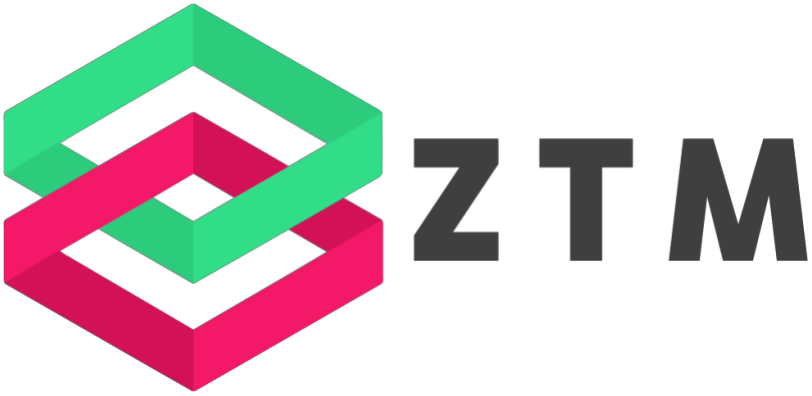

VBA PROGRAMMING CHEAT SHEET

TRAVIS CUZICK

V1.01

HEEELLLOOOOO!

I'm Andrei Neagoie, Founder and Lead Instructor of the Zero To Mastery Academy.

After working as a Senior Software Developer over the years, I now dedicate 100% of my time to teaching others valuable software development skills, help them break into the tech industry, and advance their careers.

In only a few years, **over 750,000 students** around the world have taken Zero To Mastery courses and many of them are now working at top tier companies like Apple, Google, Amazon, Tesla, IBM, Facebook, and Shopify, just to name a few.

This cheat sheet, created by our VBA Programming instructor (Travis Cuzick) provides you with the key VBA functions and shortcuts that you need to know and remember.

If you want to not only learn VBA programming but also get the exact steps to build your own projects and get hired as a Data Analyst or Data Scientist, then check out our Career Paths.

Happy Coding!
Andrei

Founder & Lead Instructor, Zero To Mastery
Andrei Neagoie

P.S. I also recently wrote a book called Principles For Programmers. You can download the first five chapters for free here.

Contents

Basic Programming Operations

Operation	Code
Declare a variable	Dim myVar As String
Set the value of a variable	myVar = “some value”
Set the value of an object variable	Set myObj = Range(“B2:C3”)
Gather user input	userInput = InputBox(“What’s your favorite color?”)
Print a message to the screen	MsgBox(“You shall not pass!”)
Execute a macro from within another macro	Call myMacro
Use a built-in worksheet function in a macro	Application.WorksheetFunction.CountA(“A:A”)
Comment out a line of code - note the apostrophe (‘)	‘VBA will ignore me!

Basic Operations on Data

Operation	Code
Addition	imFour = 2 + 2
Subtraction	imZero = 2 - 2
Multiplication	imAlsoFour = 2 * 2
Division (uses “/” operator)	MsgBox(10 / 3) ‘returns 3.333333

Integer Division (uses “\” operator)	MsgBox(10 \ 3) ‘returns 3
Concatenation	helloWorld = “Hello” & “ world”

Data Types

Data Type	Description	Example
Integer	Whole number between -32,768 and 32,767	11
Long	Whole numbers between -2,147,483,648 and 2,147,483,647	1,234,567
Single	Decimal number with seven digits of precision	3.141593
Double	Decimal number with fifteen digits of precision	3.14159265358979
Date	Date values	3/5/2021
String	Text data	“Hello world”
Boolean	Logical true/false values	False
Range	Range object in Excel	Set myRange = Range(“A1”)
Worksheet	Worksheet object in Excel	Set mySheet = Sheets(“Sheet 1”)
Workbook	Worksheet object in Excel	Set myWorkbook = Workbooks(1)
Variant	Unspecified data type	myVariant = “Anything goes!”

Logical/Comparison Operators

Operator	Symbol	Example
Equals	=	5 = 5
Not Equals	<>	5 <> 55
Greater than	>	2 > 1
Greater than or equal to	>=	2 >= 2
Less than	<	4 < 5
Less than or equal to	<=	4 <= 5
And	And	(5 = 5) And (5 <> 55) = True (5 = 5) And (5 = 55) = False (5 <> 5) And (5 = 55) = False
Or	Or	(5 = 5) Or (5 <> 55) = True (5 = 5) Or (5 = 55) = True (5 <> 5) Or (5 = 55) = False
Not	Not	Not (5 = 5) = False Not (5 = 55) = True

If Statements

Type	Example Scenario	VBA Code
Simple If statement	If the value stored in the variable “val” is greater than 1,000, print the text “Large”.	If val > 1000 Then MsgBox(“Large”)

	Otherwise, do nothing.	End If

If-Else statement	If the value stored in the variable “val” is greater than 1,000, print the text “Large”.
	Otherwise, print the text “Small”.
		If val > 1000 Then
		MsgBox(“Large”)
		Else
		MsgBox(“Small”)
		End If

If-ElseIf-Else statement	If the value stored in the variable “val” is greater than 1,000, print the text “Large”.
	If the value stored in the variable “val” is between 200 and 1,000, print the text “Medium”.
	Otherwise, print the text “Small”.
		If val > 1000 Then
		MsgBox(“Large”)
		ElseIf val >= 200 Then
		MsgBox(“Medium”)
		Else
		MsgBox(“Small”)
		End If

Loops

Type	Example Scenario	VBA Code

Do Loop	Print the first 5 integers to the screen	Dim counter As Integer
		counter = 1
		
		Do
		If counter > 5 Then
		Exit Do
		End If

		MsgBox (counter) counter = counter + 1 Loop
Do While Loop	Print the first 5 integers to the screen	Dim counter As Integer counter = 1 Do While counter <= 5 MsgBox (counter) counter = counter + 1 Loop
Do Until Loop	Print the first 5 integers to the screen	Dim counter As Integer counter = 1 Do Until counter > 5 MsgBox (counter) counter = counter + 1 Loop
For Next Loop	Print the first 5 integers to the screen	Dim counter As Integer For counter = 1 To 5 MsgBox (counter) Next counter
For Each Loop	Print the values in cells A1 through A5 to the screen	Dim cell As Range For Each cell In Range("A1:A5")

		MsgBox (cell.Value) Next cell

Arrays

Example Scenario	VBA Code
Create an empty array with 5 *integer* elements and a 0-based index	Dim myArr(5) As Integer
Set the value at the 3rd position of an array with a 0-based index	Dim myArr(5) As Integer myArr(2) = 3
Print the 3rd element of an array with a 0-based index	Dim myArr(5) myArr(2) = 3 MsgBox(myArr(2))
Create an empty 2-dimensionnal array with 3 rows and 2 columns, that can accommodate multiple data types	Dim myArr(2, 1) As Variant
Set the values of a 3x2 array	Dim myArr(2, 1) As Variant myArr(0,0) = “one” myArr(0,1) = 1 myArr(1,0) = “two” myArr(1,1) = 2 myArr(2,0) = “three” myArr(2,1) = 3

Print the maximum index (upper bound) of an array	Dim myArr(5) As String MsgBox(UBound(myArr))
Print all the elements of an array using a For Next Loop	Dim myArr(2) As Variant myArr(0) = "one" myArr(1) = 2 myArr(2) = "three" Dim counter As Integer For counter = 0 To UBound(myArr) MsgBox (myArr(counter)) Next counter
Transfer data from cells A1 through A5 to an array	Dim myArr() As Variant myArr = Range("A1:A5").Value
Transfer data from a 3-element array to an Excel range	Dim myArr(2) As Variant myArr(0) = "one" myArr(1) = 2 myArr(2) = "three" Range("A1:C1") = myArr

Transfer data from a 3-element array to a *vertical* Excel range	Dim myArr(2) As Variant myArr(0) = "one" myArr(1) = 2 myArr(2) = "three" Range("A1:A3") = Application.WorksheetFunction.Transpose(myArr)
Use the SPLIT function to convert a text string into an array of words	Dim textStr As String Dim splitStr() As String textStr = "I am a list of words" splitStr() = SPLIT(textStr)
Use the JOIN function to combine an array of values into a single string, with those values separated by spaces	Dim myArr(5) As Variant Dim combinedStr As String myArr(0) = "I" myArr(1) = “am” myArr(2) = "a" myArr(3) = "list" myArr(4) = "of" myArr(5) = "words" combinedStr = JOIN(myArr, “ ”)

The With Construct

Example Scenario	VBA Code (without using With)	VBA Code (using With)
Format cell A1 as follows: - Bold - Italic - Font-style: Roboto - Font-size: 14	Range("A1").Font.Bold = True Range("A1").Font.Italic = True Range("A1").Font.Name = "Roboto" Range("A1").Font.Size = 14	With Range("A1").Font .Bold = True .Italic = True .Name = "Roboto" .Size = 14 End With

Working With Ranges

Example Scenario	**VBA Code**
Target a single cell using a hard-coded reference	Range(“A1”)
Target multiple cells using a hard-coded reference	Range(“A1:C3”)
Print the *value* of a cell using a hard-coded reference	MsgBox(Range(“A1”).Value)
Set the value of a cell using a hard-coded reference	Range(“A1”).Value = 11
Print the value of the *active* cell	MsgBox(ActiveCell.Value)
Set the value of the active cell	ActiveCell.Value = 22
Print the value of the cell 1 row *below*, and 2 columns to the *right*, of the active cell	MsgBox(ActiveCell.Offset(1,2))

Set the value of the cell 1 row *above*, and 2 columns to the *left*, of the active cell	ActiveCell.Offset(-1,-2).Value = “I’m upset that I’ve been offset!”
Use the Cells property to print the value of cell A1	MsgBox(Cells(1,1))
Use the Cells property to *set* the value of cell D3	Cells(3,4).Value = “Row 3, column 4”
Use the Cells property within a For Next loop to print the values of the first 5 cells in column A	Dim counter As Integer For counter = 1 To 5 MsgBox (Cells(counter, 1)) Next counter

Use the Cells property within a *nested* For Next loop to print the values of all the cells in a 2-dimensional *selection* (that is, the cells currently selected by the user)	Dim a As Integer
	Dim b As Integer
	
	For a = 1 To Selection.Rows.Count
	  For b = 1 To Selection.Columns.Count
	  MsgBox (Selection.Cells(a, b))
	  Next b
	Next a

Select the *last* cell with data in a column of values starting in cell A1	Range("A1").End(xlDown).Select
Select *all* the values in a column of data starting in cell A1	Range(Range("A1"), Range("A1").End(xlDown)).Select

Working With Worksheets

Example Scenario	VBA Code
Activate a sheet by referencing its name	Sheets("Sheet 1").Activate
Activate a sheet by referencing its *index*	Sheets(1).Activate
Print the name of the *active* worksheet	MsgBox(ActiveSheet.Name)
Add a new worksheet	Sheets.Add
Add a new worksheet and specify its name	Sheets.Add.Name = "My New Sheet"
Delete a worksheet	Sheets("My New Sheet").Delete
Hide a worksheet	Sheets(2).visible = False
Unhide a worksheet	Sheets(2).visible = True
Loop through all sheets in a workbook	Dim ws As Worksheet For Each ws In ActiveWorkbook.Worksheets MsgBox (ws.Name) Next ws

Working With Workbooks

Example Scenario	VBA Code
Activate a workbook by referencing its name	Workbooks("My Workbook").Activate
Activate the first workbook that was opened, among all open workbooks	Workbooks(1).Activate
Activate the *last* workbook that was opened, among all open workbooks	Workbooks(Workbooks.Count).Activate
Print the name of the *active* workbook	MsgBox(ActiveWorkbook.Name)
Print the name of *this* workbook (the one containing the VBA code)	MsgBox(ThisWorkbook.Name)
Add a new workbook	Workbooks.Add
Open a workbook	Workbooks.Open("C:\My Workbook.xlsx")
Save a workbook	Workbooks("My Workbook").Save
Close a workbook and save changes	Workbooks("My Workbook").Close SaveChanges:=True
Close a workbook without saving changes	Workbooks("My Workbook").Close SaveChanges:=False
Loop through all open workbooks	Dim wb As Workbook For Each wb In Application.Workbooks MsgBox (wb.Name) Next wb

Useful Keyboard Shortcuts

Press this	To do this
Alt+F11	Toggle between the VBA editor and the Excel window
F2	Open the Object browser
F4	Open the Properties window
F5	Runs the current procedure (or resumes running it if it has been paused)
F8	Starts “debug mode”, in which one line of code is executed at a time
Ctrl + Break	Halts the currently running procedure
Ctrl+J	List the properties and methods for the selected object

Back To Top