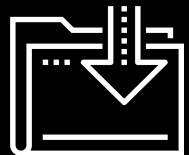


**Data Boot Camp**  
Lesson 2.1

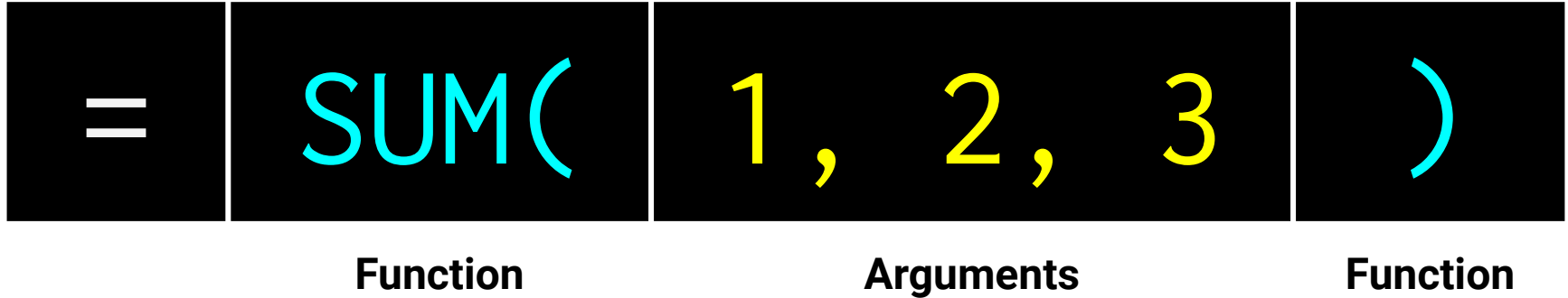


# Intro to Programming Logic

# Ooh, Coding! (Sort of...)

---

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



# How a Computer Thinks (Procedurally)

---

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



# How a Computer Thinks (Procedurally)

---

In order for a computer to interpret it, the real-world problem must be broken down into a set of procedural steps.

**Complex Real-World Problem**





# How Code Is Written (Procedurally)

---

## Code (Python)

```
# STEP 1
# -----
thingamagig = 500
doodad = 200

# STEP 2
# -----
combinedThing = thingamagig + doodad

# STEP 3
# -----
runContraption(combinedThing)

# STEP 4
# -----
resetContraption()
```



# When Procedures Aren't Enough... We Need More Tools!

## Code (Python)

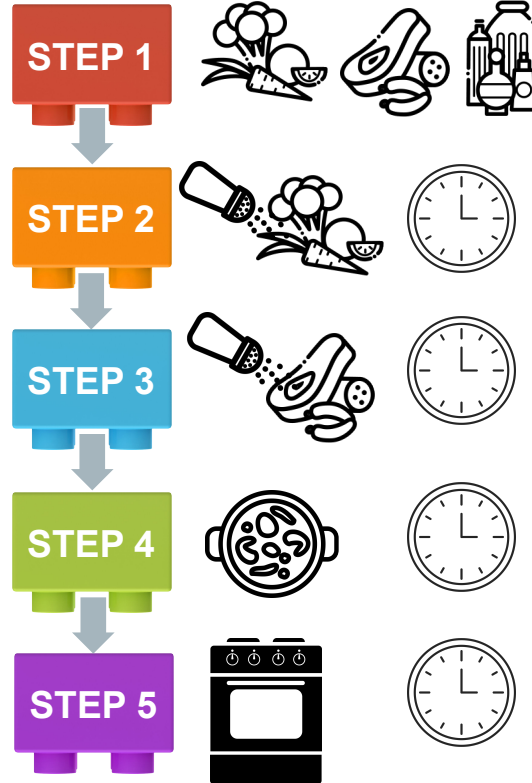
```
# STEP 1
# -----
ingredient1 = vegetables
ingredient2 = meats
ingredient3 = spices

# STEP 2
# -----
season(vegetables)

# STEP 3
# -----
season(meats)

# STEP 4
# -----
stirfry(vegetables)

# STEP 5
# -----
roast(meats)
```



# Fundamental Tools of Programming

---

These structures are found in nearly all programming languages:



Conditionals



Iterations



Functions



Variables / Arrays



# To Make a Sandwich

---



# To Make a Sandwich

---

## Logical Procedure:

01 Get bread, peanut butter, and jelly from pantry.

02 Lay out bread on table.

03 Open jars of peanut butter and jelly.

04 Get spreading knife.

05 Use knife to spread peanut butter.

06 Use knife to spread jelly.

07 Combine bread to create sandwich.

# Fundamental Tools Can Help Make the Sandwich

---

We use these tools as building blocks to make an ideal sandwich procedure:



Conditionals - If peanut butter is crunchy, use less.



Iterations - While there is more peanut butter, add more jelly.



Functions - Spread the condiment using a knife



Variables / Arrays - The ingredients are bread, peanut butter and jelly.

# VBA Building Blocks



# Variables and 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 **dim** followed by the type. Then they can be **assigned** a value.

## Variable Declaration

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

## Variable Assignment

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

# Array: A Collection of Items

---

Arrays are effectively **groups** of related items. They present another way to store and reference similar pieces of information.

Item 0

Item 1

Item 2

["Peanut Butter",	"Jelly",	"Bread"]
-------------------	----------	----------

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

```
ingredients(0) = "Peanut Butter"
```

```
ingredients(1) = "Jelly"
```

```
ingredients(2) = "Bread"
```



# Conditionals

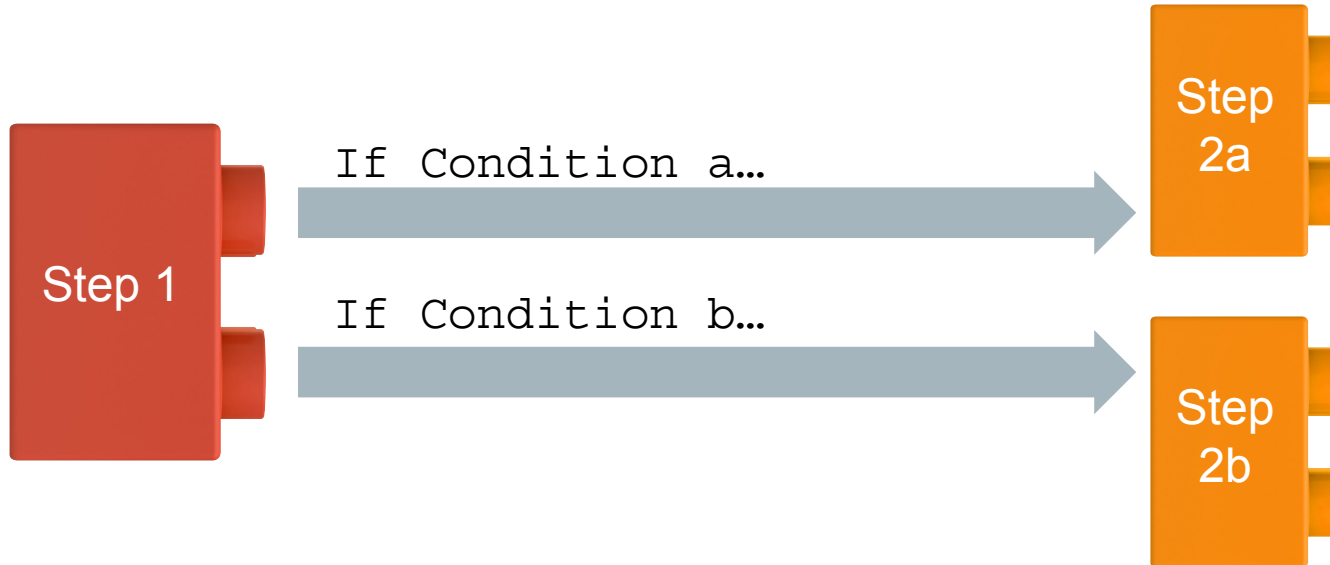
# Conditionals: If This, Then That



**Conditionals** can control the flow of logic based on certain conditions being met.



In most languages, you use **if/else** code for this purpose.



# Conditionals: If This, Then That

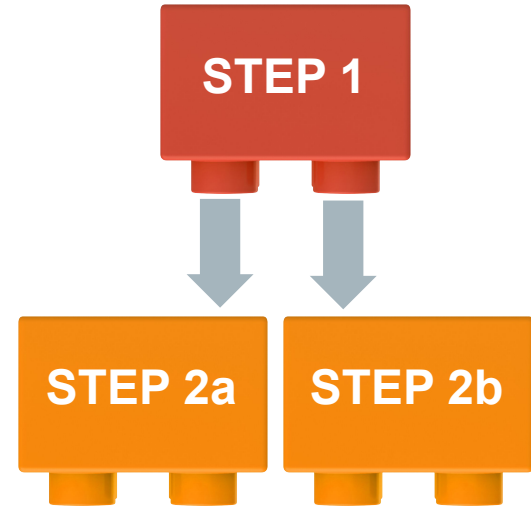


In VBA, conditionals are declared using the keywords **If**, **Then**, **Elseif**, **Else**, and **End if**.



VBA lets us create far more sophisticated conditional logic than with Excel formulas alone.

```
If (pbThickness > 1.0) Then  
    stopSpreading()  
  
Else  
    stopMore()  
  
End if
```



# Iteration (Looping)

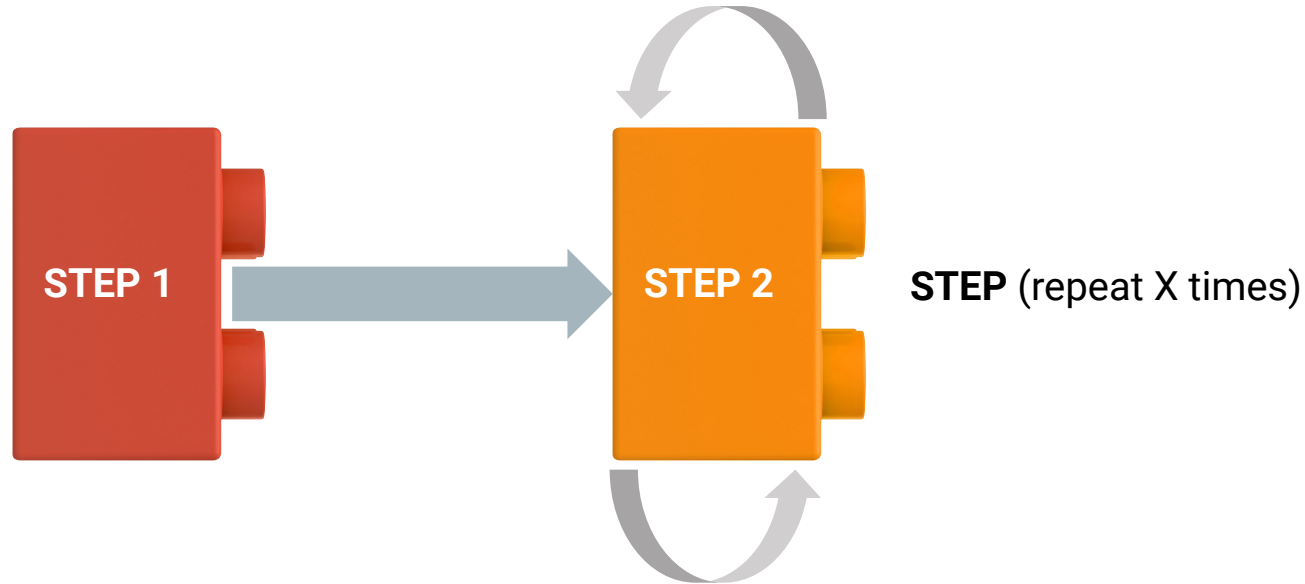
# Iteration: Round and Round We Go!



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



Almost all programming languages use **for loops** and **while loops** for iteration.



# Iteration: Round and Round We Go!

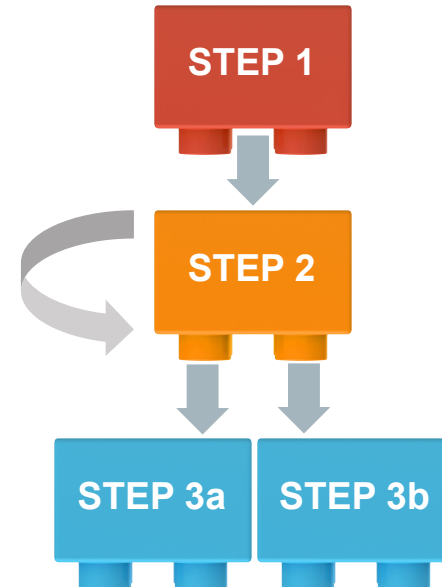
---

This code will make more sense later. 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!

```
1  ' Get Ingredients
2  dim ing1, ing2, ing3 as String
3  ing1 = "Peanut Butter"
4  ing2 = "Jelly"
5  ing3 = "Bread"
6
7  ' Repeat the spreading process a max of 5 times
8  for i = 1 to 5
9
10     ' Each time, check that you haven't spread too much.
11     if pbThickness >= 1.0 then
12
13         ' If you have spread too much, stop spreading.
14         stopSpreading()
15
16     ' Otherwise...
17     else:
18
19         ' Keep spreading.
20         spreadMore()
21     end if
22
23 next i
```



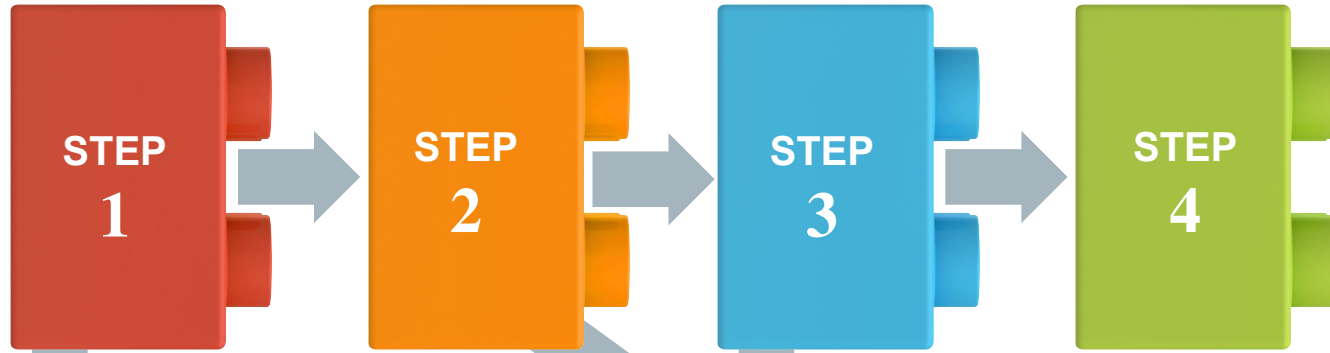


# Functions

# Functions: When One Block Can't Do It All!

In essence, **functions** are a sort of sub-process. They let you create premade, reusable blocks of code that can be called on demand.

**Main Process**



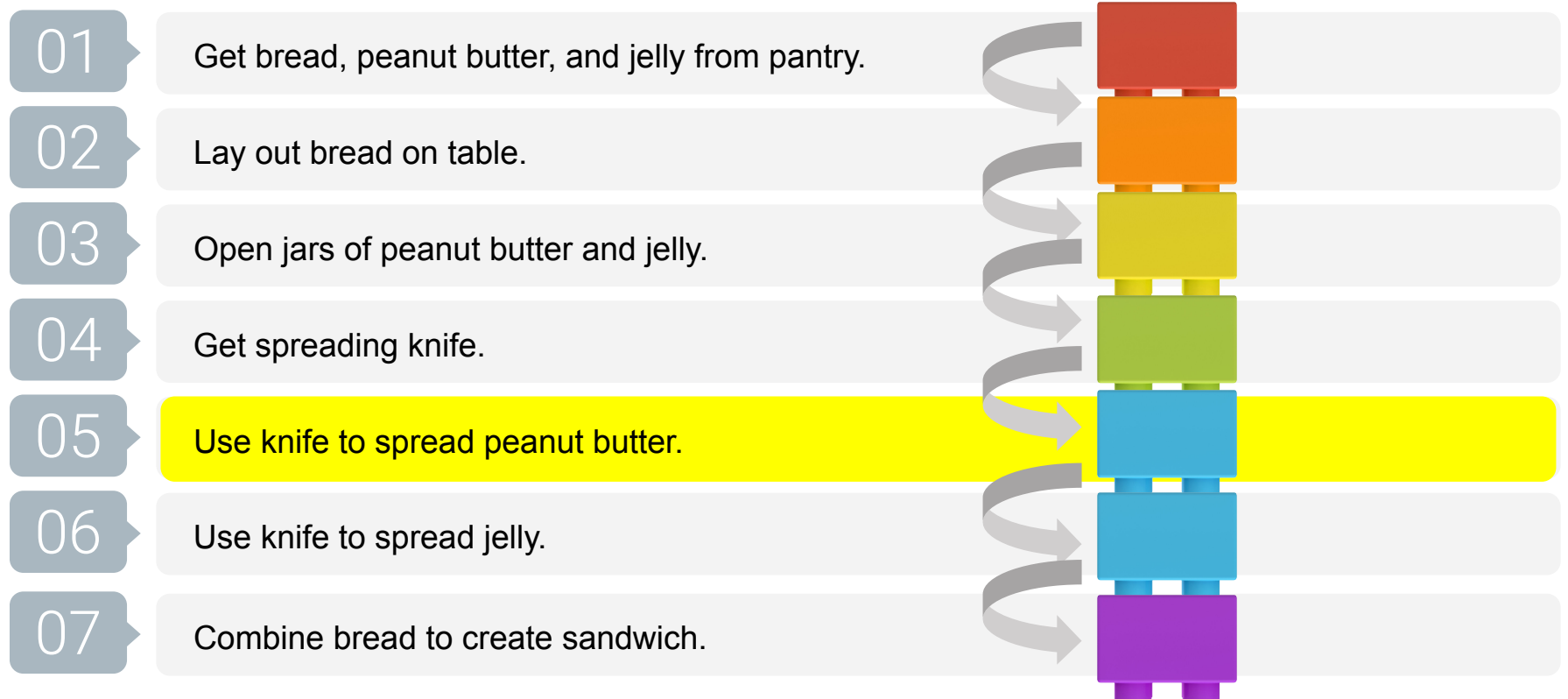
**Sub-Processes**



# Putting It All Together

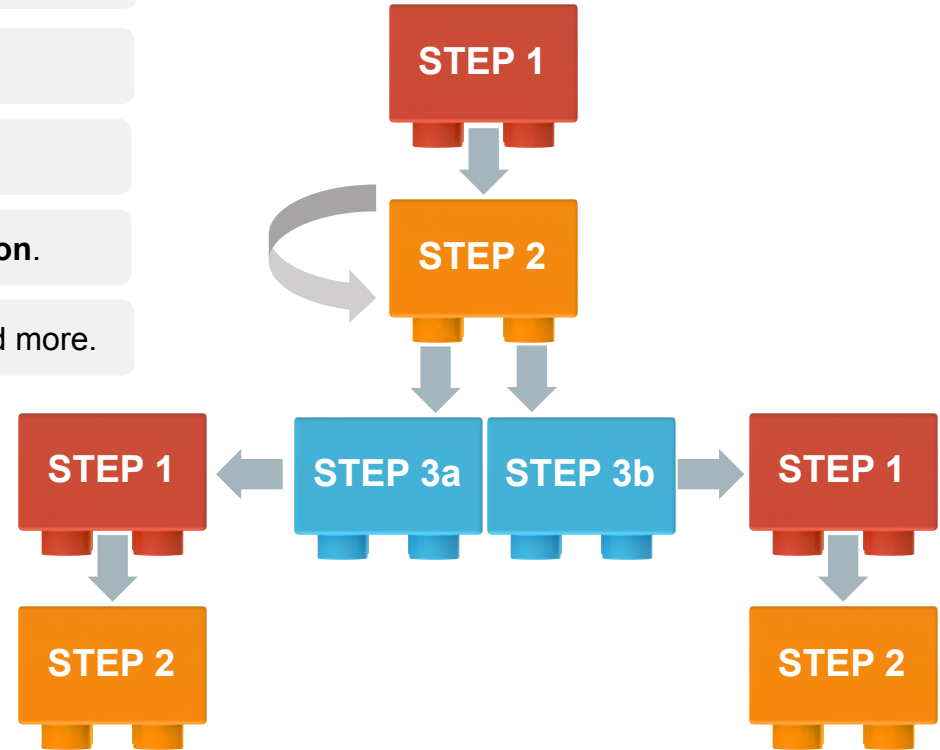
# To Make a Sandwich:

## Logical Procedure:



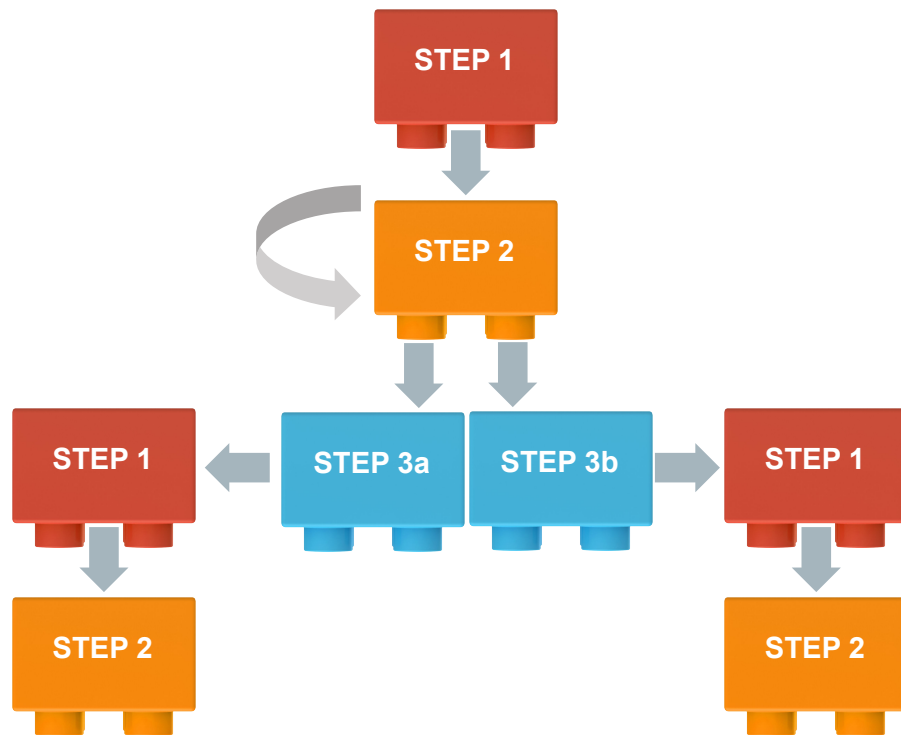
# To Make a Sandwich (Full Logic)

- 01 Get items.
- 02 **Repeatedly** “spread the peanut butter.”
- 03 Check if thickness **condition** is 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 = creating building blocks and putting them together





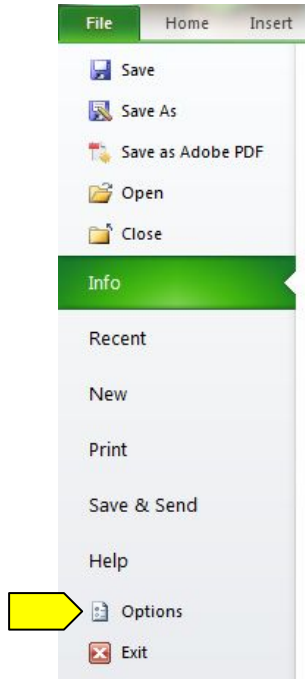


Let's Get Coding!

# Add Developer Tools: Windows

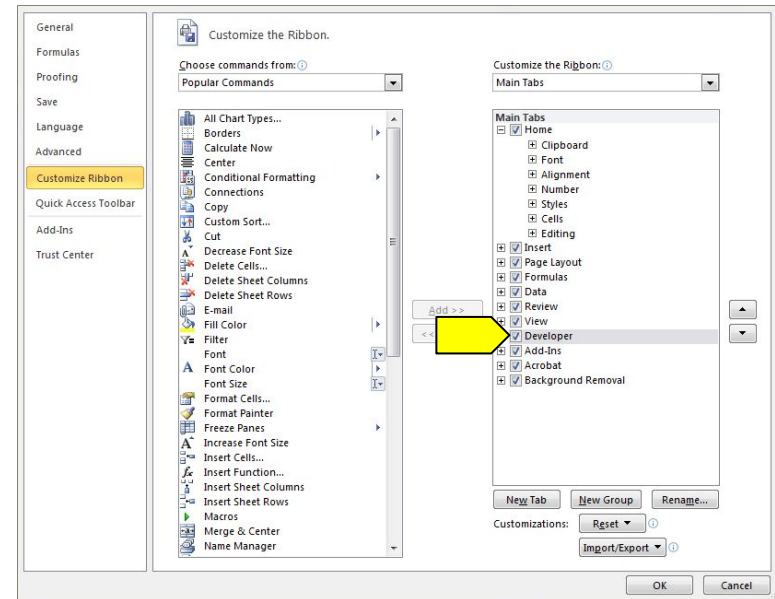
01

Go to **File > Excel Options**.



02

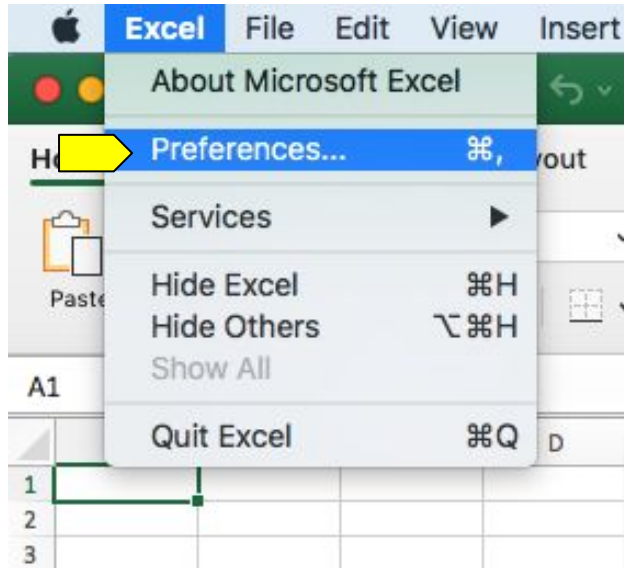
Then go to **Customize Ribbon**, choose **Main Tabs** in the right pane, and make sure **Developer** is checked.



# Add Developer Tools: Mac

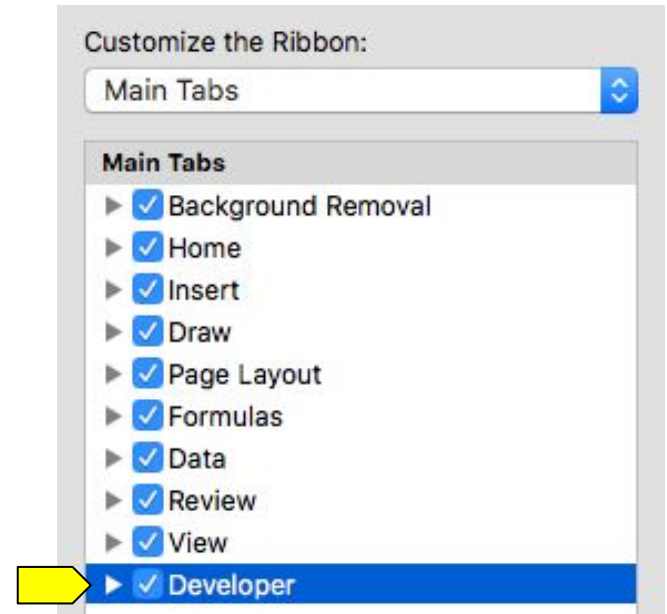
01

Go to **Excel > Preferences**.



02

Then go to **Ribbon & Toolbar**, select **Main Tabs** in the right pane, and make sure **Developer** is checked.





# Questions?