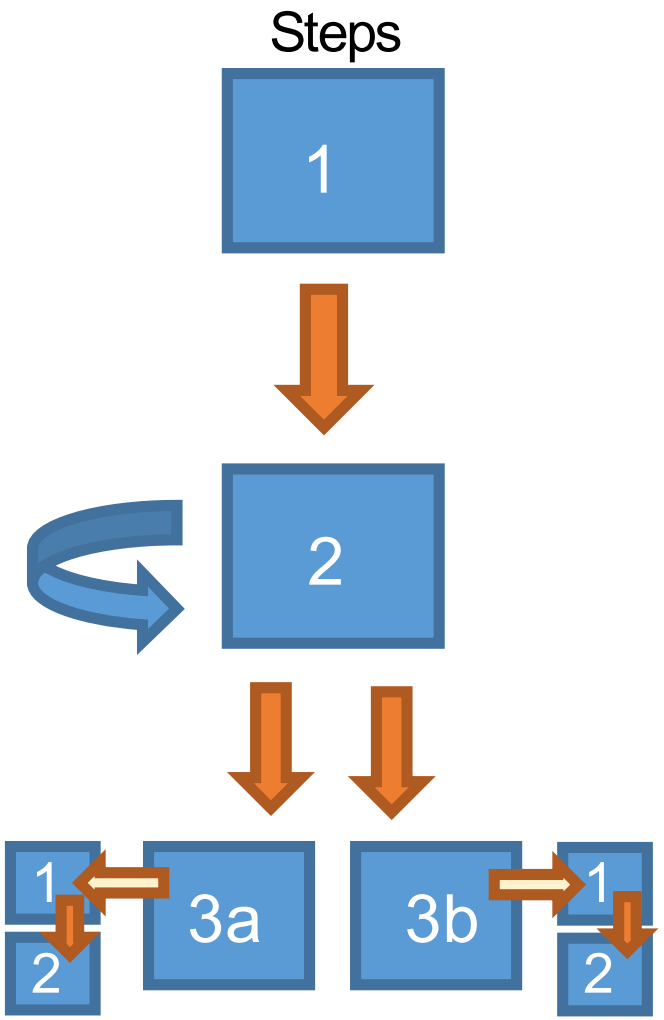
3a. If thickness condition is met run stop function.

3b. If thickness condition is *not* met then spread more.



To Make a Sandwich (In Code)…

```
Sub PeanutButter():

  ' Get Ingredients
  dim ing1, ing2 as String
  ing1 = "Peanut Butter"
  ing2 = "Jelly"

  ' Repeat the spreading process a max of five times
  for i=0 to 5

    ' Each time, check that you haven't spread too much
    if (pbThickness > 1.0){

      ' If you have spread too much, stop spreading.
      stopSpreading()
    }

    ' Otherwise
    else

      ' Keep spreading...
      keepSpreading()

    end if

  next i

End Sub

' Define the spreadMore function
Sub SpreadMore():

  ' Use another set of sub-functions to move the knife
  dipIntoPb()
  horizontalShiftKnife()

End Sub
```

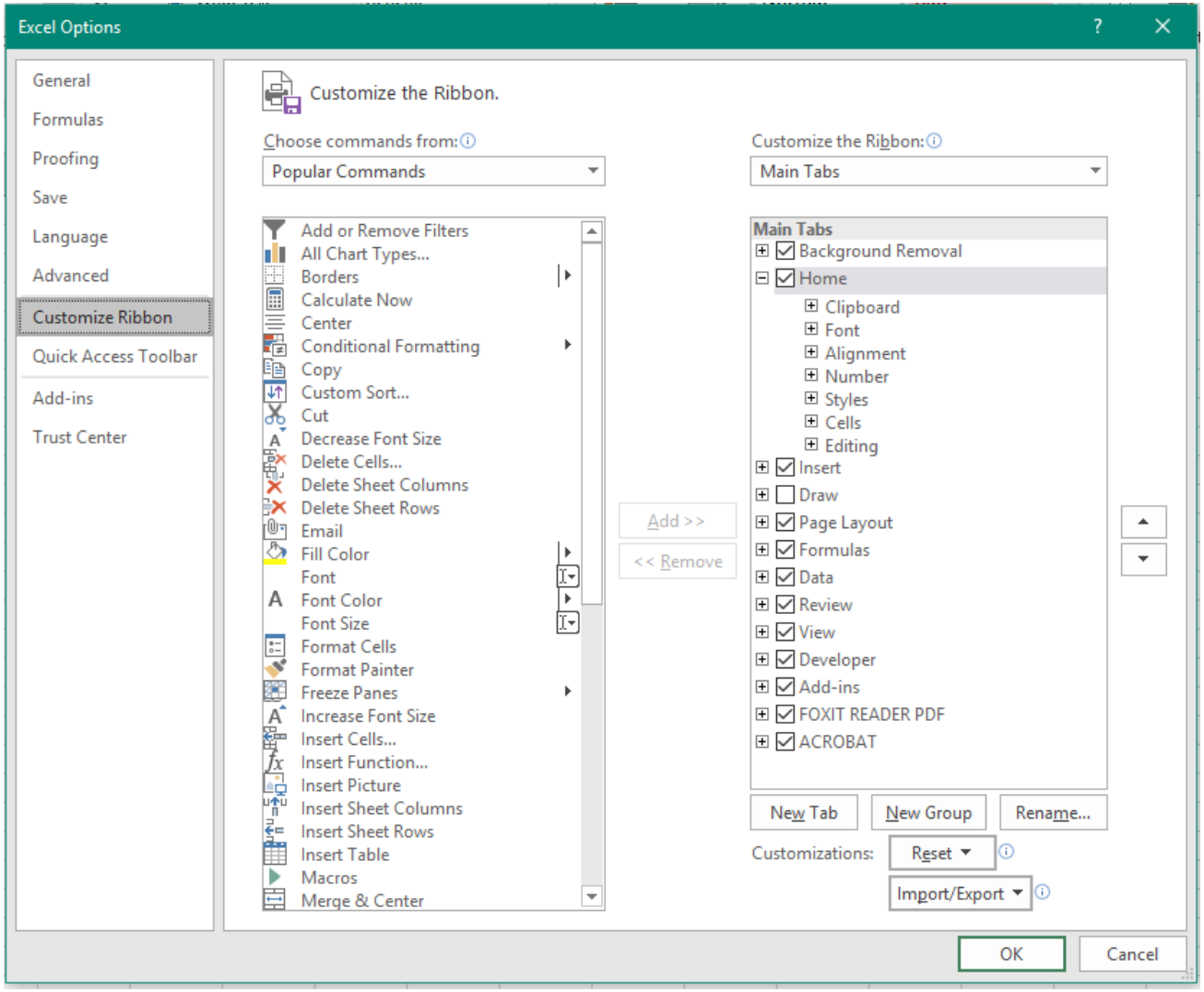


Big Picture!

Coding = Building Blocks and Putting them Together

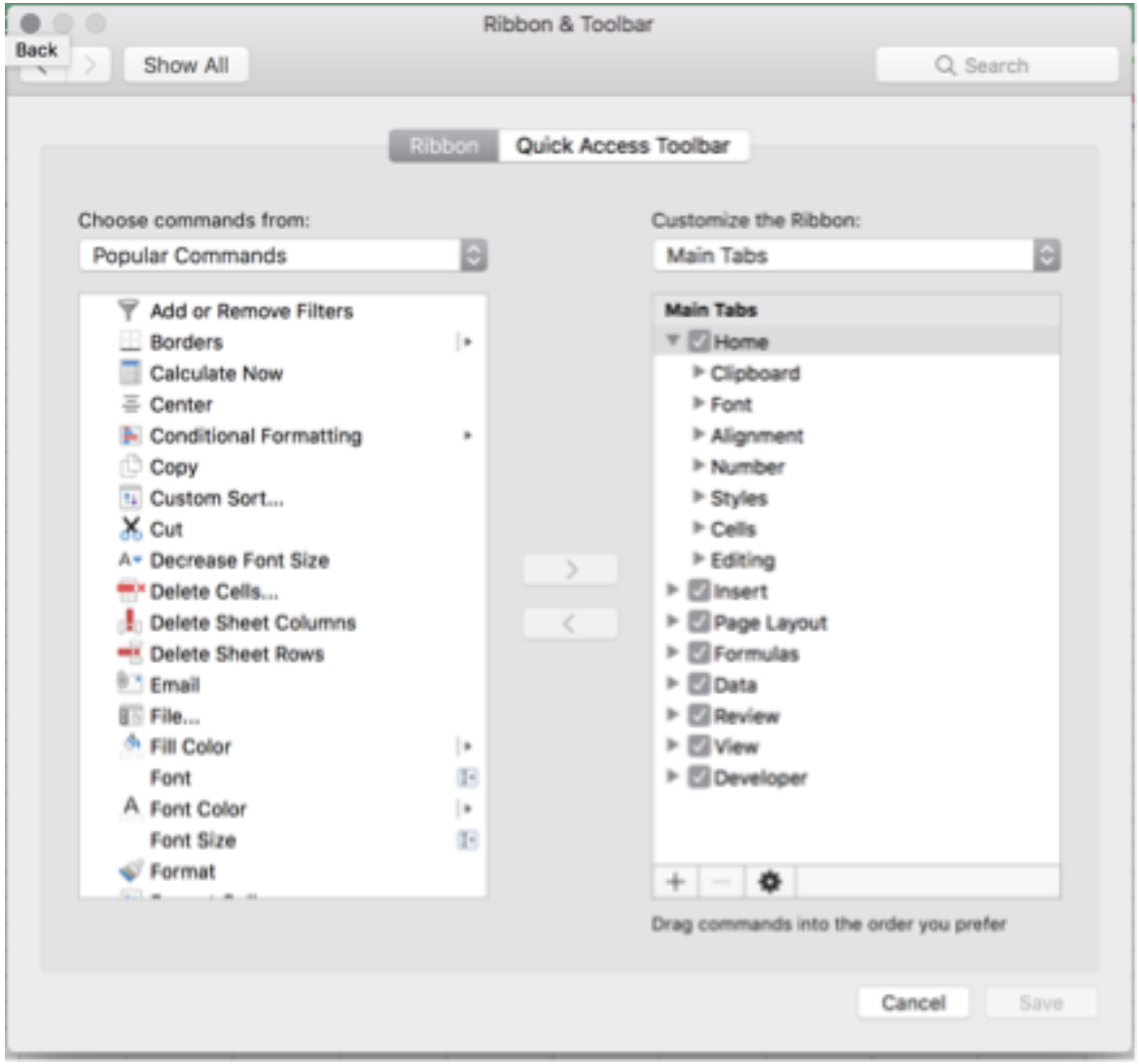
Let's Get Coding!

But First… Let's Add Developer Tools!



On a Windows machine, visit **File -> Excel Options**.
Then navigate to **Customize Ribbon** to enable the **Developer** tab.

But First… Let's Add Developer Tools!



On a Mac machine, visit **Excel -> Preferences**.
Then navigate to **Ribbon and Toolbar** to enable the **Developer** tab.

Questions / Discussion