COMP1022Q
Introduction to Computing with Excel VBA

Introduction to Looping

David Rossiter and Gibson Lam

Outcomes

- After completing this presentation, you are expected to be able to:
  1. Write while loops to run code repeatedly in VBA
  2. Write do loops to run code repeatedly in VBA

What is Looping?

- A loop is a set of code which repeats many times
- Looping is a very useful feature in all programming languages because it makes repetitive work easier
- In this presentation we will look at two types of loop:

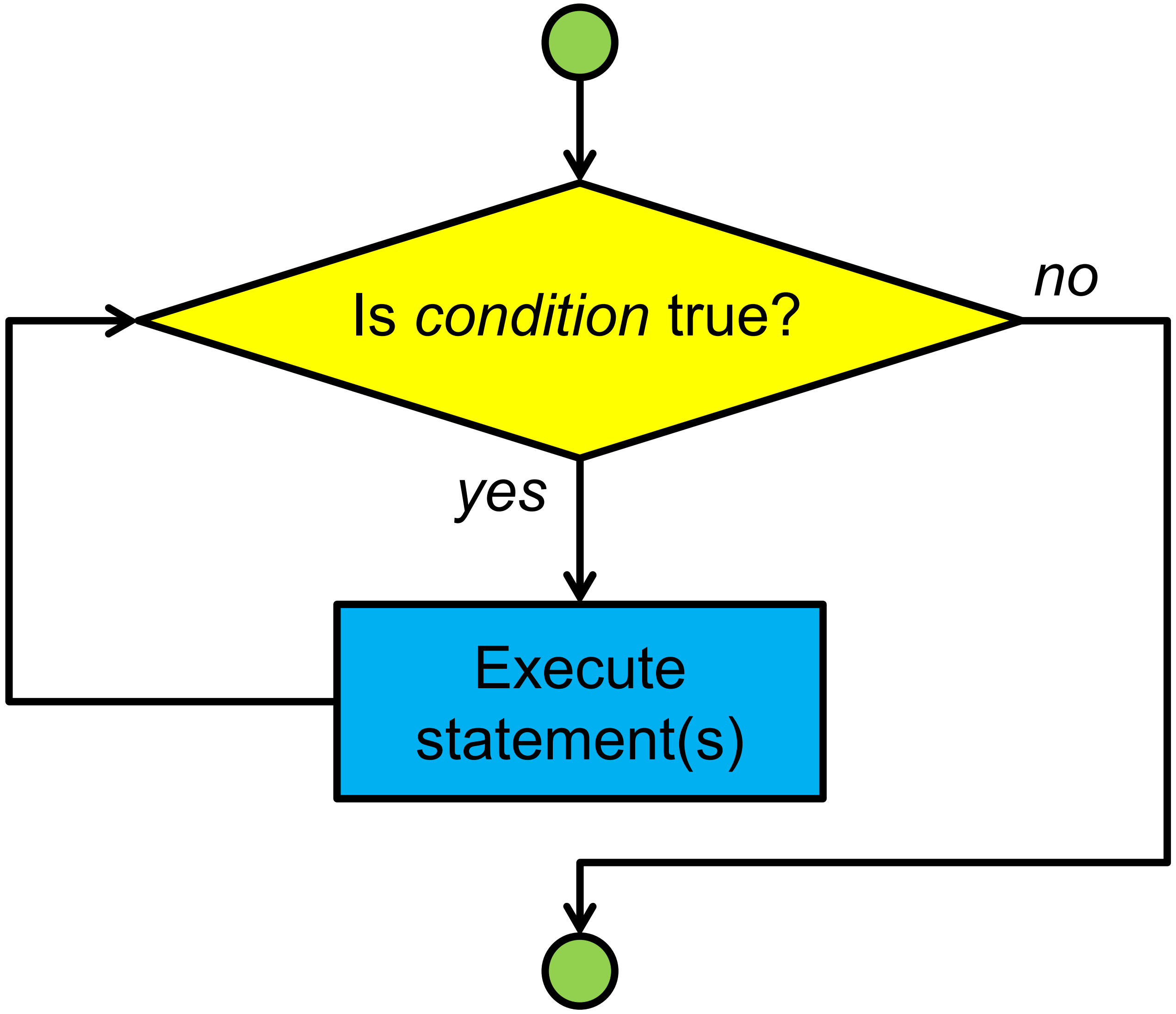
  - While Loops
  - Do Loops

While…Wend

```
While ...condition...
        ...statement(s)...
Wend
```

- While *condition* is true, execute *statement(s)* repeatedly
- When *condition* is false, the content of *While…Wend* is not executed any more

The Flow of While…Wend



Storing Things

- For the following examples we use variables that store text
- And we will also use variables that store integer numbers
- For example:

```
Dim MyFavouriteText As String 'stores text
Dim MyFavouriteNumber As Integer 'stores an integer
```

- Here’s some examples of using the variables:

```
MyFavouriteText = "you are a silly sausage"
MyFavouriteNumber = 8888
```

A Simple Example of While…Wend

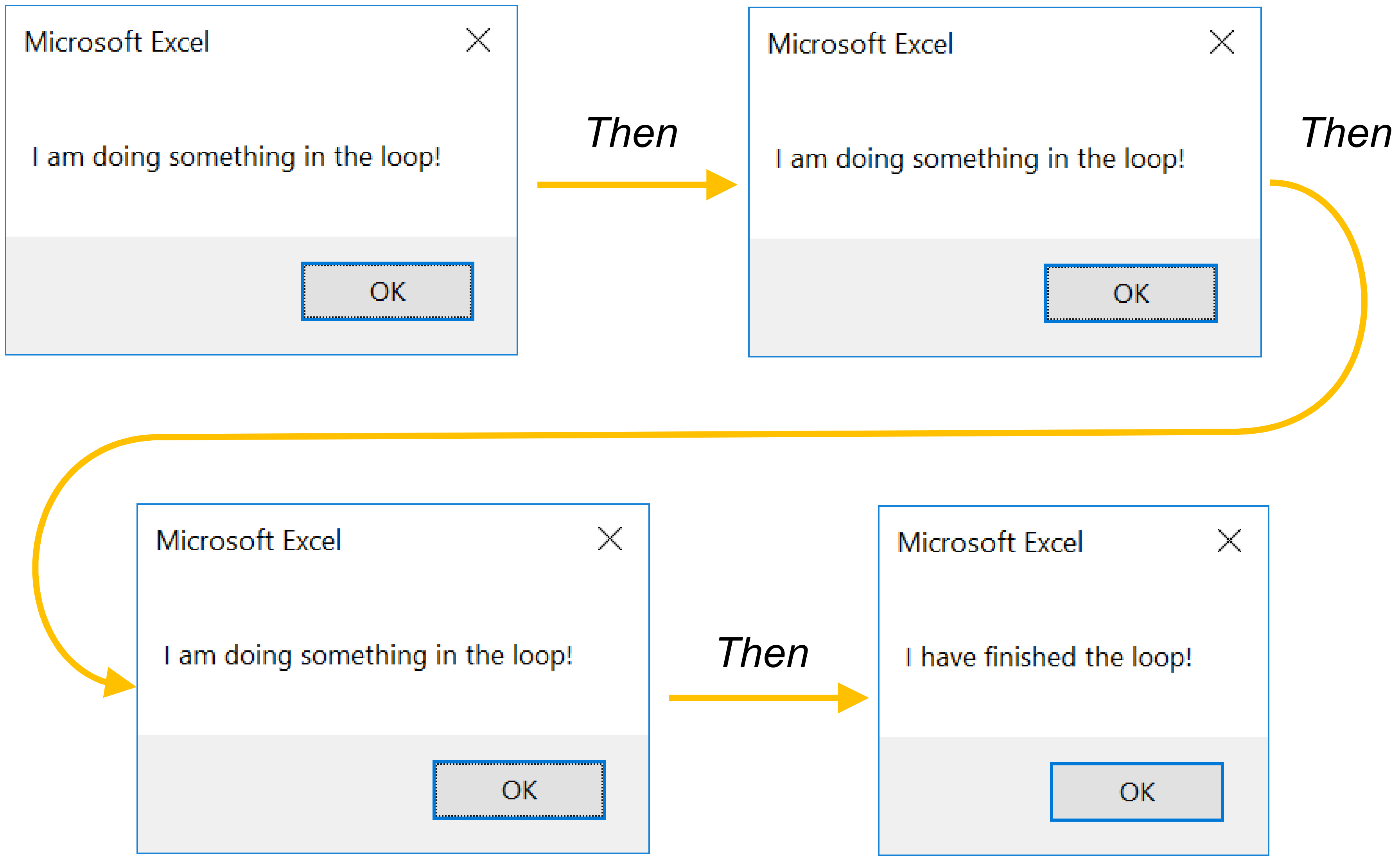
- Here is a simple example that runs the loop content three times

```
Dim Count As Integer

Count = 0
While Count < 3
    MsgBox "I am doing something in the loop!"
    Count = Count + 1
Wend

MsgBox "I have finished the loop!"
```

This is the loop condition. The loop content executes when it is True.

This is the loop content and it has two lines of code

Running The Example


I am doing something in the loop!

I am doing something in the loop!

I am doing something in the loop!

I have finished the loop!

Eating Candy

- In the following example the idea is that someone walks into a candy shop
- That person keeps buying and eating a candy bar, until there isn't enough money to buy more

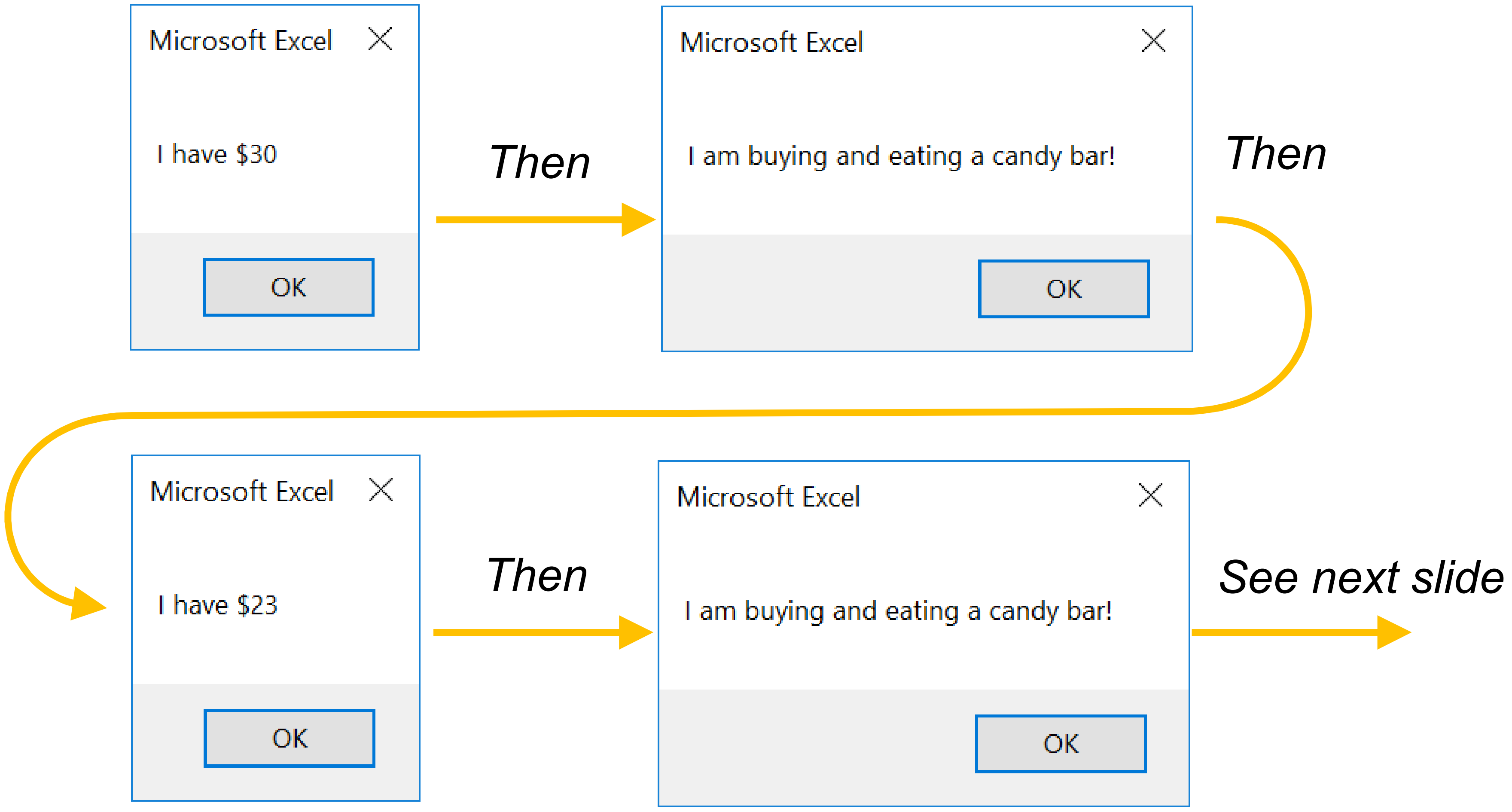
- This is the code to find how many candy bars you can eat using a loop

```
Dim Money As Integer
Dim CostOfCandyBar As Integer

Money = 30
CostOfCandyBar = 7

While Money >= CostOfCandyBar
    MsgBox "I have $" & Money
    MsgBox "I am buying and eating a candy bar!"
    Money = Money - CostOfCandyBar
Wend

MsgBox "Now, I only have $" & Money & " left."
MsgBox "I don't have enough money for any more :("
```

The loop will keep on running if you have enough money for a candy bar

The amount of money is updated after buying a candy bar

Running The Example


Microsoft Excel

I have $30

Microsoft Excel

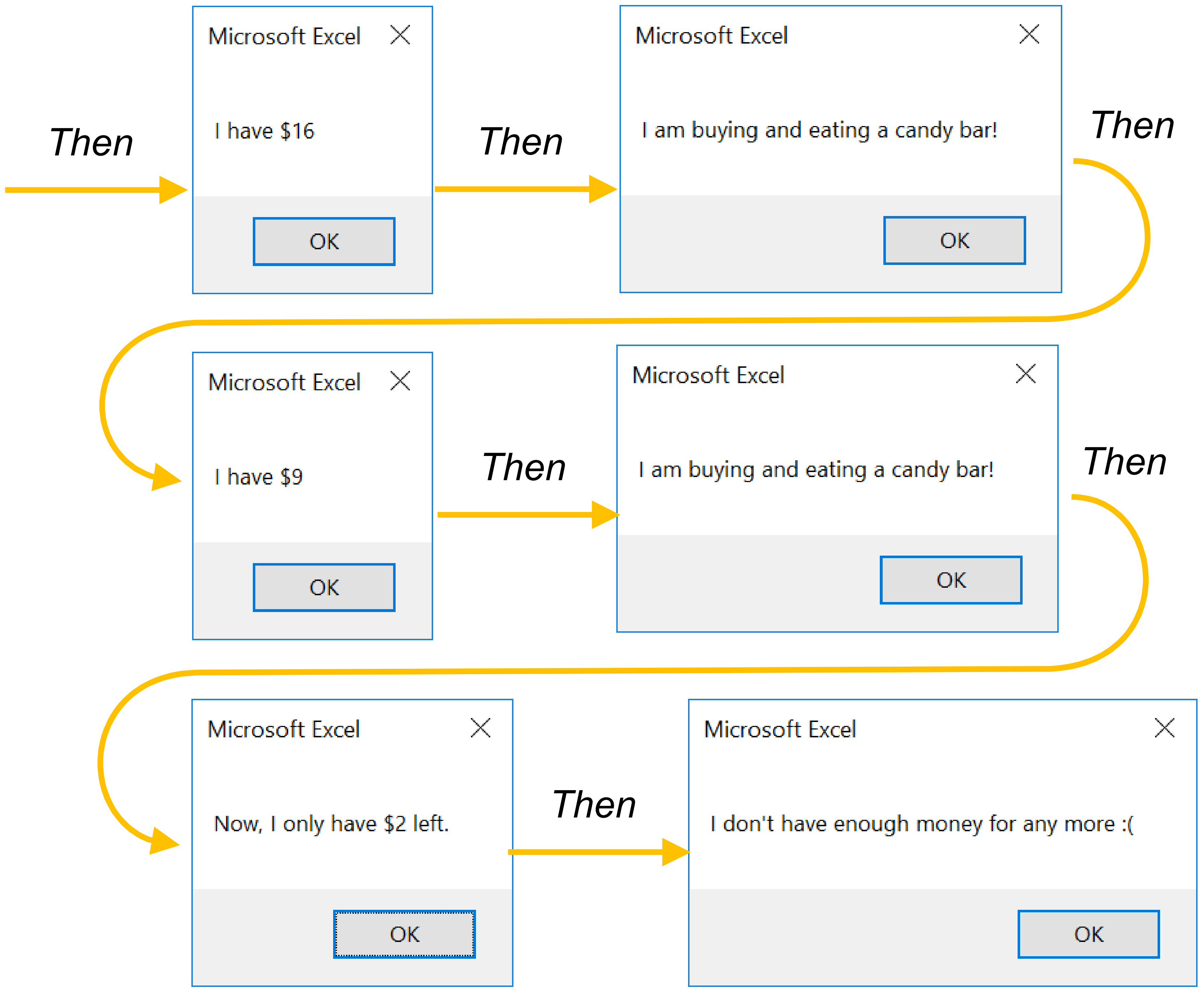
I am buying and eating a candy bar!

Microsoft Excel

I have $23

Microsoft Excel

I am buying and eating a candy bar!



```
I have $16
I am buying and eating a candy bar!
I have $9
I am buying and eating a candy bar!
Now, I only have $2 left.
I don't have enough money for any more :(
```

Improving the Example

- We can improve the example by telling you how many candy bars in total you can buy and eat
- In the improved code, it does everything the same as before, but it also counts how many bars have been bought and eaten
- The arrows ⇦ show the new 4 lines of code that have been added

```
Dim Money As Integer
Dim CostOfCandyBar As Integer
Dim Eaten As Integer
```

Create a variable to remember how many candy bars you have eaten

```
Money = 30
CostOfCandyBar = 7
Eaten = 0
```

You have not eaten any candy bar at the start

```
While Money >= CostOfCandyBar
    MsgBox "I have $" & Money
    MsgBox "I am buying and eating a candy bar!"
    Eaten = Eaten + 1
```

You have eaten one more candy bar

```
    Money = Money - CostOfCandyBar
Wend

MsgBox "Now, I only have $" & Money & " left."
MsgBox "I don't have enough money for any more :("
MsgBox "I ate " & eaten & " bars"
```

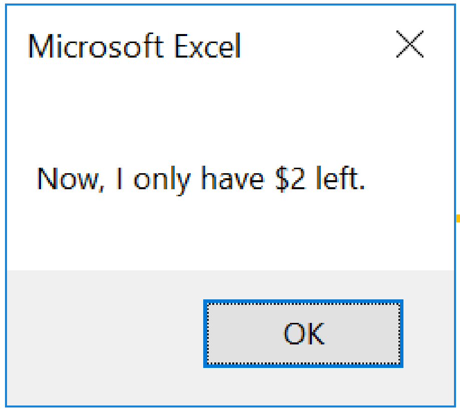
Show the count

Running The Example

The user will see the same windows as before,

with one extra window shown at the end

Same as before

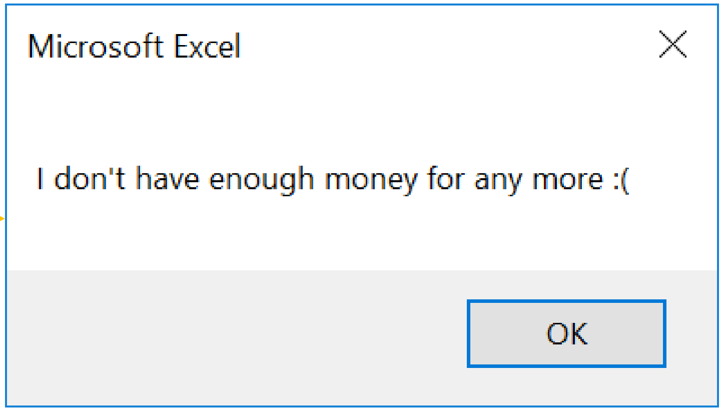


Microsoft Excel

Now, I only have $2 left.

Then

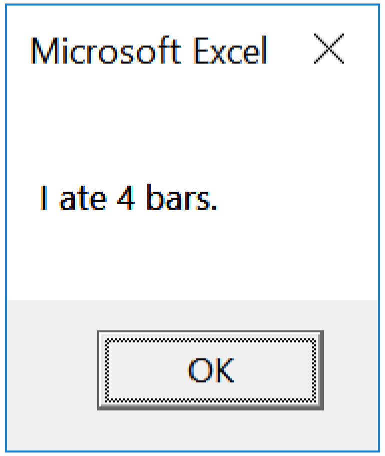
Same as before



Microsoft Excel

I don't have enough money for any more :(

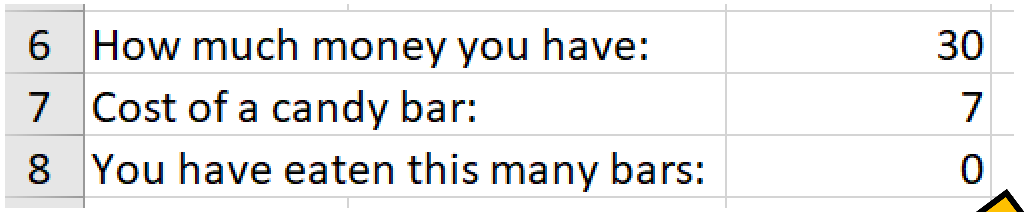
Then



Microsoft Excel

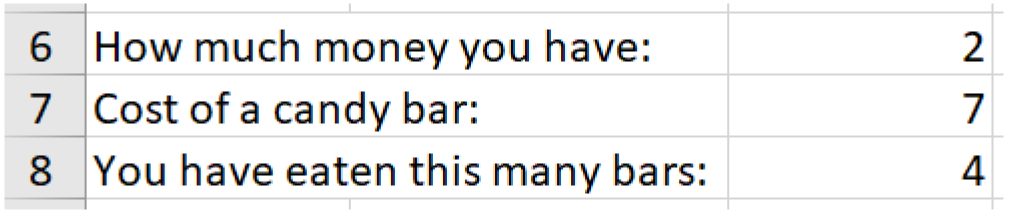
I ate 4 bars.

Using Cells Instead of Variables



6	How much money you have:	30
7	Cost of a candy bar:	7
8	You have eaten this many bars:	0

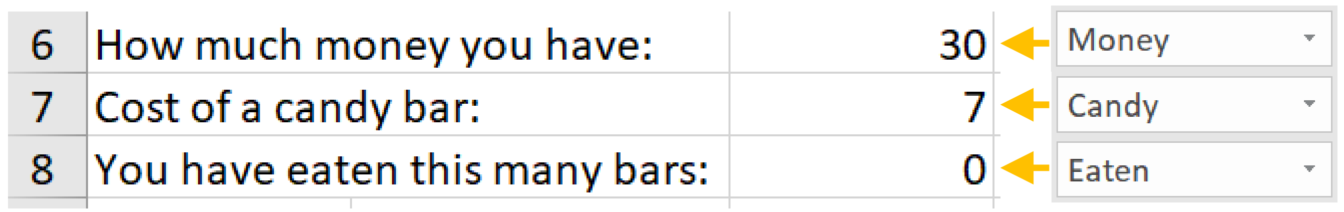
After running the macro:



6	How much money you have:	2
7	Cost of a candy bar:	7
8	You have eaten this many bars:	4

- Because we are using Excel, we could change our code so it uses cells instead of variables
- That's very useful – although if we were using another programming language, or if we were using VBA in another program such as Word or PowerPoint, we wouldn't be able to do that

Using Cells Instead of Variables



6	How much money you have:	30	Money
7	Cost of a candy bar:	7	Candy
8	You have eaten this many bars:	0	Eaten

The 3 cells used in this example are named ‘Money’, ‘Candy’ and ‘Eaten’

```
While Range("Money").Value >= Range("Candy").Value
    Range("Money").Value = _
        Range("Money").Value - _
        Range("Candy").Value

    Range("Eaten").Value = Range("Eaten").Value + 1
Wend

MsgBox "The loop has finished"
MsgBox "Take a look at the new values in the cells!"
```

The underscore (_) means the code is continued on the next line

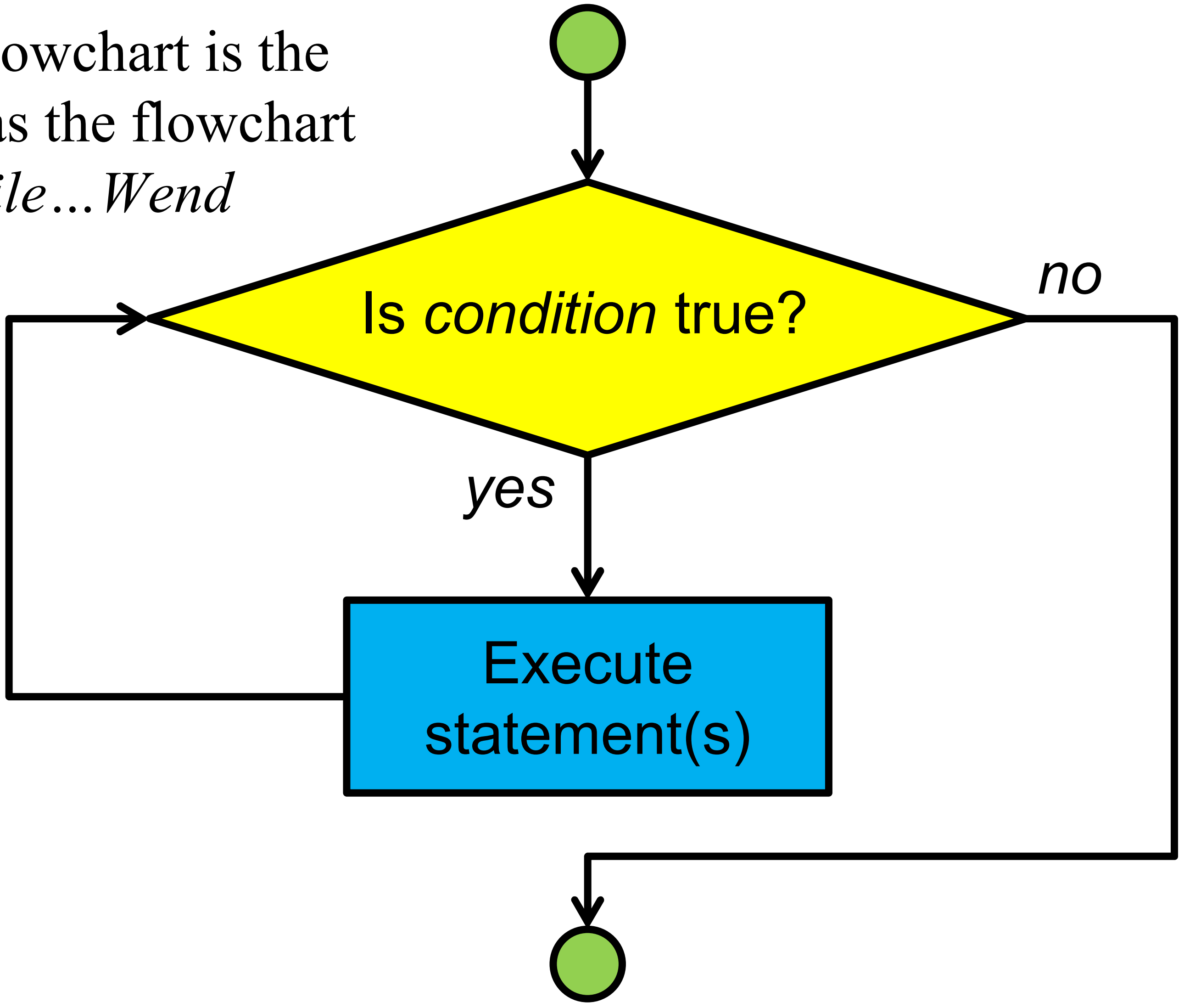
Do While…Loop

```
Do While ...condition...
        ...statement(s)...
Loop
```

- The usage of *Do While…Loop* is exactly the same as *While…Wend*
- One benefit is that the words *Do While…Loop* are perhaps more like English than *While…Wend*

The Flow of Do While…Loop

- This flowchart is the same as the flowchart of *While…Wend*



Example of Do While…Loop

- For example, we can create a program which does the same thing as the last example using *Do While…Loop*

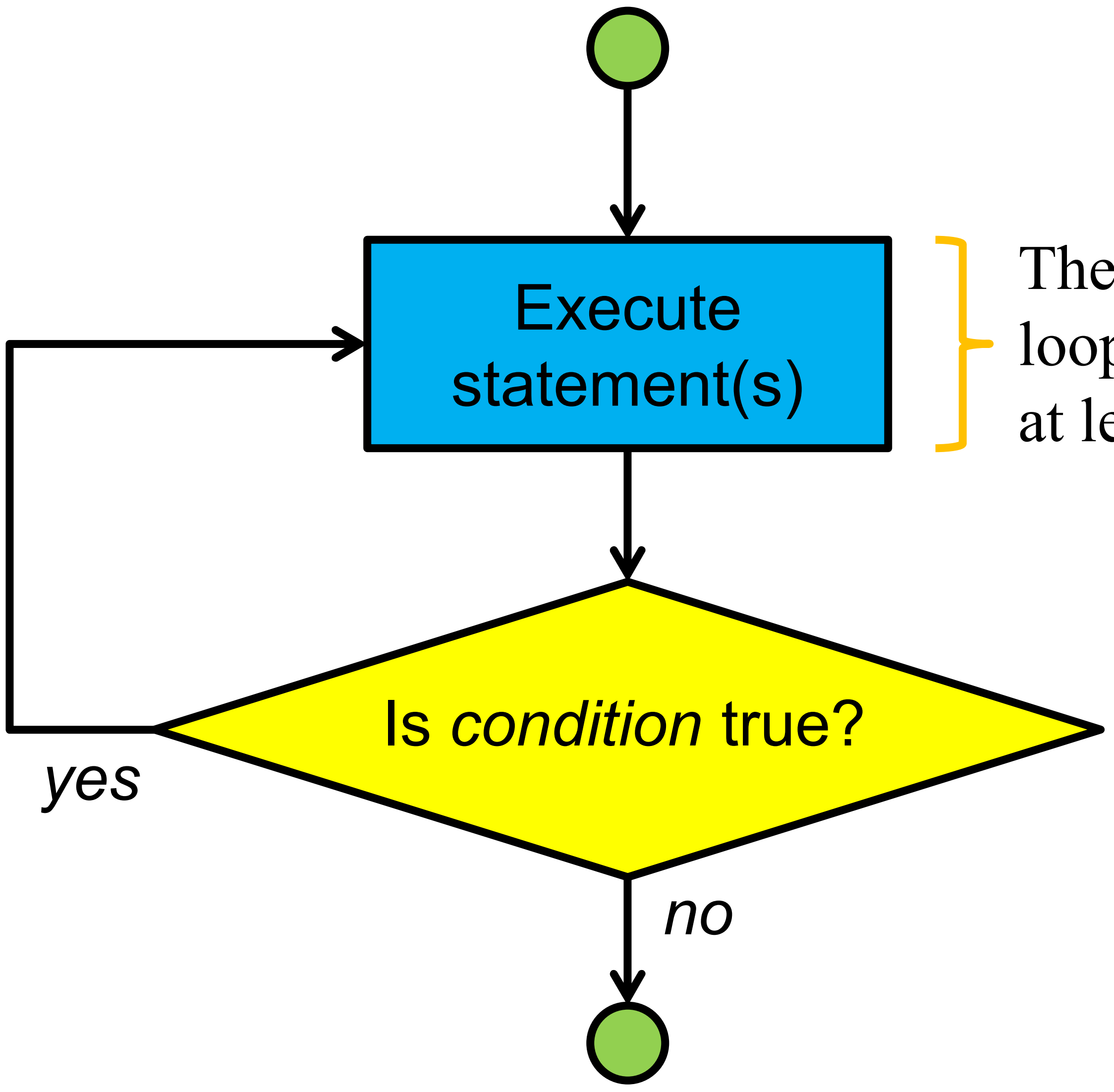
```
Do While Range("Money").Value >= Range("Candy").Value
    Range("Money").Value = _
        Range("Money").Value - _
        Range("Candy").Value
    Range("Eaten").Value = Range("Eaten").Value + 1
Loop

MsgBox "The loop has finished"
MsgBox "Take a look at the new values in the cells!"
```

Do…Loop While

```
Do
        . . .statement(s). . .
Loop While ...condition...
```

- This is similar to the previous two loops we looked at but *condition* is evaluated **after** *statement(s)* is executed
- This means that *statement(s)* will be executed **at least once**

The Flow of Do…Loop While



The Do…Loop While loop will be executed at least once

An Example of Do…Loop While 1/3

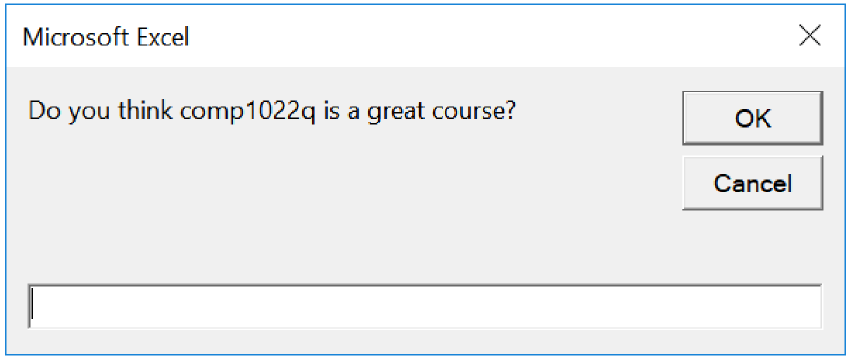
```
Dim Answer As String

' Execute the loop at least once
Do
    ' Ask a question
    Answer = InputBox( _
        "Do you think comp1022q is a great course?")

' Check the answer at the end of the loop
Loop While Answer <> "yes"

MsgBox "Thanks!"
```

Anything after ' gets ignored, so you can use it to write comments

<> means 'is not the same as'

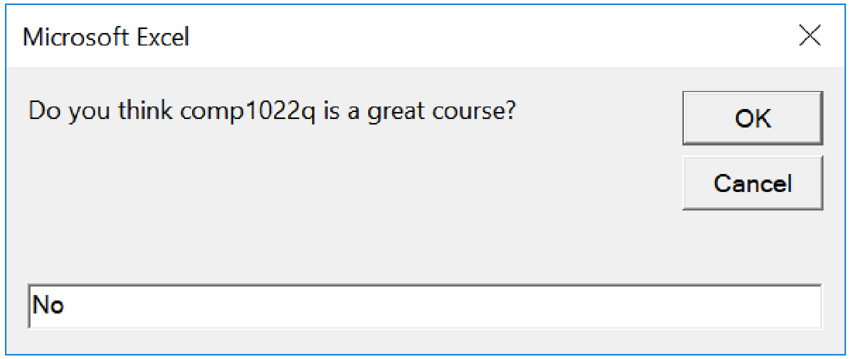
If this is True (meaning that the user did not enter "yes"), the loop will run again

An Example of Do…Loop While 1/3


Microsoft Excel

Do you think comp1022q is a great course?

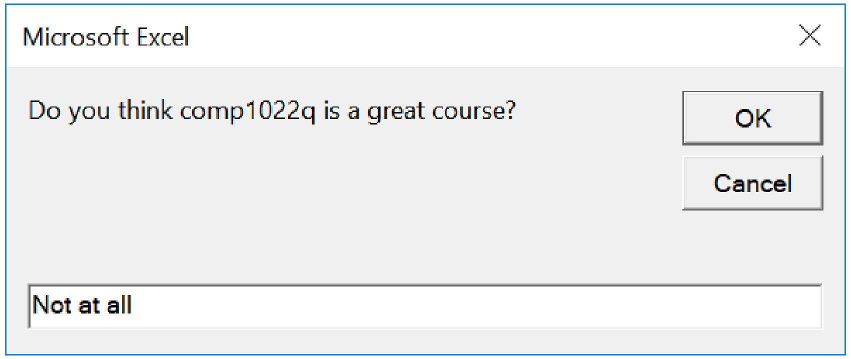
You need to answer a question in an InputBox


Microsoft Excel

Do you think comp1022q is a great course?

No

If you answer ‘No’ you will be asked again

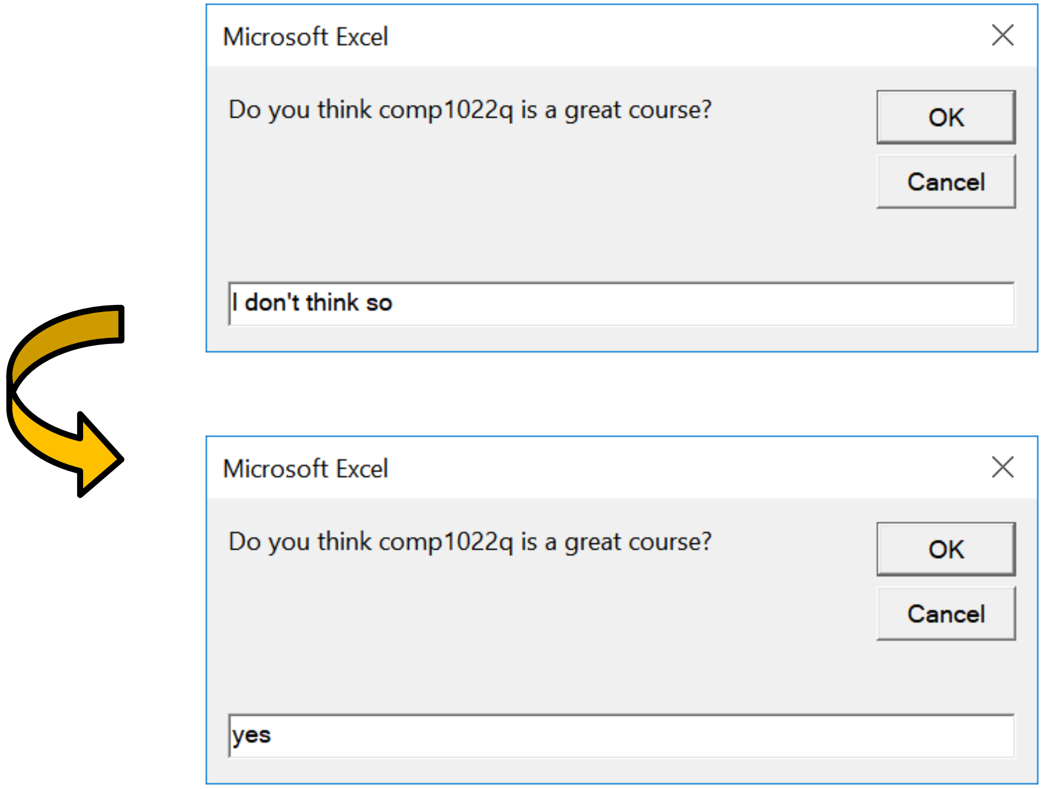

Microsoft Excel

Do you think comp1022q is a great course?

Not at all

If you answer ‘Not at all!’ you will be asked again

An Example of Do…Loop While 1/3



Do you think comp1022q is a great course?

I don't think so

Do you think comp1022q is a great course?

yes

If you answer 'I don't think so' you will be asked again

You will not be asked the question again if you answer 'yes'