CS2034B / DH2144B

# Data Analytics: Principles and Tools

Western
UNIVERSITY · CANADA

## Week 8

Programming Part 3

# Programming Part 3:

## Arrays and More Loops

# Arrays

- An array is a collection of values of the *same data type*.
- Like a Range that is stored only in memory and not on a worksheet.
- There are two types in VBA:
  - Static arrays have a fixed length that is determined when it is created (when we are writing the code).
  - Dynamic array's size is set at run time (when the code is running).

# Arrays

## Declaring Static Arrays

- Declared using `Dim` keyword like variables but have extra syntax:

```
Dim arrayName(first_index to last_index) as dataType
```

**The name of the array**
Just like a variable name

**The data type of the elements in the array**
Just like a variable data type (Integer, Double, String, etc.)

# Arrays

## Declaring Static Arrays

- Declared using `Dim` keyword like variables but have extra syntax:

```
Dim arrayName(first_index to last_index) as dataType
```

**The start of the array**

Like ranges, arrays have an index that allows you to identify the $n^{th}$ item. However, unlike ranges they do not have to start at 1. This sets at what index (number) the array will start at.

# Arrays

## Declaring Static Arrays

- Declared using `Dim` keyword like variables but have extra syntax:

```
Dim arrayName(first_index to last_index) as dataType
```

**The end of the array**

This sets the last index in the array.

The size of the array (how many elements it can hold) will be equal to: last_index – first_index + 1.

# Arrays

## Declaring Static Arrays

- Some examples:

```
Dim ints(1 To 10) As Integer
```

```
Dim dubs(5 To 8) As Double
```

```
Dim names(1 To 4) As String
```

# Arrays

## Declaring Static Arrays

- Some examples:

```
Dim ints(1 To 10) As Integer
```

**Array of 10 Integers named ints starting at ints(1) and going to ints(10)**

```
Dim dubs(5 To 8) As Double
```

```
Dim names(1 To 4) As String
```

# Arrays

## Declaring Static Arrays

- Some examples:

```
Dim ints(1 To 10) As Integer
```

```
Dim dubs(5 To 8) As Double
```

**Array of 4 Doubles named dubs starting at dubs(5) and going to dubs(8)**

```
Dim names(1 To 4) As String
```

# Arrays

## Declaring Static Arrays

- Some examples:

```
Dim ints(1 To 10) As Integer
```

```
Dim dubs(5 To 8) As Double
```

**Array of 4 Strings named names starting at names(1) and going to names(4)**

```
Dim names(1 To 4) As String
```

# Arrays

## Declaring Static Arrays

- We can also give a single number (the last_index) and it will be assumed that the first_index is 0:

```
Dim i(10) As Integer

Dim s(5) As String
```

**Would be equivalent to:**

```
Dim i(0 To 10) As Integer

Dim s(0 To 5) As String
```

# Arrays

## Assigning Values

- We assign values to an array by using the name of the array and an index number of the element to set.
- Example:

```
Dim arrMarks(1 To 4) As Long

arrMarks(1) = 5

arrMarks(3) = 46
```

arrMarks(5) = 99

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| | | | |

# Arrays

## Assigning Values

- We assign values to an array by using the name of the array and an index number of the element to set.
- Example:

```
Dim arrMarks(1 To 4) As Long
```

```
arrMarks(1) = 5
```

```
arrMarks(3) = 46
```

arrMarks(5) = 99

**By default all values are 0 when first declared**

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| 0 | 0 | 0 | 0 |

# Arrays

## Assigning Values

- We assign values to an array by using the name of the array and an index number of the element to set.
- Example:

```
Dim arrMarks(1 To 4) As Long

arrMarks(1) = 5

arrMarks(3) = 46
```

arrMarks(5) = 99

**Set 1st element to 5**

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| 5 | 0 | 0 | 0 |

# Arrays

## Assigning Values

- We assign values to an array by using the name of the array and an index number of the element to set.
- Example:

```
Dim arrMarks(1 To 4) As Long

arrMarks(1) = 5

arrMarks(3) = 46
```

arrMarks(5) = 99

**Set 3rd element to 46**

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| 5 | 0 | 46 | 0 |

# Arrays

## Assigning Values

- We assign values to an array by using the name of the array and an index number of the element to set.
- Example:

```
Dim arrMarks(1 To 4) As Long

arrMarks(1) = 5

arrMarks(3) = 46

arrMarks(5) = 99
```

**Causes a #VALUE! error as there is no 5th element. This would be outside the bounds of the array!**

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| 5 | 0 | 46 | 0 |

# Arrays

## Reading Values

- We can retrieve a value from an array in a similar manner.
- Example: Using arrMarks from before print the 2nd and 3rd values.

```
Debug.Print arrMarks(2)
Debug.Print arrMarks(3)
```

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| 5 | 0 | 46 | 0 |

# Arrays

## Reading Values

- We can retrieve a value from an array in a similar manner.
- Example: Using arrMarks from before print the 2nd and 3rd values.

```
Debug.Print arrMarks(2)
Debug.Print arrMarks(3)
```

**Output:**

```
0
46
```

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| 5 | 0 | 46 | 0 |

# Arrays

## Bounds

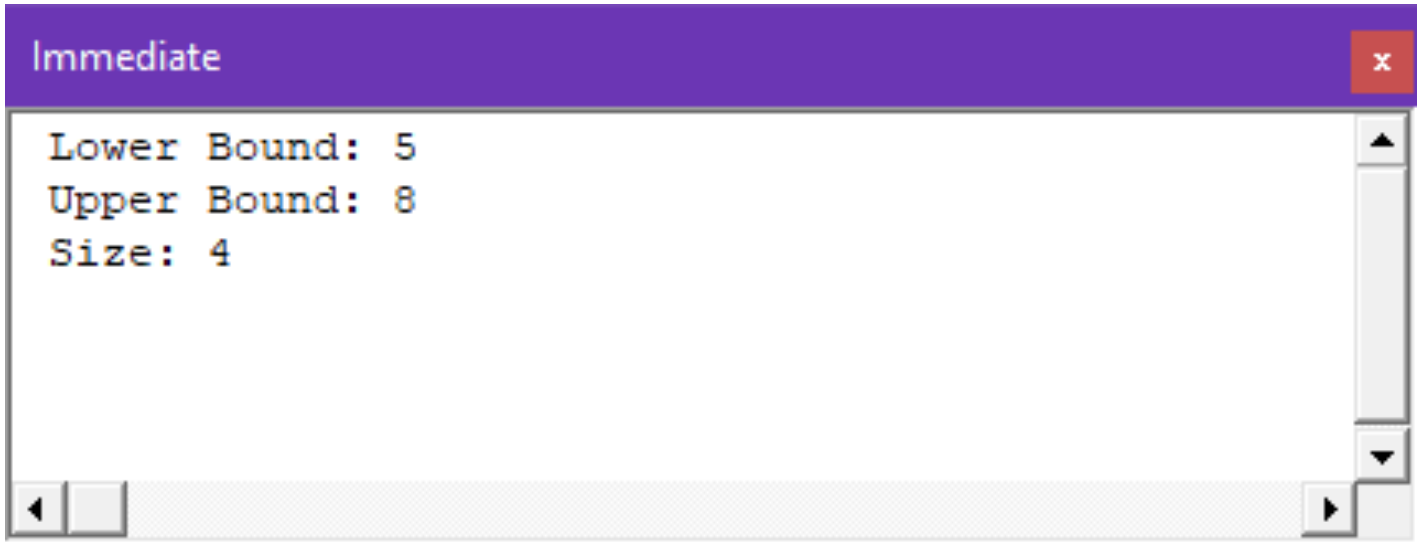
- We can find the bounds of an array using the built in `LBound` and `UBound` functions.
- `LBound(arrayName)` returns the first_index, the lowest element index.
- `UBound(arrayName)` returns the last_index, the largest element index.
- The size of an array is equal to:

```
UBound(arrayName) - LBound(arrayName) + 1
```

# Arrays

## Bounds Example:

```
Dim dubs(5 To 8) As Double

Debug.Print "Lower Bound: " & LBound(dubs)
Debug.Print "Upper Bound: " & UBound(dubs)
Debug.Print "Size: " & UBound(dubs) - LBound(dubs) + 1
```



# Arrays

## Using Arrays with Loops

- For loops are ideal for looping through each element in the array and performing some operation.
- We can use the LBound and UBound functions to set the starting and ending point of the for loop's counter such that the loops index matches the arrays.
- Example:

```
For i = LBound(arrayName) To UBound(arrayName)
```

- Inside the loop `arrayName(i)` will then be equal to the $i^{th}$ element in the array and the loop will run once for each element.

# Arrays

## Example: Print Movies

Write a function that creates an array and fills it with some of your favorite movies. Then use a for loop to print the list to the immediate window.

```
Function MoviePrint()
    Dim movies(1 To 5) As String
    movies(1) = "Inception"
    movies(2) = "The Martian"
    movies(3) = "Terminator 2"
    movies(4) = "The Matrix"
    movies(5) = "Moon"

    Debug.Print "My Top 5 Movie List:"
    For i = LBound(movies) To UBound(movies)
        Debug.Print movies(i)
    Next i
End Function
```

# Arrays

## Example: Print Movies

Write a function that creates an array and fills it with some of your favorite movies. Then use a for loop to print the list to the immediate window.

```
Function MoviePrint()
    Dim movies(1 To 5) As String
    movies(1) = "Inception"
    movies(2) = "The Martian"
    movies(3) = "Terminator 2"
    movies(4) = "The Matrix"
    movies(5) = "Moon"

    Debug.Print "My Top 5 Movie List:"
    For i = LBound(movies) To UBound(movies)
        Debug.Print movies(i)
    Next i
End Function
```

**Declare the movies array and initialize it with some movie names.**

# Arrays

## Example: Print Movies

Write a function that creates an array and fills it with some of your favorite movies. Then use a for loop to print the list to the immediate window.

```
Function MoviePrint()
    Dim movies(1 To 5) As String
    movies(1) = "Inception"
    movies(2) = "The Martian"
    movies(3) = "Terminator 2"
    movies(4) = "The Matrix"
    movies(5) = "Moon"

    Debug.Print "My Top 5 Movie List:"
    For i = LBound(movies) To UBound(movies)
        Debug.Print movies(i)
    Next i
End Function
```

**Create a for loop with index (counter) i that starts at the lowest index of the movies array and runs to the highest index.**

**This loop will visit each element of the array.**

# Arrays

## Example: Print Movies

Write a function that creates an array and fills it with some of your favorite movies. Then use a for loop to print the list to the immediate window.

```
Function MoviePrint()
    Dim movies(1 To 5) As String
    movies(1) = "Inception"
    movies(2) = "The Martian"
    movies(3) = "Terminator 2"
    movies(4) = "The Matrix"
    movies(5) = "Moon"

    Debug.Print "My Top 5 Movie List:"
    For i = LBound(movies) To UBound(movies)
        Debug.Print movies(i)
    Next i
End Function
```

**Print the value of the $i^{th}$ element of the movies array.**

# Arrays

## Example: Print Movies

Write a function that creates an array and fills it with some of your favorite movies. Then use a for loop to print the list to the immediate window.

Immediate

```
My Top 5 Movie List:
Inception
The Martian
Terminator 2
The Matrix
Moon
```

# Arrays

## Passing Arrays to Functions

- Like other variables, arrays can be passed as arguments to functions.
- If we know the data type of an array before hand, the function header would look like this (for type Integer):

```
Function functionName(arrayParamName() As Integer)
```

- If we do not know the data type ahead of time, we can define the function like so:

```
Function functionName(arrayParamName)
```

# Arrays

## Passing Arrays to Functions

- Inside the function we can access the values of the array parameter like we would any other array (using the subscript notation). E.g. `arrayName(5)` for the 5th element.
- Array arguments are not the same as Ranges (E.g. A1:A5) and we have to pass arrays to functions differently from Excel.
- **Example:** Send an array with the values “cat”, “dog” “duck” to the function myArrayFunc from Excel.

```
=myArrayFunc({"cat","dog","duck"})
```

# Arrays

## Passing Arrays to Functions

- Inside the function we can access the values of the array parameter like we would any other array (using the subscript notation). E.g. `arrayName(5)` for the 5th element.
- Array arguments are not the same as Ranges (E.g. A1:A5) and we have to pass arrays to functions differently from Excel.
- **Example:** [illegible] "duck" to [illegible]

**Denote the beginning and ending of a set of array values. This is treated as one single argument to the function.**

```
=myArrayFunc({"cat","dog","duck"})
```

# Arrays

## Passing Arrays to Functions

- Inside the function we can access the values of the array parameter like we would any other array (using the subscript notation). E.g. `arrayName(5)` for the 5th element.
- Array arguments are not the same as Ranges (E.g. A1:A5) and we have to pass arrays to functions differently from

**Header for myArrayFunc would be:**

```
Function myArrayFunc(arrayParamName)
```

```
=myArrayFunc({"cat","dog","duck"})
```

# Arrays

## Passing Arrays to Functions

- We can also pass an array from one function to another by calling a function from inside our function.
- If the function we are call, for example `func2`, returns a result we need to save or do something with that result. **For Example:**

```
Dim result As Integer
result = func2()
```

- If the function we are calling does not return a result or we do not want do anything with it. We need to use the `Call` keyword. **For Example:**

```
Call func2()
```

# Arrays

## Passing Arrays to Functions

**Example:** Create a function, func1, that creates an Integer array and passes it to function func2 which then prints its values to the immediate window.

```
Function func1()
    Dim a(1 To 4) As Integer
    a(1) = 5
    a(2) = -6
    a(3) = 1000
    a(4) = 367

    Call func2(a)
End Function
```

```
Function func2(arr() As Integer)
    Dim i As Integer

    For i = LBound(arr) To UBound(arr)
        Debug.Print arr(i)
    Next i
End Function
```

# Arrays

## Passing Arrays to Functions

**Example:** Create a function, func1, that creates an Integer array and passes it to function func2 which then prints its values to the immediate window.

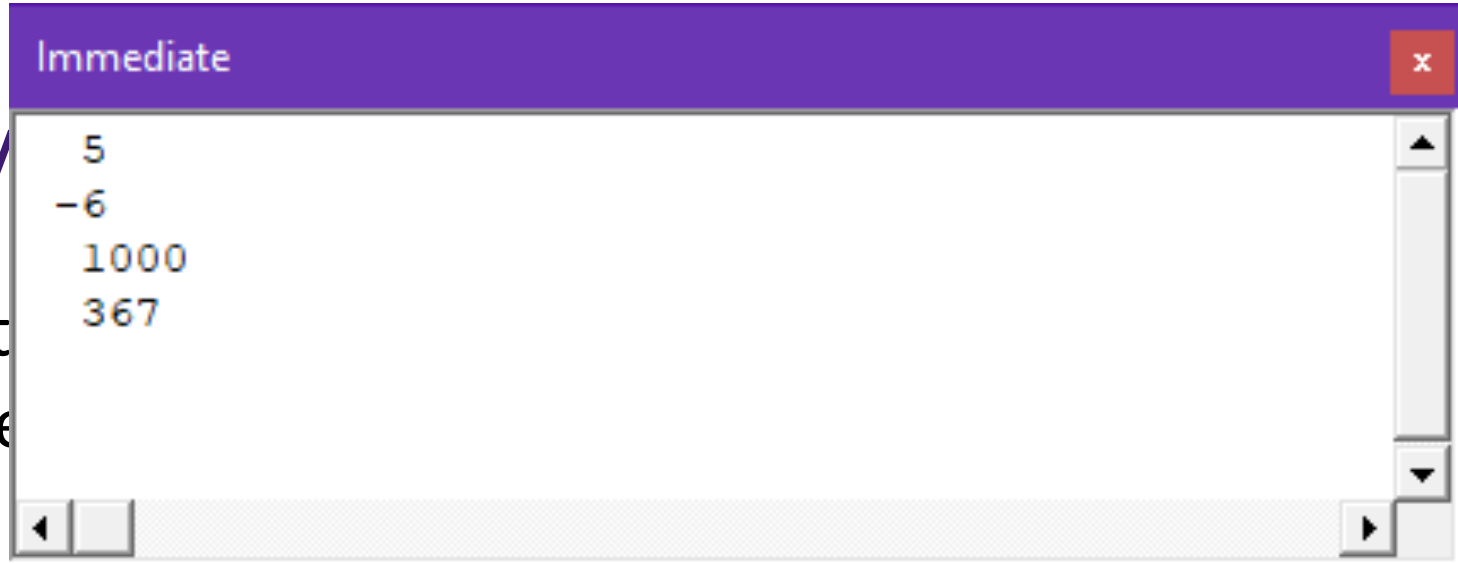
```
Function func1()
    Dim a(1 To 4) As Integer
    a(1) = 5
    a(2) = -6
    a(3) = 1000
    a(4) = 367

    Call func2(a)
End Function
```

```
Function func2(arr() As Integer)
    Dim i As Integer

    For i = LBound(arr) To UBound(arr)
        Debug.Print arr(i)
    Next i
End Function
```

# Arrays

## Passing Array

**Example:** Create a array and passes it values to the imme




```
Function func1()
    Dim a(1 To 4) As Integer
    a(1) = 5
    a(2) = -6
    a(3) = 1000
    a(4) = 367

    Call func2(a)
End Function
```

```
Function func2(arr() As Integer)
    Dim i As Integer

    For i = LBound(arr) To UBound(arr)
        Debug.Print arr(i)
    Next i
End Function
```

# Arrays

Write a function named ***SumArrays*** that takes two Integer arrays and finds the sum of the numbers in the same element of each array. Output the result to the immediate window. Assume the arrays are the same size.

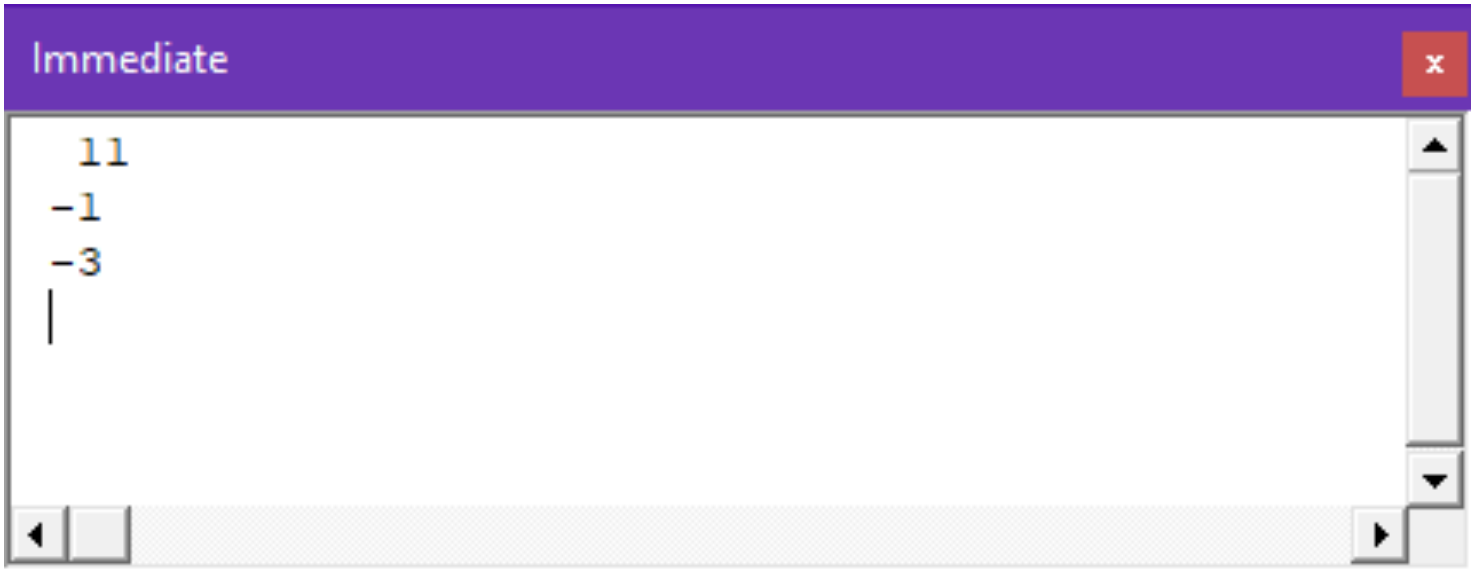
```
Function SumArrays(arrayOne, arrayTwo)
    Dim i As Integer

    For i = LBound(arrayOne) To UBound(arrayOne)
        Debug.Print arrayOne(i) + arrayTwo(i)
    Next i
End Function
```

# Arrays

Write a function named ***SumArrays*** that takes two Integer arrays and finds the sum of the numbers in the same element of each array. Output the result to the immediate window. Assume the arrays are the same size.

Can test in Excel with:

```
=SumArrays({5,-2,0},{6,1,-3})
```



# While Loop

- Similar to For loop but runs until a condition is **false**.

```
Do While condition
   statements
Loop
```

# While Loop

- Similar to For loop but runs until a condition is **false**.

```
Do While condition
   statements
Loop
```

**Boolean Expression**
Runs loop until condition is false.

**Statements**
Lines of code to run while condition is true.

# While Loop

## Example 3:

```
Function NumDig(n As Integer)
    Dim digits As Integer, x As Integer
    digits = 0
    x = n

    Do While x > 0
        x = x \ 10
        digits = digits + 1
    Loop

    NumDig = digits
End Function
```

# While Loop

## Example 3:

```
Function NumDig(n As Integer)
    Dim digits As Integer, x As Integer
    digits = 0
    x = n

    Do While x > 0
        x = x \ 10
        digits = digits + 1
    Loop

    NumDig = digits
End Function
```

**Boolean Expression**

Runs loop until condition is false.

**Statements**

Lines of code to run while condition is true.

# While Loop

## Example 3:

```
Function NumDig(n As Integer)
    Dim digits As Integer, x As Integer
    digits = 0
    x = n

    Do While x > 0
        x = x \ 10
        digits = digits + 1
    Loop

    NumDig = digits
End Function
```

**Loop will run until x is less than or equal to 0.**

# While Loop

## Example 3:

```
Function NumDig(n As Integer)
    Dim digits As Integer, x As Integer
    digits = 0
    x = n

    Do While x > 0
        x = x \ 10
        digits = dig
    Loop

    NumDig = digits
End Function
```

**We use integer division to divide x by 10 each loop. For example, if x started at 1234:**

| | |
|---|---|
| `x = 1234 \ 10` | Loop 1 |
| `x = 123 \ 10` | Loop 2 |
| `x = 12 \ 10` | Loop 3 |
| `x = 1 \ 10` | Loop 4 |
| `x = 0` | Exits Loop |

# While Loop

## Example 3:

```
Function NumDig(n As Integer)
    Dim digits As Integer, x As Integer
    digits = 0
    x = n

    Do While x > 0
        x = x \ 10
        digits = digits + 1
    Loop

    NumDig = digits
End Function
```

**Value of digits is increased by one each loop until x is less than or equal to 0.**

# Until Loop

- Just like a while loop, but we run the loop until the condition is **true**.

```
Do Until condition
   statements
Loop
```

# Until Loop

- Just like a while loop, but we run the loop until the condition is **true**.

```
Do Until condition
   statements
Loop
```

**Boolean Expression**

Runs loop until condition is true.

**Statements**

Lines of code to run while condition is false.

# Until Loop

## Example 4:

```
Function SumRow(row As Integer) As Double
    Dim sum As Double, i As Integer
    i = 1
    sum = 0

    Do Until IsEmpty(Cells(row, i))
        sum = sum + Cells(row, i)
        i = i + 1
    Loop

    SumRow = sum
End Function
```

# Until Loop

## Example 4:

```
Function SumRow
    Dim sum As
    i = 1
    sum = 0

    Do Until IsEmpty(Cells(row, i))
        sum = sum + Cells(row, i)
        i = i + 1
    Loop

    SumRow = sum
End Function
```

**IsEmpty is a built in function that returns True if the cell is empty (does not contain anything).**

# Until Loop

## Example 4:

```
Function SumRow
    Dim sum As
    i = 1
    sum = 0

    Do Until IsEmpty(Cells(row, i))
        sum = sum + Cells(row, i)
        i = i + 1
    Loop

    SumRow = sum
End Function
```

**This loop will run until it encounters an empty cell in the given row (given by the row parameter).**

# Until Loop

## Example 4:

```
Function SumRow(row As Integer) As Double
    Dim sum As Double, i As Integer
    i = 1
    sum = 0

    Do Until IsEmpty(Cells(row, i))
        sum = sum + Cells(row, i)
        i = i + 1
    Loop

    SumRow = sum
End Function
```

**The value of the cell is added to sum each loop.**

# Until Loop

## Example 4:

```
Function SumRow(row As Integer) As Double
    Dim sum As Double, i As Integer
    i = 1
    sum = 0

    Do Until IsEmpty(Cells(row, i))
        sum = sum + Cells(row, i)
        i = i + 1
    Loop

    SumRow = sum
End Function
```

**The value of i is incremented by one each loop.**

# Until Loop

## Example 4:

```
Function SumRow(row As Integer) As Double
    Dim sum As Double, i As Integer
    i = 1
    sum = 0

    Do Until IsEmpty(Cells(row, i))
        sum = sum + Cells(row, i)
        i = i + 1
    Loop

    SumRow = sum
End Function
```

```
row = 4
i = 1
sum = 0
```

C1 | =sumrow(4)

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | | | 2 | | |
| 2 | | | | | |
| 3 | | | | | |
| 4 | 10 | -35 | 3 | 24 | |
| 5 | | | | | |

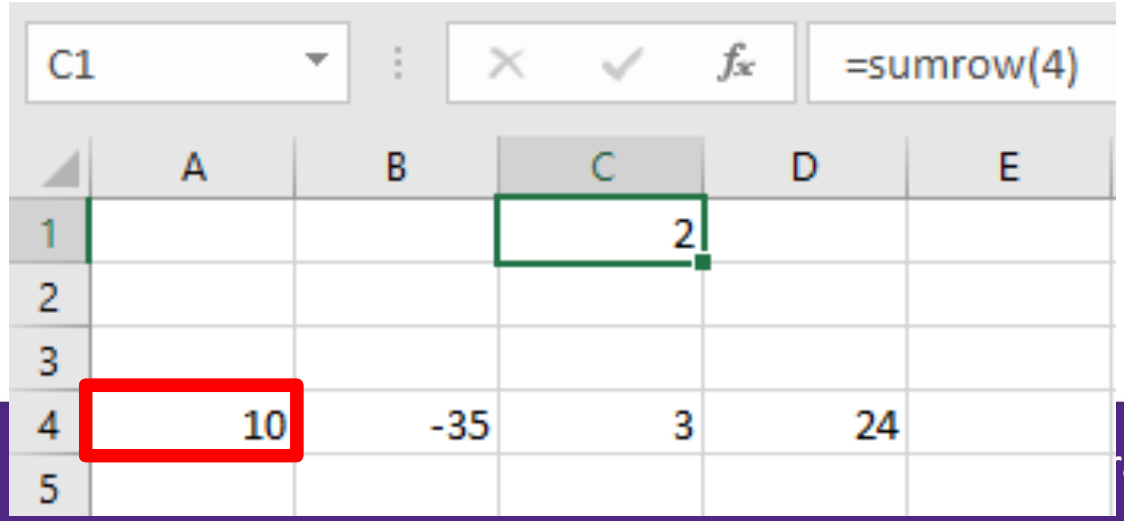
# Until Loop

## Example 4:

```
Function SumRow(row As Integer) As Double
    Dim sum As Double, i As Integer
    i = 1
    sum = 0

    Do Until IsEmpty(Cells(row, i))
        sum = sum + Cells(row, i)
        i = i + 1
    Loop

    SumRow = sum
End Function
```

**False**

```
row = 4
i = 1
sum = 0
```




**Cells(4, 1) = 10**

# Until Loop

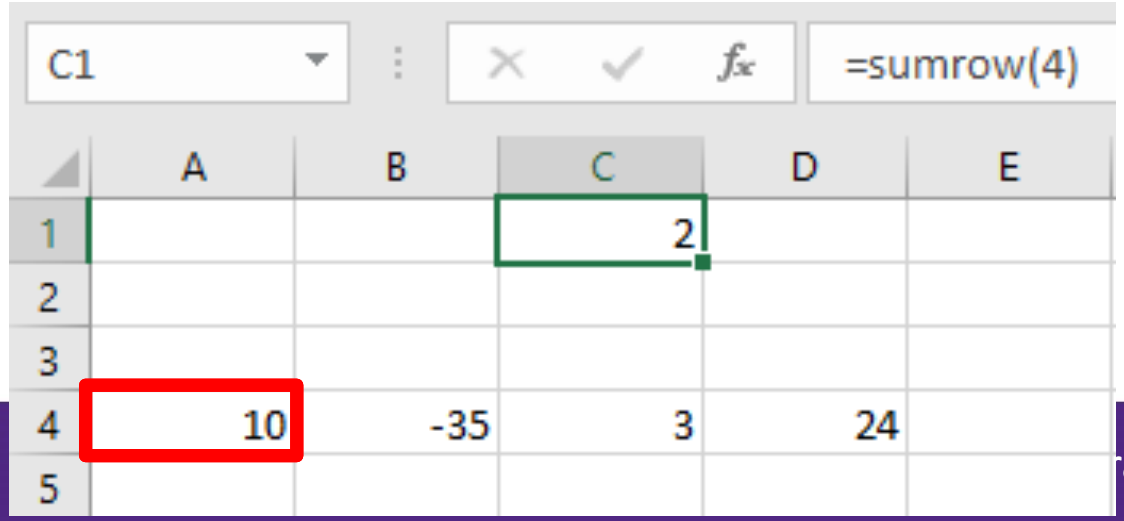
## Example 4:

```
Function SumRow(row As Integer) As Double
    Dim sum As Double, i As Integer
    i = 1
    sum = 0

    Do Until IsEmpty(Cells(row, i))
        sum = sum + Cells(row, i)
        i = i + 1
    Loop

    SumRow = sum
End Function
```

row = 4

i = 1

sum = 10




Cells(4, 1) = 10

# Until Loop

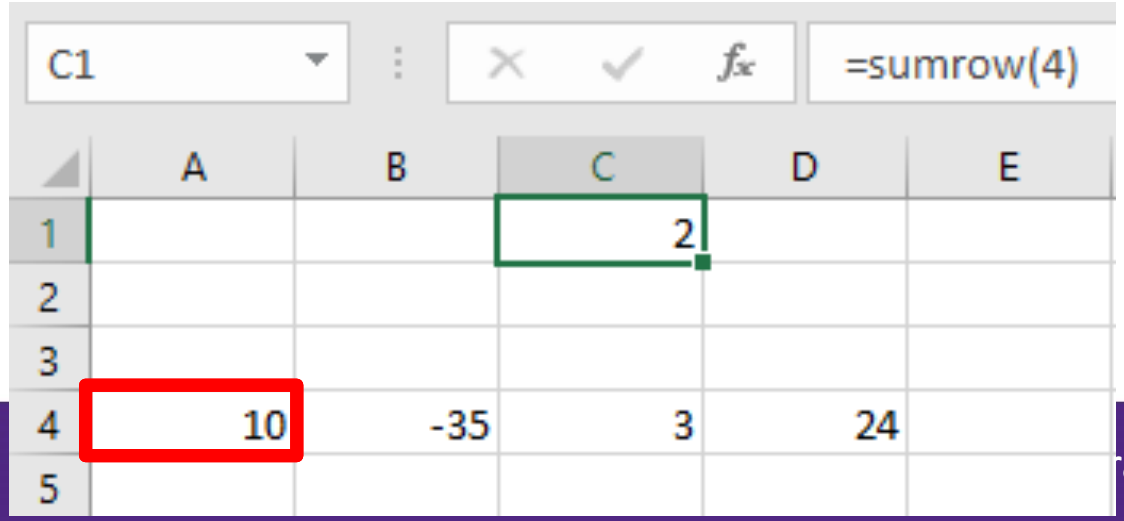
## Example 4:

```
Function SumRow(row As Integer) As Double
    Dim sum As Double, i As Integer
    i = 1
    sum = 0

    Do Until IsEmpty(Cells(row, i))
        sum = sum + Cells(row, i)
        i = i + 1
    Loop

    SumRow = sum
End Function
```

row = 4

i = 2

sum = 10



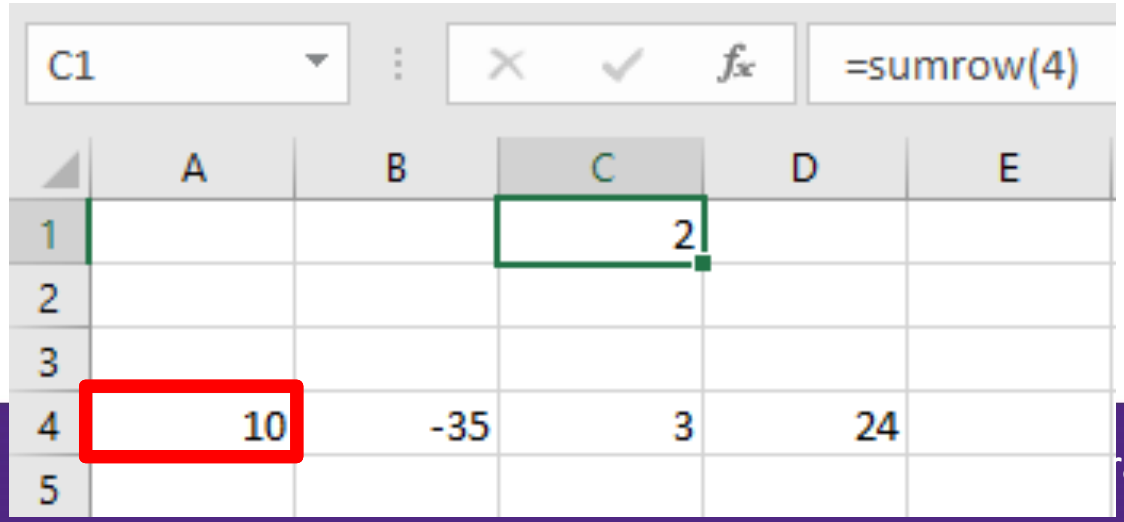

Cells(4, 1) = 10

# Until Loop

## Example 4:

```
Function SumRow(row As Integer) As Double
    Dim sum As Double, i As Integer
    i = 1
    sum = 0

    Do Until IsEmpty(Cells(row, i))
        sum = sum + Cells(row, i)
        i = i + 1
    Loop

    SumRow = sum
End Function
```

```
row = 4
i = 2
sum = 10
```




**Cells(4, 1) = 10**

# Until Loop

## Example 4:

```
Function SumRow(row As Integer) As Double
    Dim sum As Double, i As Integer
    i = 1
    sum = 0

    Do Until IsEmpty(Cells(row, i))
        sum = sum + Cells(row, i)
        i = i + 1
    Loop

    SumRow = sum
End Function
```

**False**

```
row = 4
i = 2
sum = 10
```

C1 | =sumrow(4)

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | | | 2 | | |
| 2 | | | | | |
| 3 | | | | | |
| 4 | 10 | -35 | 3 | 24 | |
| 5 | | | | | |

**Cells(4, 2) = -35**

# Until Loop

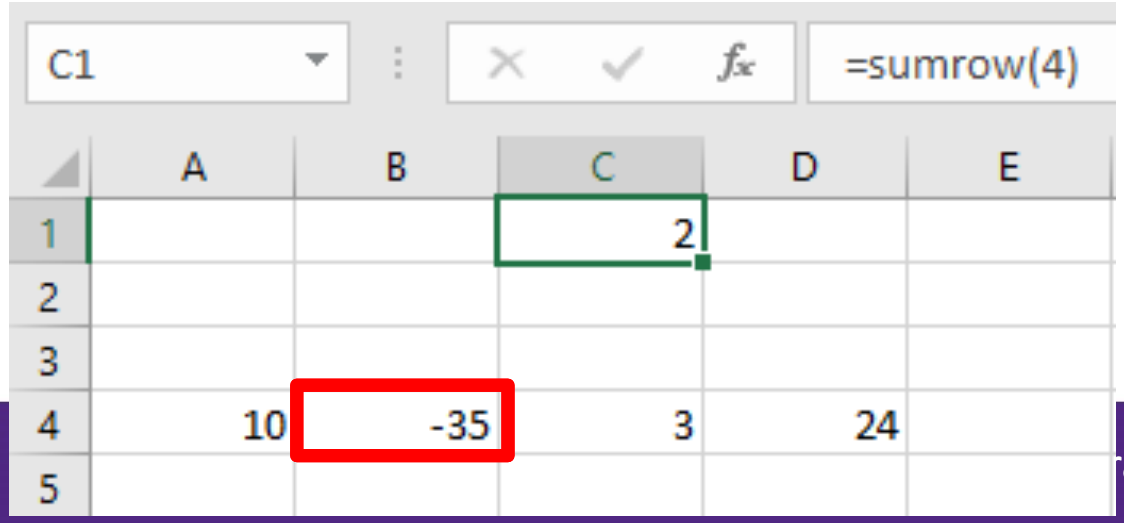
## Example 4:

```
Function SumRow(row As Integer) As Double
    Dim sum As Double, i As Integer
    i = 1
    sum = 0

    Do Until IsEmpty(Cells(row, i))
        sum = sum + Cells(row, i)
        i = i + 1
    Loop

    SumRow = sum
End Function
```

row = 4

i = 2

sum = -25




Cells(4, 2) = -35

# Until Loop

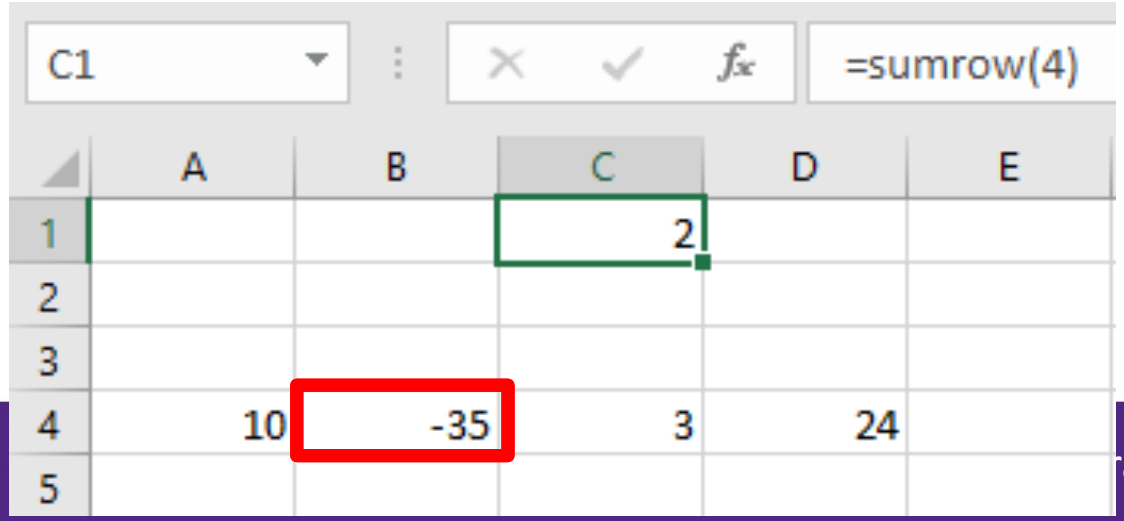
## Example 4:

```
Function SumRow(row As Integer) As Double
    Dim sum As Double, i As Integer
    i = 1
    sum = 0

    Do Until IsEmpty(Cells(row, i))
        sum = sum + Cells(row, i)
        i = i + 1
    Loop

    SumRow = sum
End Function
```

row = 4

i = 3

sum = -25



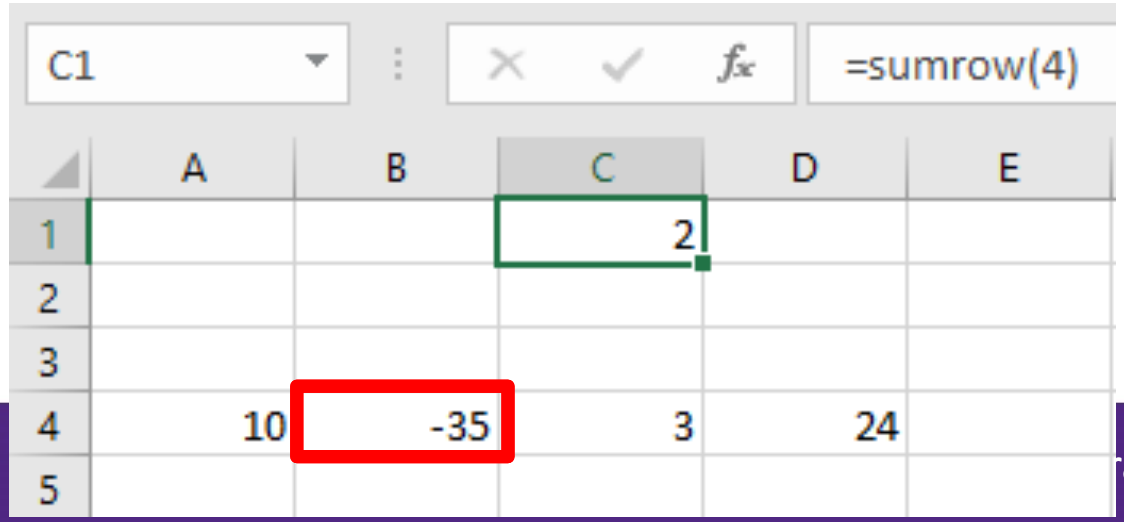

**Cells(4, 2) = -35**

# Until Loop

## Example 4:

```
Function SumRow(row As Integer) As Double
    Dim sum As Double, i As Integer
    i = 1
    sum = 0

    Do Until IsEmpty(Cells(row, i))
        sum = sum + Cells(row, i)
        i = i + 1
    Loop

    SumRow = sum
End Function
```

```
row = 4
i = 3
sum = -25
```




**Cells(4, 2) = -35**

# Until Loop

## Example 4:

```
Function SumRow(row As Integer) As Double
    Dim sum As Double, i As Integer
    i = 1
    sum = 0

    Do Until IsEmpty(Cells(row, i))
        sum = sum + Cells(row, i)
        i = i + 1
    Loop

    SumRow = sum
End Function
```

**False**

```
row = 4
i = 3
sum = -25
```

C1 | =sumrow(4)

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | | | 2 | | |
| 2 | | | | | |
| 3 | | | | | |
| 4 | 10 | -35 | 3 | 24 | |
| 5 | | | | | |

**Cells(4, 3) = 3**

# Until Loop

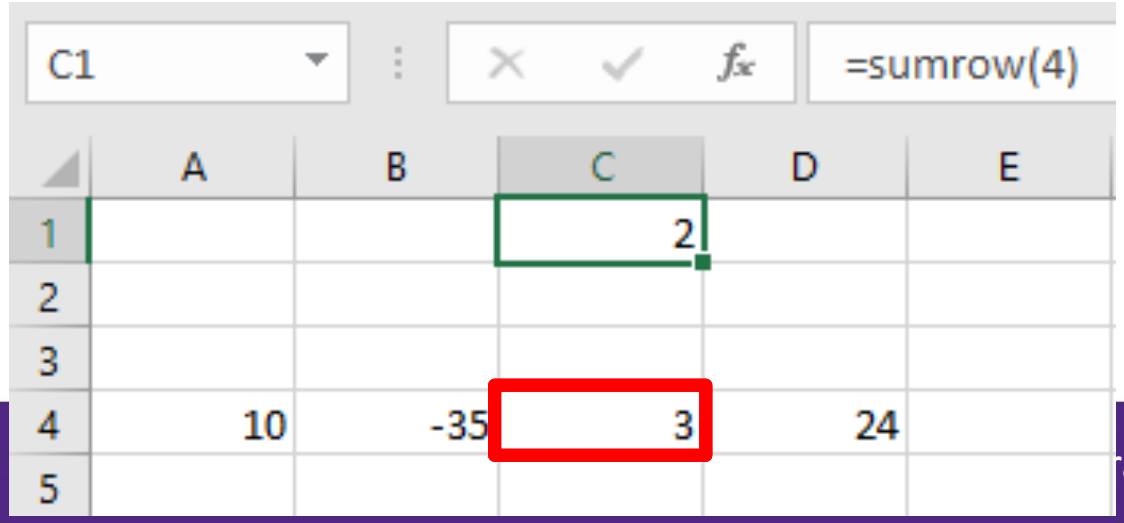
## Example 4:

```
Function SumRow(row As Integer) As Double
    Dim sum As Double, i As Integer
    i = 1
    sum = 0

    Do Until IsEmpty(Cells(row, i))
        sum = sum + Cells(row, i)
        i = i + 1
    Loop

    SumRow = sum
End Function
```

row = 4

i = 3

sum = -22




Cells(4, 3) = 3

# Until Loop

## Example 4:

```
Function SumRow(row As Integer) As Double
    Dim sum As Double, i As Integer
    i = 1
    sum = 0

    Do Until IsEmpty(Cells(row, i))
        sum = sum + Cells(row, i)
        i = i + 1
    Loop

    SumRow = sum
End Function
```

row = 4

i = 4

sum = -22

C1 | =sumrow(4)

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | | | 2 | | |
| 2 | | | | | |
| 3 | | | | | |
| 4 | 10 | -35 | 3 | 24 | |
| 5 | | | | | |

**Cells(4, 3) = 3**

# Until Loop

## Example 4:

```
Function SumRow(row As Integer) As Double
    Dim sum As Double, i As Integer
    i = 1
    sum = 0

    Do Until IsEmpty(Cells(row, i))
        sum = sum + Cells(row, i)
        i = i + 1
    Loop

    SumRow = sum
End Function
```

```
row = 4
i = 4
sum = -22
```

C1 | =sumrow(4)

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | | | 2 | | |
| 2 | | | | | |
| 3 | | | | | |
| 4 | 10 | -35 | 3 | 24 | |
| 5 | | | | | |

**Cells(4, 3) = 3**

# Until Loop

## Example 4:

```
Function SumRow(row As Integer) As Double
    Dim sum As Double, i As Integer
    i = 1
    sum = 0

    Do Until IsEmpty(Cells(row, i))
        sum = sum + Cells(row, i)
        i = i + 1
    Loop

    SumRow = sum
End Function
```

**False**

```
row = 4
i = 4
sum = -22
```

C1 | =sumrow(4)

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | | | 2 | | |
| 2 | | | | | |
| 3 | | | | | |
| 4 | 10 | -35 | 3 | 24 | |
| 5 | | | | | |

**Cells(4, 4) = 24**

# Until Loop

## Example 4:

```
Function SumRow(row As Integer) As Double
    Dim sum As Double, i As Integer
    i = 1
    sum = 0

    Do Until IsEmpty(Cells(row, i))
        sum = sum + Cells(row, i)
        i = i + 1
    Loop

    SumRow = sum
End Function
```

row = 4

i = 4

sum = 2

C1 =sumrow(4)

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | | | 2 | | |
| 2 | | | | | |
| 3 | | | | | |
| 4 | 10 | -35 | 3 | 24 | |
| 5 | | | | | |

Cells(4, 4) = 24

# Until Loop

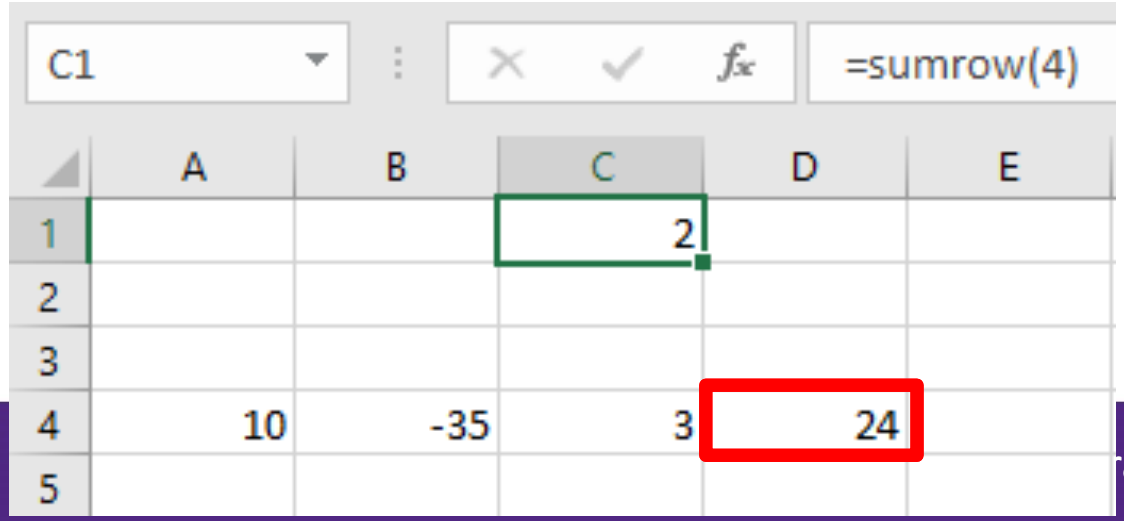
## Example 4:

```
Function SumRow(row As Integer) As Double
    Dim sum As Double, i As Integer
    i = 1
    sum = 0

    Do Until IsEmpty(Cells(row, i))
        sum = sum + Cells(row, i)
        i = i + 1
    Loop

    SumRow = sum
End Function
```

row = 4

i = 5

sum = 2



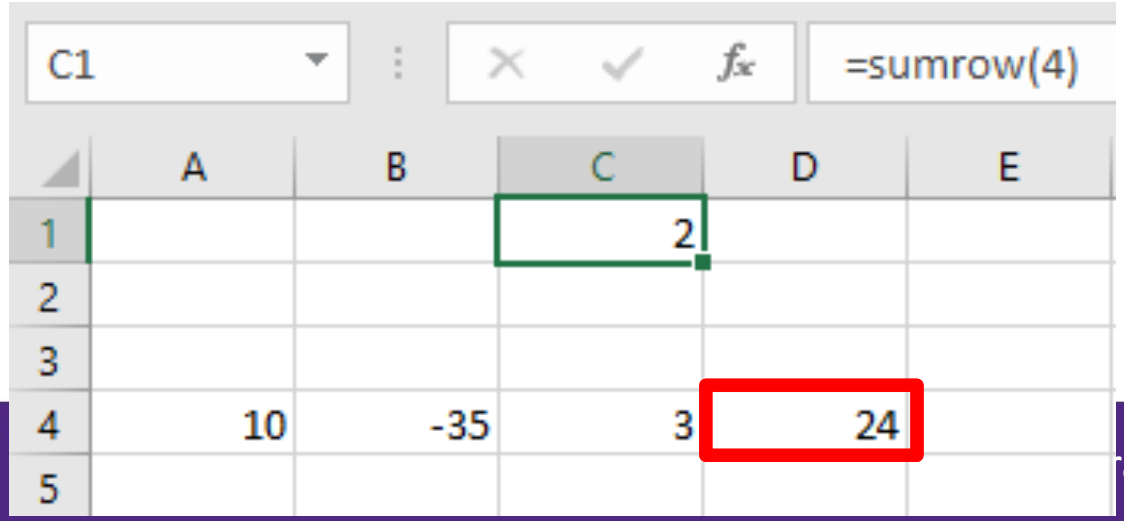

**Cells(4, 4) = 24**

# Until Loop

## Example 4:

```
Function SumRow(row As Integer) As Double
    Dim sum As Double, i As Integer
    i = 1
    sum = 0

    Do Until IsEmpty(Cells(row, i))
        sum = sum + Cells(row, i)
        i = i + 1
    Loop

    SumRow = sum
End Function
```

```
row = 4
i = 5
sum = 2
```




**Cells(4, 4) = 24**

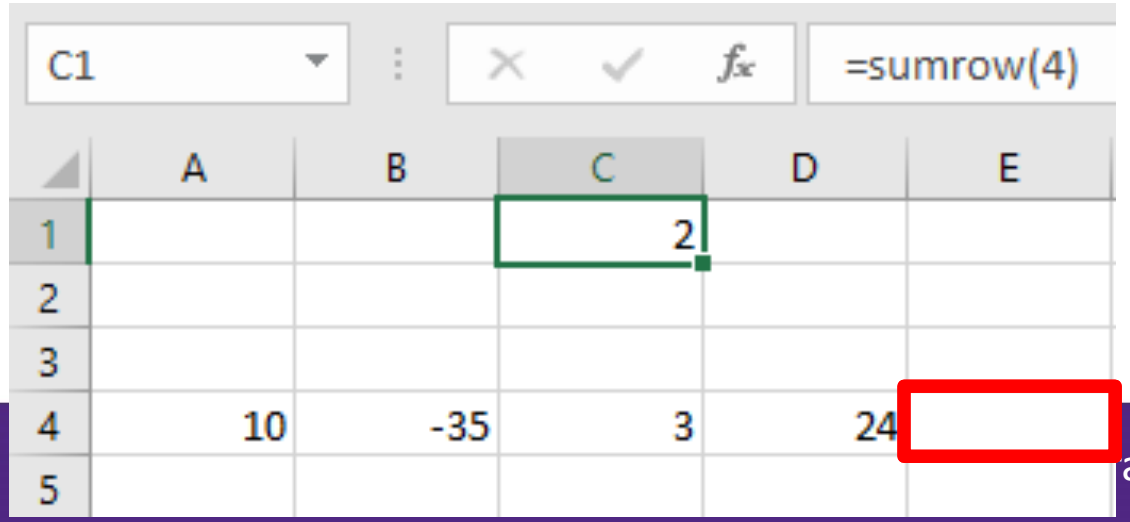
# Until Loop

## Example 4:

```
Function SumRow(row As Integer) As Double
    Dim sum As Double, i As Integer
    i = 1
    sum = 0

    Do Until IsEmpty(Cells(row, i))
        sum = sum + Cells(row, i)
        i = i + 1
    Loop

    SumRow = sum
End Function
```

True

```
row = 4
i = 5
sum = 2
```




Cells(4, 5) =

# Until Loop

## Example 4:

```
Function SumRow(row As Integer) As Double
    Dim sum As Double, i As Integer
    i = 1
    sum = 0

    Do Until IsEmpty(Cells(row, i))
        sum = sum + Cells(row, i)
        i = i + 1
    Loop

    SumRow = sum
End Function
```

```
row = 4
i = 5
sum = 2
```

C1 | =sumrow(4)

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | | | 2 | | |
| 2 | | | | | |
| 3 | | | | | |
| 4 | 10 | -35 | 3 | 24 | |
| 5 | | | | | |

**Cells(4, 5) =**

# Until Loop

## Example 4:

```
Function SumRow(row As Integer) As Double
    Dim sum As Double, i As Integer
    i = 1
    sum = 0

    Do Until IsEmpty(Cells(row, i))
        sum = sum + Cells(row, i)
        i = i + 1
    Loop

    SumRow = sum
End Function
```

row = 4
i = 5
sum = 2

SumRow = 2

C1 =sumrow(4)

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | | | 2 | | |
| 2 | | | | | |
| 3 | | | | | |
| 4 | 10 | -35 | 3 | 24 | |
| 5 | | | | | |

Cells(4, 5) =

# Exit Loops

- We can exit loops early (before the condition or end index) using the Exit command.
- This works just like `Exit Function`, but uses the keyword For or Do.

`Exit For` **Exits a for loop**

`Exit Do` **Exits a do while or do until loop**

# Exit Loops

## Example 5:

```
Function MyMatch(rng As Range, val As Double) As Integer
    MyMatch = -1

    For i = 1 To rng.Count
        If rng.Cells(i) = val Then
            MyMatch = i
        End If
    Next i
End Function
```

**Returns index of the given value (val) in the given range (rng).**

**Our own version of the MATCH function.**

# Exit Loops

## Example 5:

```
Function MyMatch(rng As Range, val As Double) As Integer
    MyMatch = -1

    For i = 1 To rng.Count
        If rng.Cells(i) = val Then
            MyMatch = i
        End If
    Next i
End Function
```

**Loop keeps running after a match is found.**

**This will return the index of the last match in the range. OK, but inefficient.**

# Exit Loops

## Example 5:

```
Function MyMatch(rng As Range, val As Double) As Integer
    MyMatch = -1

    For i = 1 To rng.Count
        If rng.Cells(i) = val Then
            MyMatch = i
            Exit For
        End If
    Next i
End Function
```

### Solution

**Exit the loop after we find the first match.**

**Exits the loop but not the function. Would run any commands after the Next i (if there were any in this case).**

# Subroutines

- Like functions but with a few important differences:
  - Are allowed to edit the worksheet directly.
  - Do not return a value/result.
  - Have to be run manually or with a button (cannot be used in an Excel formula).

```
Sub subname(paramName As type, …)
  statements
End Sub
```

# Subroutines

- Like functions but with a few important differences.

**Parameters**

**Name of Subroutine**

**Parameter Name**

**Parameter Data Type**

```
Sub subname(paramName As type, …)
   statements
End Sub
```

**Statements**

Lines of code the subroutine contains

# Subroutines

## Example 6:

```
Sub TimesTable()
    For i = 1 To 9
        For k = 1 To 9
            Cells(i, k) = i * k
        Next k
    Next i
End Sub
```

# Subroutines

## Example 6:

```
Sub TimesTable()
    For i = 1 To 9
        For k = 1 To 9
            Cells(i, k) = i * k
        Next k
    Next i
End Sub
```

### Nested for loops

The for k loop is inside of the for i loop. The for k loop will run to completion (9 times) each time the for i loop runs.

The line `Cells(i, k) = i * k` will be run 81 times (9 × 9). Once for each combintation of i and k.

# Subroutines

## Example 6:

```
Sub TimesTable()
    For i = 1 To 9
        For k = 1 To 9
            Cells(i, k) = i * k
        Next k
    Next i
End Sub
```

**Unlike in functions, we can set the value of the cell directly.**

**This updates the cell in row i, column k to the result of i * k.**

# Subroutine

With your cursor in the TimesTable sub click on the run button or press F5.

## Example 6: How to Run

Microsoft Visual Basic for Applications - week 6 ex.xlsm - [Module2 (Code)]

File Edit View Insert Format Debug Run Tools Add-Ins Window Help

Ln 3, Col 8

Project - VBAProject

VBAProject (week 6 ex.x
- Microsoft Excel Objects
  - Sheet1 (Sheet1)
  - Sheet2 (Sheet2)
  - Sheet3 (Sheet3)
  - ThisWorkbook
- Modules
  - Module1
  - Module2

(General)

Run Sub/UserForm (F5)

```
Sub TimesTable()
    For i = 1 To 9
        For k = 1 To 9
            Cells(i, k) = i * k
        Next k
    Next i
End Sub
```

# Subroutines

## Example 6: Output

| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | |
| 2 | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | |
| 3 | 3 | 6 | 9 | 12 | 15 | 18 | 21 | 24 | 27 | |
| 4 | 4 | 8 | 12 | 16 | 20 | 24 | 28 | 32 | 36 | |
| 5 | 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 | |
| 6 | 6 | 12 | 18 | 24 | 30 | 36 | 42 | 48 | 54 | |
| 7 | 7 | 14 | 21 | 28 | 35 | 42 | 49 | 56 | 63 | |
| 8 | 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 | 72 | |
| 9 | 9 | 18 | 27 | 36 | 45 | 54 | 63 | 72 | 81 | |
| 10 | | | | | | | | | | |

**Updates the current worksheet with the times table generated by the TimesTable subroutine.**

# Subroutines

## Example 7: Color the odd cells from Example 6

```
Sub ColourOdd()
    For i = 1 To 9
        For k = 1 To 9
            If Cells(i, k) Mod 2 <> 0 Then
                Cells(i, k).Interior.ColorIndex = 3
            End If
        Next k
    Next i
End Sub
```

# Subroutines

## Example 7: Color the odd cells from Example 6

```
Sub ColourOdd()
    For i = 1 To 9
        For k = 1 To 9
            If Cells(i, k) Mod 2 <> 0 Then
                Cells(i, k).Interior.ColorIndex = 3
            End If
        Next k
    Next i
End Sub
```

**Check if cell value is odd (has a remainder when divided by 2).**

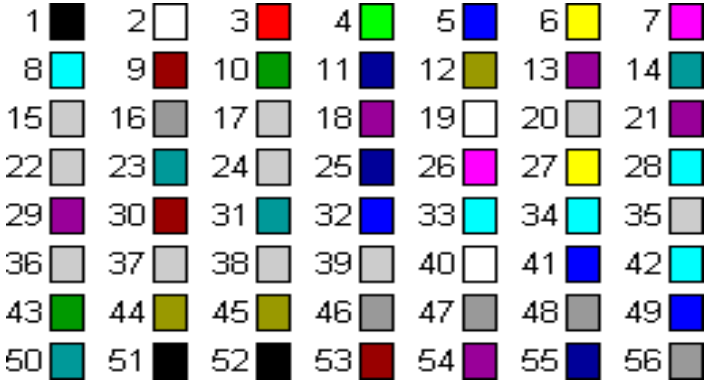
# Subroutines

## Example 7: Color the odd cells from Example 6

```
Sub ColourOdd()
    For i = 1 To 9
        For k = 1 To 9
            If Cells(i, k) Mod 2 <> 0 Then
                Cells(i, k).Interior.ColorIndex = 3
            End If
        Next
    Next i
End Sub
```

**.Interior.ColorIndex sets the background colour of the cell.**

**3 is for red:**



# Subroutines

## Example 7: Output

| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | |
| 2 | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | |
| 3 | 3 | 6 | 9 | 12 | 15 | 18 | 21 | 24 | 27 | |
| 4 | 4 | 8 | 12 | 16 | 20 | 24 | 28 | 32 | 36 | |
| 5 | 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 | |
| 6 | 6 | 12 | 18 | 24 | 30 | 36 | 42 | 48 | 54 | |
| 7 | 7 | 14 | 21 | 28 | 35 | 42 | 49 | 56 | 63 | |
| 8 | 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 | 72 | |
| 9 | 9 | 18 | 27 | 36 | 45 | 54 | 63 | 72 | 81 | |
| 10 | | | | | | | | | | |

**Requires time table data to already be in the active worksheet.**

# String Functions

- VBA has a number of functions to help us deal with Strings:

| Function | Description |
|---|---|
| `Len(str)` | Returns the length of the string. |
| `Trim(str)` | Returns a copy of the string with leading and trailing spaces removed (i.e. "  hello  " to "hello"). |
| `Space(int)` | Creates a string that contains the given number of spaces. |
| `LCase(str) and UCase(str)` | Returns a copy of the string with all letters converted to lower or upper case. |
| `StrComp(str1, str2, mode)` | Compares two strings. Can be used to do case insensitive comparisons. |
| `Split(str, delim)` | Splits a string around a given character into an array of smaller strings. |
| `Replace(str, find, replacewith)` | Simple find and replace using only literal characters (not regular expression). |

# String Functions

## Replace

- Replace ( str, find, replacewith)
- Replaces all occurrences of the string find with the string replacewith in the string str.
- **Examples:**

```
Replace("Hello World!", "Hello", "Good bye")
Result: "Good bye World!"

Replace("Hello. How are you?", "o", "")
Result: "Hell. How are yu?"

Replace("Hello. How are you?", " ", "")
Result: "Hello.Howareyou?"
```

# String Functions

## Replace

- Replace ( str, find, replacewith)
- Replaces all occurrences of the string find with the string replacewith in the string str.
- **Examples:**

**When the replacewith string is empty it deletes the character.**

```
Replace("Hello. How are you?", "o", "")
Result: "Hell. How are yu?"
```

Deletes the character o

```
Replace("Hello. How are you?", " ", "")
Result: "Hello.Howareyou?"
```

Deletes spaces

# String Functions

## StrComp

- StrComp ( string1, string2 [, compare ] )
- Compare
  - There are three compares in VBA we will use the **vbTextCompare** which does textual comparison. Ignores capitalization.
- Return Value
  - If *string1* is **equal** to *string2*, the StrComp function will return **0**.
  - If *string1* is **less than** *string2*, the StrComp function will return **-1**.
  - If *string1* is **greater than** *string2*, the StrComp function will return **1**.

# String Functions

## StrComp Examples:

```
StrComp("abc", "ABC", vbTextCompare)
```
Result: 0

```
StrComp("abc", "abc", vbTextCompare)
```
Result: 0

```
StrComp("abc", "ab", vbTextCompare)
```
Result: 1

```
StrComp("abc", "zxy", vbTextCompare)
```
Result: -1

# String Functions

## Split

- The `Split` function is used to split a **string** into a string array based on a **delimiter**. A **delimiter** is a character such as a comma or space that separates the items.

```
Split(str, [delim])
```

**String**

The string we are splitting.

**Delimiter**

A string that contains the delimiter to use. This argument is optional, if omitted a space is used.

# String Functions

## `Split` Examples:

```
Split("Tech on the Net")
Result: {"Tech", "on", "the", "Net"}

Split("172.23.56.4", ".")
Result: {"172", "23", "56", "4"}

Split("A;B;C;D", ";")
Result: {"A", "B", "C", "D"}
```

# String Functions

**Example 8:** Write a function named CountWord that counts the number of occurrences of a given word in a given string.

## For example:

```
CountWord("Programming is a skill best acquired by practice
and example and not programming books.", "programming")
```

Would return 2.

# String Functions

## Example 8:

```
Function CountWord(str As String, word As String) As Integer
```

**First we create our function header.**

This function takes in a string (`str`) that we will be counting words in and a single word (`word`) that we will be looking for.

It returns an Integer (the number of times we found `word` in `str`).

```
End Function
```

# String Functions

## Example 8:

```
Function CountWord(str As String, word As String) As Integer
    Dim i As Integer, count As Integer
    Dim words() As String
```

**Declare the variables we will use in this function.**

`i` will be the index for a loop that goes through each word in `str`.

`count` will store the number of times we found `word` in `str`.

`words` is a dynamic array of strings in which we will store the result of calling the `Split` function on `str`.

```
End Function
```

# String Functions

## Example 8:

```
Function CountWord(str As String, word As String) As Integer
    Dim i As Integer, count As Integer
    Dim words() As String

    count = 0
    words = Split(str, " ")
```

**This delimiter tells the Split function how to divide up the string. In this case, Split will split the string each time it encounters a space character.**

**Initialize the variables.**

Set `count` to 0 to start with. So far we have not found any occurrences of `word` in `str`.

We use the `Split` function to break up the string `str` into individual words and store them in the array `words`.

```
End Function
```

# String Functions

## Example 8:

```
Function CountWord(str As String, word As String) As Integer
    Dim i As Integer, count As Integer
    Dim words() As String

    count = 0
    words = Split(str, " ")

    For i = LBound(words) To UBound(words)


    Next i

End Function
```

**Loop through each word.**

Create a for loop that runs through each word in the words array.

# String Functions

## Example 8:

```
Function CountWord(str As String, word As String) As Integer
    Dim i As Integer, count As Integer
    Dim words() As String

    count = 0
    words = Split(str, " ")

    For i = LBound(words) To UBound(words)
        If StrComp(words(i), word, vbTextCompare) = 0 Then

        End If
    Next i


End Function
```

**Compare the current word in words with the word argument (the word we are looking for).**

Use StrComp to ignore case differences.

# String Functions

## Example 8:

```
Function CountWord(str As String, word As String) As Integer
    Dim i As Integer, count As Integer
    Dim words() As String

    count = 0
    words = Split(str, " ")

    For i = LBound(words) To UBound(words)
        If StrComp(words(i), word, vbTextCompare) = 0 Then
            count = count + 1
        End If
    Next i

End Function
```

**If the words match, increment our counter (count) by 1.**

# String Functions

## Example 8:

```
Function CountWord(str As String, word As String) As Integer
    Dim i As Integer, count As Integer
    Dim words() As String

    count = 0
    words = Split(str, " ")

    For i = LBound(words) To UBound(words)
        If StrComp(words(i), word, vbTextCompare) = 0 Then
            count = count + 1
        End If
    Next i

    CountWord = count
End Function
```

**Lastly, return our result.**