**Excelgraduate**

***Copyright 2023 excelgraduate.com | All Rights Reserved.***

***Web View:***[*https://excelgraduate.com/advanced-useful-vba-codes-for-excel/*](https://excelgraduate.com/advanced-useful-vba-codes-for-excel/)

**40 Advanced VBA Codes for Excel**

Excel is one of the most widely used programs in the world, and it's no secret that its [**VBA programming language**](https://excelgraduate.com/vba-in-excel/) is a powerful tool for automating tasks and increasing productivity. With advanced VBA codes, you can take your Excel skills to the next level and streamline your workflows even further. That's why I've compiled a list of 40 advanced useful VBA codes for Excel.

So without further ado, let's dive into the world of advanced VBA codes for Excel and see what's in store.

**Table of Contents**

[**File Export**](#_uapza8eka5xi) **2**

[Export Each Worksheet in a Workbook as Separate Excel Files](#_rkym4csonane) 2

[Export All Worksheets in a Workbook as Separate PDF Files](#_p8nkurfiostu) 2

[Export Worksheet as a PDF File Using Current Date & Time in the Filename with a Prompt](#_dp7p0cybdom0) 3

[Export Charts from Excel to PowerPoint](#_yee7zsi31wy8) 5

[Select and Export Range as PDF in Excel](#_akgyekh3zzt1) 6

[**Range Manipulation**](#_asn7m45pdcgl) **7**

[Select a Range to Apply Alternate Row Colors in Excel](#_crykokyau3oc) 7

[Remove Blank Rows in the Active Worksheet in Excel](#_v0flzky7kn58) 7

[Unhide All Rows and Columns in the Active Worksheet](#_a28xs9u9yg38) 8

[Unmerge All Merged Cells in Excel](#_vfkvzcl19sfy) 8

[**Sheet Manipulation**](#_stid5dcz0up2) **8**

[Delete Multiple Sheets Without Any Warning Prompt in Excel](#_ut5ixyhyvnf8) 8

[Unhide All Worksheets in Your Excel Workbook](#_j01t5af77mm7) 9

[Sort Worksheets Alphabetically in Excel](#_eokq827972mt) 10

[Check Whether a Specific Sheet Exists in a Workbook](#_j4mmyga23pou) 11

[**Workbook Manipulation**](#_gr9vfjo6lc5g) **11**

[Combine Multiple Excel Workbooks into a Single Workbook](#_5vdrjy6axrx0) 11

[Delete All Blank Worksheets from an Excel Workbook](#_dooydevqzeg2) 12

[Refresh All Pivot Tables in the Active Workbook](#_8hsdwcxmd5xl) 13

[Activate R1C1 Reference Style in Excel](#_l9qfey3kn8gi) 13

[Activate A1 Reference Style in Excel](#_kpk37nq6pxuh) 13

[**Data Manipulation**](#_9w5xi36ryt9v) **14**

[Create a List of All Sheets [Table of Contents] in Excel](#_e0dp1uup8bp2) 14

[Transfer Data from Excel to Powerpoint](#_49yoefpq4qse) 15

[Remove All Extra Spaces from a Selected Range in Excel](#_hq8rwq1fav4p) 16

[Search Value on Multiple Sheets in Excel](#_vrxezakt3bos) 17

[**Formatting**](#_sfbqcojhpz9f) **18**

[AutoFit All Non-Blank Columns in the Active Worksheet in Excel](#_cwi9ew9w5jad) 18

[AutoFit All Non-Blank Rows in the Active Worksheet in Excel](#_itoze3hmeexb) 19

[Highlight All the Cells Having Formulas in Excel](#_bmz6zgivgh05) 19

[Change Letter Case in Excel](#_gcs39kn0xogj) 19

[Highlight Cells with Wrongly Spelled Words in Excel](#_rxoo4dja519x) 20

[Change Font Size of All Sheets of an Entire Workbook](#_hxnp8dpyrprf) 21

[Remove All Text Wraps in the Active Worksheet](#_w3rju1z7w5bb) 22

[**Print File**](#_4rbdkhntlzd6) **22**

[Select and Print Multiple Ranges On Separate Pages](#_l4ys811px5r0) 22

[Print Selected Sheets Using Sheet Numbers](#_c1wjhwsgnwff) 23

[Print Selected Sheets By Mentioning the Sheet Names](#_ph88r8hibey1) 23

[Print the Active Worksheet with Comments](#_yh9n05tsv6e8) 24

[Print All the Hidden As Well As Visible Worksheets](#_8zyj8ye7cz16) 24

[**Miscellaneous**](#_kk9ev2guohme) **25**

[Select All Non-Blanks Cells in the Active Worksheet](#_322mcd8se5b8) 25

[Remove Page Breaks from the Active Worksheet](#_3qtx9tz8erdv) 25

[Count Total Number of Non-Blank Rows in a Selected Range in Excel](#_rpnxthyorqie) 25

[Count Total Number of Non-Blank Columns in the Active WorkSheet in Excel](#_5epe5bszhw6a) 26

[Read Contents of a Selected Range Using Text To Speech](#_xpohk6z4bxn5) 27

[Search on Google from Your Excel Worksheet](#_21agr22c1azw) 27

[**Conclusion**](#_dpb9j3dlonf) **28**

## **File Export**

### **Export Each Worksheet in a Workbook as Separate Excel Files**

*This code will allow you to export all the sheets in your workbook as separate Excel files. You will get a prompt window to choose a location to save all the Excel files.*

| Sub CopySheetsToNewWorkbooks()   Dim sheetToCopy As Worksheet  Dim saveFolder As String    ' Prompt the user to choose a directory to save the new sheets in  With Application.FileDialog(msoFileDialogFolderPicker)  .Title = "Select a folder to save the sheets in"  .Show  If .SelectedItems.Count > 0 Then  saveFolder = .SelectedItems(1) & "\"  Else  ' User canceled the dialog, exit the subroutine  Exit Sub  End If  End With    Application.ScreenUpdating = False    For Each sheetToCopy In ActiveWorkbook.Sheets  sheetToCopy.Copy  ActiveWorkbook.SaveAs Filename:=saveFolder & sheetToCopy.Name & ".xlsx"  ActiveWorkbook.Close saveChanges:=False  Next    Application.ScreenUpdating = True  End Sub |
| --- |

### **Export All Worksheets in a Workbook as Separate PDF Files**

*This code will allow you to export all the sheets in your workbook as separate pdf files. You will get a prompt window to choose a location to save all the pdf files.*

| Sub CopySheetsToNewPDFs()   Dim sheetToCopy As Worksheet  Dim saveFolder As String    ' Prompt the user to choose a directory to save the new PDFs in  With Application.FileDialog(msoFileDialogFolderPicker)  .Title = "Select a folder to save the PDFs in"  .Show  If .SelectedItems.Count > 0 Then  saveFolder = .SelectedItems(1) & "\"  Else  ' User cancelled the dialog, exit the subroutine  Exit Sub  End If  End With    Application.ScreenUpdating = False    For Each sheetToCopy In ActiveWorkbook.Sheets  sheetToCopy.ExportAsFixedFormat Type:=xlTypePDF, Filename:=saveFolder & sheetToCopy.Name & ".pdf"  Next    Application.ScreenUpdating = True  End Sub |
| --- |

### **Export Worksheet as a PDF File Using Current Date & Time in the Filename with a Prompt**

*This code can export a worksheet as a pdf file. The file name will start with the sheet name followed by the current date & time. You will get a prompt to choose a specific location to save the pdf file. Also you will be allowed to edit the file name while saving it.*

| Sub SavePDFWithDateAndTime()  Dim ws As Worksheet Dim wb As Workbook Dim timeStr As String Dim nameStr As String Dim pathStr As String Dim fileStr As String Dim pathAndFileStr As String Dim saveAsResult As Variant  On Error GoTo errorHandler  Set wb = activeWorkbook Set ws = ActiveSheet timeStr = Format(Now(), "mm.dd.yyyy\_hh.mm\_AM/PM")  pathStr = wb.Path If pathStr = "" Then pathStr = Application.DefaultFilePath End If pathStr = pathStr & ""  nameStr = Replace(ws.Name, " ", "") nameStr = Replace(nameStr, ".", "\_")  fileStr = nameStr & "\_" & timeStr & ".pdf" pathAndFileStr = pathStr & fileStr  saveAsResult = Application.GetSaveAsFilename \_ (InitialFileName:=pathAndFileStr, \_ FileFilter:="PDF Format (\*.pdf), \*.pdf", \_ Title:="Choose a folder & name")  If saveAsResult <> "False" Then ws.ExportAsFixedFormat \_ Type:=xlTypePDF, \_ fileName:=saveAsResult, \_ Quality:=xlQualityStandard, \_ IncludeDocProperties:=True, \_ IgnorePrintAreas:=False, \_ OpenAfterPublish:=False MsgBox "Pdf successfully saved at:" \_ & vbCrLf \_ & saveAsResult End If  exitHandler: Exit Sub errorHandler: MsgBox "Failed to save the PDF file." Resume exitHandler  End Sub |
| --- |

### **Export Charts from Excel to PowerPoint**

*This code exports a selected chart in Excel to a new* [***PowerPoint***](https://www.microsoft.com/en-ww/microsoft-365/powerpoint) *slide. It first checks if a chart is selected and displays a message if one isn't. If a chart is selected, it adds a slide with a title only layout, copies the selected chart, and pastes it onto the PowerPoint slide.*

| Sub ExportChartToPowerPoint()  ' Declare and initialize variables Dim pptApp As Object ' PowerPoint application Dim pptPres As Object ' PowerPoint presentation Dim pptSlide As Object ' PowerPoint slide Dim pptShape As Object ' PowerPoint shape  ' Check if a chart is selected If ActiveChart Is Nothing Then  MsgBox "Please select a chart to export.", vbExclamation, "No Chart Selected"  Exit Sub End If  ' Create a PowerPoint application if one doesn't exist If pptApp Is Nothing Then  Set pptApp = CreateObject("PowerPoint.Application") End If  On Error GoTo 0 ' Disable error handling  Application.ScreenUpdating = False ' Disable screen updating  ' Create a new PowerPoint presentation Set pptPres = pptApp.Presentations.Add  ' Add a slide with a title only layout Set pptSlide = pptPres.Slides.Add(1, 11) '11 = ppLayoutTitleOnly  ' Copy the selected chart and paste it onto the PowerPoint slide ActiveChart.ChartArea.Copy pptSlide.Shapes.Paste Set pptShape = pptSlide.Shapes(pptSlide.Shapes.Count)  ' Position the chart on the slide pptShape.Left = 200 pptShape.Top = 200  ' Show the PowerPoint application pptApp.Visible = True pptApp.Activate  Application.CutCopyMode = False ' Clear the clipboard  ' Enable screen updating Application.ScreenUpdating = True  End Sub |
| --- |

### **Select and Export Range as PDF in Excel**

*This code allows you to select and insert a range and then export the range as a pdf file. You will also get a prompt to manually choose a destination to save and rename the file.*

| Sub ExportRangeAsPDF()  ' Allow user to select a range  Dim selectedRange As Range  Set selectedRange = Application.InputBox("Select a range", Type:=8)    ' Define the filename and path for the exported PDF  Dim savePath As Variant  savePath = Application.GetSaveAsFilename(FileFilter:="PDF (\*.pdf), \*.pdf")    ' Export the range as a PDF file  If savePath <> False Then  selectedRange.ExportAsFixedFormat Type:=xlTypePDF, Filename:=savePath, Quality:=xlQualityStandard, IncludeDocProperties:=True, IgnorePrintAreas:=False  End If  End Sub |
| --- |

## **Range Manipulation**

### **Select a Range to Apply Alternate Row Colors in Excel**

*This VBA code allows you to select a range and apply alternate row colors to that range.*

| Sub ApplyRowColors()  ' Allow user to select a range  Dim selectedRange As Range  Set selectedRange = Application.InputBox("Select a range", Type:=8)    ' Define the colors to alternate between  Dim color1 As Long  color1 = RGB(242, 242, 242) ' light gray  Dim color2 As Long  color2 = RGB(255, 255, 255) ' white    ' Apply the colors to each row in the range  Dim numRows As Long  numRows = selectedRange.Rows.Count  Dim i As Long  For i = 1 To numRows  If i Mod 2 = 0 Then  selectedRange.Rows(i).Interior.Color = color1  Else  selectedRange.Rows(i).Interior.Color = color2  End If  Next i  End Sub |
| --- |

### **Remove Blank Rows in the Active Worksheet in Excel**

*This code can remove all the blank rows inside your data in Excel when you run it.*

| Sub RemoveBlankRows()   Dim rng As Range  Dim i As Long    'Set the range of cells to the used range of the active worksheet  Set rng = ActiveSheet.UsedRange    'Loop through each row in the range  For i = rng.Rows.Count To 1 Step -1  If WorksheetFunction.CountA(rng.Rows(i)) = 0 Then  'If the row is completely empty, delete it  rng.Rows(i).Delete  End If  Next i   End Sub |
| --- |

### **Unhide All Rows and Columns in the Active Worksheet**

*This code will let you unhide all the hidden rows and columns in one go.*

| Sub UnhideAllRowsColumns()   ActiveSheet.Cells.EntireRow.Hidden = False  ActiveSheet.Cells.EntireColumn.Hidden = False   End Sub |
| --- |

### **Unmerge All Merged Cells in Excel**

*This VBA code will unmerge all the merged cells in your active worksheet in Excel.*

| Sub UnmergeAllCells() ActiveSheet.Cells.UnMerge End Sub |
| --- |

## **Sheet Manipulation**

### **Delete Multiple Sheets Without Any Warning Prompt in Excel**

*This subroutine deletes multiple sheets without any warning prompt with the given names. Just run the code, insert sheet names to delete, separated by commas and see the magic.*

| Sub DeleteSheetsWithNames()   ' Declare variables  Dim currentSheet As Worksheet  Dim sheetNamesToDelete As Variant  Dim i As Long    ' Prompt the user to enter the sheet names to delete, separated by commas  sheetNamesToDelete = Split(InputBox("Enter the sheet names to delete, separated by commas"), ",")    ' Disable alerts to avoid confirmation messages  Application.DisplayAlerts = False    ' Loop through each worksheet in the workbook  For Each currentSheet In ThisWorkbook.Worksheets    ' Check if the current sheet name is in the array of sheets to delete  For i = LBound(sheetNamesToDelete) To UBound(sheetNamesToDelete)  If Trim(sheetNamesToDelete(i)) = currentSheet.Name Then  ' Delete the sheet and exit the inner loop once a match is found  currentSheet.Delete  Exit For  End If  Next i    Next currentSheet    ' Enable alerts again  Application.DisplayAlerts = True    ' Display a message to confirm the deletion  MsgBox "The sheets have been deleted successfully."   End Sub |
| --- |

### **Unhide All Worksheets in Your Excel Workbook**

*This code allows you to unhide all the worksheets at the same time.*

| Sub UnhideAllSheets()   Dim ws As Worksheet    For Each ws In ActiveWorkbook.Worksheets  ws.Visible = xlSheetVisible  Next ws   End Sub |
| --- |

### **Sort Worksheets Alphabetically in Excel**

*This VBA code sorts the worksheets in an Excel workbook alphabetically based on your input. It asks you to choose whether you want to sort the worksheets in ascending order (A-Z), descending order (Z-A), or exit the sorting process.*

| Sub AlphabeticallySortWorksheets()  Application.ScreenUpdating = False Dim sheetCount As Integer, i As Integer, j As Integer Dim sortOrder As VbMsgBoxResult  sortOrder = MsgBox("Click Yes to sort A-Z, No to sort Z-A, or Cancel to exit.", vbYesNoCancel) sheetCount = Sheets.Count For i = 1 To sheetCount - 1  For j = i + 1 To sheetCount  If sortOrder = vbYes Then  If UCase(Sheets(j).Name) < UCase(Sheets(i).Name) Then  Sheets(j).Move before:=Sheets(i)  End If  ElseIf sortOrder = vbNo Then  If UCase(Sheets(j).Name) > UCase(Sheets(i).Name) Then  Sheets(j).Move before:=Sheets(i)  End If  ElseIf sortOrder = vbCancel Then  MsgBox "Sorting worksheets cancelled."  Exit Sub  End If  Next j Next i Application.ScreenUpdating = True MsgBox "Worksheets have been sorted " & IIf(sortOrder = vbYes, "in ascending order (A-Z).", "in descending order (Z-A).")  End Sub |
| --- |

### **Check Whether a Specific Sheet Exists in a Workbook**

*This VBA code helps you check whether a sheet with a specific name exists in your Excel workbook or not. If it finds a match, it displays a message box telling you that the sheet exists and exits the sub. If it doesn't find a match, it displays another message box informing you that the sheet does not exist.*

| Sub CheckIfSheetExists()   Dim sheetName As String  Dim ws As Worksheet  sheetName = InputBox("Enter the name of the sheet you want to check.")    For Each ws In ThisWorkbook.Worksheets  If ws.Name = sheetName Then  MsgBox "The sheet " & sheetName & " exists in this workbook."  Exit Sub  End If  Next ws    MsgBox "The sheet " & sheetName & " does not exist in this workbook."   End Sub |
| --- |

## **Workbook Manipulation**

### **Combine Multiple Excel Workbooks into a Single Workbook**

*The following sub routine combines multiple Excel workbooks into a single workbook. It prompts the user to select multiple files using the File Dialog Box. It then opens each file, copies all worksheets and pastes them into the destination workbook.*

| Sub CombineWorkbooks()  ' Declare variables Dim fileCount, g As Integer Dim fileDialog As fileDialog Dim destinationWorkbook, sourceWorkbook As Workbook Dim sourceWorksheet As Worksheet  ' Set the destination workbook as the active workbook Set destinationWorkbook = Application.ActiveWorkbook  ' Open the File Dialog Box to allow the user to select multiple files Set fileDialog = Application.fileDialog(msoFileDialogFilePicker) fileDialog.AllowMultiSelect = True fileCount = fileDialog.Show  ' Loop through each selected file For g = 1 To fileDialog.SelectedItems.Count  ' Open the file and set it as the source workbook  Workbooks.Open fileDialog.SelectedItems(g)  Set sourceWorkbook = ActiveWorkbook    ' Loop through each worksheet in the source workbook and copy it to the destination workbook  For Each sourceWorksheet In sourceWorkbook.Worksheets  sourceWorksheet.Copy after:=destinationWorkbook.Sheets(destinationWorkbook.Worksheets.Count)  Next sourceWorksheet    ' Close the source workbook  sourceWorkbook.Close Next g  End Sub |
| --- |

### **Delete All Blank Worksheets from an Excel Workbook**

*This VBA code loops through all the worksheets in the active workbook and checks if each worksheet is blank. If a worksheet is blank, it is deleted without any confirmation message.*

| Sub DeleteBlankWorksheets()   Dim ws As Worksheet    Application.DisplayAlerts = False 'Disable alerts    For Each ws In ThisWorkbook.Worksheets  If Application.WorksheetFunction.CountA(ws.Cells) = 0 Then  ws.Delete  End If  Next ws    Application.DisplayAlerts = True 'Enable alerts  End Sub |
| --- |

### **Refresh All Pivot Tables in the Active Workbook**

*This code loops through all pivot tables in the active workbook using a For Each loop and then refreshes each pivot table using the RefreshTable method.*

| Sub RefreshAllPivotTables()  Dim pt As PivotTable    For Each pt In ActiveWorkbook.PivotTables  pt.RefreshTable  Next pt  End Sub |
| --- |

### **Activate R1C1 Reference Style in Excel**

*This code sets the reference style of Excel from A1 reference style to R1C1 reference style.*

| Sub ActivateR1C1ReferenceStyle()  Application.ReferenceStyle = xlR1C1 End Sub |
| --- |

### **Activate A1 Reference Style in Excel**

*This code sets the reference style of Excel from R1C1 reference style to A1 reference style.*

| Sub ActivateA1ReferenceStyle()  Application.ReferenceStyle = xlA1 End Sub |
| --- |

## **Data Manipulation**

### **Create a List of All Sheets [Table of Contents] in Excel**

*This VBA code creates a sheet named “Table of Contents” that lists all the other worksheets in the workbook, excluding the “Table of Contents” sheet itself. Then the code loops through all the worksheets in the workbook, excluding the “Table of Contents” sheet, and adds their names by inserting a hyperlink to each sheet.*

| Sub CreateTableOfContents()   Dim ws As Worksheet  Dim tocSheet As Worksheet  Dim lastRow As Long  Dim sheetName As String  Dim i As Long    ' Create a new sheet for the table of contents  Set tocSheet = ThisWorkbook.Sheets.Add(After:= \_  ThisWorkbook.Sheets(ThisWorkbook.Sheets.Count))  tocSheet.Name = "Table of Contents"    ' Set the column headings and format the table of contents  With tocSheet  .Range("A1").Value = "List of All Sheets"  .Range("A1").Font.Bold = True  .Range("A1").Font.Size = 12  .Range("A1").HorizontalAlignment = xlCenter  .Columns("A").AutoFit  .Range("A2:A" & .Rows.Count).Font.Size = 12  End With    ' Loop through all worksheets and add their names to the table of contents  i = 2 ' Start adding sheet names in row 2  For Each ws In ThisWorkbook.Worksheets  If ws.Name <> tocSheet.Name Then ' Exclude the table of contents sheet  sheetName = ws.Name    ' Add a hyperlink to the sheet in the table of contents  tocSheet.Hyperlinks.Add Anchor:=tocSheet.Range("A" & i), Address:="", \_  SubAddress:="'" & sheetName & "'!A1", TextToDisplay:=sheetName    i = i + 1 ' Move to the next row  End If  Next ws    ' Move the table of contents sheet to the first position in the workbook  tocSheet.Move Before:=ThisWorkbook.Sheets(1)   End Sub |
| --- |

### **Transfer Data from Excel to Powerpoint**

*This VBA code asks you to select a range of cells in your Excel sheet. Then it opens the PowerPoint application, adds a new presentation and slide to it, and pastes the selected range as a table onto the new PowerPoint slide.*

| Sub TransferDataToPowerPoint()    ' Declare variables  Dim xlRange As Range  Dim pptApp As Object  Dim pptPres As Object  Dim pptSlide As Object  Dim pptShape As Object    ' Prompt user to select a range in Excel  On Error Resume Next  Set xlRange = Application.InputBox(prompt:="Select a range to transfer to PowerPoint.", Type:=8)  On Error GoTo 0    ' Check if a range was selected  If xlRange Is Nothing Then  MsgBox "No range was selected. Please try again.", vbCritical  Exit Sub  End If    ' Create new PowerPoint presentation and add a new slide  Set pptApp = CreateObject("PowerPoint.Application")  pptApp.Visible = True  Set pptPres = pptApp.Presentations.Add  Set pptSlide = pptPres.Slides.Add(1, 12)    ' Copy range to clipboard  xlRange.Copy    ' Paste range onto PowerPoint slide as a table  Set pptShape = pptSlide.Shapes.PasteSpecial(DataType:=2)  pptShape.Left = 50  pptShape.Top = 100    ' Cleanup  Set pptShape = Nothing  Set pptSlide = Nothing  Set pptPres = Nothing  Set pptApp = Nothing  Set xlRange = Nothing   End Sub |
| --- |

### **Remove All Extra Spaces from a Selected Range in Excel**

*This code allows you to manually select a range of cells in an Excel workbook using a prompt. Then it removes any leading and trailing spaces from the cell values, as well as any extra spaces between words.*

| Sub RemoveSpaces()   Dim rng As Range  Dim cell As Range    'Prompt the user to select the range of cells to remove spaces from  On Error Resume Next  Set rng = Application.InputBox("Please select the range of cells to remove spaces from:", "Select Range", Type:=8)  On Error GoTo 0    'Check if the user cancelled the selection  If rng Is Nothing Then  MsgBox "No range was selected.", vbInformation  Exit Sub  End If    'Loop through each cell in the range  For Each cell In rng    'Remove leading and trailing spaces from the cell value  cell.Value = Trim(cell.Value)    'Remove any extra spaces between words  Do While InStr(cell.Value, " ") > 0  cell.Value = Replace(cell.Value, " ", " ")  Loop    Next cell   End Sub |
| --- |

### **Search Value on Multiple Sheets in Excel**

*This code allows you to search for a specific value on all the worksheets in the current workbook. It will prompt you to enter the search term and then loops through each worksheet in the workbook to find the first occurrence of the value. If it finds the value, it will display a message box showing the worksheet name and cell address where the value is found. If it does not find the value on any sheet, it displays a message indicating that the value was not found.*

| Sub SearchValueOnSheets()    ' Declare variables  Dim ws As Worksheet  Dim rngSearch As Range  Dim strSearch As String  Dim foundCell As Range    ' Prompt user to enter a search term  strSearch = InputBox("Enter the value you want to search for:")    ' Loop through all worksheets in the workbook  For Each ws In ThisWorkbook.Worksheets    ' Search for the value on the sheet  Set rngSearch = ws.Cells.Find(What:=strSearch, LookIn:=xlValues, LookAt:=xlWhole)    ' Check if the value was found  If Not rngSearch Is Nothing Then  Set foundCell = rngSearch  Exit For  End If    Next ws    ' Check if the value was found on any sheet  If Not foundCell Is Nothing Then  MsgBox "The value was found on sheet " & foundCell.Worksheet.Name & " in cell " & foundCell.Address & ".", vbInformation  Else  MsgBox "The value was not found on any sheet.", vbInformation  End If   End Sub |
| --- |

## **Formatting**

### **AutoFit All Non-Blank Columns in the Active Worksheet in Excel**

*This VBA code will autofit all non-blank columns in the active worksheet of an Excel workbook.*

| Sub AutoFitNonBlankColumns()   Dim lastCol As Long  Dim i As Long    ' Get the last column with data in the current worksheet  lastCol = Cells.Find("\*", SearchOrder:=xlByColumns, SearchDirection:=xlPrevious).Column    ' Loop through each column and autofit if there is non-blank data  For i = 1 To lastCol  If WorksheetFunction.CountA(Columns(i)) > 1 Then  Columns(i).AutoFit  End If  Next i  End Sub |
| --- |

### **AutoFit All Non-Blank Rows in the Active Worksheet in Excel**

*This VBA code will autofit all non-blank rows in the active worksheet of an Excel workbook.*

| Sub AutoFitNonBlankRows()   Dim lastRow As Long  Dim i As Long    lastRow = ActiveSheet.Cells(Rows.Count, 1).End(xlUp).Row    For i = 1 To lastRow  If WorksheetFunction.CountA(Rows(i)) > 0 Then  Rows(i).EntireRow.AutoFit  End If  Next i   End Sub |
| --- |

### **Highlight All the Cells Having Formulas in Excel**

*This VBA code loops through all the cells in the used range of the active sheet and checks if each cell contains a formula. If a cell contains a formula (i.e., if the first character of the cell is "="), then it changes the cell color to yellow.*

| Sub HighlightFormulaCells()   Dim cell As Range  For Each cell In ActiveSheet.UsedRange  If Left(cell.Formula, 1) = "=" Then  cell.Interior.Color = RGB(255, 255, 0) 'Set highlight color to yellow  End If  Next cell  End Sub |
| --- |

### **Change Letter Case in Excel**

*This subroutine changes the letter case of the selected cells based on user input. To change the case, you have to select the cells first and then run the code. It will work like magic.*

| Sub UpdateSelectedCellsCase()  ' Prompt the user to input a letter to indicate the desired case Dim caseType As String  caseType = InputBox("Enter 'a' for lowercase, 'b' for UPPERCASE, or 'c' for Proper Case." \_  & vbCrLf & vbCrLf & "Note: Only the alphabetic characters will be affected.")  ' Apply the selected case to each cell in the selection Select Case caseType  Case "a", "A"  For Each selectedCell In Application.Selection  selectedCell.Value = LCase(selectedCell.Value)  Next selectedCell   Case "b", "B"  For Each selectedCell In Application.Selection  selectedCell.Value = UCase(selectedCell.Value)  Next selectedCell   Case "c", "C"  For Each selectedCell In Application.Selection  selectedCell.Value = WorksheetFunction.Proper(selectedCell.Value)  Next selectedCell   Case Else  ' Display an error message and exit the subroutine  MsgBox "Invalid input. Please enter 'a', 'b', or 'c'.", vbExclamation, "Error"  Exit Sub  End Select  ' Display a completion message MsgBox "Case updated successfully!", vbInformation, "Complete"  End Sub |
| --- |

### **Highlight Cells with Wrongly Spelled Words in Excel**

*This code highlights the cells that have misspelled words in the active worksheet.*

| Sub HighlightMisspelledCells()  'This subroutine highlights the cells that have misspelled words.    Dim cell As Range  For Each cell In ActiveSheet.UsedRange  ' Check if the cell text has any misspelled words.  If Not Application.CheckSpelling(word:=cell.Text) Then  ' Highlight the cell with red color.  cell.Interior.Color = vbRed  End If  Next cell    ' Inform the user that the highlighting process has completed.  MsgBox "Misspelled cells have been highlighted.", vbInformation, "Highlight Misspelled Cells"  End Sub |
| --- |

### **Change Font Size of All Sheets of an Entire Workbook**

*This code prompts you to enter a font size, loops through all worksheets in the workbook. Then it changes the font size of all cells in each sheet to the entered font size. If you cancel or enter an invalid input (e.g. a negative number), the macro exits without making any changes.*

| Sub ChangeFontSize()  Dim ws As Worksheet  Dim fontSize As Integer    'Prompt user for font size  fontSize = InputBox("Enter font size:", "Font Size")    'Exit if user cancels or enters invalid input  If fontSize <= 0 Then Exit Sub    'Loop through all worksheets in the workbook  For Each ws In ThisWorkbook.Worksheets  'Change font size of all cells in the sheet  ws.Cells.Font.Size = fontSize  Next ws  End Sub |
| --- |

### **Remove All Text Wraps in the Active Worksheet**

*This code removes all the text wraps in your active worksheet in Excel.*

| Sub RemoveTextWrap()  Cells.WrapText = False End Sub |
| --- |

## **Print File**

### **Select and Print Multiple Ranges On Separate Pages**

*This piece of code allows you to specify the number of ranges first and then input those ranges by selecting cells using a prompt window. After that, it will tell you to save the ranges as separate pdf files to start printing each range on separate pages.*

| Sub PrintSelectedRanges()  'Declare variables Dim numRanges As Integer Dim currentRange As Integer Dim rangeAddress As Object Dim currentSheet As Worksheet Dim printArea As Object Dim Preview As Boolean  'Get the number of ranges to print from the user numRanges = InputBox("Enter the number of ranges to print:")  'Loop through each range and prompt the user to select and insert it For currentRange = 1 To numRanges  'Prompt the user to select and insert the current range  Set printArea = Application.InputBox("Select range " & currentRange & ":", Type:=8)    'Add the selected range to the overall print area  If currentRange = 1 Then  Set rangeAddress = printArea  Else  Set rangeAddress = Union(rangeAddress, printArea)  End If Next currentRange  'Set the print area for the active sheet and print it With ActiveSheet.PageSetup  .printArea = rangeAddress.Address  Preview = False  ActiveWindow.SelectedSheets.PrintOut Preview:=Preview End With  End Sub |
| --- |

### **Print Selected Sheets Using Sheet Numbers**

*This code will allow you to print a number of selected sheets. After running the code, you will be given the option to insert the starting and ending sheet numbers on your workbook. Based on your given sheet numbers, it will save and print those sheets one by one. Keep in mind that, this code only works on consecutive sheets.*

| Sub PrintSelectedSheets()  Dim sheetStart As Integer Dim sheetEnd As Integer  sheetStart = InputBox("Enter the starting sheet number:") sheetEnd = InputBox("Enter the ending sheet number:")  For i = sheetStart To sheetEnd  Worksheets(i).PrintOut Next i  End Sub |
| --- |

### **Print Selected Sheets By Mentioning the Sheet Names**

*By using this code, you can print a number of selected sheets by mentioning the sheet names on the code.*

| Sub PrintSheetsByName()  Worksheets("January").PrintOut Worksheets("February").PrintOut Worksheets("May").PrintOut Worksheets("August").PrintOut  End Sub |
| --- |

### **Print the Active Worksheet with Comments**

*This code will allow you to print out your active worksheet with all the comments in it.*

| Sub PrintSheetsWithComments()  'Display comments with comment indicators Application.DisplayCommentIndicator = xlCommentAndIndicator  'Set up printing options to include comments With ActiveSheet  .PageSetup.PrintComments = xlPrintInPlace  .PrintOut 'Print the active sheet with comments End With  End Sub |
| --- |

### **Print All the Hidden As Well As Visible Worksheets**

*This code can print out all the hidden as well as visible worksheets on your workbook. It will allow you to save all the sheets one by one first and then print them out respectively.*

| Sub PrintAllHiddenAndVisibleSheets()  'Declare variables Dim currentVisible As Long Dim workingSheet As Worksheet  'Loop through each worksheet in the active workbook For Each workingSheet In ActiveWorkbook.Worksheets  With workingSheet  'Save the current visibility state of the worksheet  currentVisible = .Visible  'Set the worksheet to be visible  .Visible = xlSheetVisible  'Print the worksheet  .PrintOut  'Restore the previous visibility state of the worksheet  .Visible = currentVisible  End With Next workingSheet  End Sub |
| --- |

## **Miscellaneous**

### **Select All Non-Blanks Cells in the Active Worksheet**

*This code selects all the cells with data in the active worksheet.*

| Sub SelectCellsWithData()   Dim ws As Worksheet  Set ws = ActiveSheet    Dim lastRow As Long  Dim lastColumn As Long  lastRow = ws.Cells.Find("\*", SearchOrder:=xlByRows, SearchDirection:=xlPrevious).Row  lastColumn = ws.Cells.Find("\*", SearchOrder:=xlByColumns, SearchDirection:=xlPrevious).Column    Dim dataRange As Range  Set dataRange = ws.Range(ws.Cells(1, 1), ws.Cells(lastRow, lastColumn))    dataRange.SpecialCells(xlCellTypeConstants).Select   End Sub |
| --- |

### **Remove Page Breaks from the Active Worksheet**

*This code lets you remove page breaks from the current worksheet in just one click.*

| Sub DisablePageBreaks()  ActiveSheet.DisplayPageBreaks = False End Sub |
| --- |

### **Count Total Number of Non-Blank Rows in a Selected Range in Excel**

*This subroutine counts the number of non-blank rows in the selected range. Just select a range and then run the code. It will show the count of all non-blank rows in a popped-up dialog box.*

| Sub CountNonBlankRows()  ' Declare and initialize variables Dim rowCount As Integer  rowCount = 0  ' Loop through each row in the selection For i = 1 To Selection.Rows.Count   ' Check if the first cell in the row is not blank  If Selection.Cells(i, 1) <> "" Then  rowCount = rowCount + 1 ' Increment the row count  End If  Next i  ' Display the row count in a message box MsgBox "Number of non-blank rows: " & rowCount  End Sub |
| --- |

### **Count Total Number of Non-Blank Columns in the Active WorkSheet in Excel**

*This code can count the total number of non-blank columns in the active worksheet in Excel. Just run the code and you will get count figure in a popped-up dialog box.*

| Sub CountNonBlankColumns()   ' Declare and initialize variables  Dim colCount As Integer  colCount = 0    ' Get the range of cells in the active worksheet  Dim dataRange As Range  Set dataRange = ActiveSheet.UsedRange    ' Loop through each column in the range  Dim col As Range  For Each col In dataRange.Columns    ' Check if the column has any non-blank cells  If Application.WorksheetFunction.CountA(col) > 0 Then  colCount = colCount + 1 ' Increment the column count  End If    Next col    ' Display the column count in a message box  MsgBox "Number of non-blank columns: " & colCount  End Sub |
| --- |

### **Read Contents of a Selected Range Using Text To Speech**

*This code will prompt you to select a range. Next, it will read the contents of each cell using text to speech.*

| Sub SpeakSelectedRange()  Dim myRange As Range  Set myRange = Application.InputBox(prompt:="Please select a range to speak", Type:=8)    For Each cell In myRange  SpeakText (cell.Value)  Next cell End Sub  Sub SpeakText(TextToSpeak As String)  Dim objVoice As Object  Set objVoice = CreateObject("SAPI.SpVoice")  objVoice.Speak TextToSpeak  End Sub |
| --- |

### **Search on Google from Your Excel Worksheet**

*This code will prompt you to enter a search query in an input box. Upon entering the query and clicking OK, the code replaces any spaces in the query with a + sign, creates a Google search URL by appending the q parameter (which represents the search query) to the base URL* [*https://www.google.com/search?q=*](https://www.google.com/search?q=)*. Finally, it opens the constructed URL in your default web browser.*

| Sub GoogleSearch()  Dim query As String  Dim url As String    query = InputBox("Enter your Google search query:")    If query <> "" Then  query = Replace(query, " ", "+")  url = "https://www.google.com/search?q=" & query  ActiveWorkbook.FollowHyperlink url  End If  End Sub |
| --- |

## **Conclusion**

I hope that this collection of 40 advanced useful VBA codes for Excel has been helpful to you and that you're able to apply them in your daily work. Remember, these codes are just a starting point, and there's always more to learn. Keep exploring the world of VBA and see how you can further customize Excel to suit your needs.

**Excelgraduate**

***Copyright 2023 excelgraduate.com | All Rights Reserved.***

***Web View:***[*https://excelgraduate.com/advanced-useful-vba-codes-for-excel/*](https://excelgraduate.com/advanced-useful-vba-codes-for-excel/)