COMP1022Q
Introduction to Computing with Excel VBA

Looping Part 2

David Rossiter, Gibson Lam, Eddie Chan

This Presentation

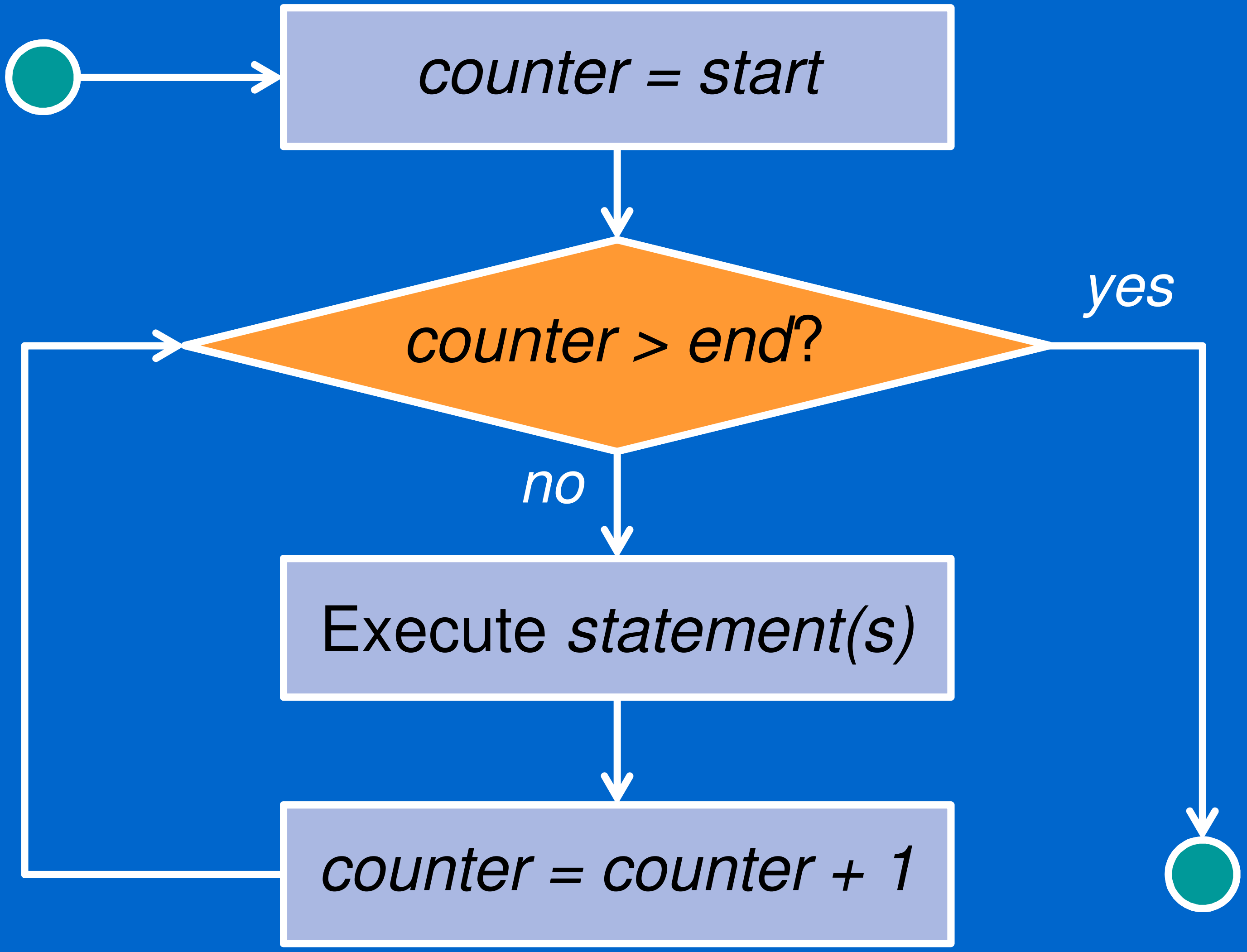
- Previously we discussed the use of while loops and two types of do loops
- In this presentation we will introduce for loops, and we look further at do loops

For…Next

```
For counter = start To end
        . . .statement(s). . .
Next counter
```

- *For…Next* uses a *counter* that is equal to *start* at the start of the loop
- The *counter* increases after each iteration of the loop
- The loop executes up to and including the iteration when the value of *counter* is equal to *end*
- That means the number of times the loop repeats itself is (*end* - *start* + 1)

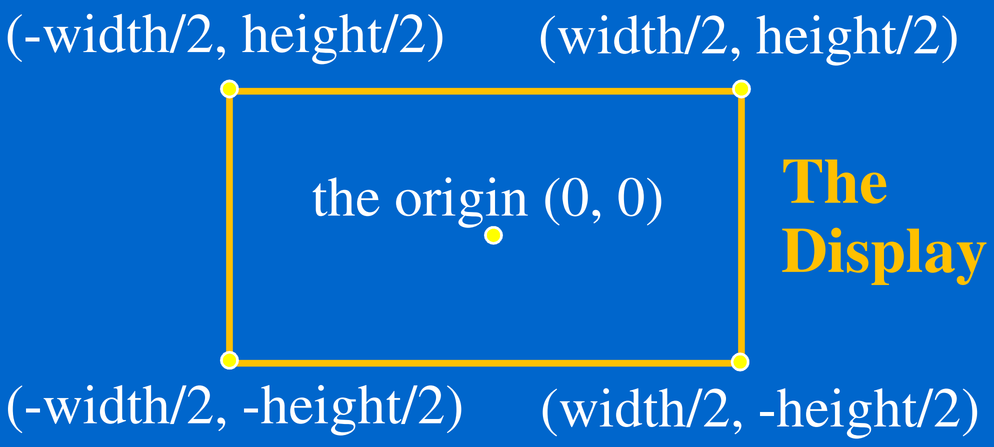
The Flow of For…Next



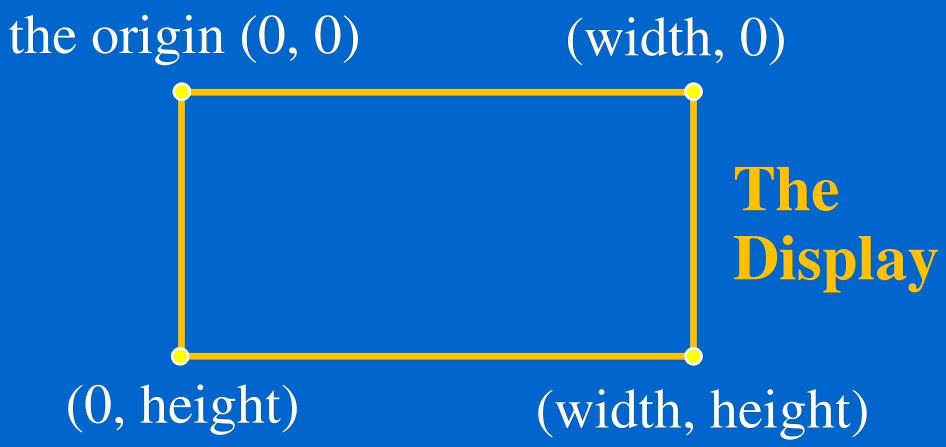
Coordinate System in VBA

- We will draw shapes using VBA in some of the examples later so let’s look at the coordinate system in VBA

- The cartesian coordinate system:
  - You probably used this system when you learned Maths at school



- The VBA coordinate system:
  - For example, when you do some programming with VBA shape objects, all the x and y values are positive



Adding a Shape Using VBA

- You can use the following code to add a shape in the currently selected worksheet:

Top left corner of the shape

```
ActiveSheet.Shapes.AddShape Shape, X, Y, _
                            Width, Height
```

This is the currently selected worksheet

- The `Shape` parameter is a number representing the shape that you want to draw
- If you don’t know the shape number an alternative way is to use some shape names for the `Shape` parameter such as `msoShapeRectangle` and `msoShapeOval`

An Example of For…Next (1/2)

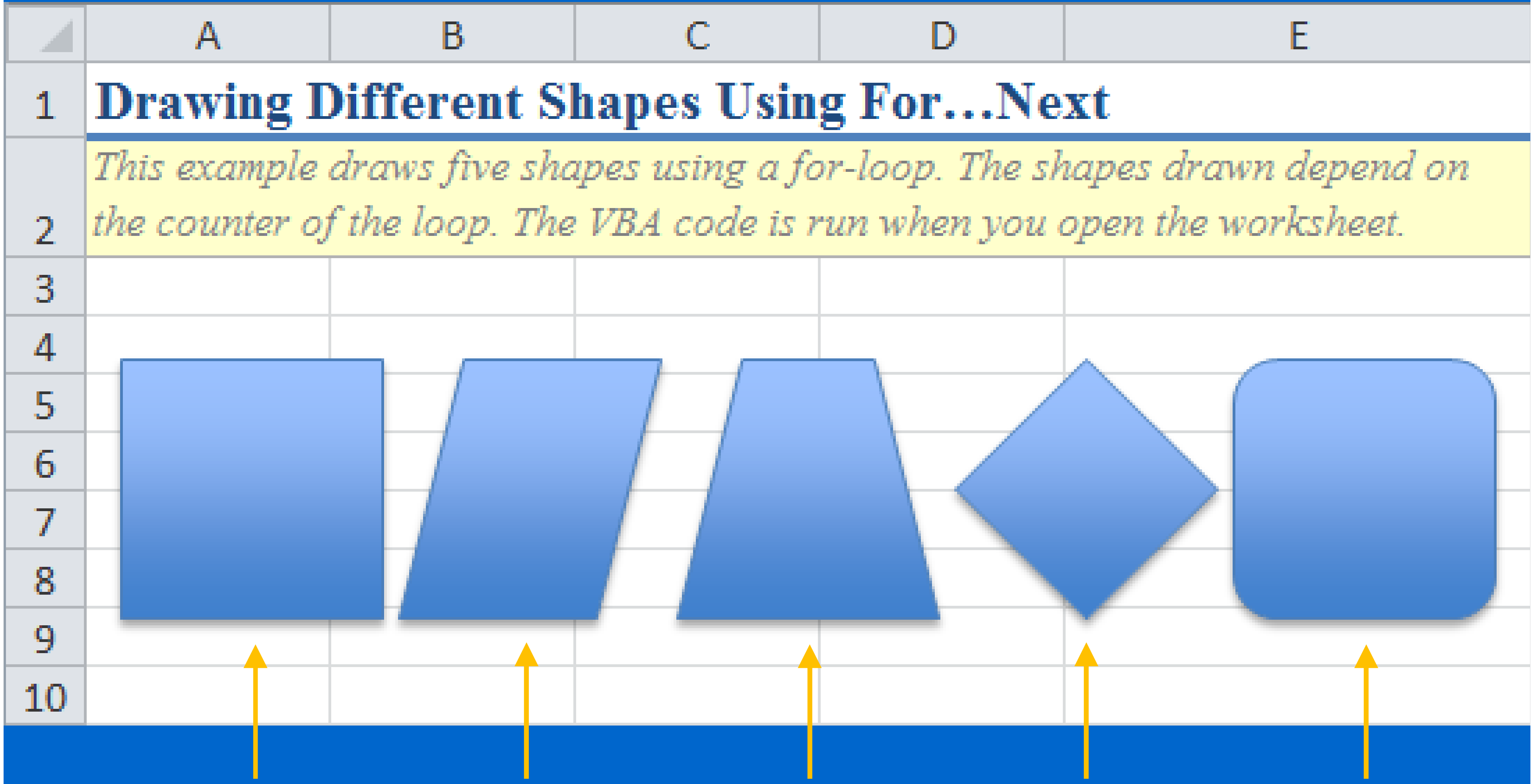
```
X = 10 ' x pos of the first shape

' Draw different shapes specified
' by the loop counter
For Shape = 1 To 5
    ' Draw a shape on the worksheet
    ActiveSheet.Shapes.AddShape Shape, _
                                X, 80, 70, 70

    ' Set the x position of the next shape
    X = X + 75
Next Shape
```

Loop counter

Loop body

Add a shape at (X, 80) with a size of 70 by 70 and the shape type specified by the loop counter

An Example of For…Next (2/2)



Drawing Different Shapes Using For…Next

This example draws five shapes using a for-loop. The shapes drawn depend on the counter of the loop. The VBA code is run when you open the worksheet.

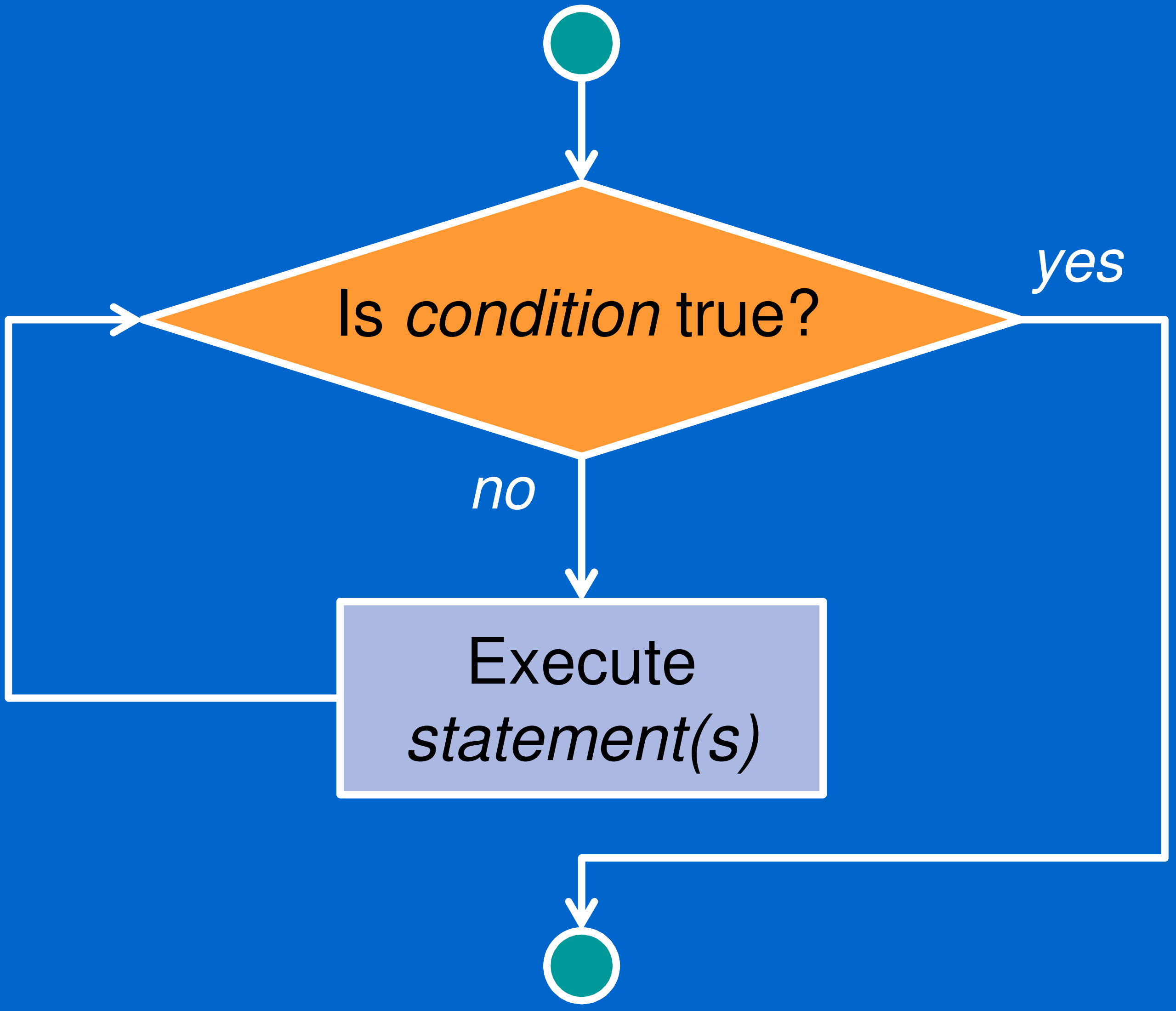
Shape = 1 *Shape = 2* *Shape = 3* *Shape = 4* *Shape = 5*

Do Until…Loop

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

- This is similar to *Do While…Loop* we saw before
- However, the evaluation of the stopping condition is the opposite of *Do While…Loop*, i.e. *Do Until…Loop* stops when *condition* is true
- Remember *Do While…Loop* stops when *condition* is false

The Flow of Do Until…Loop



An Example of Do Until…Loop (1/3)

```
Angle = 0

' Draw squares until nine are drawn
Do Until ActiveSheet.Shapes.Count = 9
    ' Draw an unfilled square
    Set Square = ActiveSheet.Shapes.AddShape( _
        msoShapeRectangle, 125, 100, 150, 150)
    Square.Fill.Visible = msoFalse

    ' Rotate the square by an angle
    Square.Rotation = Angle

    ' Increase the angle by 10
    Angle = Angle + 10
Loop
```

Loop condition

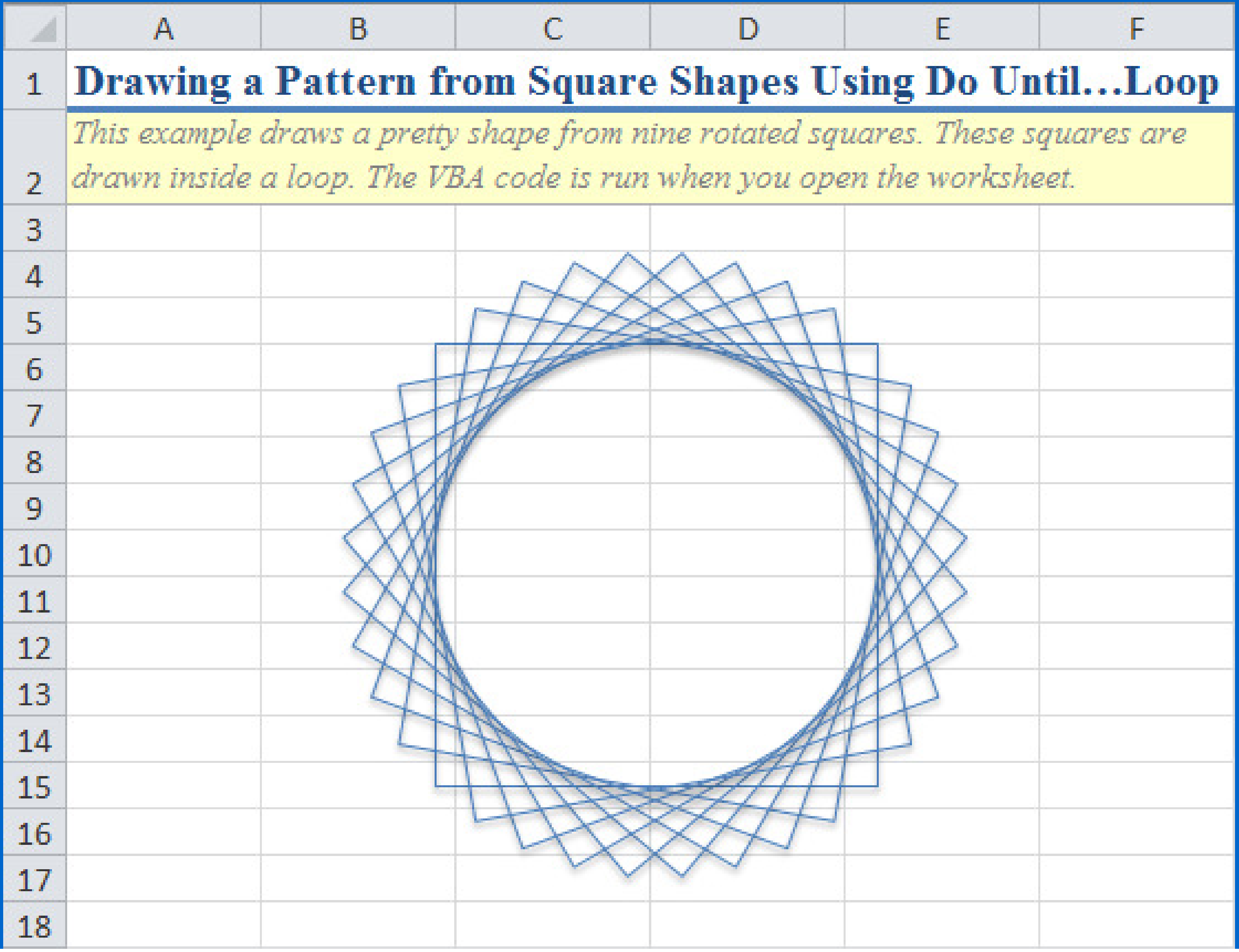
Loop body

'mso' means 'Microsoft Object'

Add a hollow square at (125, 100) with a size of 150 by 150, and then store the shape into a variable

Rotate the newly drawn square

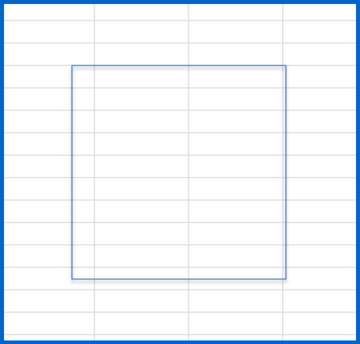
An Example of Do Until…Loop (2/3)



Drawing a Pattern from Square Shapes Using Do Until…Loop

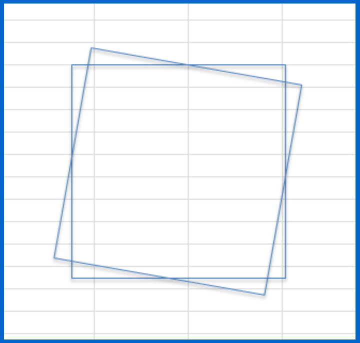
This example draws a pretty shape from nine rotated squares. These squares are drawn inside a loop. The VBA code is run when you open the worksheet.

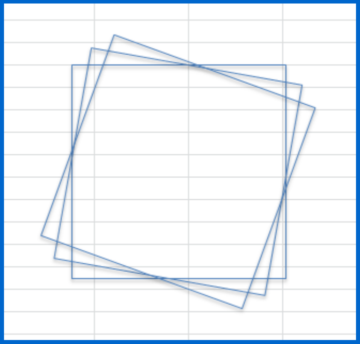
An Example of Do Until…Loop (3/3)



ActiveSheet.
Shapes.Count = 1
(Angle = 0)

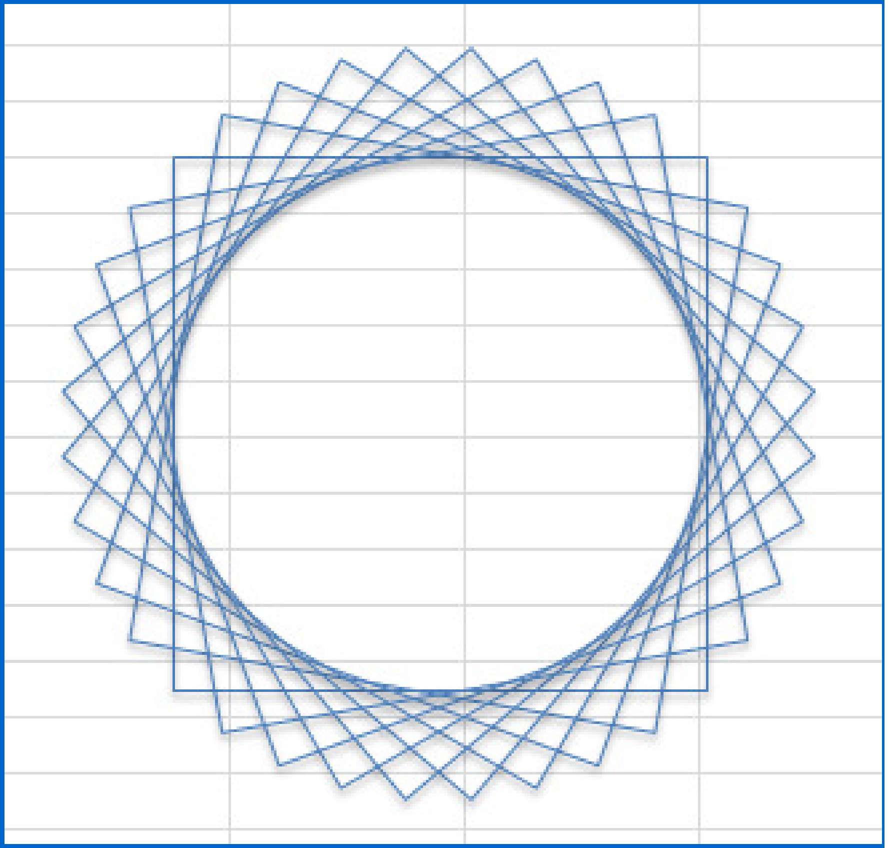
ActiveSheet.
Shapes.Count = 2
(Angle = 10)





ActiveSheet.
Shapes.Count = 3
(Angle = 20)

ActiveSheet.Shapes.Count = 9
(Angle = 80)



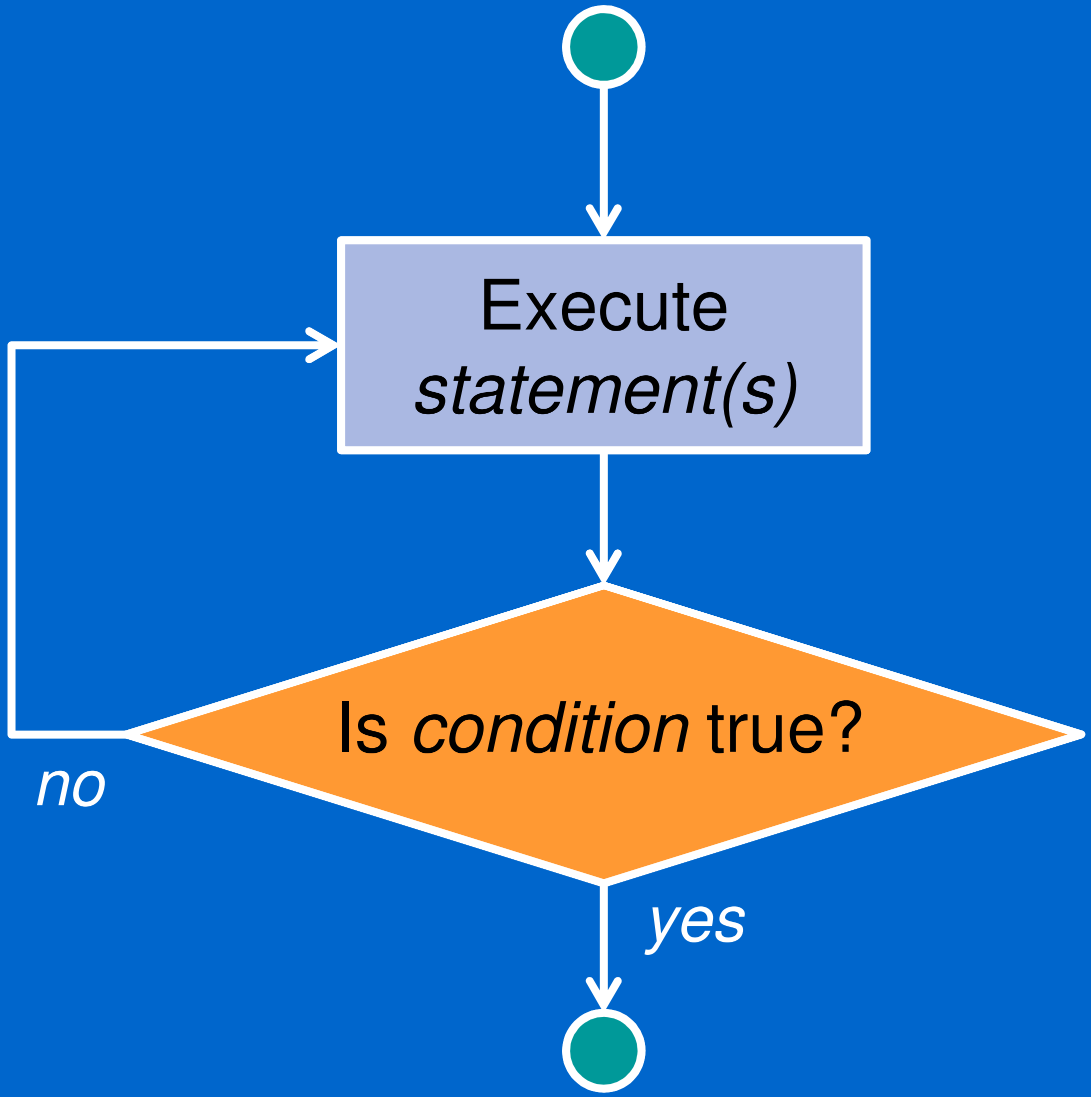
Repeat loop 9 times in total

Do…Loop Until

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

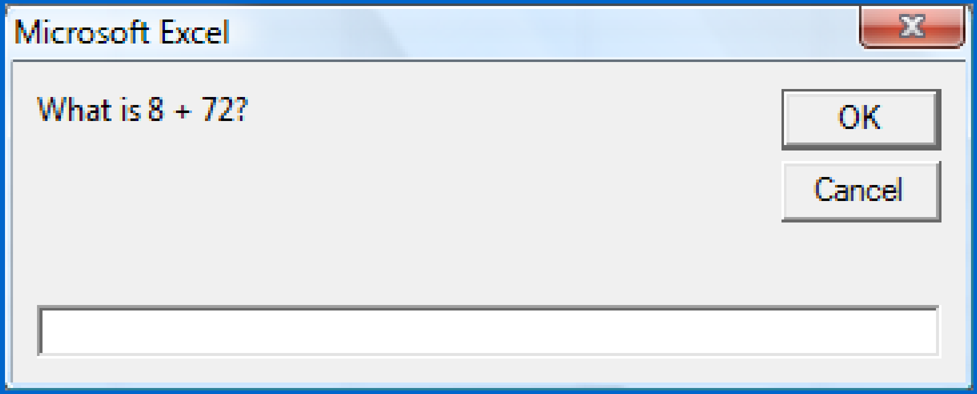
- This is similar to *Do Until…Loop* but *condition* is evaluated **after** *statement(s)* is executed
- This means that *statement(s)* will be executed at least once before *condition* is evaluated

The Flow of Do…Loop Until



An Example of Do…Loop Until

- This example creates a simple math addition question using random numbers
- Two random integers between the range of 1 to 100 are generated
- The user is then asked what the sum of these two numbers is, like this:



What is 8 + 72?

Random Numbers in VBA

- Random numbers can be generated in VBA using the *Rnd* function
- The *Rnd* function generates a real number smaller than 1 and bigger than or equal to 0, for example:

```
    RandomNumber = Rnd()
```

- However, in this example we want to generate a random integer between in the range 1 to 100

Generating Random Integers

- To generate a random integer in the range of 1 to 100 you will need to do these steps:
  1. Generate a number between 0 to 0.99999 using the *Rnd* function

```
RandomNumber = Rnd()           'range = [0,1)
```

  2. Multiply the generated number by 100

```
RandomNumber = Rnd()*100       'range = [0,100)
```

  3. Convert the number to an integer using the *Int* function

```
RandomNumber = Rnd()*100       'range = [0,99]
```

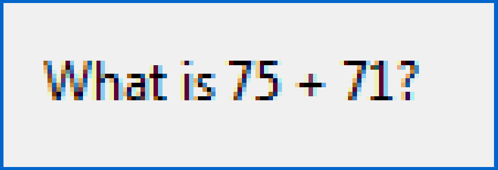
  4. Add 1 to the number

```
RandomNumber = Rnd()*100 +1    'range = [1,100]
```

Randomness of the Rnd Function

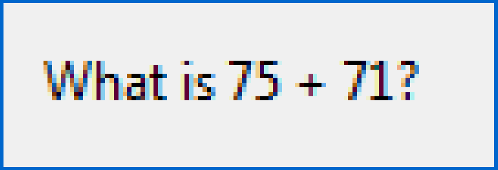
- You will find that every time you run your code you will get the same series of random numbers!
  - For example,
    - The first time your program asks a random math question:
    - Later you run the program the second time it will ask the same question again!

1st time you run it:



```
What is 75 + 71?
```

2nd time you run it:



```
What is 75 + 71?
```

- That means any game which uses the random numbers will be the same every time you play it
- To change this, you need to use *Randomize*

Simple Math Test (1/2)

```
' Randomize the random number generated by Rnd
Randomize
```

You need to call Randomize to ensure that the numbers generated are really random every time

```
' Create the first number in the range 1 to 100
Number1 = Rnd() * 100 + 1
```

In computers, a multiply is handled before an addition

```
' Create the second number in the range 1 to 100
Number2 = Rnd() * 100 + 1

' Calculate the answer and store it as string
Answer = Number1 + Number2
```

Continued on the next slide…

Simple Math Test (2/2)

Continued from the previous slide…

```
' Execute the loop at least once
Do
```

Loop body

```
    ' Ask the question
    Guess = InputBox("What is " & Number1 & _
                          " + " & Number2 & "?")
```

```
' Check the answer at the end of the loop
Loop Until Answer = Guess
```

Loop condition

```
MsgBox "Excellent, you have got the " & _
       "correct answer!"
```