



無料電子ブック

学習

excel-vba

Free unaffiliated eBook created from
Stack Overflow contributors.

#excel-vba

.....	1
1: excel-vba	2
.....	2
.....	2
VB.....	2
Excel.....	3
Examples.....	3
.....	3
.....	4
Visual Basic EditorVBE.....	5
.....	9
.....	14
Excel.....	16
2: Excel VBASQL -	20
Examples.....	20
VBAADODB.Connection.....	20
.....	20
.....	20
.....	20
a Windows.....	20
b SQL Server.....	21
SQL	21
.....	21
.....	21
.....	21
.....	22
3: Excel VBA	23
.....	23
Examples.....	23
xlVeryHidden.....	23

.Name.Index.CodeName	24
.....	26
Excel.....	26
-	27
4: Excel-VBA.....	29
.....	29
.....	29
Examples.....	29
.....	29
.....	29
With.....	30
-	31
Excel.....	32
.....	33
5: SheetWorksheet.....	36
.....	36
Examples.....	36
.....	36
6: VBAExcel.....	37
.....	37
Examples.....	37
ListObject.....	37
ListRows / ListColumns.....	37
Excel.....	38
7: VBAPowerPoint.....	39
.....	39
Examples.....	39
VBAPowerPoint.....	39
8: VBA.....	41
Examples.....	41
VBA.....	41
9: VBA.....	42

.....	42
Examples.....	42
"Option Explicit".....	42
.....	44
VB.....	45
.....	46
.....	47
GoTo 0.....	47
.....	47
GoTo <line>.....	48
.....	49
.....	50
ExcelActiveCellActiveSheet.....	51
.....	52
SELECTACTIVATE.....	53
.....	54
WorksheetFunctionUDF.....	55
.....	56
10: VBA.....	58
.....	58
Examples.....	58
FormatConditions.Add.....	58
.....	58
.....	58
XIFormatConditionType enumeration.....	58
.....	59
.....	59
.....	59
.....	60
.....	60
.....	60

.....	61
.....	61
.....	61
FormatConditions.AddUniqueValues.....	61
.....	61
.....	61
FormatConditions.AddTop10.....	61
5.....	61
FormatConditions.AddAboveAverage.....	62
.....	62
FormatConditions.AddIconSetCondition.....	62
IconSet.....	63
.....	64
.....	64
.....	64
11:	66
Examples.....	66
.....	66
.....	66
12:	68
.....	68
Examples.....	68
Excel.....	68
ExcelVBE.....	68
13:	69
.....	69
Examples.....	69
.....	69
14: ;.....	71
.....	71
.....	71

Examples.....	71
Smartfilter.....	71
15:	77
.....	77
Examples.....	77
.....	77
2.....	78
16:	81
Examples.....	81
.....	81
.....	82
SERIES.....	83
.....	85
17:	89
.....	89
Examples.....	89
Debug.Print.....	89
.....	89
.....	89
.....	90
.....	91
.....	91
18:	94
Examples.....	94
.....	94
19:	96
.....	96
Examples.....	96
.....	96
.....	98
.....	98
.....	98

20:	100
Examples	100
.....	100
.....	100
.....	100
.....	100
.....	100
.....	100
.....	100
.....	101
.....	101
.....	101
.....	101
.....	101
.....	101
.....	101
.....	102
.....	102
.....	102
.....	102
.....	102
21: /	104
Examples	104
/2	104
Merged Range	104
22:	105
Examples	105
.....	105
23: UDF	107
.....	107
.....	107

Examples.....	107
UDF - Hello World.....	107
.....	109
.....	110
24:	111
Examples.....	111
.....	111
.....	112
ActiveWorkbookThisWorkbook.....	112
.....	113
25:	116
.....	116
Examples.....	116
.....	116
.....	116
.....	117
.....	117
Range.CurrentRegion.....	118
.....	118
.....	119
.....	120
26:	122
Examples.....	122
.....	122
ActiveWorkbookThisWorkbook.....	122
A.....	123
.....	124
.....	124
27:	125
.....	125
Examples.....	125
.....	125

VBA.....	125
.....	126
.....	128
28: CustomDocumentProperties	130
.....	130
Examples.....	130
.....	130
29:	133
Examples.....	133
If.....	133
30:	135
.....	135
.....	135
Examples.....	135
.....	135
.....	136
.....	137
.....	137
.....	138
31:	139
Examples.....	139
.....	139
.....	139
Array.....	139
.....	139
Evaluate2D.....	140
Split.....	140
.....	140
.....	140
.....	140
[].....	141
.....	142

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [excel-vba](#)

It is an unofficial and free excel-vba ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official excel-vba.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

1: excel-vbaをいめる

Microsoft Excelには、VBAというマクロプログラミングがまれています。このプログラミングは、なくとも3つのリソースをします。

1. マクロをしてコードからExcelをにします。ほとんどの、ユーザーインターフェイスからExcelをしてユーザーができるは、Excel VBAでコードをすることによってできます。
2. しいカスタムワークシートをします。
3. Microsoft Word、PowerPoint、Internet Explorer、メモなどのアプリケーションとのインタラクトExcel

VBAはVisual Basic for Applicationsのです。これは、1990ば、Microsoft Excelのマクロをサポートする、できるVisual Basicプログラミングのカスタムバージョンです。

excel-vbaタグにされたやトピックは、Microsoft ExcelでのVBAのにでしていることをしてください。のをけるために、VBAのなされたトピックやはすべてされるべきです。

- トピックにする
 - ✓ ワークシート・オブジェクトのと
 - ✓ `WorksheetFunction`クラスとそれぞれの`xldirection`をしてをナビゲートする
- トピックの
 - "for each"ループをする
 - ✗ `MsgBox`クラスとどのようにメッセージをするには
 - VBAでのWinAPIの

バージョン

VB

バージョン	
VB6	1998-10-01
VB7	200166
WIN32	1998-10-01
WIN64	200166

バージョン	
マック	1998-10-01

Excel

バージョン	
16	2016-01-01
15	2013-01-01
14	2010-01-01
12	2007-01-01
11	2003-01-01
10	2001-01-01
9	1999-01-01
8	1997-01-01
7	1995-01-01
5	1993-01-01
2	1987-01-01

Examples

の

VBAでにをするには、`Dim`ステートメントをし、そのにとをします。がされずにされる、またはがされていないは`Variant`がりてられます。

モジュールのので`Option Explicit`ステートメントをすると、にすべてののをにできます「に[Option Explicit](#)」を。

`Option Explicit`をすると、をぐことができ、オブジェクトがされたタイプのままになるため、に`Option Explicit`をすることをくおめします。

```
Option Explicit

Sub Example()
    Dim a As Integer
    a = 2
```

```

Debug.Print a
'Outputs: 2

Dim b As Long
b = a + 2
Debug.Print b
'Outputs: 4

Dim c As String
c = "Hello, world!"
Debug.Print c
'Outputs: Hello, world!
End Sub

```

りとしてカンマをしてのを1にできますが、それぞれのをにするか、デフォルトでVariantにするがあります。

```

Dim Str As String, IntOne, IntTwo As Integer, Lng As Long
Debug.Print TypeName(Str) 'Output: String
Debug.Print TypeName(IntOne) 'Output: Variant <--- !!!
Debug.Print TypeName(IntTwo) 'Output: Integer
Debug.Print TypeName(Lng) 'Output: Long

```

データの\$@をしてをすることもできますが、これをするとはますますけられます。

```

Dim this$ 'String
Dim this% 'Integer
Dim this& 'Long
Dim this! 'Single
Dim this# 'Double
Dim this@ 'Currency

```

をするそののはのとおりです。

- Staticなもの Static CounterVariable as Integer

DimステートメントのわりにStaticステートメントをすると、されたはびしでそのをし
ます。

- Publicように Public CounterVariable as Integer

パブリックは、プロジェクトののプロシージャでできます。パブリックがモジュール
またはクラスモジュールでされているは、パブリックがされているプロジェクトをす
るすべてのプロジェクトですることできます。

- Privateような Private CounterVariable as Integer

プライベートは、じモジュールのプロシージャでのみできます。

ソースと

MSDN-

タイプVisual Basic

Visual Basic EditorVBEをく

ステップ1ワークブックをく

File Home Insert Page Layout Formulas Data Review View Developer Tell me what you want to do

Clipboard: Paste, Cut, Copy, Format Painter

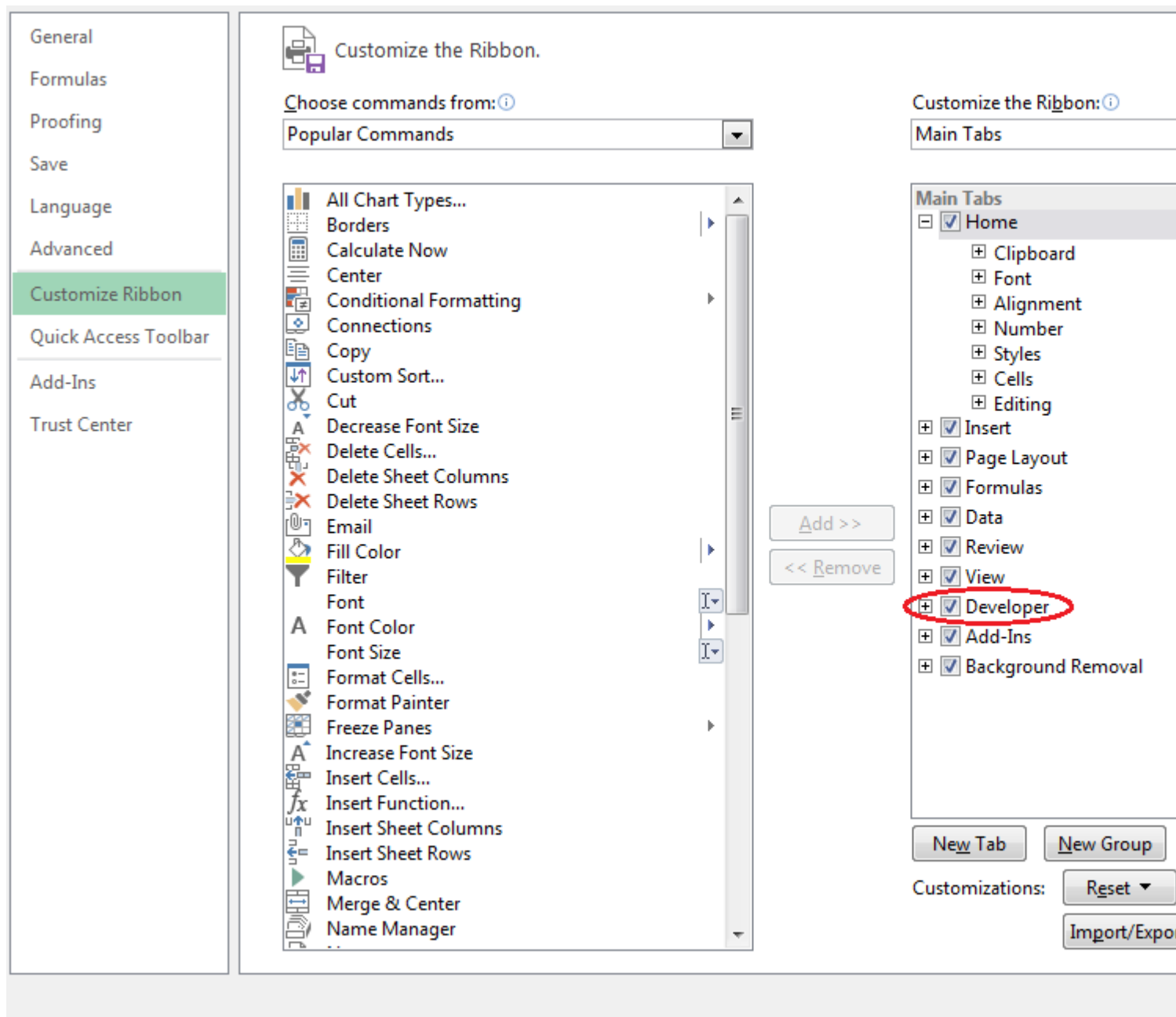
Font: Century gothic, 10, Bold, Italic, Underline, Text Color, Background Color

Alignment: Left, Center, Right, Justify, Indent, Decrease Indent, Increase Indent, Merge & Center, Wrap Text

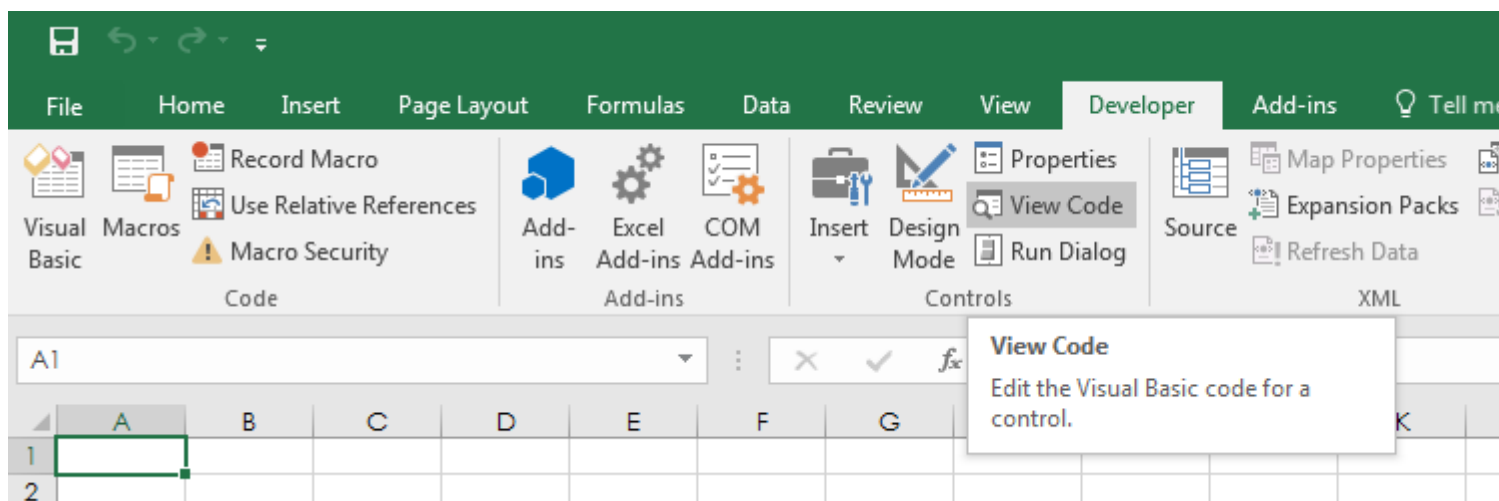
Number: General, Percentage, Currency, Accounting, Date, Time, Text, Fraction, Scientific, More Numbering

	A	B	C	D	E	F	G	H	I	J	K
1											
2											
3											
4											
5											
6											
7											
8											
9											
10											
11											
12											
13											
14											
15											
16											
17											
18											
19											
20											
21											
22											
23											
24											
25											
26											
27											
28											
29											
30											
31											
32											
33											
34											
35											
36											
37											
38											
39											
40											
41											
42											
43											
44											
45											

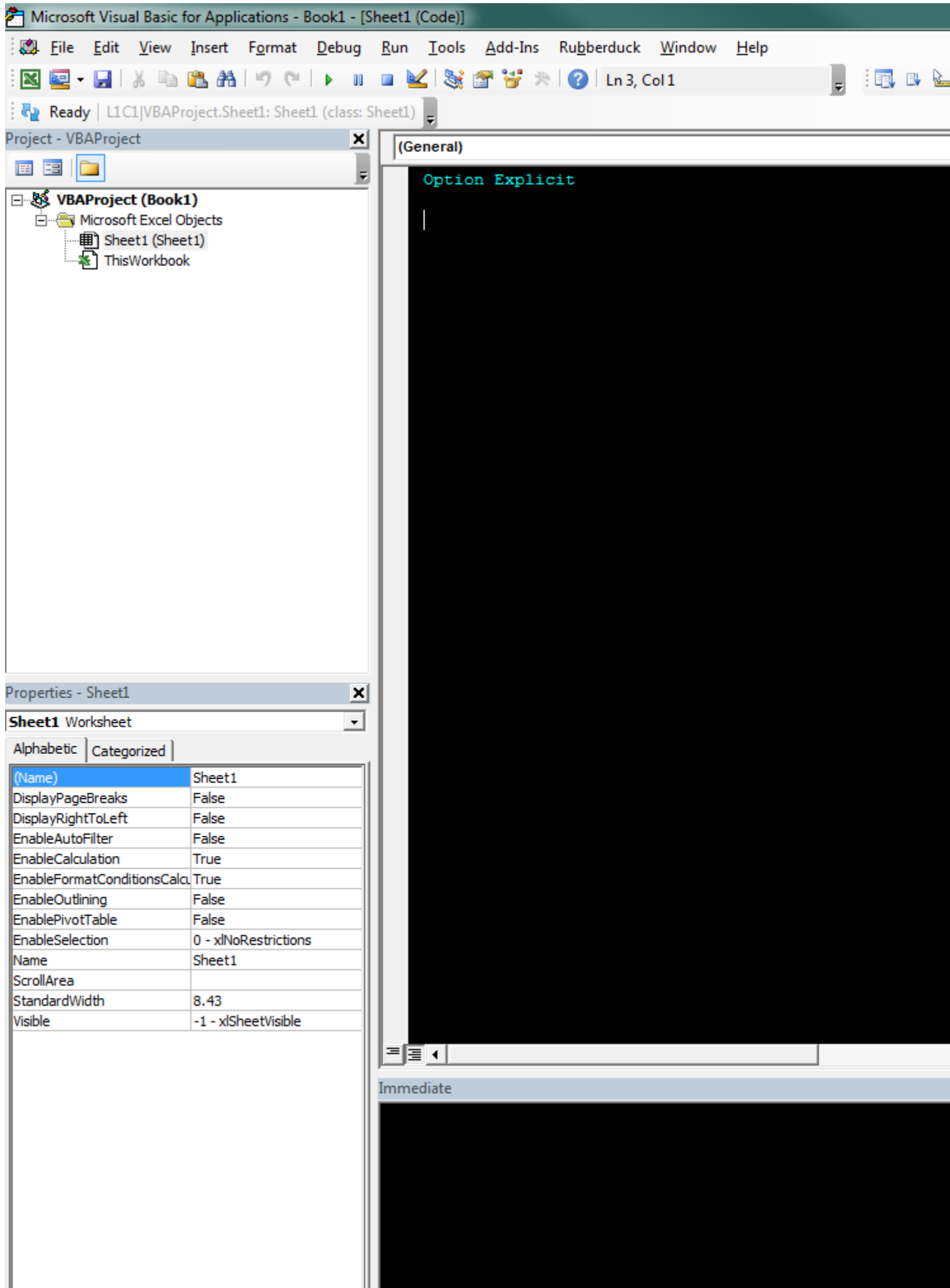
カスタマイズ」にみ、のチェックボックスをオンにします。



に、タブにし、[コードの]または[Visual Basic]をクリックします。

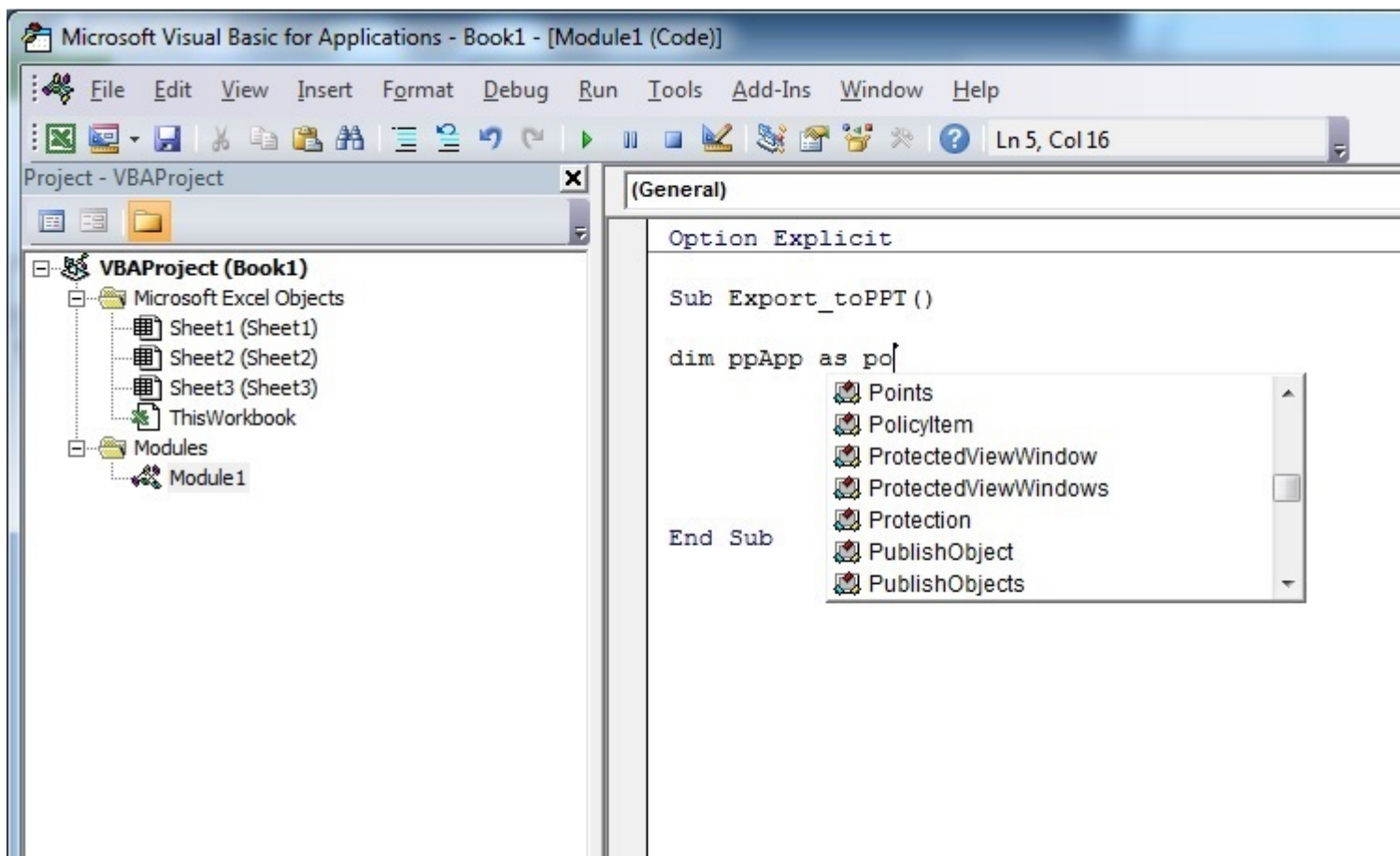


ステップ2オプションC[]タブ>[マクロ]>[]ボタンをクリックして、のマクロをきます
これら3つのオプションはすべて、Visual Basic EditorVBEをきます。

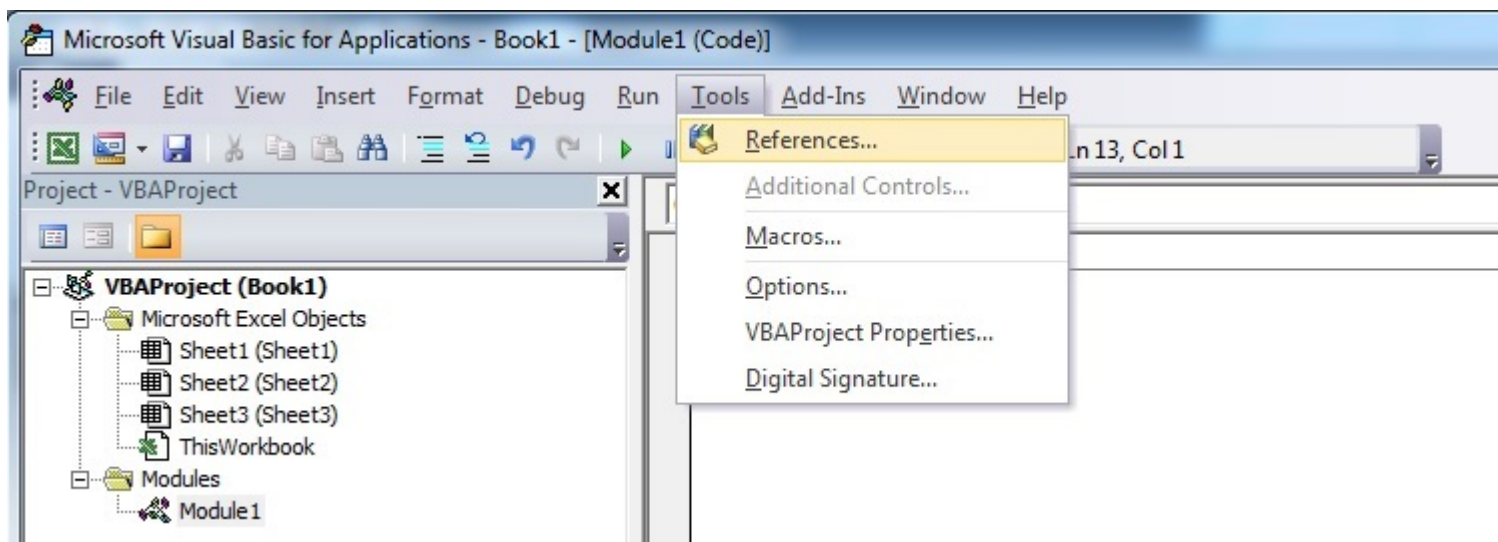


をするについてします。

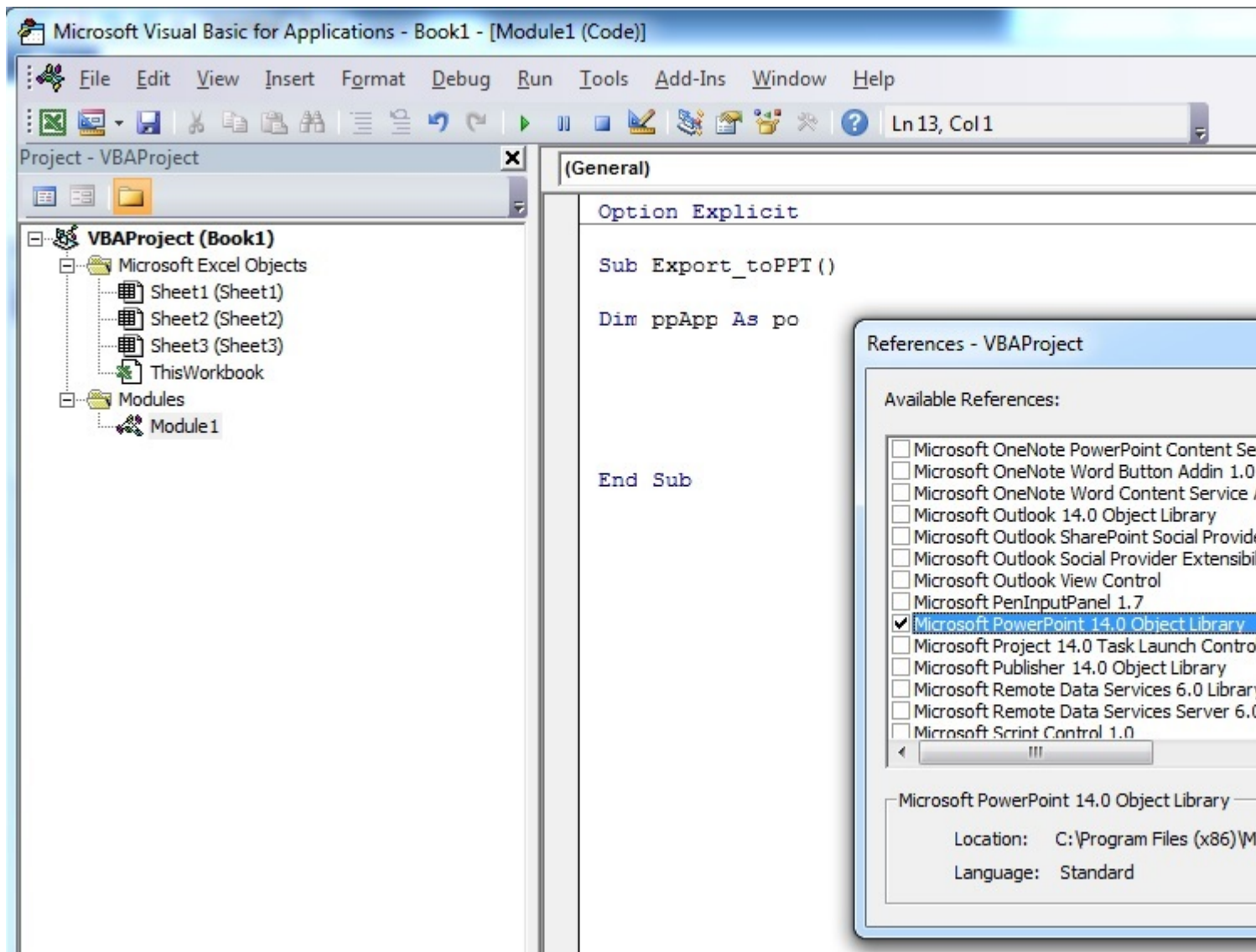
のは、のVBプロジェクトに*PowerPoint*ライブラリをするをしています。このとおり、PowerPoint Object Libraryはできません。



ステップ1 メニューツール -> を...

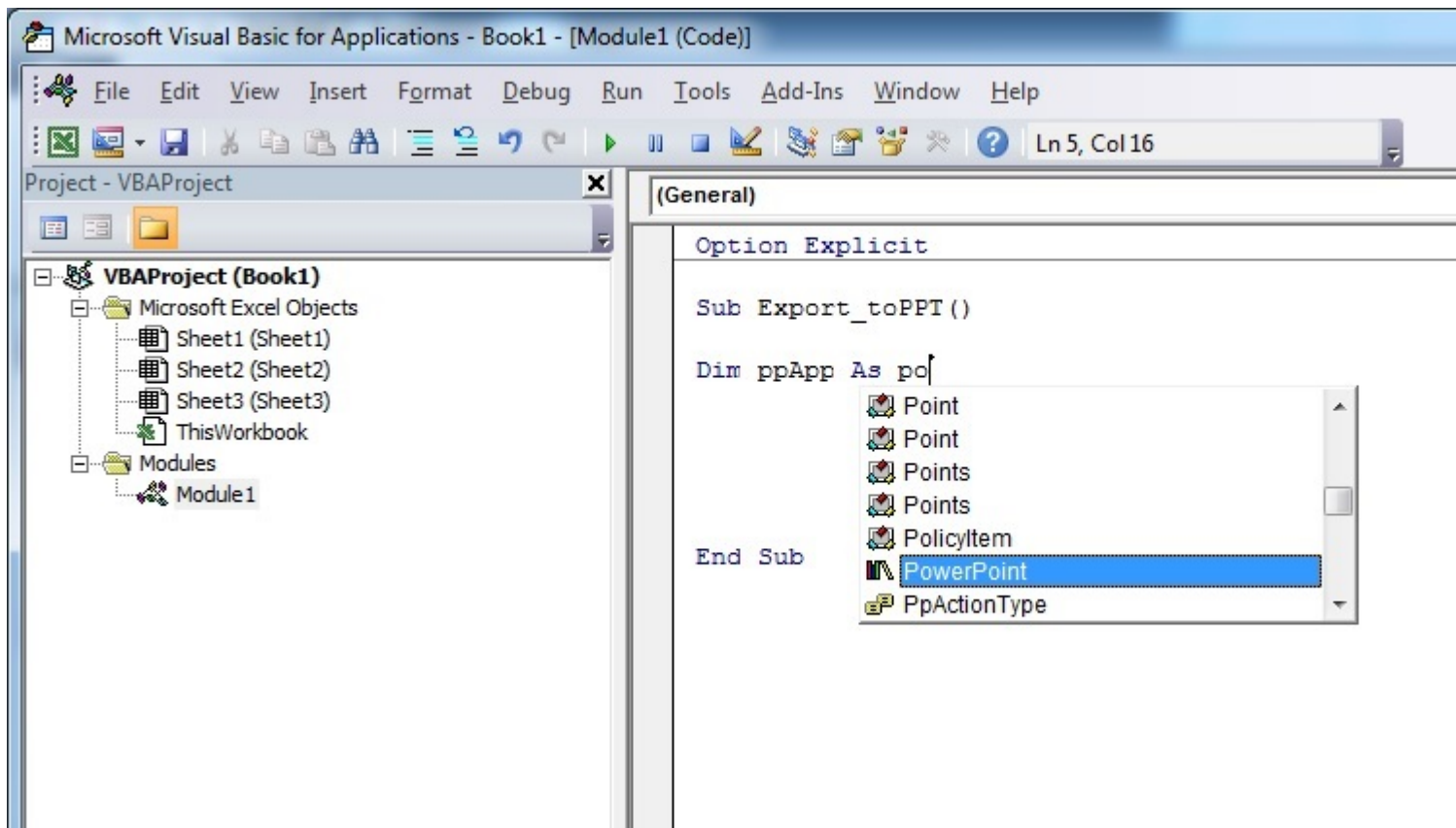


ステップ2 するをします。このでは、「**Microsoft PowerPoint 14.0 Object Library**」をして「**OK**」をします。

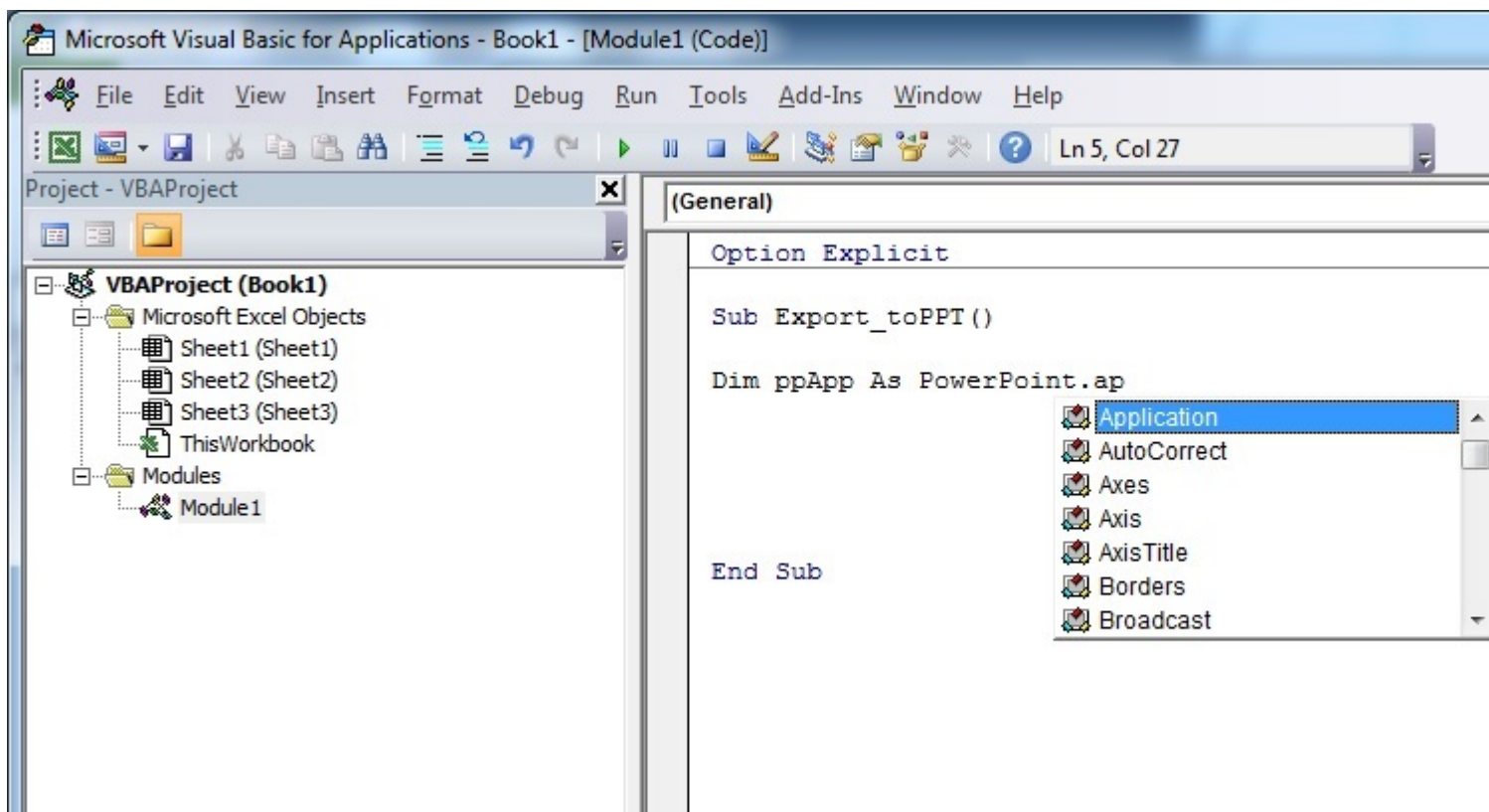


PowerPoint 14.0は、Office 2010のバージョンがPCにインストールされていることをします。

ステップ3 VBエディタで**Ctrl + Space**をすと、PowerPointのオートコンプリートオプションがされます。

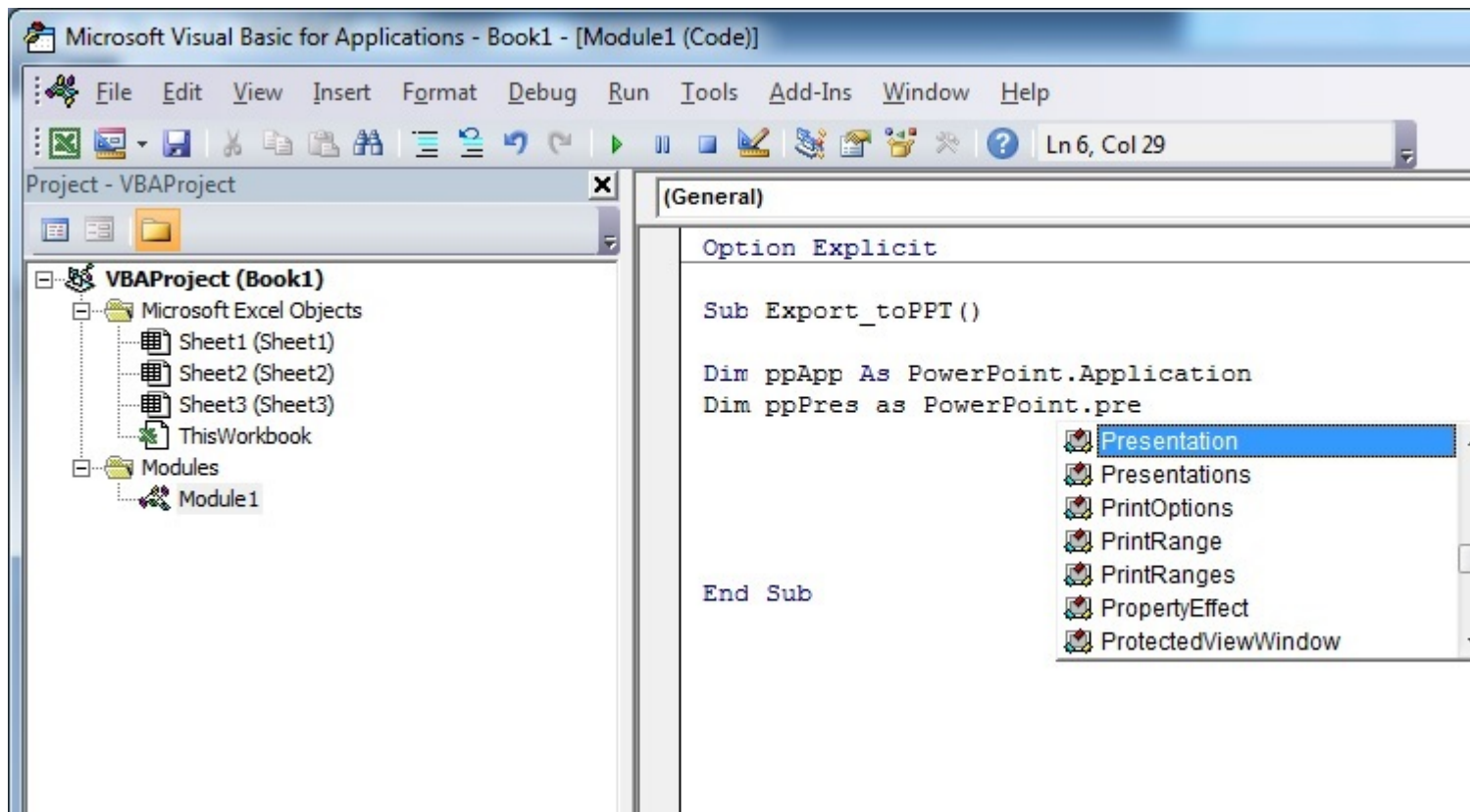


PowerPoint をしてをす . PowerPoint Object Libraryにするすべてのオブジェクトオプションをむのメニューがされます。このは、PowerPointのオブジェクトApplication をするをしています。

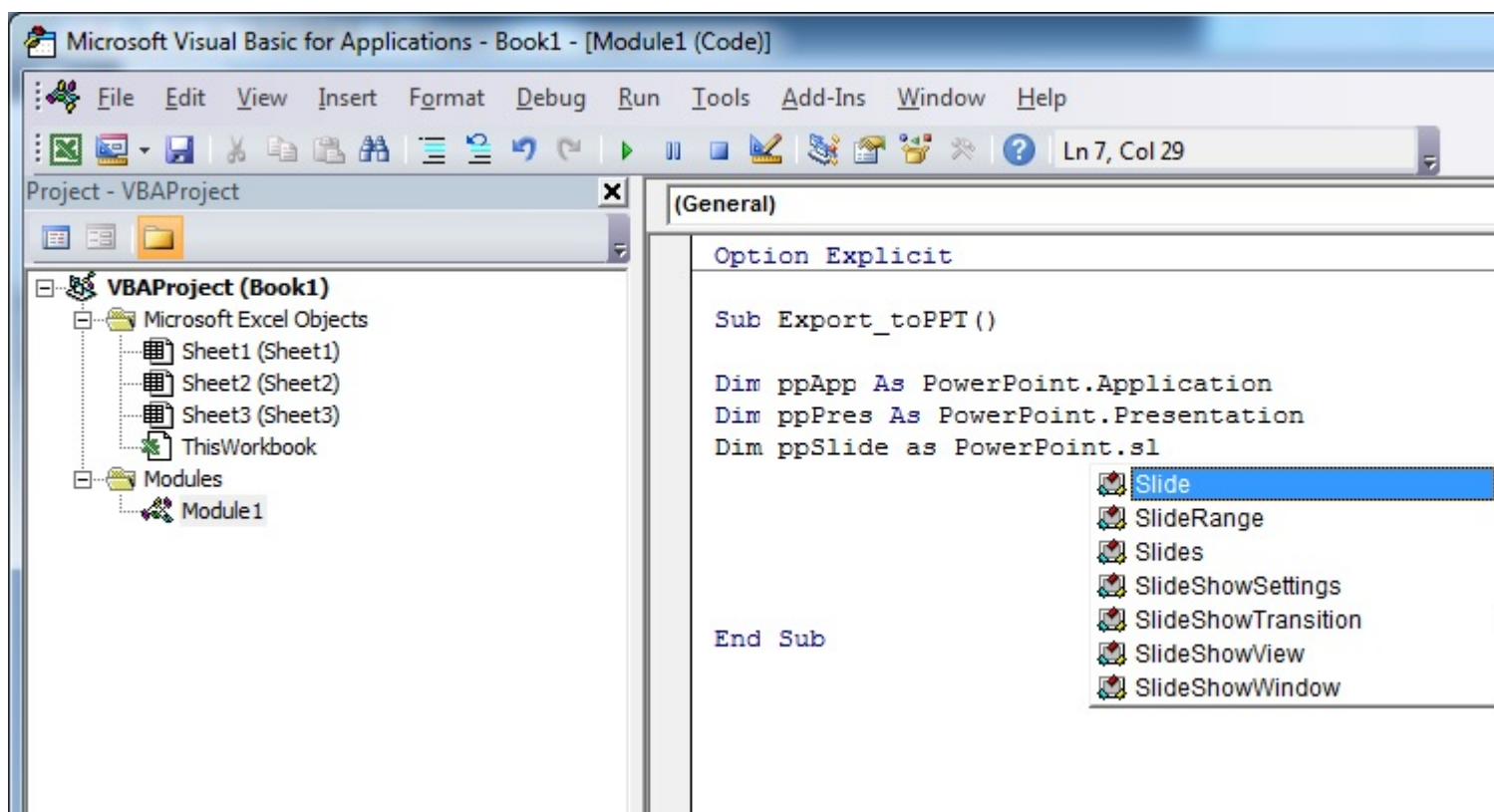


ステップ4 これで、ユーザーはPowerPointオブジェクトライブラリをしてさらにくのできます。

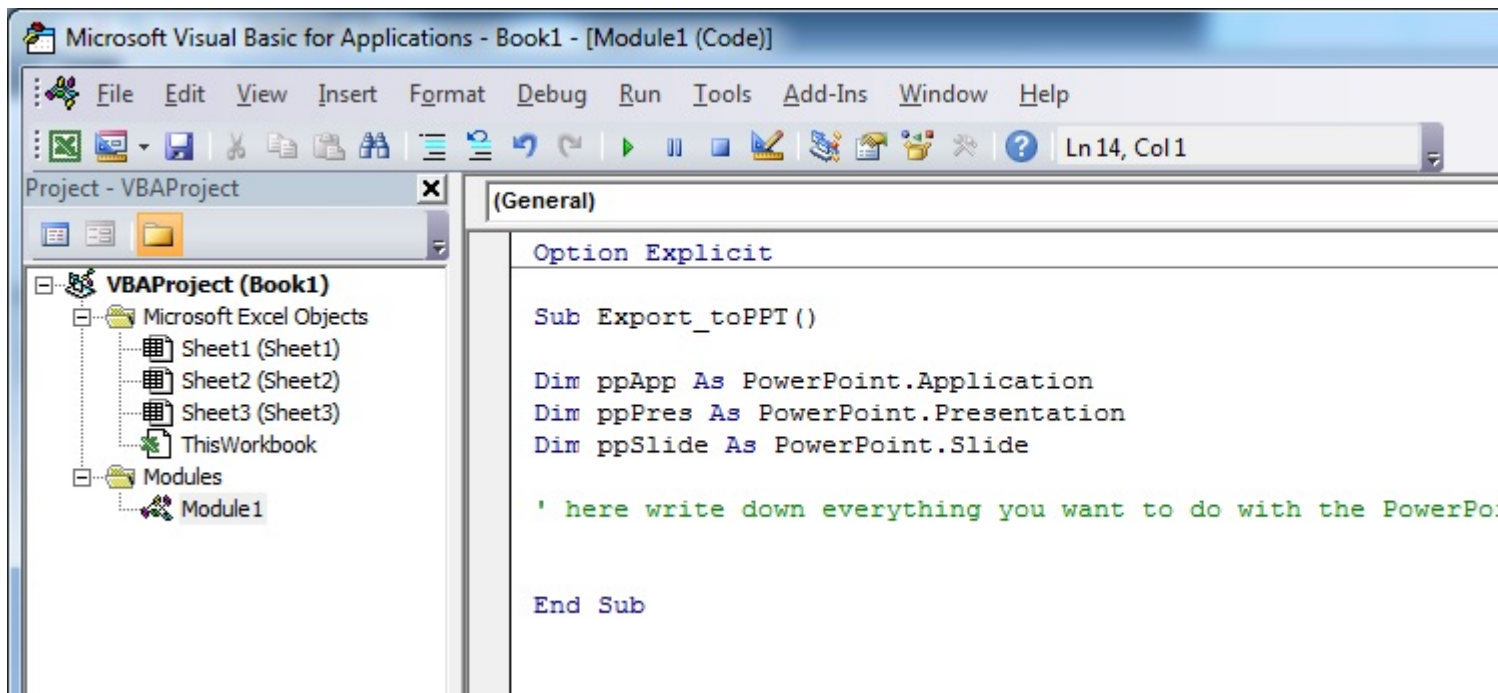
PowerPointオブジェクトライブラリのPresentationオブジェクトをしているをします。



PowerPointオブジェクトライブラリのSlideオブジェクトをしているのをします。



セクションはのスクリーンショットのようになり、ユーザーはコードでこれらのをすることができます。



このチュートリアルのコードバージョン

```
Option Explicit

Sub Export_toPPT()

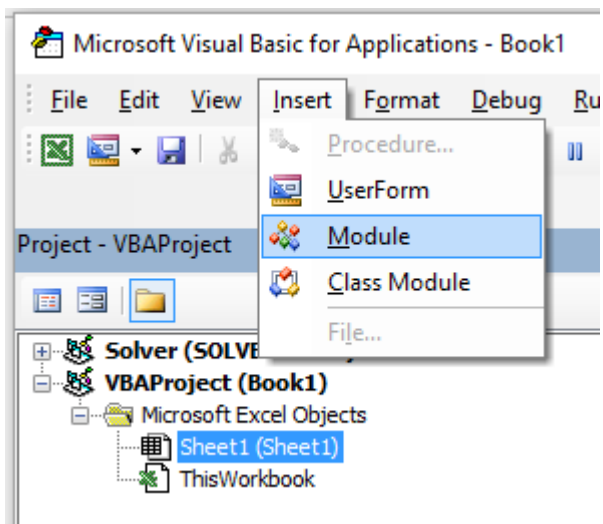
    Dim ppApp As PowerPoint.Application
    Dim ppPres As PowerPoint.Presentation
    Dim ppSlide As PowerPoint.Slide

    ' here write down everything you want to do with the PowerPoint Class and objects

End Sub
```

こんにちは

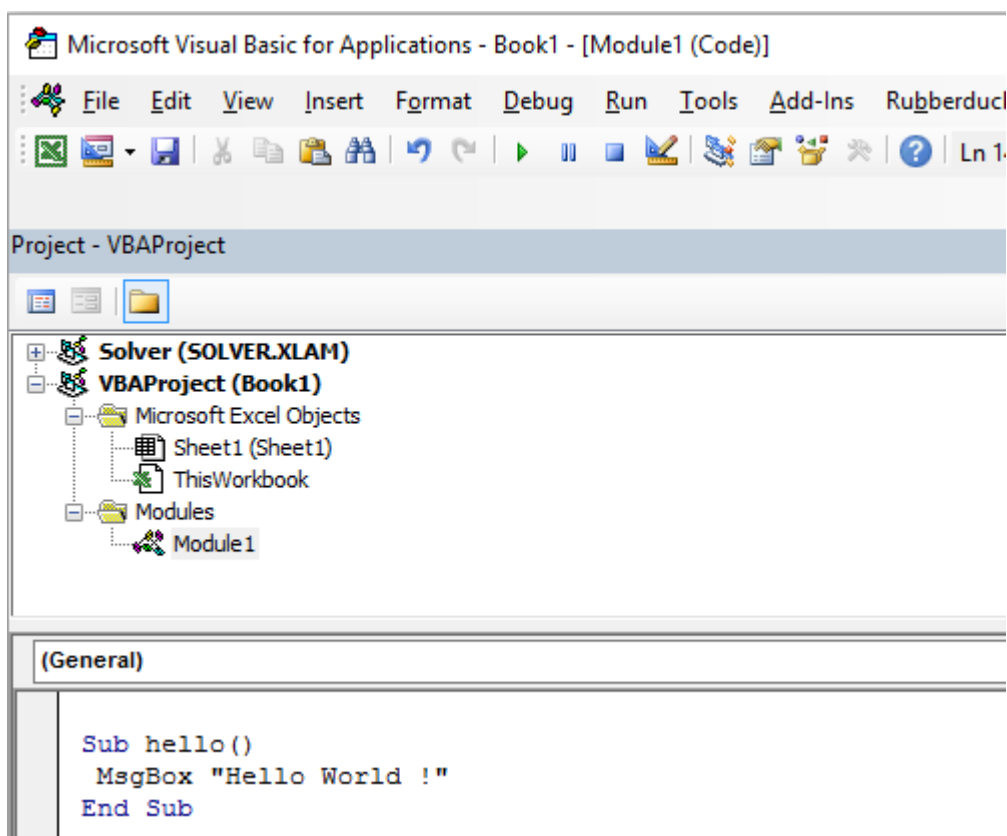
1. Visual Basic EditorをきますVisual Basic Editorをくをしてください
2. [] -> [モジュール]をクリックして、しいモジュールをします。



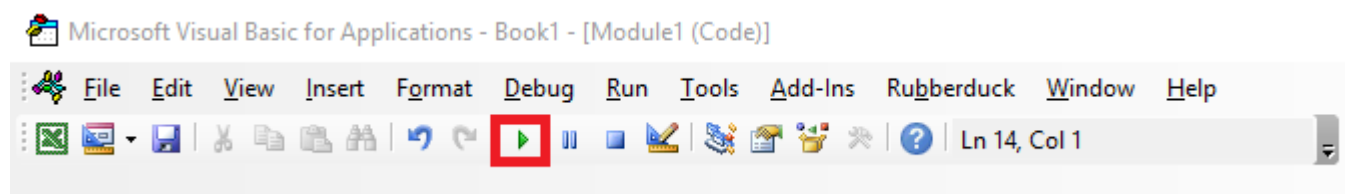
3. のコードをしいモジュールにコピーしてりけます。

```
Sub hello()  
    MsgBox "Hello World !"  
End Sub
```

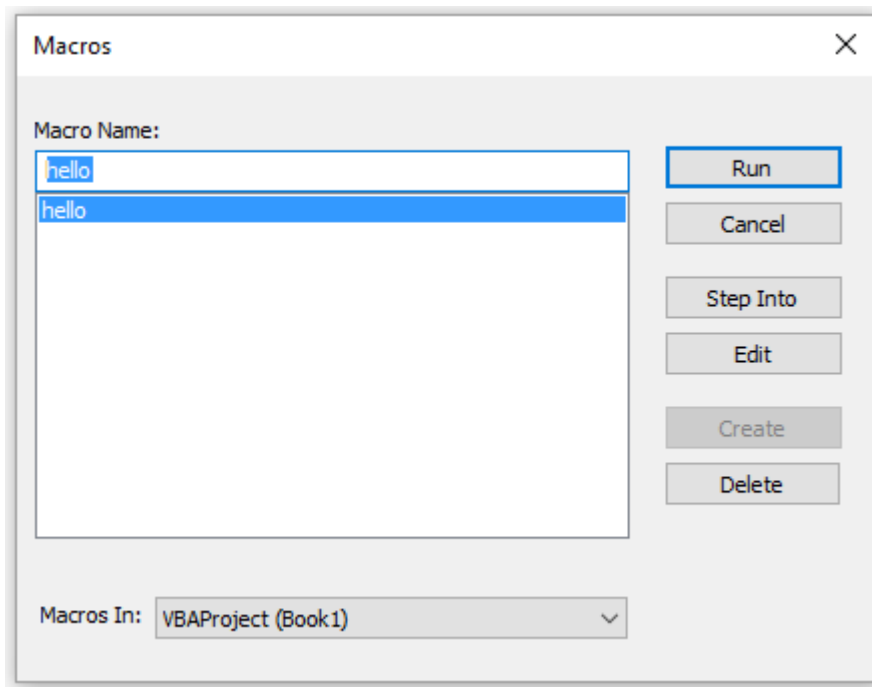
するには



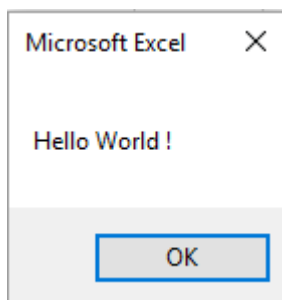
4. Visual Basicツールバーの「」またはF5キーをクリックして、プログラムをします。



5. にした「hello」サブをし、「_{Run}」をクリックし_{Run}。



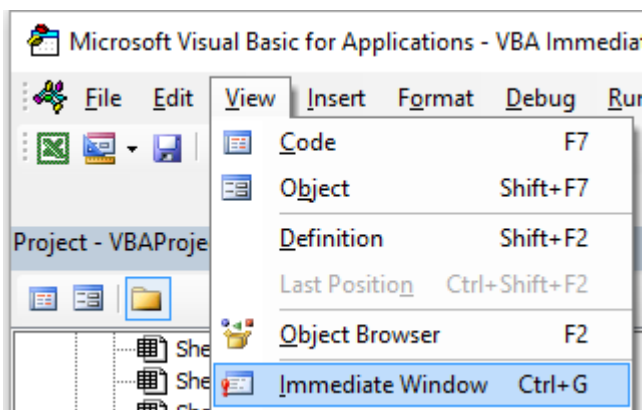
6. すると、のウィンドウがされます。



Excelオブジェクトモデル

このは、のための Excelオブジェクトモデルをやかにするです。

1. Visual Basic EditorVBEをきます。
2. [] - [イミディエイトウィンドウ]をクリックしてイミディエイトウィンドウまたは_{ctrl + G}をきます。



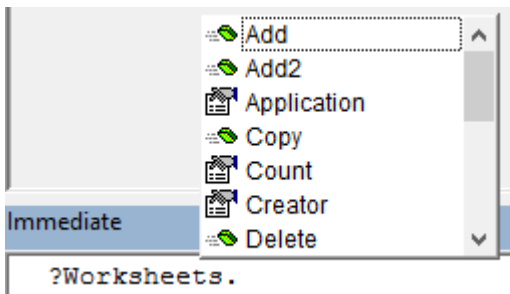
3. VBEののにイミディエイトウィンドウがされます。



このウィンドウでは、いくつかのVBAコードをテストすることができます。それでは、このコンソールにのようになしてみましよう。

```
?Worksheets.
```

VBEはインテリセンスをっており、ののようにツールチップをくがあります。



リストから.Countをするか、.Coutを.Countしてします。

```
?Worksheets.Count
```

4. Enterキーをします。がされ、1がされます。これは、ワークブックにするワークシートのをします。?は、Debug.Printのエイリアスです。

ワークシートはオブジェクトであり、カウントはメソッドです。Excelにはのオブジェクト Workbook、Worksheet、Range、Chart ..があり、それぞれにはのメソッドとプロパティがまれています。ExcelのVBAリファレンスには、オブジェクトのなリストがあります。ワークシートオブジェクトがここにされます。

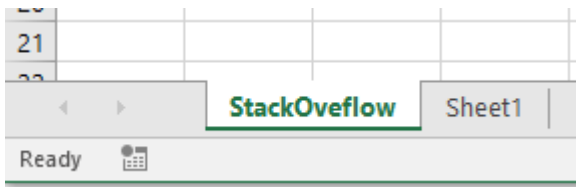
このExcel VBAリファレンスは、Excelオブジェクトモデルにするなになるはずです。

5. はせずに、の、をしてみましよう?

```
Worksheets.Add().Name = "StackOveflow"
```

6. Enterをします。これにより、StackOverflow.というしいワークシートがされます

```
StackOverflow.
```



このをするには、のExcelリファレンスのAddをみむがあります。あなたはをつけるでしょう

Add: Creates a new worksheet, chart, or macro sheet.
The new worksheet becomes the active sheet.
Return Value: An Object value that represents the new worksheet, chart, or macro sheet.

したがって、Worksheets.Add() はしいワークシートをしてします。sなし ワークシートオブジェクトでつけることができるドキュメントにし、Name そのプロパティの1ですこちら。これはのよ

Worksheet.Name Property: Returns or sets a String value that represents the object name.

したがって、なるオブジェクトをべることによって、このコードをすることができます

```
Worksheets.Add().Name = "StackOveflow"
```

Add() はしいワークシートをしてし、そのワークシートにをし、Name プロパティを "StackOverflow" にします

さて、Excelにはいくつかのオブジェクトがまれています。これらのオブジェクトは、じクラスの1つまたはのExcelオブジェクトのコレクションでされています。Worksheet オブジェクトのコレクションであるWorksheets のです。オブジェクトには、プログラマができるいくつかのプロパティとメソッドがあります。

Excelオブジェクトモデルは、Excel オブジェクトをします。

すべてのオブジェクトのにApplicationオブジェクトがあり、Excelインスタンスをします。メソッドをびすか、プロパティを/できるようにするためには、オブジェクトへののがとなるため、VBAでのプログラミングではこのをよくするがあります。

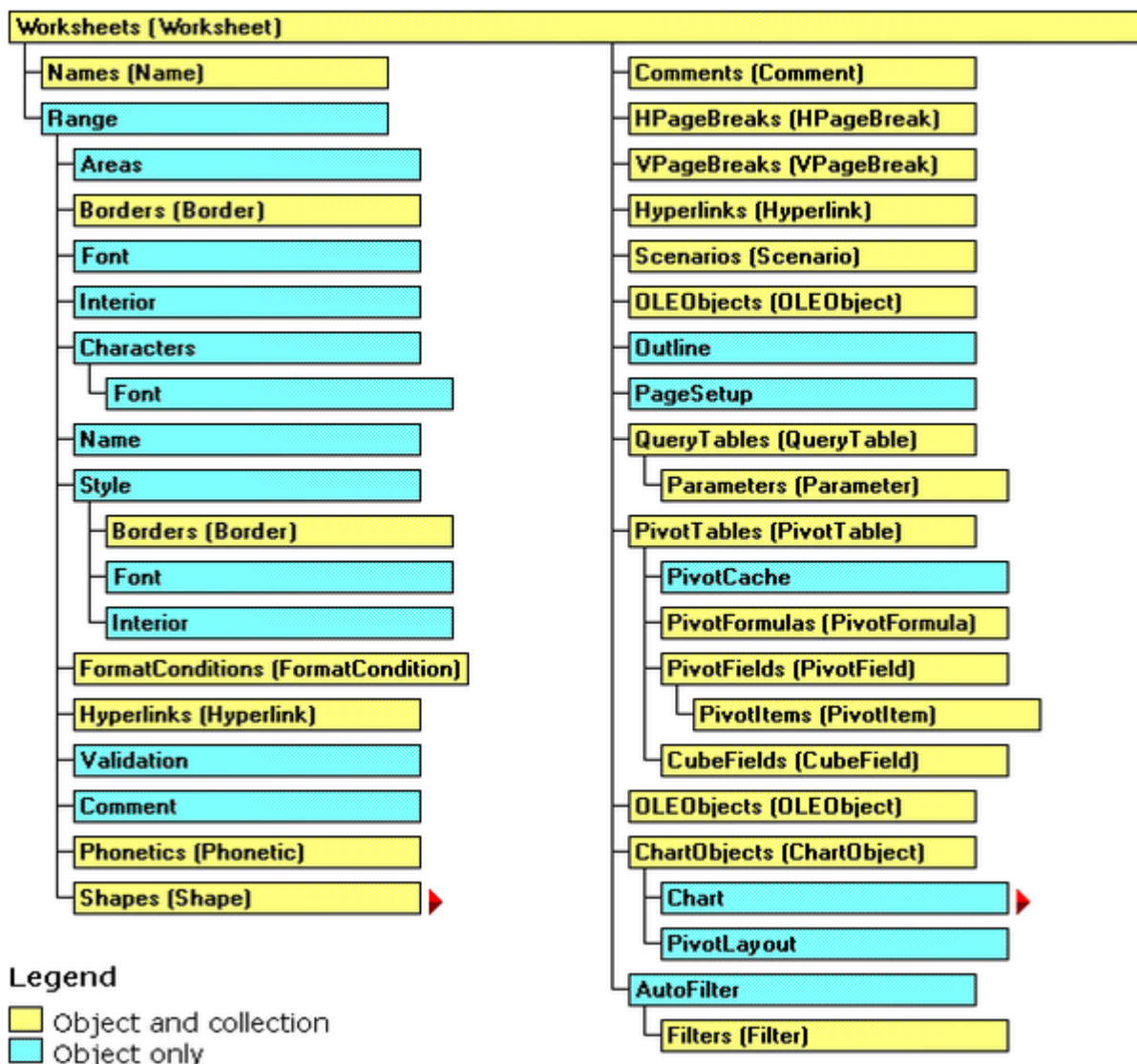
にされたExcelオブジェクトモデルは、

```
Application
Workbooks
Workbook
Worksheets
Worksheet
Range
```

Excel 2007のように、ワークシートオブジェクトのなバージョンをにします。

Microsoft Excel Objects (Worksheet)

See Also



なExcelオブジェクトモデルは[ここに](#)あります。

に、Excelオブジェクトモデルのである `events Workbook.WindowActivate` があるオブジェクトもあり `events`。

オンラインでexcel-vbaをいめるをむ <https://riptutorial.com/ja/excel-vba/topic/777/excel-vbaをいめる>

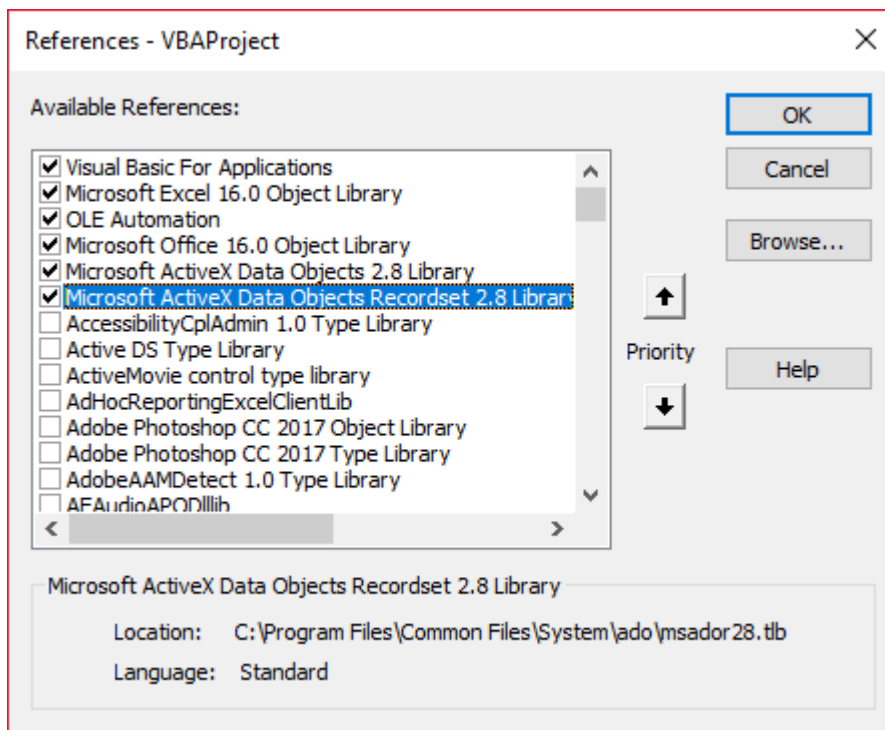
2: Excel VBAのSQL - ベストプラクティス

Examples

VBAでADODB.Connectionをするには

プロジェクトへののをします。

- Microsoft ActiveXデータオブジェクト2.8ライブラリ
- Microsoft ActiveXデータオブジェクトRecordset 2.8ライブラリ



をする

```
Private mDataBase As New ADODB.Connection
Private mRS As New ADODB.Recordset
Private mCmd As New ADODB.Command
```

をする

a。 Windowsで

```
Private Sub OpenConnection(pServer As String, pCatalog As String)
    Call mDataBase.Open("Provider=SQLOLEDB;Initial Catalog=" & pCatalog & ";Data Source=" &
pServer & ";Integrated Security=SSPI")
    mCmd.ActiveConnection = mDataBase
End Sub
```

b。 SQL Serverをする

```
Private Sub OpenConnection2(pServer As String, pCatalog As String, pUser As String, pPsw As
String)
    Call mDataBase.Open("Provider=SQLOLEDB;Initial Catalog=" & pCatalog & ";Data Source=" &
pServer & ";Integrated Security=SSPI;User ID=" & pUser & ";Password=" & pPsw)
    mCmd.ActiveConnection = mDataBase
End Sub
```

SQLコマンドをする

```
Private Sub ExecuteCmd(sql As String)
    mCmd.CommandText = sql
    Set mRS = mCmd.Execute
End Sub
```

レコードセットからデータをみむ

```
Private Sub ReadRS()
    Do While Not (mRS.EOF)
        Debug.Print "ShipperID: " & mRS.Fields("ShipperID").Value & " CompanyName: " &
mRS.Fields("CompanyName").Value & " Phone: " & mRS.Fields("Phone").Value
        Call mRS.MoveNext
    Loop
End Sub
```

をじる

```
Private Sub CloseConnection()
    Call mDataBase.Close
    Set mRS = Nothing
    Set mCmd = Nothing
    Set mDataBase = Nothing
End Sub
```

どうやってうのですか

```
Public Sub Program()
```

```
Call OpenConnection("ServerName", "NORTHWND")
Call ExecuteCmd("INSERT INTO [NORTHWND].[dbo].[Shippers] ([CompanyName],[Phone]) Values
('speedy shipping','(503) 555-1234')")
Call ExecuteCmd("SELECT * FROM [NORTHWND].[dbo].[Shippers]")
Call ReadRS
Call CloseConnection
End Sub
```

ShipperID1Speedy Express Phone503555-9831

ShipperID2United Package503555-3199

ShipperID3 CompanyName503555-9931

ShipperID4 CompanyNameスピーディーな503555-1234

オンラインでExcel VBAのSQL - ベストプラクティスをむ <https://riptutorial.com/ja/excel-vba/topic/9958/excel-vbaのsql----ベストプラクティス>

3: Excel VBAのヒントとテクニック

このトピックは、SOユーザーがコーディングでしたことでされたさまざまなヒントとテクニックからされています。これらは、くの、なや、Excelをよりなでするをするのです。

Examples

xlVeryHiddenシートの

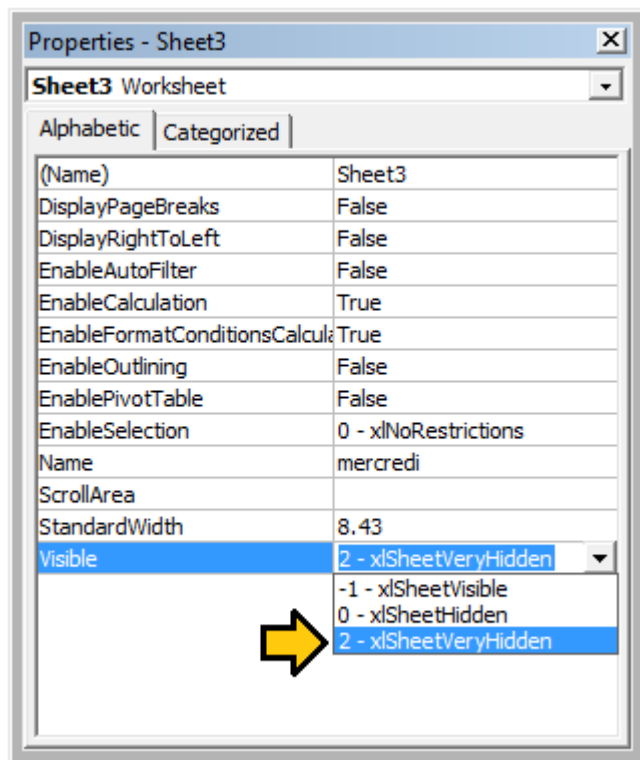
ExcelのワークシートにはVisibleプロパティの3つのオプションがあります。これらのオプションは、xlSheetVisibilityのでされ、のとおりです。

1. xlVisibleまたはxlSheetVisible -1 しいシートのデフォルト
2. xlHiddenまたはxlSheetHidden 0
3. xlVeryHidden xlSheetVeryHidden 2

シートは、シートのデフォルトのをします。シートタブバーにされ、にしています。のシートはシートタブバーからされているためできません。ただし、シートのタブをクリックして[]をすると、のシートをExcelウィンドウからすることが出来ます

、にのシートは、Visual Basic Editorからのみアクセスできます。これにより、Excelのインスタンスでデータをし、エンドユーザーからにするのあるデータをするためのになツールになります。VBAコードのきでシートにアクセスでき、されたデータをにできます。

ワークシートの.VisibleプロパティをxlSheetVeryHiddenにでするには、VBEのPropertiesウィンドウ `F4` をき、するワークシートをし、13のドロップダウンをしてします。



コードでワークシートの.VisibleプロパティをxlSheetVeryHidden¹にするには、.Visibleプロパティにアクセスしてしをりてます。

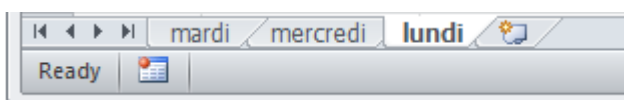
```
with Sheet3
    .Visible = xlSheetVeryHidden
end with
```

¹xlVeryHiddenとxlSheetVeryHiddenのが2のをしますこれらはがあります。

ワークシート.Name、.Indexまたは.CodeName

「ベストプラクティス」は、オブジェクトがワークシートをにするようにしています。ワークシートは、.Nameプロパティ、の.Indexプロパティ、または.CodeNameプロパティでできますが、ユーザーはタブをドラッグするか、じタブをダブルクリックしてワークシートのをするだけでワークシートのキューをべえることができます。されていないブックにします。

の3つのワークシートをえてみましょう。、のに3つのワークシートのをし、これらをするVBAのサブプロシージャをコードしました。ここで、1のユーザーがて、がワークシート・キューのにし、のユーザーがるとし、ワークシートがフランスでよくえるとしたとします。のようなワークシートのタブ・キューをつワークブックがされました。



のいずれかのワークシートメソッドをしていたは、コードがれてしまいます。

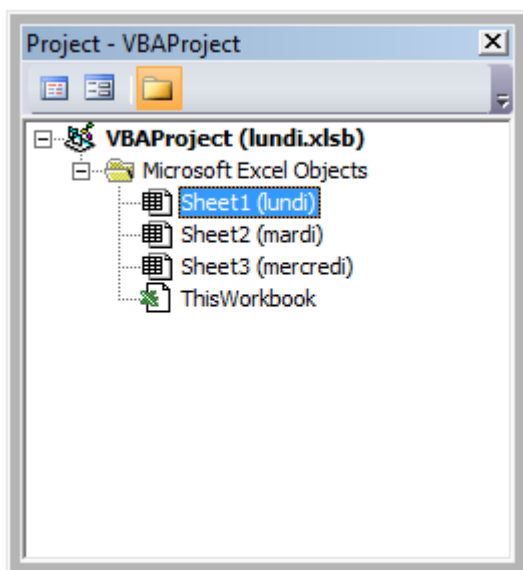
```
'reference worksheet by .Name
with worksheets("Monday")
    'operation code here; for example:
    .Range(.Cells(2, "A"), .Cells(.Rows.Count, "A").End(xlUp)) = 1
end with

'reference worksheet by ordinal .Index
with worksheets(1)
    'operation code here; for example:
    .Range(.Cells(2, "A"), .Cells(.Rows.Count, "A").End(xlUp)) = 1
end with
```

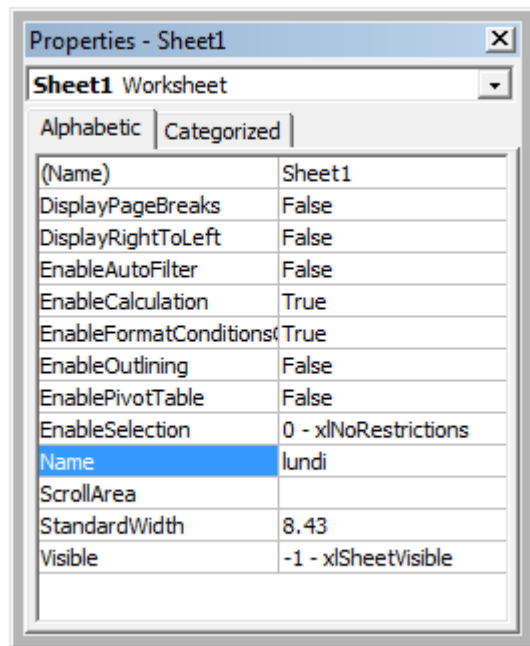
のこのワークシートのがざんされています。ただし、ワークシートの.CodeNameプロパティをしていた、サブプロシーヤはききです

```
with Sheet1
    'operation code here; for example:
    .Range(.Cells(2, "A"), .Cells(.Rows.Count, "A").End(xlUp)) = 1
end with
```

のは、VBA Projectウィンドウ[Ctrl] + Rをしています.CodeName、.Nameのワークシートがリストされています。されるはされません。の.Indexは、ワークシート・ウィンドウのタブ・キューにされるでされます。



.CodeNameのをすることはいいことではありませんが、ではありません。 VBEの[プロパティ]ウィンドウ[F4]をください。



ワークシート。コードは、のにあります。ワークシートの.Nameは1/10です。どちらもです。

のわりにデリミタをしたの

VBAでダイナミックアレイをすると、になデータセットにして、にえめでが加かることがあります。なデータをStrings、Numbers、Booleansなどにする、 Split() をしてVBAでダイナミックになReDim Preserveステートメントをけることができます。たとえば、からのをについでにし、そのをしてListBoxのをするループをてみましょう。

```
Private Sub UserForm_Initialize()

Dim Count As Long, DataString As String, Delimiter As String

For Count = 1 To ActiveSheet.UsedRows.Count
    If ActiveSheet.Range("A" & Count).Value <> "Your Condition" Then
        RowString = RowString & Delimiter & ActiveSheet.Range("A" & Count).Value
        Delimiter = "><" 'By setting the delimiter here in the loop, you prevent an extra
occurrence of the delimiter within the string
    End If
Next Count

ListBox1.List = Split(DataString, Delimiter)

End Sub
```

Delimiterはのにできますが、セットでにしないをすることはです。えは、あなたはのをしていたとしましょう。そのは、をしてください、 -、 /またはりとしてはではありません。これらのいづれかをするようにをできるため、のデータポイントがされます。

このメソッドのにはがありますつまりの。にきなデータセットのはがです。これはずしもVBAでダイナミックアレイをするためのまたはもなではありませんが、なです。

Excelのダブルクリックイベント

では、Excelのには、シングルクリックダブルクリックをするのがなく、クリックをできる「OnAction」プロパティのみがまれています。しかし、コードでは、ダブルクリックでなるまたはにするがあるがあります。のサブルーチンをVBAプロジェクトにすることができます。また、OnActionルーチンとしてすると、ダブルクリックにOnActionすることができます。

```
Public Const DOUBLECLICK_WAIT As Double = 0.25 'Modify to adjust click delay
Public LastClickObj As String, LastClickTime As Date

Sub ShapeDoubleClick()

    If LastClickObj = "" Then
        LastClickObj = Application.Caller
        LastClickTime = CDBl(Timer)
    Else
        If CDBl(Timer) - LastClickTime > DOUBLECLICK_WAIT Then
            LastClickObj = Application.Caller
            LastClickTime = CDBl(Timer)
        Else
            If LastClickObj = Application.Caller Then
                'Your desired Double Click code here
                LastClickObj = ""
            Else
                LastClickObj = Application.Caller
                LastClickTime = CDBl(Timer)
            End If
        End If
    End If

End Sub
```

このルーチンは、シェイプがのクリックをにするようにします。したに2のクリックでのコードをするだけです。

ファイルをくダイアログのファイル

このサブルーチンは、ユーザーがのファイルをし、それらのファイルパスをとってファイルをし、debug.printでコンソールにするなど、かをするをにしたものです。

```
Option Explicit

Sub OpenMultipleFiles()
    Dim fd As FileDialog
    Dim fileChosen As Integer
    Dim i As Integer
    Dim basename As String
    Dim fso As Variant
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set fd = Application.FileDialog(msoFileDialogFilePicker)
    basename = fso.GetBaseName(ActiveWorkbook.Name)
    fd.InitialFileName = ActiveWorkbook.Path & " Set Default Location to the Active Workbook Path
    fd.InitialView = msoFileDialogViewList
    fd.AllowMultiSelect = True

    fileChosen = fd.Show
```

```
If fileChosen = -1 Then
    'open each of the files chosen
    For i = 1 To fd.SelectedItems.Count
        Debug.Print (fd.SelectedItems(i))
        Dim fileName As String
        ' do something with the files.
        fileName = fso.GetFileName(fd.SelectedItems(i))
        Debug.Print (fileName)
    Next i
End If

End Sub
```

オンラインでExcel VBAのヒントとテクニックをむ <https://riptutorial.com/ja/excel-vba/topic/2240/excel-vbaのヒントとテクニック>

4: Excel-VBAの

き

Excel-VBAのは、ドキュメントとのによるエラーのコーディングにもされます。これをここにします。

*はで、-32,76832,767ののき16ビットデータです。そうでないと、オーバーフローがします。、は、コードのまたはモジュールのすべてのプロシージャに10のステップでされます。

Examples

ワークシートのをにする

ワークシートのをにすると、マクロのがにされます。さらに、イベント、、ページりをにすることはです。のSubは、こののためにどのマクロでもできます。

```
Sub OptimizeVBA(isOn As Boolean)
    Application.Calculation = IIf(isOn, xlCalculationManual, xlCalculationAutomatic)
    Application.EnableEvents = Not(isOn)
    Application.ScreenUpdating = Not(isOn)
    ActiveSheet.DisplayPageBreaks = Not(isOn)
End Sub
```

のためにのコードにってください

```
Sub MyCode()

    OptimizeVBA True

    'Your code goes here

    OptimizeVBA False

End Sub
```

の

なるでもじがられますが、なるがされます。どちらがいかをべるために、のようなコードをうことができます

```
timel = Timer

For Each iCell In MyRange
    iCell = "text"
Next iCell
```

```

time2 = Timer

For i = 1 To 30
    MyRange.Cells(i) = "text"
Next i

time3 = Timer

debug.print "Proc1 time: " & cStr(time2-time1)
debug.print "Proc2 time: " & cStr(time3-time2)

```

MicroTimer

```

Private Declare PtrSafe Function getFrequency Lib "Kernel32" Alias "QueryPerformanceFrequency"
(cyFrequency As Currency) As Long
Private Declare PtrSafe Function getTickCount Lib "Kernel32" Alias "QueryPerformanceCounter"
(cyTickCount As Currency) As Long

Function MicroTimer() As Double
    Dim cyTicks1 As Currency
    Static cyFrequency As Currency

    MicroTimer = 0
    If cyFrequency = 0 Then getFrequency cyFrequency           'Get frequency
    getTickCount cyTicks1                                     'Get ticks
    If cyFrequency Then MicroTimer = cyTicks1 / cyFrequency 'Returns Seconds
End Function

```

With ブロックの

ブロックをすると、マクロをするがされます。、チャート、ワークシートなどをくわりに、のよ
うなwith-blocksをうことができます。

```

With ActiveChart
    .Parent.Width = 400
    .Parent.Height = 145
    .Parent.Top = 77.5 + 165 * step - replacer * 15
    .Parent.Left = 5
End With

```

これはこれよりいい

```

ActiveChart.Parent.Width = 400
ActiveChart.Parent.Height = 145
ActiveChart.Parent.Top = 77.5 + 165 * step - replacer * 15
ActiveChart.Parent.Left = 5

```

ノート

- Withブロックがされると、オブジェクトはできません。その、1つのWithをしてのなるオブ
ジェクトにをえることはできません

- **With** ブロックののにジャンプしないでください。 **With** ブロックのステートメントがされ、**With** ステートメントまたは **End With** ステートメントがされない、プロシージャーをするまで、オブジェクトへのをむはメモリーにります
- に **With** キャッシュされたオブジェクトがイテレータとしてされている、**With** を **With** ループしないでください。
- **With** をのブロックにすることで、**With** をネストすることができます。ただし、**Out With** ブロックのメンバーは **With With** ブロックでマスクされるため、**In With** ブロックのオブジェクトを **Out With** ブロックのオブジェクトののメンバーにするがあります。

れの

このでは、**With** ステートメントをして、のオブジェクトにしてのステートメントをします。オブジェクトとそのプロパティは、でのみされるです。

```
With MyObject
    .Height = 100                'Same as MyObject.Height = 100.
    .Caption = "Hello World"    'Same as MyObject.Caption = "Hello World".
    With .Font
        .Color = Red            'Same as MyObject.Font.Color = Red.
        .Bold = True            'Same as MyObject.Font.Bold = True.
        MyObject.Height = 200   'Inner-most With refers to MyObject.Font (must be qualified)
    End With
End With
```

MSDNの

の - パフォーマンス

- をするのがく、にセルをループしたり、を1つずつするとき
- のとして、オートフィルタをしてするをにするがあります
- をコピーしてしいワークシートにりけます
- のシートをにりく
- このメソッドをすると、するがいほどくなります

```
Option Explicit

'Deleted rows: 775,153, Total Rows: 1,000,009, Duration: 1.87 sec

Public Sub DeleteRows()
    Dim oldWs As Worksheet, newWs As Worksheet, wsName As String, ur As Range

    Set oldWs = ThisWorkbook.ActiveSheet
    wsName = oldWs.Name
    Set ur = oldWs.Range("F2", oldWs.Cells(oldWs.Rows.Count, "F").End(xlUp))
```



```

Application.ScreenUpdating = False
Set newWs = Sheets.Add(After:=oldWs)      'Create a new WorkSheet

With ur      'Copy visible range after Autofilter (modify Criterial and 2 accordingly)
    .AutoFilter Field:=1, Criterial:="<>0", Operator:=xlAnd, Criteria2:="<>"
    oldWs.UsedRange.Copy
End With
'Paste all visible data into the new WorkSheet (values and formats)
With newWs.Range(oldWs.UsedRange.Cells(1).Address)
    .PasteSpecial xlPasteColumnWidths
    .PasteSpecial xlPasteAll
    newWs.Cells(1, 1).Select: newWs.Cells(1, 1).Copy
End With

With Application
    .CutCopyMode = False
    .DisplayAlerts = False
    oldWs.Delete
    .DisplayAlerts = True
    .ScreenUpdating = True
End With
newWs.Name = wsName
End Sub

```

すべてのExcelをにするマクロをするに

のでは、WorkbookおよびWorksheetレベルですべてのExcelをににします

- FastWBは、OnまたはOffフラグをけけるトグルです
- FastWSは、オプションのワークシートオブジェクトをけれるか、またはもけけません
- wsパラメータがされていないは、コレクションのすべてのワークシートのすべてのをオンまたはオフにします
 - カスタムタイプをしてすべてのをキャプチャしてからオフにすることができます
 - プロセスのに、をすることができます

```

Public Sub FastWB(Optional ByVal opt As Boolean = True)
    With Application
        .Calculation = IIf(opt, xlCalculationManual, xlCalculationAutomatic)
        If .DisplayAlerts <> Not opt Then .DisplayAlerts = Not opt
        If .DisplayStatusBar <> Not opt Then .DisplayStatusBar = Not opt
        If .EnableAnimations <> Not opt Then .EnableAnimations = Not opt
        If .EnableEvents <> Not opt Then .EnableEvents = Not opt
        If .ScreenUpdating <> Not opt Then .ScreenUpdating = Not opt
    End With
    FastWS , opt
End Sub

```

```

Public Sub FastWS(Optional ByVal ws As Worksheet, Optional ByVal opt As Boolean = True)
    If ws Is Nothing Then
        For Each ws In Application.ThisWorkbook.Sheets
            OptimiseWS ws, opt
        Next ws
    End If
End Sub

```

```

        Next
    Else
        OptimiseWS ws, opt
    End If
End Sub
Private Sub OptimiseWS(ByVal ws As Worksheet, ByVal opt As Boolean)
    With ws
        .DisplayPageBreaks = False
        .EnableCalculation = Not opt
        .EnableFormatConditionsCalculation = Not opt
        .EnablePivotTable = Not opt
    End With
End Sub

```

すべてのExcelをデフォルトにす

```

Public Sub XlResetSettings()    'default Excel settings
    With Application
        .Calculation = xlCalculationAutomatic
        .DisplayAlerts = True
        .DisplayStatusBar = True
        .EnableAnimations = False
        .EnableEvents = True
        .ScreenUpdating = True
        Dim sh As Worksheet
        For Each sh In Application.ThisWorkbook.Sheets
            With sh
                .DisplayPageBreaks = False
                .EnableCalculation = True
                .EnableFormatConditionsCalculation = True
                .EnablePivotTable = True
            End With
        Next
    End With
End Sub

```

デバッグによるエラーの

をして...エラーのにする「Errをることの」

どのラインがエラーをさせるかをすることは、デバッグのなであり、のをめる。されたエラーをなでするには、モジュールとプロシージャーのとともに、エラーをさせるがあります。のでは、これらのデータをログファイルにしています。

バックグラウンド

エラーオブジェクトはエラーErr.NumberとエラーErr.Descriptionをしますが、エラーのをにはいません。**Err**のは、しかし、ありませんが、あなたは、コードに、のいくつかののBTW 1に*をすることをしてに。

エラーがまったくない、Errは0をします。がなは、プロシージャーののをします。

```
Option Explicit
```

```

Public Sub MyProc1()
Dim i As Integer
Dim j As Integer
On Error GoTo LogErr
10      j = 1 / 0      ' raises an error
okay:
Debug.Print "i=" & i
Exit Sub

LogErr:
MsgBox LogErrors("MyModule", "MyProc1", Err), vbExclamation, "Error " & Err.Number
Stop
Resume Next
End Sub

Public Function LogErrors( _
    ByVal sModule As String, _
    ByVal sProc As String, _
    Err As ErrObject) As String
' Purpose: write error number, description and Erl to log file and return error text
Dim sLogFile As String: sLogFile = ThisWorkbook.Path & Application.PathSeparator &
"LogErrors.txt"
Dim sLogTxt As String
Dim lFile As Long

' Create error text
sLogTxt = sModule & "|" & sProc & "|Erl " & Erl & "|Err " & Err.Number & "|" &
Err.Description

On Error Resume Next
lFile = FreeFile

Open sLogFile For Append As lFile
Print #lFile, Format$(Now(), "yy.mm.dd hh:mm:ss "); sLogTxt
Print #lFile,
Close lFile
' Return error text
LogErrors = sLogTxt
End Function

```

・ログファイルをするためのコード

```

Sub ShowLogFile()
Dim sLogFile As String: sLogFile = ThisWorkbook.Path & Application.PathSeparator &
"LogErrors.txt"

On Error GoTo LogErr
Shell "notepad.exe " & sLogFile, vbNormalFocus

okay:
On Error Resume Next
Exit Sub

LogErr:
MsgBox LogErrors("MyModule", "ShowLogFile", Err), vbExclamation, "Error No " & Err.Number
Resume okay
End Sub

```

オンラインでExcel-VBAのをむ <https://riptutorial.com/ja/excel-vba/topic/9798/excel-vbaの>

5: Sheetオブジェクトではなく、Worksheetオブジェクトをする

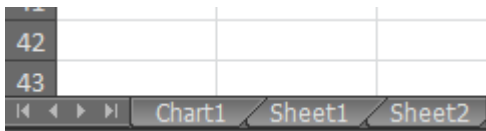
き

多くのVBAユーザーは、WorksheetsとSheetsオブジェクトのをしています。ではない。

シートオブジェクトは、ワークシートとグラフのでされています。したがって、Excelワークブックにチャートがあるは、 SheetsとWorksheetsをとしてしないようにするがあります。

Examples

のオブジェクトのをする



```
Option Explicit

Sub CheckWorksheetsDiagram()

    Debug.Print Worksheets(1).Name
    Debug.Print Charts(1).Name
    Debug.Print Sheets(1).Name

End Sub
```

```
Sheet1
Chart1
Chart1
```

オンラインでSheetオブジェクトではなく、Worksheetオブジェクトをするをむ

<https://riptutorial.com/ja/excel-vba/topic/9996/sheetオブジェクトではなく-worksheetオブジェクトをする>

6: VBAでのExcelテーブルの

き

このトピックでは、VBAのテーブルのについてし、Excelテーブルのをとしています。VBA、またはExcelオブジェクトモデルでは、テーブルはListObjectsとしてられています。ListObjectのにもされるプロパティは、ListRows、ListColumns、DataBodyRange、RangeおよびHeaderRowRangeです。

Examples

ListObjectのインスタンス

```
Dim lo as ListObject
Dim MyRange as Range

Set lo = Sheet1.ListObjects(1)

'or

Set lo = Sheet1.ListObjects("Table1")

'or

Set lo = MyRange.ListObject
```

ListRows / ListColumnsの

```
Dim lo as ListObject
Dim lr as ListRow
Dim lc as ListColumn

Set lr = lo.ListRows.Add
Set lr = lo.ListRows(5)

For Each lr in lo.ListRows
    lr.Range.ClearContents
    lr.Range(1, lo.ListColumns("Some Column").Index).Value = 8
Next

Set lc = lo.ListColumns.Add
Set lc = lo.ListColumns(4)
Set lc = lo.ListColumns("Header 3")

For Each lc in lo.ListColumns
    lc.DataBodyRange.ClearContents 'DataBodyRange excludes the header row
    lc.Range(1,1).Value = "New Header Name" 'Range includes the header row
Next
```

Excelテーブルをのにする

```
Dim lo as ListObject  
  
Set lo = Sheet1.ListObjects("Table1")  
lo.Unlist
```

オンラインでVBAでのExcelテーブルのをむ <https://riptutorial.com/ja/excel-vba/topic/9753/vbaでのexcelテーブルの>

7: VBAによるPowerPointの

このセクションでは、VBAをしてPowerPointとするさまざまなをします。スライドのデータのからチャートのまで、PowerPointはExcelとみわけてするとなツールです。したがって、このセクションでは、VBAをしてこのをするさまざまなをします。

Examples

VBAからPowerPointをする

なパラメータとのにじてできるバリエーションがありますが、このではPowerPointをするためのなフレームワークをしています。

このコードでは、PowerPointのがアクティブなVBAプロジェクトにされているがあります。をにするについては、[References Documentation](#) エントリをしてください。

まず、アプリケーション、プレゼンテーション、およびスライドオブジェクトのをします。これはレイトバインドでうことができますが、なにはバインディングをすることがにベストです。

```
Dim PPApp As PowerPoint.Application
Dim PPTPres As PowerPoint.Presentation
Dim PPSlide As PowerPoint.Slide
```

に、PowerPointアプリケーションのしいインスタンスをくかします。ここで `On Error Resume Next` コールは、PowerPointがまだかれていないに `GetObject` によってスローされるエラーをするためにされます。なについては、ベストプラクティストピックの[エラーの](#)をしてください。

```
'Open PPT if not running, otherwise select active instance
On Error Resume Next
Set PPApp = GetObject(, "PowerPoint.Application")
On Error GoTo ErrHandler
If PPApp Is Nothing Then
    'Open PowerPoint
    Set PPApp = CreateObject("PowerPoint.Application")
    PPApp.Visible = True
End If
```

アプリケーションがされると、しいプレゼンテーションとそれにくスライドがのためにされます。

```
'Generate new Presentation and slide for graphic creation
Set PPTPres = PPApp.Presentations.Add
Set PPSlide = PPTPres.Slides.Add(1, ppLayoutBlank)

'Here, the slide type is set to the 4:3 shape with slide numbers enabled and the window
'maximized on the screen. These properties can, of course, be altered as needed

PPApp.ActiveWindow.ViewType = ppViewSlide
```



```
PPPres.PageSetup.SlideOrientation = msoOrientationHorizontal  
PPPres.PageSetup.SlideSize = ppSlideSizeOnScreen  
PPPres.SlideMaster.HeadersFooters.SlideNumber.Visible = msoTrue  
PPApp.ActiveWindow.WindowState = ppWindowMaximized
```

このコードがすると、のスライドをむしいPowerPointウィンドウがきます。オブジェクトをすることで、にじて、テキスト、グラフィックス、およびExcelのをできます

オンラインでVBAによるPowerPointのをむ <https://riptutorial.com/ja/excel-vba/topic/2327/vbaによるpowerpointの>

8: VBAのセキュリティ

Examples

パスワードであなたのVBAをする

によっては、ユーザーにアクセスさせたくないVBAのパスワードなどがあります。VBAプロジェクトをパスワードですること、こののなセキュリティをできます。

のをします

1. Visual Basic Editorを Alt + F11
2. [ツール] -> [VBAProjectのプロパティ]にします。
3. []タブにします。
4. 「するプロジェクトをロックする」チェックボックスをオフにします。
5. [パスワード]および[パスワードの]テキストボックスにパスワードをします

これで、Officeアプリケーションでコードにアクセスしたいは、まずパスワードをするがあります。しかし、なVBAプロジェクトのパスワードでさえ、れにくいことにしてください。

オンラインでVBAのセキュリティをむ <https://riptutorial.com/ja/excel-vba/topic/7642/vbaのセキュリティ>

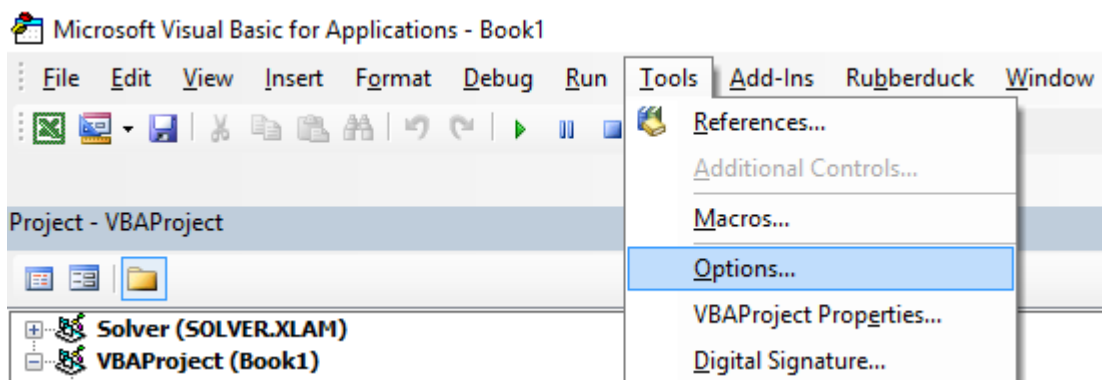
9: VBA ベストプラクティス

私たちはそれを行っています、これらのVBAでプログラムをめるにとってははるかにらかではありません。

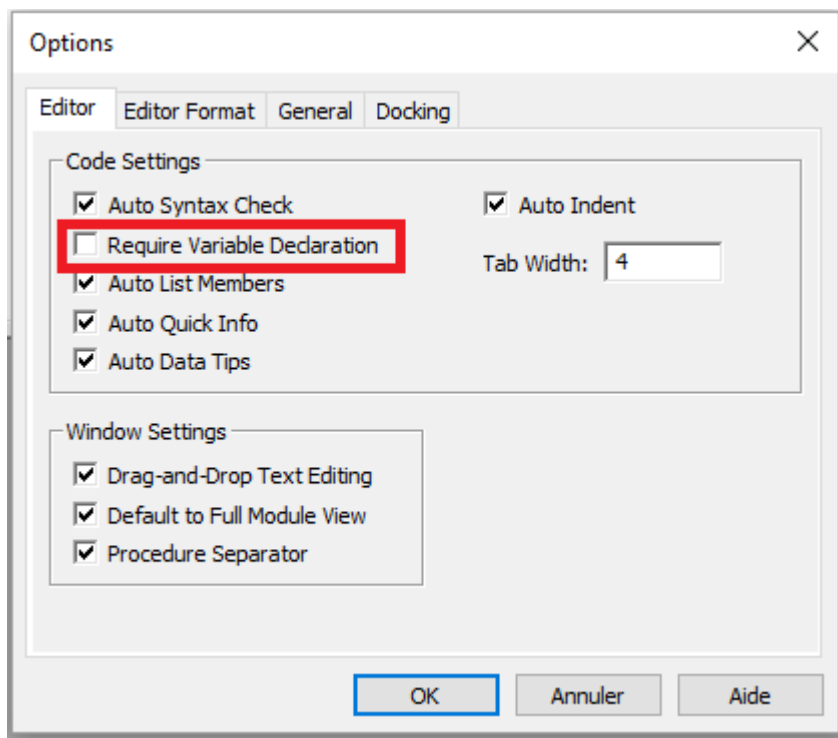
Examples

に "Option Explicit" をしてください。

VBAエディタウィンドウで、[ツール]メニューから[オプション]をします。



に、[エディタ]タブで、[か]チェックボックスがオンになっていることをします。



このオプションをすると、Option ExplicitがすべてのVBAモジュールののににされます。

さなメモこれはこれまでにかれていないモジュール、クラスモジュールなどにてはま

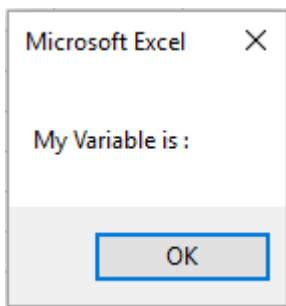
ります。たとえば、オプション "Require Variable Declaration"をにするにSheet1のコードをた、 Option Explicitはされません

Option Explicitでは、すべてのをにするがあります Dimなど。 Option Explicitにしないと、できないはVBAコンパイラによってVariantのしいとみなされ、にするにしいバグがします。 Option Explicitにすると、できないがあればコンパイルエラーがスローされ、のあるがされます。

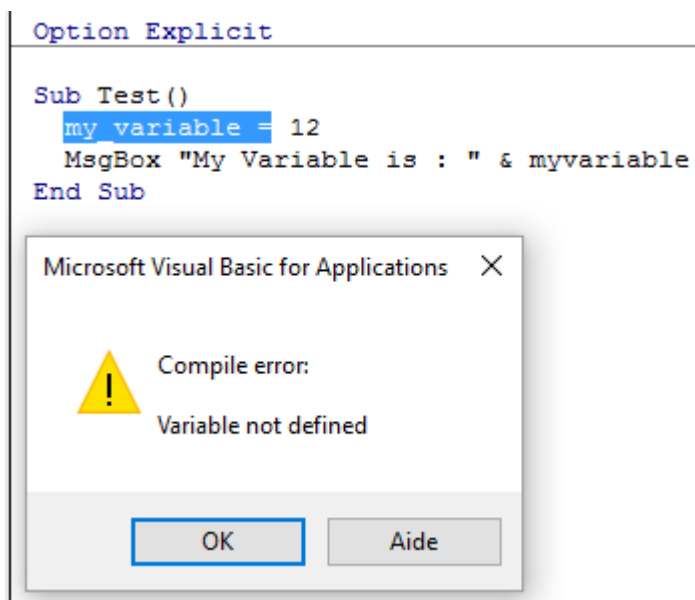
のコードをする

```
Sub Test()  
    my_variable = 12  
    MsgBox "My Variable is : " & myvariable  
End Sub
```

のメッセージがされます。



myvariableわりにmy_variableしてエラーがした、メッセージボックスにはのがされます。 Option Explicitをすると、をすコンパイルエラーメッセージがされるため、このエラーはしません。



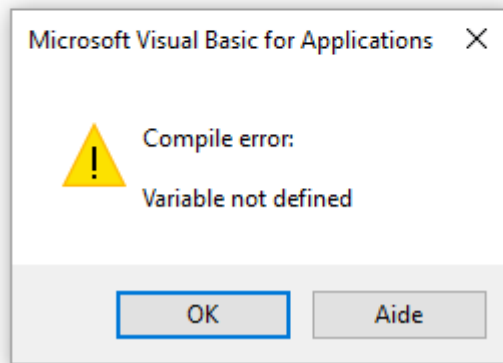
しいをすると、のようになります。

```
Sub Test()  
    Dim my_variable As Integer  
    my_variable = 12  
    MsgBox "My Variable is : " & myvariable  
End Sub
```

myvariableのエラーをにすエラーメッセージがされmyvariable。

Option Explicit

```
Sub Test()  
    Dim my_variable As Integer  
    my_variable = 12  
    MsgBox "My Variable is : " & myvariable  
End Sub
```



Option Explicit and Arrays のにする

ReDimステートメントをして、プロシージャーでにをすることができます。

- ReDimステートメントをするとき、ののスペルをえないようにしてください
- Option Explicitステートメントがモジュールにまれていても、しいがされます

```
Dim arr() as Long
```

```
ReDim ar() 'creates new array "ar" - "ReDim ar()" acts like "Dim ar()"
```

ではなくです

Office ブログ - Excel VBA パフォーマンスコーディングのベストプラクティス

なりRangeのをけることによって、のパフォーマンスがられることがよくあります。このでは、Rangeオブジェクトをにみみ、のをにしてから、をRangeします。Rangeは、Range 2しかアクセスしませんが、ループは、みり/きみのために20アクセスします。

```
Option Explicit  
Sub WorkWithArrayExample()  
  
    Dim DataRange As Variant  
    Dim Irow As Long  
    Dim Icol As Integer  
    DataRange = ActiveSheet.Range("A1:A10").Value ' read all the values at once from the Excel  
    grid, put into an array  
  
    For Irow = LBound(DataRange,1) To UBound(DataRange, 1) ' Get the number of rows.  
        For Icol = LBound(DataRange,2) To UBound(DataRange, 2) ' Get the number of columns.  
            DataRange(Irow, Icol) = DataRange(Irow, Icol) * DataRange(Irow, Icol) ' cell.value^2  
        Next Icol  
    Next Irow
```

```

Next Icol
Next Irow
ActiveSheet.Range("A1:A10").Value = DataRange ' writes all the results back to the range at
once

End Sub

```

タイムリーににするヒントやは、 [チャールズ・ウィリアムズの「なVBA UDFの」1](#)や[そのので](#)ごい
ただけます。

なは**VB**をする

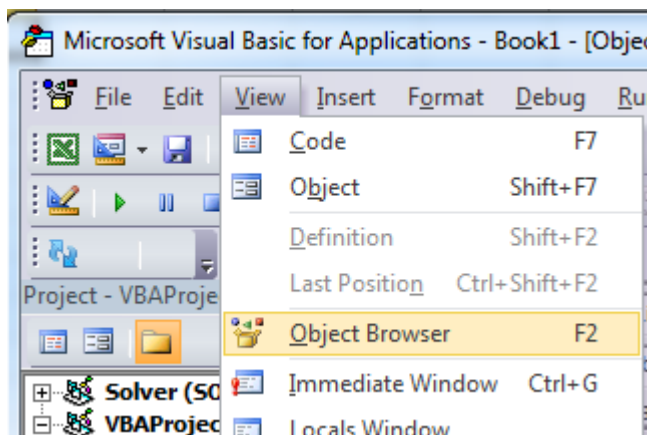
```
If MsgBox("Click OK") = vbOK Then
```

のわりにすることができます

```
If MsgBox("Click OK") = 1 Then
```

みやすさをさせるためです。

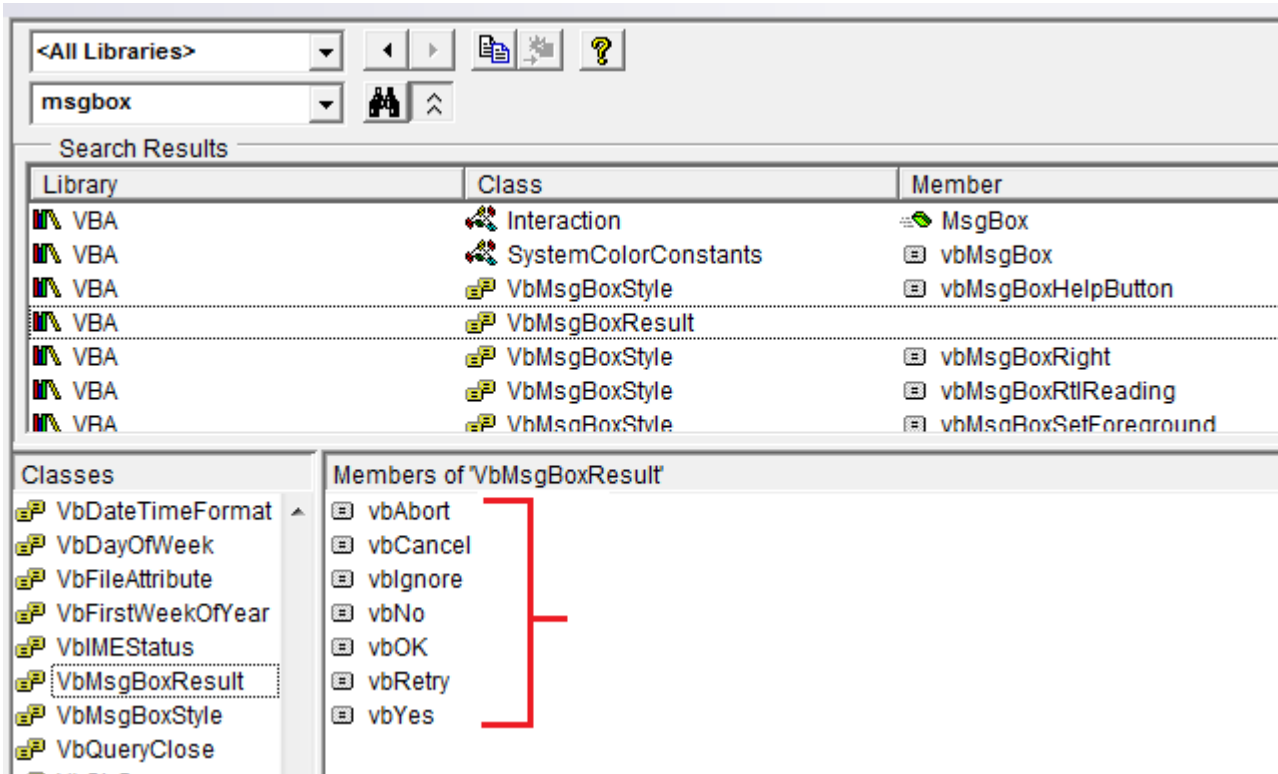
オブジェクトブラウザをして、なVBをします。 ビュー→オブジェクトブラウザまたはVBエディ
タからのF2



するクラスを



なメンバーを



なをする

コードのなとは、コメントをにするのにちます

```
Dim ductWidth As Double
Dim ductHeight As Double
Dim ductArea As Double

ductArea = ductWidth * ductHeight
```

よりもい

```
Dim a, w, h

a = w * h
```

これは、セル、、ワークシート、またはワークブックのいずれであっても、あるからのにデータをコピーするににちます。のようなをしてをけてください

```
Dim myWB As Workbook
Dim srcWS As Worksheet
Dim destWS As Worksheet
Dim srcData As Range
Dim destData As Range

Set myWB = ActiveWorkbook
Set srcWS = myWB.Sheets("Sheet1")
Set destWS = myWB.Sheets("Sheet2")
Set srcData = srcWS.Range("A1:A10")
Set destData = destWS.Range("B11:B20")
destData = srcData
```

1つののにのをするは、ののようにのをしてください。

```
Dim ductWidth As Double, ductHeight As Double, ductArea As Double
```

はののみをし、のはVariantままです。

```
Dim ductWidth, ductHeight, ductArea As Double
```

エラー

なエラーにより、エンドユーザーはVBAランタイムエラーをすることができなくなり、はエラーをにしてすることができます。

VBAでは3つのなエラーがありますが、コードでにとされているをき、2つはプログラムではけるべきです。

```
On Error GoTo 0 'Avoid using
```

または

```
On Error Resume Next 'Avoid using
```

を

```
On Error GoTo <line> 'Prefer using
```

エラーGoTo 0

コードにエラーがされていない、`On Error GoTo 0`はデフォルトのエラーハンドラです。このモードでは、ランタイムエラーがしてなVBAエラーメッセージがされ、コードをするか、ソースをするdebugモードにすることができます。コードをくには、このがもですが、エンドユーザーにされるコードのはけるべきです。エンドユーザーがしにくく、しいものです。

エラーのにへ

`On Error Resume Next`では、エラーハンドラがされるまで、エラーコールのすべてののについて、にスローされるエラーをVBAがします。になでは、このはですが、これらのはけるべきです。たとえば、Excelマクロからのプログラムをする、`On Error Resume Next`びしは、プログラムがにいていのかどうかはなにです。

```
'In this example, we open an instance of Powerpoint using the On Error Resume Next call
```



```

Dim PPAApp As PowerPoint.Application
Dim PPPres As PowerPoint.Presentation
Dim PPSlide As PowerPoint.Slide

'Open PPT if not running, otherwise select active instance
On Error Resume Next
Set PPAApp = GetObject(, "PowerPoint.Application")
On Error GoTo ErrHandler
If PPAApp Is Nothing Then
    'Open PowerPoint
    Set PPAApp = CreateObject("PowerPoint.Application")
    PPAApp.Visible = True
End If

```

On Error Resume Next びしをせず、Powerpointアプリケーションがまだいていない、GetObject メソッドはエラーをスローします。したがって、On Error Resume Next は、アプリケーションの2つのインスタンスをしないようにするためにでした。

また、エラーハンドラをすぐにリセットするには、On Error Resume Next コールがになったらすぐにリセットすることをおめします

エラーに GoTo <line>

このエラーは、のユーザーにされているすべてのコードにしておめします。これにより、プログラマは、されたにコードをすることにより、VBAのエラーをにすることができます。タグはのをむでめられ、するのにコロンがくコードにされます。On Error GoTo <line>なるびしをうことで、のエラーブロックをできます。のサブルーチンは、On Error GoTo <line>びしのをしています。

Exit Sub ラインは、のエラーハンドラのにされ、のエラーハンドラのかれて、エラーがしてもコードがにブロックにまないようにするがあります。したがって、コードブロックのにエラーハンドラをすることは、にとってはベストプラクティスです。

```

Sub YourMethodName()
    On Error GoTo errorHandler
    ' Insert code here
    On Error GoTo secondErrorHandler

    Exit Sub 'The exit sub line is essential, as the code will otherwise
            'continue running into the error handling block, likely causing an error

errorHandler:
    MsgBox "Error " & Err.Number & ": " & Err.Description & " in " & _
        VBE.ActiveCodePane.CodeModule, vbOKOnly, "Error"
    Exit Sub

secondErrorHandler:
    If Err.Number = 424 Then 'Object not found error (purely for illustration)
        Application.ScreenUpdating = True
        Application.EnableEvents = True
    Exit Sub
Else

```

```

        MsgBox "Error " & Err.Number & ": " & Err.Description
        Application.ScreenUpdating = True
        Application.EnableEvents = True
        Exit Sub
    End If
    Exit Sub

End Sub

```

エラーコードをしてメソッドをするは、をクリーンアップしてください。

- にしたものをにす
- ファイルをじる
- のリセットをリセットする
- モードをリセットする
- イベントをリセットする
- マウスポインタをリセットする
- End Sub にくオブジェクトのインスタンスにしてアンロードメソッドをびす
- ステータスバーをリセットする

あなたのをする

にワークロードをコーディングするは、ですのためにをすることをおめします。いコメントは、コードがをしているのかをし、コードがをしているのかをしてはいけません。

```

Function Bonus(EmployeeTitle as String) as Double
    If EmployeeTitle = "Sales" Then
        Bonus = 0      'Sales representatives receive commission instead of a bonus
    Else
        Bonus = .10
    End If
End Function

```

あなたのコードがそれがをしているのかをするためにコメントがなほどあいまいであるならば、コメントをしてそれをするのではなく、もっとにきえることをしてください。たとえば、のわりに

```

Sub CopySalesNumbers
    Dim IncludeWeekends as Boolean

    'Boolean values can be evaluated as an integer, -1 for True, 0 for False.
    'This is used here to adjust the range from 5 to 7 rows if including weekends.
    Range("A1:A" & 5 - (IncludeWeekends * 2)).Copy
    Range("B1").PasteSpecial
End Sub

```

のように、コードをにする。

```

Sub CopySalesNumbers
    Dim IncludeWeekends as Boolean
    Dim DaysinWeek as Integer

```

```

If IncludeWeekends Then
    DaysinWeek = 7
Else
    DaysinWeek = 5
End If
Range("A1:A" & DaysinWeek).Copy
Range("B1").PasteSpecial
End Sub

```

マクロにプロパティをオフにする

のをけるために、どのプログラミングでもベストプラクティスです。ただし、テストでコードのがすぎるとしたは、にアプリケーションのプロパティのをオフにすることがあります。このコードをモジュールにする

```

Public Sub SpeedUp( _
    SpeedUpOn As Boolean, _
    Optional xlCalc as XlCalculation = xlCalculationAutomatic _
)
    With Application
        If SpeedUpOn Then
            .ScreenUpdating = False
            .Calculation = xlCalculationManual
            .EnableEvents = False
            .DisplayStatusBar = False 'in case you are not showing any messages
            ActiveSheet.DisplayPageBreaks = False 'note this is a sheet-level setting
        Else
            .ScreenUpdating = True
            .Calculation = xlCalc
            .EnableEvents = True
            .DisplayStatusBar = True
            ActiveSheet.DisplayPageBreaks = True
        End If
    End With
End Sub

```

Office ブログの- Excel VBAパフォーマンスコーディングのベストプラクティス

そして、それをマクロのめとわりにびします

```

Public Sub SomeMacro
    'store the initial "calculation" state
    Dim xlCalc As XlCalculation
    xlCalc = Application.Calculation

    SpeedUp True

    'code here ...

    'by giving the second argument the initial "calculation" state is restored
    'otherwise it is set to 'xlCalculationAutomatic'
    SpeedUp False, xlCalc
End Sub

```

これらはにのPublic Sub プロシージャの「」となすことができますが、Application.EnableEvents = False イベントをにすることは、Worksheet_Change をするWorksheet_Change および Workbook_SheetChange プライベートイベントマクロにとってとなすがあります。イベントトリガをにしないと、がされたときにイベントマクロがにされ、""ブックになるがあります。イベントマクロをするにイベントをのにし、おそらくは「な」エラーハンドラをしてください。

```
Option Explicit

Private Sub Worksheet_Change(ByVal Target As Range)
    If Not Intersect(Target, Range("A:A")) Is Nothing Then
        On Error GoTo bm_Safe_Exit
        Application.EnableEvents = False

        'code that may change a value on the worksheet goes here

    End If
bm_Safe_Exit:
    Application.EnableEvents = True
End Sub
```

これらのをにするとがするため、アプリケーションのデバッグがはるかにしくなるがあります。コードがしくしていないは、をSpeedUp TrueまでSpeedUp Trueびしをコメントアウトしてください。

xlCalculationManual がワークブックのをげているため、ワークシートのセルにきんだり、ワークシートからをみしたりするは、にです。SpeedUp をにしないでこれをするには、Application.Calculate をインクルードして、のポイントでをすることができます。

これらはApplicationのプロパティなので、マクロがするににするがあります。これにより、エラーハンドラをしたり、のつまり、EndまたはUnload Me をけることがにになります。

エラー

```
Public Sub SomeMacro()
    'store the initial "calculation" state
    Dim xlCalc As XlCalculation
    xlCalc = Application.Calculation

    On Error GoTo Handler
    SpeedUp True

    'code here ...
    i = 1 / 0
CleanExit:
    SpeedUp False, xlCalc
    Exit Sub
Handler:
    'handle error
    Resume CleanExit
End Sub
```

ExcelでActiveCellまたはActiveSheetをしないようにする

らかのでコードがったでされた、 `ActiveCell` または `ActiveSheet` をするといのになることがあります。

```
ActiveCell.Value = "Hello"
'will place "Hello" in the cell that is currently selected
Cells(1, 1).Value = "Hello"
'will always place "Hello" in A1 of the currently selected sheet

ActiveSheet.Cells(1, 1).Value = "Hello"
'will place "Hello" in A1 of the currently selected sheet
Sheets("MySheetName").Cells(1, 1).Value = "Hello"
'will always place "Hello" in A1 of the sheet named "MySheetName"
```

- `Active*`すると、ユーザーがきてのワークシートをクリックしたり、のブックをいたりすると、されるマクロでがするがあります。
- コードがいたり、のブックをししたりすると、がするがあります。
- あなたのコードが `Sheets("MyOtherSheet").Select` している、がするがあります
`Sheets("MyOtherSheet").Select`、みきをするにどのシートにいるかをれてしまいました。

ワークシートをしてわない

すべてののがのワークシートにされているでも、コードにワークシートをにすることは、にいです。このにより、でコードをししたり、 `Sub` や `Function` またはすべてをののですするほうがずっとになります。くのは、コードのワークシートにじローカルをするというをし、そのコードのをさらににします。

として、のコードはあいまいですが、します - かのワークシートをアクティブまたはしないり、

```
Option Explicit
Sub ShowTheTime()
    '--- displays the current time and date in cell A1 on the worksheet
    Cells(1, 1).Value = Now() ' don't refer to Cells without a sheet reference!
End Sub
```

`Sheet1` がアクティブなは、 `Sheet1` A1セルにののがされます。しかし、ユーザーがらかのでワークシートをした、コードはワークシートがアクティブであるものをします。ワークシートはあいまいです。

ベストプラクティスは、コードがするワークシートをにすることです。

```
Option Explicit
Sub ShowTheTime()
    '--- displays the current time and date in cell A1 on the worksheet
    Dim myWB As Workbook
    Set myWB = ThisWorkbook
    Dim timestampSH As Worksheet
    Set timestampSH = myWB.Sheets("Sheet1")
    timestampSH.Cells(1, 1).Value = Now()
End Sub
```

のコードは、ワークブックとワークシートのをにしています。のようにえるかもしれませんが、にするいをすることで、のからあなたをうことができます。

SELECTやACTIVATEのをける

コードで `Select` または `Activate` をすることはにまれますが、のExcelメソッドでは、ワークシートまたはワークブックをアクティブにしてから、どおりにさせるがあります。

VBAのをめたばかりのは、マクロレコーダーを使ってアクションをし、コードを試みることをおすすめします。たとえば、Sheet2のセルD3にをするためにされたアクションをし、マクロコードはのようになります。

```
Option Explicit
Sub Macro1()
'
' Macro1 Macro
'
'
    Sheets("Sheet2").Select
    Range("D3").Select
    ActiveCell.FormulaR1C1 = "3.1415"      '(see **note below)
    Range("D4").Select
End Sub
```

しかし、マクロレコーダーはあなたのユーザーアクションごとに1のコードをすることを覚えておいてください。これは、ワークシート・タブをクリックしてシート2 `Sheets("Sheet2").Select` を `Sheets("Sheet2").Select`、D3をするにセルD3をクリックして `Range("D3").Select` し、されているセルののセルをする `Range("D4").Select`。

.Selectにはのがあります

- ワークシートはにされているわけではありません。これは、にワークシートをりえるがないにし、さまざまなアクティブなワークシートにしてなるをもたらすことをします。
- `.Select()` はいです。 `Application.ScreenUpdating` が `False` にされていても、これはされるなです。
- `.Select()` は `.Select()` です。 `Application.ScreenUpdating` が `True` ままになっていると、Excel にはセル、ワークシート、フォームなどをします。これはにストレスがあり、るのがにです。
- `.Select()` はリスナーをトリガーします。これはもうしんでいますが、しなければ、`Worksheet_SelectionChange()` ようながトリガされます。

VBAでコーディングしているときは、「タイプする」アクションつまりSelectステートメントはすべてなくなりました。コードを1つのステートメントにらして、をセルにれることができます。

```
'--- GOOD
ActiveWorkbook.Sheets("Sheet2").Range("D3").Value = 3.1415

'--- BETTER
```

```

Dim myWB      As Workbook
Dim myWS      As Worksheet
Dim myCell    As Range

Set myWB = ThisWorkbook      '*** see NOTE2
Set myWS = myWB.Sheets("Sheet2")
Set myCell = myWS.Range("D3")

myCell.Value = 3.1415

```

のよりいではをってセルのなるをっています、GOODのはにうまくしますが、ずっとコードモジュールではいにくく、の1つがっているとデバッグがしくなります。

**マクロレコーダーは、するデータのについてくのいます。この、をすることとしてをします。あなたのコードはこれをうはなく、のようになセルにをりてることができます。

** NOTE2されるは、`ActiveWorkbook`わりにローカルワークブックを`ThisWorkbook`にすることですになをきます。なぜなら、マクロは、VBAコードがどのようなワークブックであっても、そのワークブックのにはえないようなリソースをとします。Excelでのワークブックをいている、`ActiveWorkbook`は、VBAエディタでされているワークブックとはなるフォーカスをつワークブックです。つまり、にのワークブックをしているときに1つのワークブックでしているといします。`ThisWorkbook`は、されているコードをむブックをします。

すべてのワークブックとシートへのをにおよびする

のいているワークブックでするは、それぞれにのシートがあるは、すべてのワークブックとシートをしてするのがもです。

`ActiveWorkbook`または`ActiveSheet`は、ユーザーによってされるがあるため、しないでください。

のコードは、「Data.xlsx」ワークブックの「RAW_DATA」シートから「Results.xlsx」ワークブックの「Refined_Data」シートのをコピーするをします。

また、`Select`メソッドをせずにコピーしてりけるもしています。

```

Option Explicit

Sub CopyRanges_BetweenShts()

    Dim wbSrc      As Workbook
    Dim wbDest     As Workbook
    Dim shtCopy    As Worksheet
    Dim shtPaste   As Worksheet

    ' set reference to all workbooks by name, don't rely on ActiveWorkbook
    Set wbSrc = Workbooks("Data.xlsx")
    Set wbDest = Workbooks("Results.xlsx")

    ' set reference to all sheets by name, don't rely on ActiveSheet
    Set shtCopy = wbSrc.Sheet1 '// "Raw_Data" sheet
    Set shtPaste = wbDest.Sheet2 '// "Refined_Data" sheet

```

```
' copy range from "Data" workbook to "Results" workbook without using Select
shtCopy.Range("A1:C10").Copy _
Destination:=shtPaste.Range("A1")
```

```
End Sub
```

WorksheetFunction オブジェクトは、の **UDF** よりもにされます。

VBAはにコンパイルされます。これはパフォーマンスにきなをえます。みみのすべてがになり、それらをしようとします。

として、はSUMとCOUNTIFをしています、WorkSheetFunctionsでできるものなら、ifをうことができます。

それらののみは、をループし、セルごとにセルをすることですをして。

```
Sub UseRange()
    Dim rng as Range
    Dim Total As Double
    Dim CountLessThan01 As Long

    Total = 0
    CountLessThan01 = 0
    For Each rng in Sheets(1).Range("A1:A100")
        Total = Total + rng.Value2
        If rng.Value < 0.1 Then
            CountLessThan01 = CountLessThan01 + 1
        End If
    Next rng
    Debug.Print Total & ", " & CountLessThan01
End Sub
```

1つののは、をにし、のようにするこです。

```
Sub UseArray()
    Dim DataToSummarize As Variant
    Dim i As Long
    Dim Total As Double
    Dim CountLessThan01 As Long

    DataToSummarize = Sheets(1).Range("A1:A100").Value2 'faster than .Value
    Total = 0
    CountLessThan01 = 0
    For i = 1 To 100
        Total = Total + DataToSummarize(i, 1)
        If DataToSummarize(i, 1) < 0.1 Then
            CountLessThan01 = CountLessThan01 + 1
        End If
    Next i
    Debug.Print Total & ", " & CountLessThan01
End Sub
```

しかし、ループをするのではなく、なをするのにになApplication.Worksheetfunctionをうことがで

きます

```
Sub UseWorksheetFunction()  
    Dim Total As Double  
    Dim CountLessThan01 As Long  
  
    With Application.WorksheetFunction  
        Total = .Sum(Sheets(1).Range("A1:A100"))  
        CountLessThan01 = .CountIf(Sheets(1).Range("A1:A100"), "<0.1")  
    End With  
  
    Debug.Print Total & ", " & CountLessThan01  
End Sub
```

また、なをうでも、 `Application.Evaluate` することもでき `Application.Evaluate`。

```
Sub UseEvaluate()  
    Dim Total As Double  
    Dim CountLessThan01 As Long  
  
    With Application  
        Total = .Evaluate("SUM(" & Sheet1.Range("A1:A100").Address( _  
            external:=True) & ")")  
        CountLessThan01 = .Evaluate("COUNTIF('Sheet1'!A1:A100, "<0.1")")  
    End With  
  
    Debug.Print Total & ", " & CountLessThan01  
End Sub
```

そして、に、25,000のSubsをえてっているここでは、5のテストがミリでされますもちろん、PCでなるでしょうが、おいにべてにします。

1. UseWorksheetFunction2156ミリ
2. UseArray2219 ms+ 3
3. UseEvaluate4693 ms+ 118
4. UseRange6530 ms+ 203

プロパティまたはメソッドのをとしてしないでください

プロパティまたはメソッドのされたを、のプロシージャおよびのとしてすることは、に「ベストプラクティス」とはみなされません。

いフォーム - はにえばなコードですが、 `Find` メソッドと `Row`、 `Column`、 `Address` プロパティのはのあいまいさにをきこすがあり、にします。

```
Option Explicit  
  
Sub find()  
    Dim row As Long, column As Long  
    Dim find As String, address As Range  
  
    find = "something"
```

```

With ThisWorkbook.Worksheets("Sheet1").Cells
    Set address = .SpecialCells(xlCellTypeLastCell)
    row = .find(what:=find, after:=address).row      '< note .row not capitalized
    column = .find(what:=find, after:=address).column '< note .column not capitalized

    Debug.Print "The first 'something' is in " & .Cells(row, column).address(0, 0)
End With
End Sub

```

グッドフォーム - すべてののが、オリジナルにいがユニークなにされた、なのはされています。

```

Option Explicit

Sub myFind()
    Dim rw As Long, col As Long
    Dim wht As String, lastCell As Range

    wht = "something"

    With ThisWorkbook.Worksheets("Sheet1").Cells
        Set lastCell = .SpecialCells(xlCellTypeLastCell)
        rw = .Find(What:=wht, After:=lastCell).Row      '◀ note .Find and .Row
        col = .Find(What:=wht, After:=lastCell).Column  '◀ .Find and .Column

        Debug.Print "The first 'something' is in " & .Cells(rw, col).Address(0, 0)
    End With
End Sub

```

のメソッドやプロパティをのににきえたいときがあるかもしれませんが、そのようなはほとんどありません。ほとんどの、ののされたをししないでください。

オンラインでVBAベストプラクティスをむ <https://riptutorial.com/ja/excel-vba/topic/1107/vbaベストプラクティス>

10: VBAをしたき

あるにして3つのきをすることはできません。のきをするには、Modifyメソッドをします。しいをするに、のをするには、Deleteメソッドをします。

Examples

FormatConditions.Add

```
FormatConditions.Add(Type, Operator, Formula1, Formula2)
```

パラメーター

	Iオプション	データ・タイプ
タイプ	XlFormatConditionType	
オペレーター	オプション	バリエント
1	オプション	バリエント
フォーミュラ2	オプション	バリエント

XlFormatConditionType enumeration

xlAboveAverageCondition	
xlBlanksCondition	ブランク
xlCellValue	セル
xlColorScale	カ ラース ケール
xlDatabar	データバー
xlErrorsCondition	エラー
xl	
XlIconSet	アイコンセット

xlNoBlanksCondition	ブランクなし
xlNoErrorsCondition	エラーなし
xlTextString	テキスト
xlTimePeriod	
xlTop10	10の
xlUniqueValues	の

セルによる

```
With Range("A1").FormatConditions.Add(xlCellValue, xlGreater, "=100")
    With .Font
        .Bold = True
        .ColorIndex = 3
    End With
End With
```

オペレーター

xlBetween
xlEqual
xlGreater
xlGreaterEqual
xlLess
xlLessEqual
xlNotBetween
xlNotEqual

TypeがxlExpressionの、Operatorはされます。

テキストによるには、

```
With Range("a1:a10").FormatConditions.Add(xlTextString, TextOperator:=xlContains,
String:="egg")
    With .Font
        .Bold = True
        .ColorIndex = 3
    End With
End With
```

オペレーター

xlBeginsWith	されたでします。
xlコンテナ	されたをみます。
xlDoesNotContain	されたをんでいません。
xlEndsWith	されたで



```
With Range("a1:a10").FormatConditions.Add(xlTimePeriod, DateOperator:=xlToday)
    With .Font
        .Bold = True
        .ColorIndex = 3
    End With
End With
```

オペレーター

xl
xlLast7Days
xlLastWeek
xlThisWeek
xlNextWeek
xlLastMonth
xlThisMonth
xlNextMonth

きフォーマットをする

のすべてのきフォーマットをする

```
Range("A1:A10").FormatConditions.Delete
```

ワークシートのすべてのきフォーマットをします。

```
Cells.FormatConditions.Delete
```

FormatConditions.AddUniqueValues

したをする

```
With Range("E1:E100").FormatConditions.AddUniqueValues
    .DupeUnique = xlDuplicate
    With .Font
        .Bold = True
        .ColorIndex = 3
    End With
End With
```

ユニークなの

```
With Range("E1:E100").FormatConditions.AddUniqueValues
    With .Font
        .Bold = True
        .ColorIndex = 3
    End With
End With
```

FormatConditions.AddTop10

5つのをする

```
With Range("E1:E100").FormatConditions.AddTop10
    .TopBottom = xlTop10Top
    .Rank = 5
    .Percent = False
    With .Font
        .Bold = True
        .ColorIndex = 3
    End With
End With
```

End With

FormatConditions.AddAboveAverage

```
With Range("E1:E100").FormatConditions.AddAboveAverage
    .AboveBelow = xlAboveAverage
    With .Font
        .Bold = True
        .ColorIndex = 3
    End With
End With
```

オペレーター

XIAboveAverage	
XIAboveStdDev	よりきい
XIBelowAverage	の
XIBelowStdDev	
XIEqualAboveAverage	
XIEqualBelowAverage	

FormatConditions.AddIconSetCondition

	A
1	↓ 13
2	→ 22
3	→ 33
4	→ 30
5	→ 23
6	↑ 40
7	↑ 50
8	↓ 4
9	→ 20
10	↓ 13
11	↓ 5
12	↑ 45
13	→ 30
14	↑ 37
15	↓ 12

```
Range("a1:a10").FormatConditions.AddIconSetCondition
With Selection.FormatConditions(1)
    .ReverseOrder = False
    .ShowIconOnly = False
    .IconSet = ActiveWorkbook.IconSets(xl3Arrows)
End With
```

```

End With

With Selection.FormatConditions(1).IconCriteria(2)
    .Type = xlConditionValuePercent
    .Value = 33
    .Operator = 7
End With

With Selection.FormatConditions(1).IconCriteria(3)
    .Type = xlConditionValuePercent
    .Value = 67
    .Operator = 7
End With

```

IconSet

xl3Arrows
xl3ArrowsGray
xl3Flags
xl3
xl3stars
xl3Symbols
xl3Symbols2
xl3TrafficLights1
xl3TrafficLights2
xl3
xl4Arrows
xl4ArrowsGray
xl4CRV
xl4RedToBlack
xl4TrafficLights
xl5Arrows
xl5ArrowsGray
xl5Boxes

xl5CRV

xl5クォーターズ



タイプ

xlConditionValuePercent

xlConditionValueNumber

xlConditionValuePercentile

xlConditionValueFormula

オペレーター

xlGreater

5

xlGreaterEqual

7

きでアイコンのしきいをすかします。

オンラインでVBAをしたきをむ <https://riptutorial.com/ja/excel-vba/topic/9912/vbaをしたき>

11: アクティブブックのすべてのシートをループする

Examples

アクティブなブックのすべてのワークシートをします。

```
Option Explicit

Sub LoopAllSheets()

Dim sht As Excel.Worksheet
' declare an array of type String without committing to maximum number of members
Dim sht_Name() As String
Dim i As Integer

' get the number of worksheets in Active Workbook , and put it as the maximum number of
members in the array
ReDim sht_Name(1 To ActiveWorkbook.Worksheets.count)

i = 1

' loop through all worksheets in Active Workbook
For Each sht In ActiveWorkbook.Worksheets
    sht_Name(i) = sht.Name ' get the name of each worksheet and save it in the array
    i = i + 1
Next sht

End Sub
```

フォルダのすべてのファイルのすべてのシートをループする

```
Sub Theloopofloops()

Dim wbk As Workbook
Dim Filename As String
Dim path As String
Dim rCell As Range
Dim rRng As Range
Dim wsO As Worksheet
Dim sheet As Worksheet

path = "pathtofile(s)" & "\"
Filename = Dir(path & "*.xl??")
Set wsO = ThisWorkbook.Sheets("Sheet1") 'included in case you need to differentiate_
between workbooks i.e currently opened workbook vs workbook containing code

Do While Len(Filename) > 0
    DoEvents
    Set wbk = Workbooks.Open(path & Filename, True, True)
    For Each sheet In ActiveWorkbook.Worksheets 'this needs to be adjusted for
```

```

specifiying sheets. Repeat loop for each sheet so thats on a per sheet basis
    Set rRng = sheet.Range("a1:a1000") 'OBV needs to be changed
    For Each rCell In rRng.Cells
        If rCell <> "" And rCell.Value <> vbNullString And rCell.Value <> 0 Then

            'code that does stuff

        End If
    Next rCell
Next sheet
wbk.Close False
Filename = Dir
Loop
End Sub

```

オンラインでアクティブブックのすべてのシートをループするをむ <https://riptutorial.com/ja/excel-vba/topic/1144/アクティブブックのすべてのシートをループする>

12: アプリケーションオブジェクト

Excel VBAには、のExcelアプリケーションののをするためにできるクラスとオブジェクトをむなオブジェクトモデルがしています。するもなオブジェクトの1つが**Application**オブジェクトです。これは、Excelののインスタンスをすキャッチオールです。のExcelワークブックにされていないものは、すべて**Application**オブジェクトにあります。

トップレベルオブジェクトとしての**Application**オブジェクトは、Excelのあらゆるをするためにできるりものプロパティ、メソッド、およびイベントをちます。

Examples

なアプリケーションオブジェクトの**Excel**ウィンドウをする

このコードは、トップレベルの**Application**オブジェクトをしてメインのExcelウィンドウをします。

```
Sub MinimizeExcel()  
  
    Application.WindowState = xlMinimized  
  
End Sub
```

なアプリケーションオブジェクトの**Excel**と**VBE**のバージョンをする

```
Sub DisplayExcelVersions()  
  
    MsgBox "The version of Excel is " & Application.Version  
    MsgBox "The version of the VBE is " & Application.VBE.Version  
  
End Sub
```

Application.Versionプロパティのは、のあるバージョンのExcelでのみコードがするようにするのにです。

オンラインでアプリケーションオブジェクトをむ <https://riptutorial.com/ja/excel-vba/topic/5645/アプリケーションオブジェクト>

13: あるのするの

き

ので、データのをすることになり、したデータをそのにするがあります。よりきなデータセットの、VBAコードまたはきのいずれかをして、さまざまなアプローチをすることができます。このでは、ネストされた2つのfor-nextループでなif-thenをして、のセルがそののセルとしいかどうかをテストします。

Examples

あるでをつける

のテストでは、するについてA2からA7までのです。これは、ソリューションへののアプローチとしてえられるソリューションをしています。よりもをうがく、コレクションややxmlメソッドを使ってをチェックすることもできます。

```
Sub find_duplicates()
' Declare variables
Dim ws As Worksheet           ' worksheet
Dim cell As Range             ' cell within worksheet range
Dim n As Integer              ' highest row number
Dim bFound As Boolean          ' boolean flag, if duplicate is found
Dim sFound As String: sFound = "|" ' found duplicates
Dim s As String                ' message string
Dim s2 As String               ' partial message string
' Set Sheet to memory
Set ws = ThisWorkbook.Sheets("Duplicates")

' loop thru FULLY QUALIFIED REFERENCE
For Each cell In ws.Range("A2:A7")
    bFound = False: s2 = "" ' start each cell with empty values
    ' Check if first occurrence of this value as duplicate to avoid further searches
    If InStr(sFound, "|" & cell & "|") = 0 Then

        For n = cell.Row + 1 To 7 ' iterate starting point to avoid REDUNDANT SEARCH
            If cell = ws.Range("A" & n).Value Then
                If cell.Row <> n Then ' only other cells, as same cell cannot be a duplicate
                    bFound = True ' boolean flag
                    ' found duplicates in cell A{n}
                    s2 = s2 & vbNewLine & " -> duplicate in A" & n
                End If
            End If
        Next
    End If
' notice all found duplicates
If bFound Then
    ' add value to list of all found duplicate values
    ' (could be easily split to an array for further analyze)
    sFound = sFound & cell & "|"
    s = s & cell.Address & " (value=" & cell & ")" & s2 & vbNewLine & vbNewLine
End If
End Sub
```

```
End If
Next
' MessageBox with final result
MsgBox "Duplicate values are " & sFound & vbNewLine & vbNewLine & s, vbInformation, "Found
duplicates"
End Sub
```

にじて、このをすることができます。たとえば、nのは、のデータをむのセルののにすることができます。True Ifののアクションをしてをすることができますのどこかの。しかし、ルーチンのみはわからないだろう。

オンラインであるのするのをむ <https://riptutorial.com/ja/excel-vba/topic/8295/あるのするの>

14: オートフィルタ;とベストプラクティス

き

オートフィルタのは、々がしたいアイテムにをするために、またはのデータからなりなデータマイニングをすることです。「テキスト//」などのパラメータをけることができ、にみねることができます。コネクタとルールセットについて、ごとに2つのをできます。オートフィルタはをフィルタリングすることし、カラムをフィルタリングするオートフィルタはありませんなくともネイティブではありません。

VBAでオートフィルタをするには、なくとものパラメータをしてびすがあります。

```
"MySheet"Range "MyRange". Autofilterフィールド=ColumnNumberWithin "MyRange"  
ToBeFilteredInNumericValueCriteria1= "WhatIWantToFilter"
```

ウェブや[ここではstackoverflow](#)でたくさんがあります

Examples

Smartfilter

の

は、でされるすべてのロジスティックスのきがされているシート「」をとっていますが、にじてフィルタリングすることができますが、これはにがかかり、いわせをよりにするためにプロセスをしたいラックで「パルプ」はですかパルプはっていますかラック5フィルタはらしいツールですが、でこれらのにえるのにはいくらかされています。

	A	B	C	D	E	F	G	H
1	Control Num	DESCRIPTION	QUANTITY	LOCATION	DATE	ACTION		1. How many "Pulp" do we have now? (Total)
2	9005124	Pulp	42	Rack #5	4-Oct-16	In		
15	9005137	Pulp	67	Rack #1	21-Nov-15	Out		
16	9005138	Pulp	92	Rack #3	19-Jun-15	Out		
42	9005164	Pulp	48	Rack #5	1-Dec-15	In		
45	9005167	Pulp	53	Rack #5	17-Mar-15	Out		
50	9005172	Pulp	13	Rack #3	5-Dec-15	In		
55	9005177	Pulp	30	Rack #2	15-Sep-16	In		
56	9005178	Pulp	90	Rack #3	27-Jan-16	Out		
68	9005190	Pulp	67	Rack #7	25-Aug-16	Out		
70	9005192	Pulp	62	Rack #6	7-Nov-15	Out		
71	9005193	Pulp	46	Rack #7	1-Dec-15	Out		
72	9005194	Pulp	6	Rack #2	18-Dec-16	Out		
83	9005205	Pulp	86	Rack #6	30-Mar-16	Out		
102	9005224	Pulp	78	Rack #3	7-Sep-16	Out		
109	9005231	Pulp	19	Rack #1	21-May-15	In		
115	9005237	Pulp	33	Rack #6	14-Jan-15	Out		
121	9005243	Pulp	46	Rack #1	25-Sep-15	Out		
124	9005246	Pulp	48	Rack #1	3-Jan-15	In		
125	9005247	Pulp	39	Rack #3	8-May-16	Out		
142	9005264	Pulp	68	Rack #1	15-Nov-15	In		
146	9005268	Pulp	50	Rack #2	30-Nov-16	In		
154	9005276	Pulp	11	Rack #4	8-Dec-15	In		
156	9005278	Pulp	40	Rack #1	5-Jun-16	In		
169	9005291	Pulp	84	Rack #4	21-Sep-16	Out		
174	9005296	Pulp	31	Rack #1	3-May-16	In		
182	9005304	Pulp	61	Rack #7	9-Apr-16	Out		
190	9005312	Pulp	57	Rack #1	2-Jul-15	Out		
192	9005314	Pulp	56	Rack #2	12-Feb-15	In		
200	9005322	Pulp	43	Rack #7	27-Sep-16	Out		
202	9005324	Pulp	97	Rack #1	16-Apr-16	In		
205	9005327	Pulp	80	Rack #6	8-Nov-16	In		
214	9005336	Pulp	82	Rack #5	27-Jul-15	In		
215	9005337	Pulp	27	Rack #4	17-Sep-16	In		
218	9005340	Pulp	51	Rack #3	16-Nov-15	Out		
		Record						

マクロソリューション

これらのシナリオでは、ワークシートにデータがすでにしており、そのがユーザーによってにできるため、オートフィルタは、このようなシナリオではのどのソリューションであることがわかります。

このアプローチでは、がにじてのデータをにフィルタリングし、にをできる「SmartFilter」というシートをします。

はこのにして2つのモジュールと `Worksheet_Change` イベントをします

SmartFilterワークシートのコード

```
Private Sub Worksheet_Change(ByVal Target As Range)
Dim ItemInRange As Range
Const CellsFilters As String = "C2,E2,G2"
    Call ExcelBusy
    For Each ItemInRange In Target
        If Not Intersect(ItemInRange, Range(CellsFilters)) Is Nothing Then Call Inventory_Filter
    Next ItemInRange
    Call ExcelNormal
End Sub
```

"General_Functions"とばれるモジュール1のコード

```
Sub ExcelNormal()
    With Excel.Application
        .EnableEvents = True
        .Cursor = xlDefault
        .ScreenUpdating = True
        .DisplayAlerts = True
        .StatusBar = False
        .CopyObjectsWithCells = True
    End With
End Sub
Sub ExcelBusy()
    With Excel.Application
        .EnableEvents = False
        .Cursor = xlWait
        .ScreenUpdating = False
        .DisplayAlerts = False
        .StatusBar = False
        .CopyObjectsWithCells = True
    End With
End Sub
Sub Select_Sheet(NameSheet As String, Optional VerifyExistanceOnly As Boolean)
    On Error GoTo Err01Select_Sheet
    Sheets(NameSheet).Visible = True
    If VerifyExistanceOnly = False Then ' 1. If VerifyExistanceOnly = False
        Sheets(NameSheet).Select
        Sheets(NameSheet).AutoFilterMode = False
        Sheets(NameSheet).Cells.EntireRow.Hidden = False
        Sheets(NameSheet).Cells.EntireColumn.Hidden = False
    End If ' 1. If VerifyExistanceOnly = False
    If 1 = 2 Then '99. If error
Err01Select_Sheet:
        MsgBox "Err01Select_Sheet: Sheet " & NameSheet & " doesn't exist!", vbCritical: Call
ExcelNormal: On Error GoTo -1: End
        End If '99. If error
End Sub
Function General_Functions_Find_Title(InSheet As String, TitleToFind As String, Optional
InRange As Range, Optional IsNeededToExist As Boolean, Optional IsWhole As Boolean) As Range
Dim DummyRange As Range
    On Error GoTo Err01General_Functions_Find_Title
    If InRange Is Nothing Then ' 1. If InRange Is Nothing
        Set DummyRange = IIf(IsWhole = True, Sheets(InSheet).Cells.Find(TitleToFind,
LookAt:=xlWhole), Sheets(InSheet).Cells.Find(TitleToFind, LookAt:=xlPart))
    Else ' 1. If InRange Is Nothing
        Set DummyRange = IIf(IsWhole = True,
Sheets(InSheet).Range(InRange.Address).Find(TitleToFind, LookAt:=xlWhole),
```

```

Sheets(InSheet).Range(InRange.Address).Find(TitleToFind, LookAt:=xlPart))
    End If ' 1. If InRange Is Nothing
    Set General_Functions_Find_Title = DummyRange
    If 1 = 2 Or DummyRange Is Nothing Then '99. If error
Err01General_Functions_Find_Title:
    If IsNeededToExist = True Then MsgBox "Err01General_Functions_Find_Title: Title '" &
TitleToFind & "' was not found in sheet '" & InSheet & "'", vbCritical: Call ExcelNormal: On
Error GoTo -1: End
    End If '99. If error
End Function

```

「Inventory_Handling」とばれるモジュール2のコード

```

Const TitleDesc As String = "DESCRIPTION"
Const TitleLocation As String = "LOCATION"
Const TitleActn As String = "ACTION"
Const TitleQty As String = "QUANTITY"
Const SheetRecords As String = "Record"
Const SheetSmartFilter As String = "SmartFilter"
Const RowFilter As Long = 2
Const ColDataToPaste As Long = 2
Const RowDataToPaste As Long = 7
Const RangeInResult As String = "K1"
Const RangeOutResult As String = "K2"
Sub Inventory_Filter()
Dim ColDesc As Long: ColDesc = General_Functions_Find_Title(SheetSmartFilter, TitleDesc,
IsNeededToExist:=True, IsWhole:=True).Column
Dim ColLocation As Long: ColLocation = General_Functions_Find_Title(SheetSmartFilter,
TitleLocation, IsNeededToExist:=True, IsWhole:=True).Column
Dim ColActn As Long: ColActn = General_Functions_Find_Title(SheetSmartFilter, TitleActn,
IsNeededToExist:=True, IsWhole:=True).Column
Dim ColQty As Long: ColQty = General_Functions_Find_Title(SheetSmartFilter, TitleQty,
IsNeededToExist:=True, IsWhole:=True).Column
Dim CounterQty As Long
Dim TotalQty As Long
Dim TotalIn As Long
Dim TotalOut As Long
Dim RangeFiltered As Range
    Call Select_Sheet(SheetSmartFilter)
    If Cells(Rows.Count, ColDataToPaste).End(xlUp).Row > RowDataToPaste - 1 Then
Rows(RowDataToPaste & ":" & Cells(Rows.Count, "B").End(xlUp).Row).Delete
    Sheets(SheetRecords).AutoFilterMode = False
    If Cells(RowFilter, ColDesc).Value <> "" Or Cells(RowFilter, ColLocation).Value <> "" Or
Cells(RowFilter, ColActn).Value <> "" Then ' 1. If Cells(RowFilter, ColDesc).Value <> "" Or
Cells(RowFilter, ColLocation).Value <> "" Or Cells(RowFilter, ColActn).Value <> ""
        With Sheets(SheetRecords).UsedRange
            If Sheets(SheetSmartFilter).Cells(RowFilter, ColDesc).Value <> "" Then .AutoFilter
Field:=General_Functions_Find_Title(SheetRecords, TitleDesc, IsNeededToExist:=True,
IsWhole:=True).Column, Criterial:=Sheets(SheetSmartFilter).Cells(RowFilter, ColDesc).Value
            If Sheets(SheetSmartFilter).Cells(RowFilter, ColLocation).Value <> "" Then .AutoFilter
Field:=General_Functions_Find_Title(SheetRecords, TitleLocation, IsNeededToExist:=True,
IsWhole:=True).Column, Criterial:=Sheets(SheetSmartFilter).Cells(RowFilter, ColLocation).Value
            If Sheets(SheetSmartFilter).Cells(RowFilter, ColActn).Value <> "" Then .AutoFilter
Field:=General_Functions_Find_Title(SheetRecords, TitleActn, IsNeededToExist:=True,
IsWhole:=True).Column, Criterial:=Sheets(SheetSmartFilter).Cells(RowFilter, ColActn).Value
            'If we don't use a filter we would need to use a cycle For/to or For/Each Cell in range
            'to determine whether or not the row meets the criteria that we are looking and then
            'save it on an array, collection, dictionary, etc
            'IG: For CounterRow = 2 To TotalRows
            'If Sheets(SheetSmartFilter).Cells(RowFilter, ColDesc).Value <> "" and

```

```

Sheets(SheetRecords).cells(CounterRow, ColDescInRecords).Value=
Sheets(SheetSmartFilter).Cells(RowFilter, ColDesc).Value then
'Redim Preserve MyUnnecessaryArray(UnnecessaryNumber) ''Save to array:
(UnecessaryNumber)=MyUnnecessaryArray. Or in a dictionary, etc. At the end, we would transpose
this values into the sheet, at the end
'both are the same, but, just try to see the time invested on each logic.
If .Cells(1, 1).End(xlDown).Value <> "" Then Set RangeFiltered = .Rows("2:" &
Sheets(SheetRecords).Cells(Rows.Count, "A").End(xlUp).Row).SpecialCells(xlCellTypeVisible)
'If it is not <>"" means that there was not filtered data!
If RangeFiltered Is Nothing Then MsgBox "Err01Inventory_Filter: No data was found with the
given criteria!", vbCritical: Call ExcelNormal: End
RangeFiltered.Copy Destination:=Cells(RowDataToPaste, ColDataToPaste)
TotalQty = Cells(Rows.Count, ColQty).End(xlUp).Row
For CounterQty = RowDataToPaste + 1 To TotalQty
If Cells(CounterQty, ColActn).Value = "In" Then ' 2. If Cells(CounterQty, ColActn).Value =
"In"
TotalIn = Cells(CounterQty, ColQty).Value + TotalIn
ElseIf Cells(CounterQty, ColActn).Value = "Out" Then ' 2. If Cells(CounterQty,
ColActn).Value = "In"
TotalOut = Cells(CounterQty, ColQty).Value + TotalOut
End If ' 2. If Cells(CounterQty, ColActn).Value = "In"
Next CounterQty
Range(RangeInResult).Value = TotalIn
Range(RangeOutResult).Value = -(TotalOut)
End With
End If ' 1. If Cells(RowFilter, ColDesc).Value <> "" Or Cells(RowFilter,
ColLocation).Value <> "" Or Cells(RowFilter, ColActn).Value <> ""
End Sub

```

テストと

	A	B	C	D	E	F	G	H	I	J	K
912	9013034	Batch weight	21	Rack #1	9-Jun-16	Out					
913	9013035	Pectin	72	Rack #7	22-Jun-16	In					
914	9013036	Sugar	28	Rack #1	5-Aug-15	In					
915	9013037	Solids content	51	Rack #7	11-Sep-16	In					
916	9013038	Pulp	45	Rack #3	9-Apr-16	Out					
917	9013039	Batch weight	19	Rack #4	6-Apr-15	Out					
918	9013040	Citric Acid	98	Rack #4	17-Jun-16	Out					
919	9013041	Citric Acid	97	Rack #1	29-Feb-16	In					
920	9013042	Pulp	57	Rack #5	25-Nov-16	Out					
921	9013043	Citric Acid	42	Rack #2	27-Feb-16	In					
922	9013044	Batch weight	54	Rack #1	16-Sep-15	Out					
923	9013045	Solids content	12	Rack #4	13-Jul-15	In					
924	9013046	Pulp	79	Rack #4	13-Jul-15	Out					
925	9013047	Citric Acid	36	Rack #4	15-Nov-16	Out					
926	9013048	Sugar	35	Rack #3	5-Feb-16	Out					
927	9013049	Pulp	63	Rack #6	16-Dec-16	Out					
928	9013050	Solids content	48	Rack #4	1-Mar-15	In					
929	9013051	Pulp	39	Rack #4	31-May-16	Out					
930	9013052	Pulp	47	Rack #6	26-Feb-16	In					
931	9013053	Sugar	6	Rack #6	3-Mar-16	Out					
932	9013054	Pulp	53	Rack #2	11-Sep-15	Out					
933	9013055	Solids content	87	Rack #4	19-Jan-15	Out					
934	9013056	Sugar	✚	48	Rack #7	23-Nov-16	In				
935	9013057	Solids content	62	Rack #6	15-May-16	Out					
936	9013058	Batch weight	61	Rack #3	3-Dec-16	Out					
937	9013059	Citric Acid	64	Rack #7	7-Feb-16	Out					
938	9013060	Sugar	91	Rack #7	23-Sep-15	Out					
939	9013061	Citric Acid	29	Rack #1	7-Jul-16	Out					
940	9013062	Citric Acid	31	Rack #6	17-Feb-16	In					
941	9013063	Batch weight	53	Rack #1	5-Apr-15	Out					
942	9013064	Citric Acid	25	Rack #6	30-Jul-15	Out					
943	9013065	Citric Acid	68	Rack #4	22-Mar-16	Out					
944	9013066	Boiling time	22	Rack #6	17-Jun-15	In					
945	9013067	Pectin	99	Rack #2	2-Nov-16	Out					
946	9013068	Solids content	79	Rack #2	17-Nov-16	Out					
SmartFilter		Record	+								

のでたように、このはにされました。 オートフィルタをすることで、にかかり、ユーザにするのがで、このコマンドにしている、コードにをしたソリューションがされました。

オンラインでオートフィルタ-とベストプラクティスをむ <https://riptutorial.com/ja/excel-vba/topic/8645/オートフィルタ-とベストプラクティス>

15: コンボボックスでアクティブワークシートのドロップダウンメニューをする

き

これは、コンボボックスActiveXオブジェクトをシートにすることにより、ブックのアクティブシートにドロップダウンメニューをするをすなです。 Jimi Hendrixの5のうちの1を、シートのアクティブなセルにして、それにじてすることができます。

Examples

ジミヘンドリックスメニュー

に、コードはシートのもジュールにされます。

これは、アクティブシートでなるセルがされるたびに、Worksheet_SelectionChangeイベントです。コードウィンドウののドロップダウンメニューから「ワークシート」をし、そのにあるドロップダウンメニューから「Selection_Change」をすることができます。この、セルをアクティブにするたびに、コードはコンボボックスのコードにリダイレクトされます。

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)

    ComboBox1_Change

End Sub
```

ここでは、ComboBoxのルーチンは、デフォルトでChangeイベントにコードされています。そのには、すべてのオプションがされたがあります。ののCLEARオプションではなく、セルのをするためにされます。その、はコンボボックスにされ、をうルーチンにされます。

```
Private Sub ComboBox1_Change()

    Dim myarray(0 To 5)
    myarray(0) = "Hey Joe"
    myarray(1) = "Little Wing"
    myarray(2) = "Voodoo Child"
    myarray(3) = "Purple Haze"
    myarray(4) = "The Wind Cries Mary"
    myarray(5) = "CLEAR"

    With ComboBox1
        .List = myarray()
    End With

    FillACell myarray()

End Sub
```

は、セルをにするためにソングまたはヌルでセルをりつぶすルーチンにされます。まず、には、ユーザーがしたのがえられます。に、コンボボックスは、ユーザーがアクティブにしたセルののにされ、そのがされて、よりくのがられます。アクティブなセルには、ユーザーのをするのにがりてられます。ユーザーがオプションからCLEARをすると、セルはになります。

されたセルごとにルーチンがりされます。

```
Sub FillACell(MyArray As Variant)

Dim n As Integer

n = ComboBox1.ListIndex

ComboBox1.Left = ActiveCell.Left
ComboBox1.Top = ActiveCell.Top
Columns(ActiveCell.Column).ColumnWidth = ComboBox1.Width * 0.18

ActiveCell = MyArray(n)

If ComboBox1 = "CLEAR" Then
    Range(ActiveCell.Address) = ""
End If

End Sub
```

2 まれていないオプション

このは、なのデータベースとそれにするアメニティにまれないのあるオプションのにされます。

のをにしていますが、いくつかのいがあります。

1. のプロシージャにコードをみわせることで、のコンボボックスで2つのプロシージャがになりました。
2. LinkedCellプロパティをして、ユーザーのしいをにする
3. アクティブなセルをにするためのバックアップがまれており、のについたエラーコードがまれています。ここでは、アクティブなセルにされるとがとしてフォーマットされます。

```
Private Sub cboNotIncl_Change()

Dim n As Long
Dim notincl_array(1 To 9) As String

n = myTarget.Row

If n >= 3 And n < 10000 Then

    If myTarget.Address = "$G$" & n Then

        'set up the array elements for the not included services
        notincl_array(1) = "Central Air"
        notincl_array(2) = "Hot Water"
        notincl_array(3) = "Heater Rental"
        notincl_array(4) = "Utilities"
        notincl_array(5) = "Parking"
```

```

        notincl_array(6) = "Internet"
        notincl_array(7) = "Hydro"
        notincl_array(8) = "Hydro/Hot Water/Heater Rental"
        notincl_array(9) = "Hydro and Utilities"

        cboNotIncl.List = notincl_array()

Else

    Exit Sub

End If

With cboNotIncl

    'make sure the combo box moves to the target cell
    .Left = myTarget.Left
    .Top = myTarget.Top

    'adjust the size of the cell to fit the combo box
    myTarget.ColumnWidth = .Width * 0.18

    'make it look nice by editing some of the font attributes
    .Font.Size = 11
    .Font.Bold = False

    'populate the cell with the user choice, with a backup guarantee that it's in
column G

    If myTarget.Address = "$G$" & n Then

        .LinkedCell = myTarget.Address

        'prevent an error where a numerical value is formatted as text
        myTarget.EntireColumn.TextToColumns

    End If

End With

End If 'ensure that the active cell is only between rows 3 and 1000

End Sub

```

のマクロは、セルがワークシートモジュールのSelectionChangeイベントでアクティブされるたびにされます。

```

Public myTarget As Range

Private Sub Worksheet_SelectionChange(ByVal Target As Range)

    Set myTarget = Target

    'switch for Not Included
    If Target.Column = 7 And Target.Cells.Count = 1 Then

        Application.Run "Module1.cboNotIncl_Change"

    End If

```


オンラインでコンボボックスでアクティブワークシートのドロップダウンメニューをするをむ
<https://riptutorial.com/ja/excel-vba/topic/8929/コンボボックスでアクティブワークシートのドロップダウンメニューをする>

16: チャートとチャート

Examples

とによるチャートの

チャートは、チャートデータをするSeriesオブジェクトですることできます。のグラフをたないSeriesにするには、ChartObjectをWorksheet ChartObjectをしてからChartオブジェクトをします。SeriesオブジェクトをすることのXValues、RangeオブジェクトをしてValuesおよびXValuesですることです。これらのデータプロパティは、これらのへのをつSeriesをにします。このアプローチのは、Nameするときにはじがされないことです。です。のRangeのデータとはされません。SERIESをチェックして、そのがされていることは確かです。これは、SERIESをすることによってするがあります。

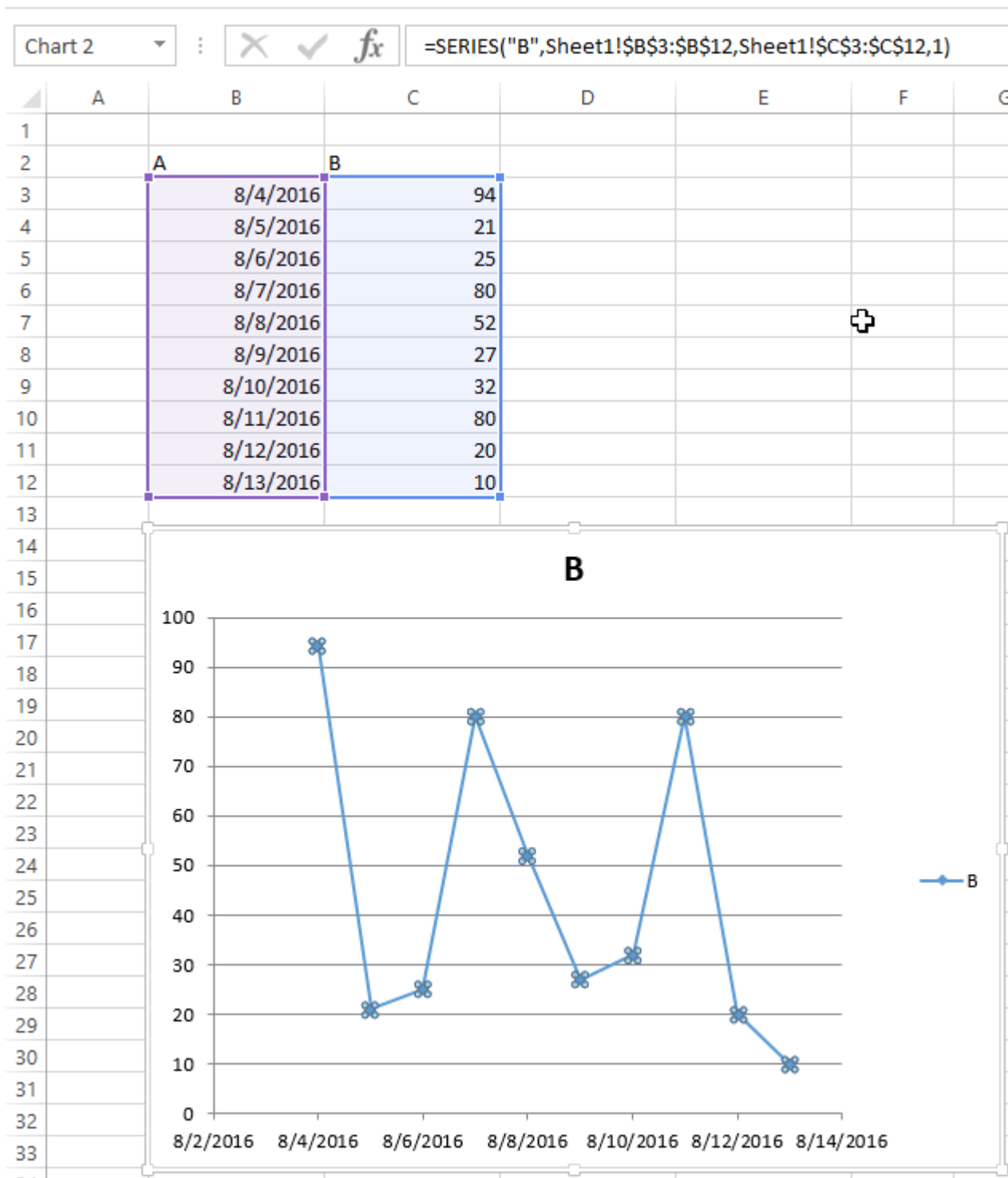
チャートのにされるコード

このコードには、ChartおよびWorksheetのためのながまれています。これらはしないはすることができます。ただし、スタイルやそののチャートのプロパティをしているはです。

```
Sub CreateChartWithRangesAndFixedName()  
  
    Dim xData As Range  
    Dim yData As Range  
    Dim serName As Range  
  
    'set the ranges to get the data and y value label  
    Set xData = Range("B3:B12")  
    Set yData = Range("C3:C12")  
    Set serName = Range("C2")  
  
    'get reference to ActiveSheet  
    Dim sht As Worksheet  
    Set sht = ActiveSheet  
  
    'create a new ChartObject at position (48, 195) with width 400 and height 300  
    Dim chtObj As ChartObject  
    Set chtObj = sht.ChartObjects.Add(48, 195, 400, 300)  
  
    'get reference to chart object  
    Dim cht As Chart  
    Set cht = chtObj.Chart  
  
    'create the new series  
    Dim ser As Series  
    Set ser = cht.SeriesCollection.NewSeries  
  
    ser.Values = yData  
    ser.XValues = xData  
    ser.Name = serName  
  
    ser.ChartType = xlXYScatterLines
```

コードのデータ/とのchart

SERIESには、それをしたRangeへのではなく、シリーズの"B"まれていることにしてください。



のグラフをする

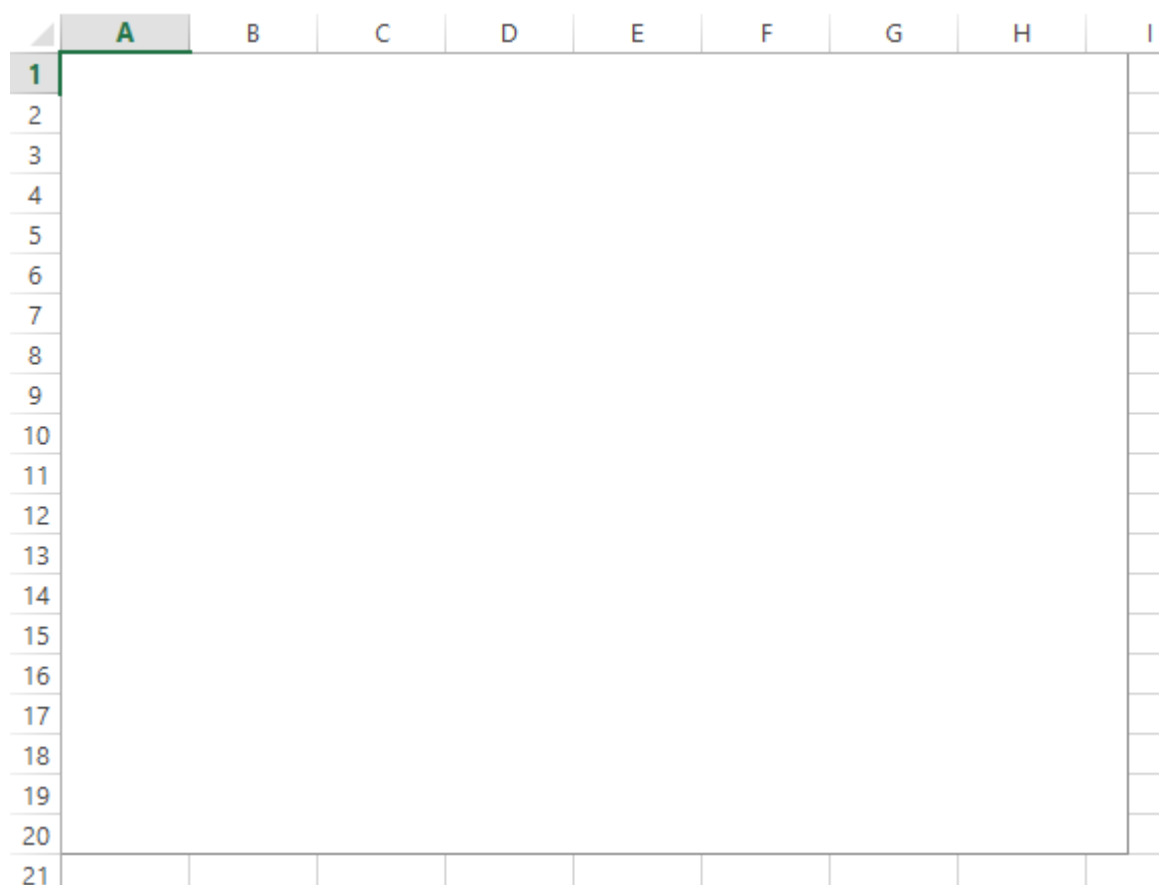
のチャートコードのは、のChartをすることです。このChartは、アクティブなデフォルトチャートテンプレートであり、にはではないテンプレートがされているがあります。

ChartObjectのキーは、そのをすることです。コールのはChartObjects.Add(Left, Top, Width, Height)です。ChartObjectをしたら、ChartオブジェクトをしてChartをにすることができます。ChartObjectは、Shapeようにし、チャートをシートにします。

のグラフをするコード

```
Sub CreateEmptyChart()  
  
    'get reference to ActiveSheet  
    Dim sht As Worksheet  
    Set sht = ActiveSheet  
  
    'create a new ChartObject at position (0, 0) with width 400 and height 300  
    Dim chtObj As ChartObject  
    Set chtObj = sht.ChartObjects.Add(0, 0, 400, 300)  
  
    'get refernce to chart object  
    Dim cht As Chart  
    Set cht = chtObj.Chart  
  
    'additional code to modify the empty chart  
    '...  
  
End Sub
```

のチャート



SERIESをしてグラフをする

しいChartおよびSeriesオブジェクトにダイナミックSeriesのをにするには、SERIESをするがあります。Rangeオブジェクトをするプロセスはで、なはにSERIESのです。

SERIESのはのとおりです。

```
=SERIES (Name,XValues,Values,Order)
```

これらは、データのまたはとしてできます。Orderは、チャートののをします。シートでされていないと、データへのはしません。のについては、のシリーズをクリックし、バーをしてください。

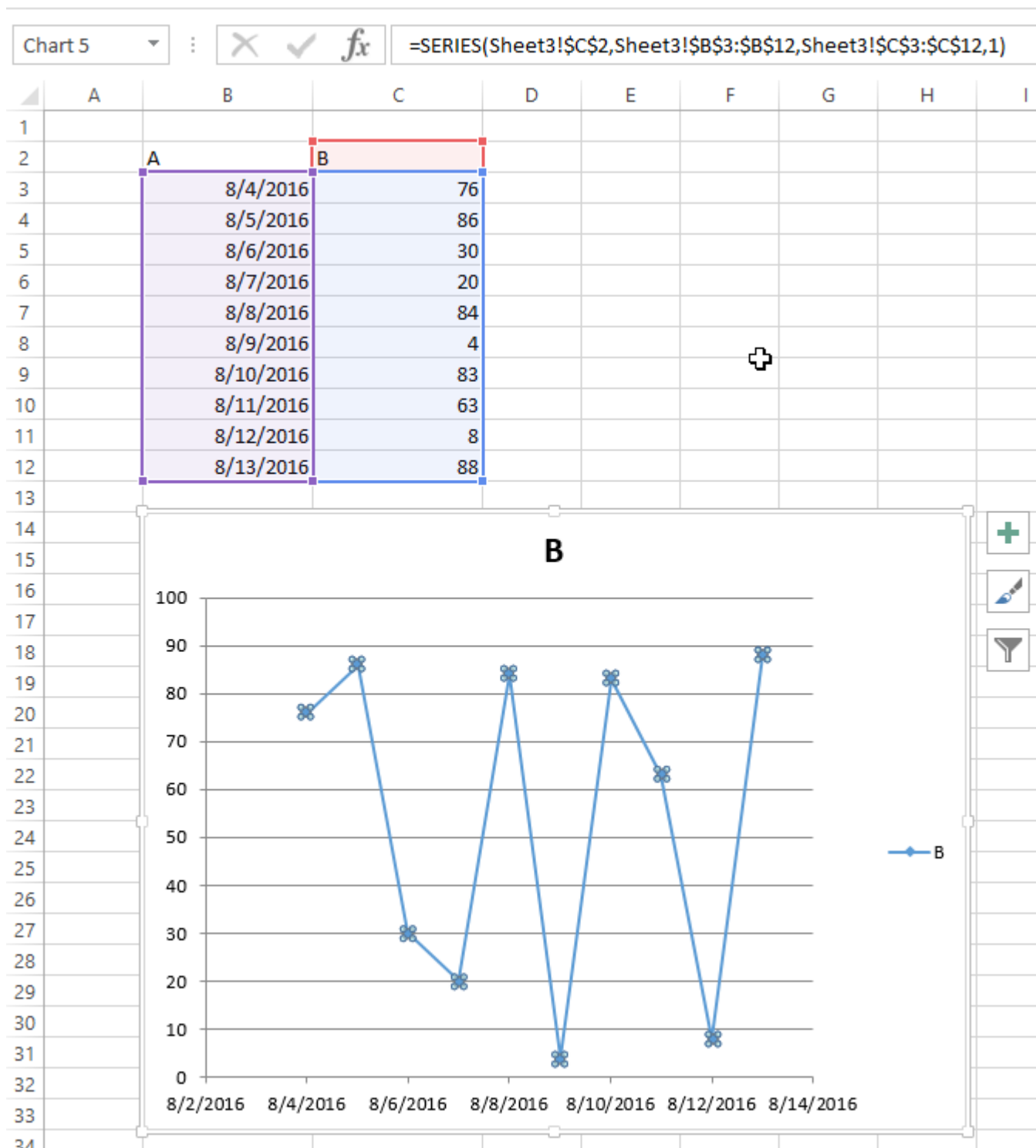
SERIESをしてグラフをし、データをするためのコード

SERIESをするでは、.Address(,,,True)されます。これにより、シートのアドレスがまれるようにされます。シートがされているとエラーになります。

```
Sub CreateChartUsingSeriesFormula()  
  
    Dim xData As Range  
    Dim yData As Range  
    Dim serName As Range  
  
    'set the ranges to get the data and y value label  
    Set xData = Range("B3:B12")  
    Set yData = Range("C3:C12")  
    Set serName = Range("C2")  
  
    'get reference to ActiveSheet  
    Dim sht As Worksheet  
    Set sht = ActiveSheet  
  
    'create a new ChartObject at position (48, 195) with width 400 and height 300  
    Dim chtObj As ChartObject  
    Set chtObj = sht.ChartObjects.Add(48, 195, 400, 300)  
  
    'get refernce to chart object  
    Dim cht As Chart  
    Set cht = chtObj.Chart  
  
    'create the new series  
    Dim ser As Series  
    Set ser = cht.SeriesCollection.NewSeries  
  
    'set the SERIES formula  
    '=SERIES(name, xData, yData, plotOrder)  
  
    Dim formulaValue As String  
    formulaValue = "=SERIES(" & _  
        serName.Address(, , , True) & "," & _  
        xData.Address(, , , True) & "," & _  
        yData.Address(, , , True) & ",1)"  
  
    ser.Formula = formulaValue  
    ser.ChartType = xlXYScatterLines  
  
End Sub
```

のデータとチャート

このグラフでは、シリーズがのセルののしくされていることにしてください。これは、がChartすることを行います。



チャートをグリッドにする

Excelでのチャートのなは、のチャートのサイズとレイアウトをシートでしています。では、グラフのサイズをしたり、セルのに「スティック」するようにグラフをしながら、Altキーをしたままにすることができます。これはカップルチャートでもしますが、VBAのアプローチはずっと

です。

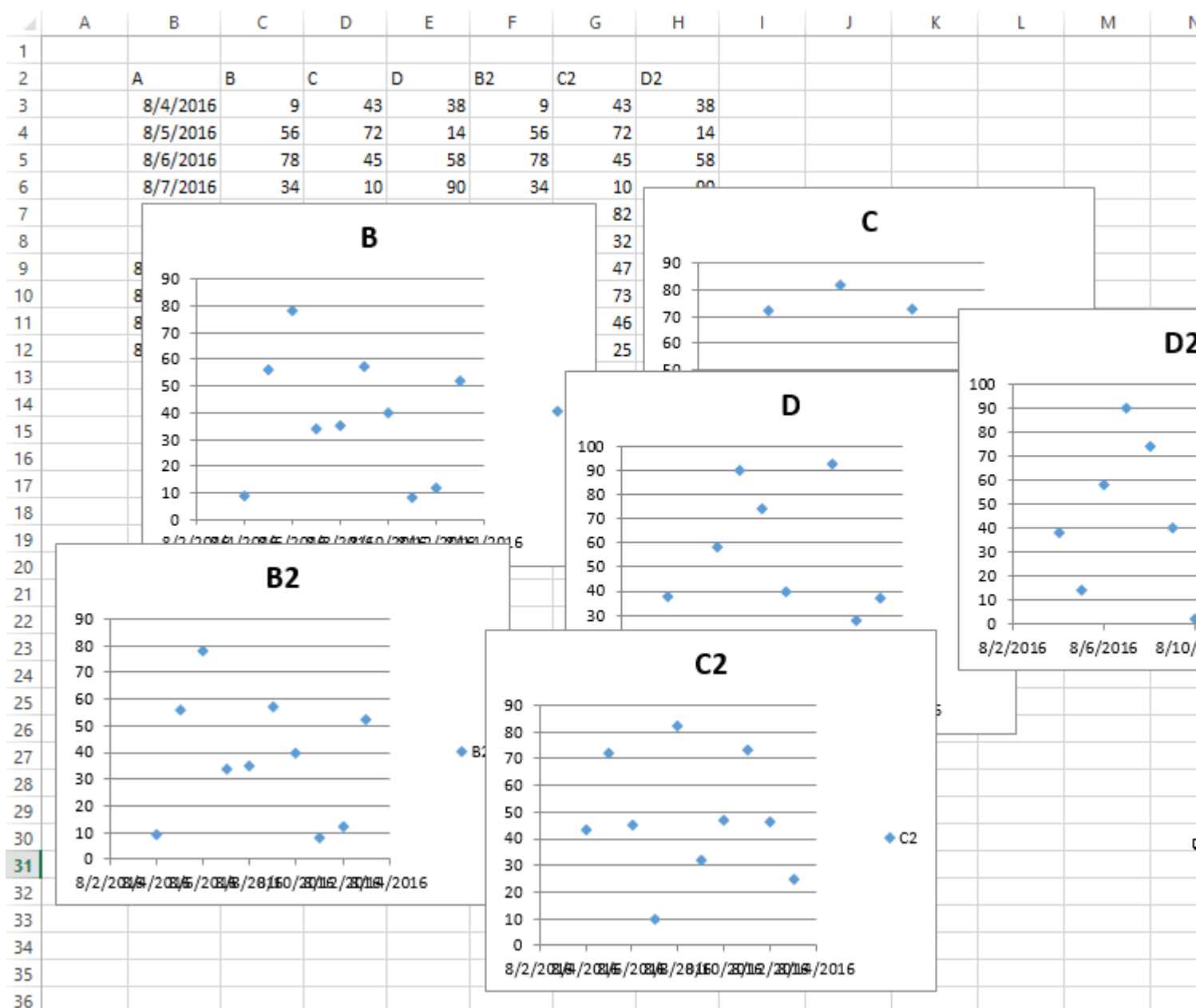
グリッドをするコード

このコードは、された、からし、されたのと、されたグラフのサイズで、グラフのグリッドをします。チャートはされたにされ、エッジをんでしいをします。

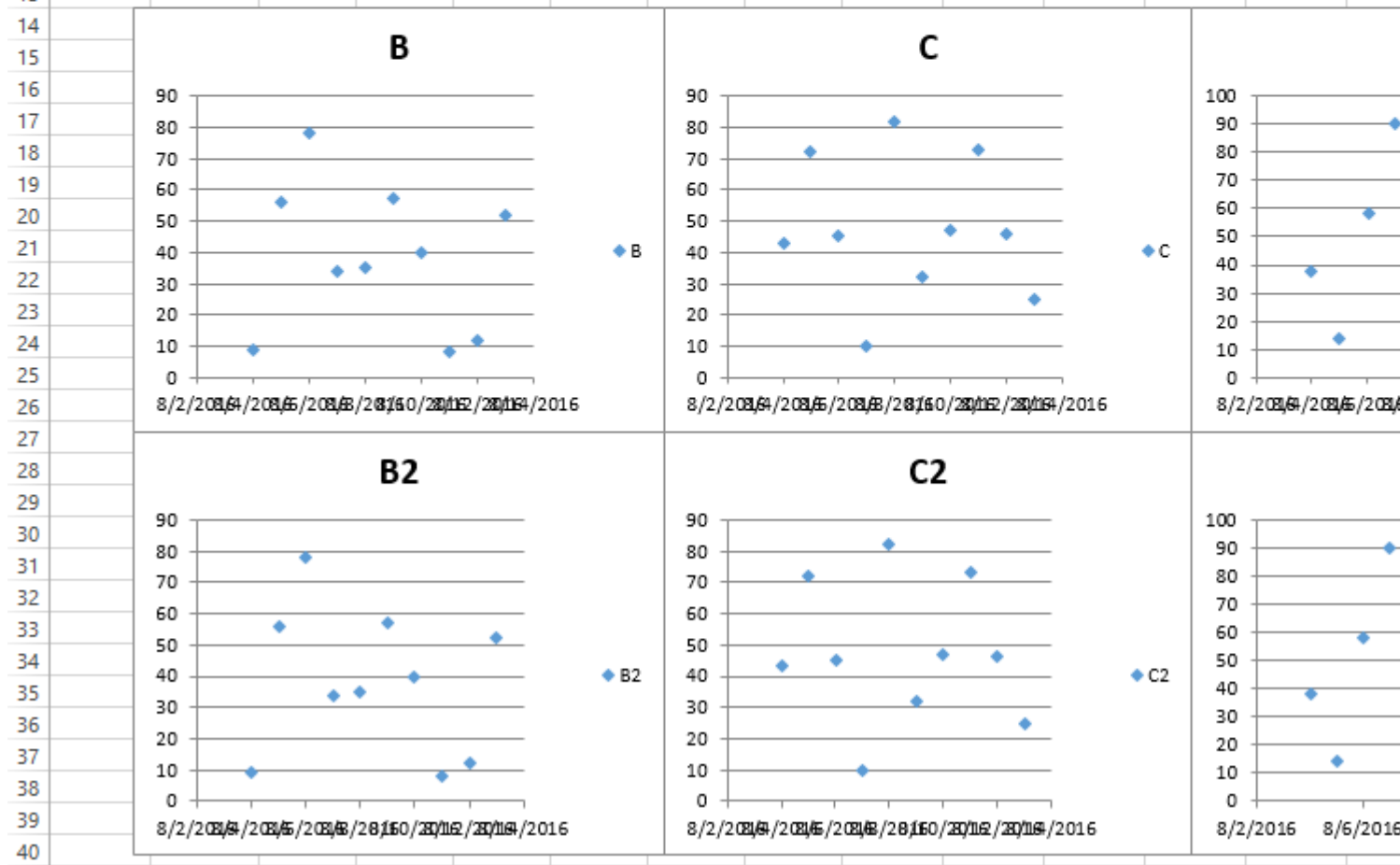
```
Sub CreateGridOfCharts()  
  
    Dim int_cols As Integer  
    int_cols = 3  
  
    Dim cht_width As Double  
    cht_width = 250  
  
    Dim cht_height As Double  
    cht_height = 200  
  
    Dim offset_vertical As Double  
    offset_vertical = 195  
  
    Dim offset_horz As Double  
    offset_horz = 40  
  
    Dim sht As Worksheet  
    Set sht = ActiveSheet  
  
    Dim count As Integer  
    count = 0  
  
    'iterate through ChartObjects on current sheet  
    Dim cht_obj As ChartObject  
    For Each cht_obj In sht.ChartObjects  
  
        'use integer division and Mod to get position in grid  
        cht_obj.Top = (count \ int_cols) * cht_height + offset_vertical  
        cht_obj.Left = (count Mod int_cols) * cht_width + offset_horz  
        cht_obj.Width = cht_width  
        cht_obj.Height = cht_height  
  
        count = count + 1  
  
    Next cht_obj  
End Sub
```

いくつかのチャートの

これらのは、チャートののランダムなレイアウトと、のコードをするグリッドをしています。



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1														
2		A	B	C	D	B2	C2	D2						
3		8/4/2016		9	43	38	9	43	38					
4		8/5/2016		56	72	14	56	72	14					
5		8/6/2016		78	45	58	78	45	58					
6		8/7/2016		34	10	90	34	10	90					
7		8/8/2016		35	82	74	35	82	74					
8		8/9/2016		57	32	40	57	32	40					
9		8/10/2016		40	47	2	40	47	2					
10		8/11/2016		8	73	93	8	73	93					
11		8/12/2016		12	46	28	12	46	28					
12		8/13/2016		52	25	37	52	25	37					



オンラインでチャートとチャートをむ <https://riptutorial.com/ja/excel-vba/topic/4968/チャートとチャート>

17: デバッグとトラブルシューティング

- Debug.Printstring
- /

Examples

Debug.Print

イミディエイトウィンドウにエラーコードののをするには、 Debug.PrintにDebug.Printます。

```
Private Sub ListErrCodes()  
    Debug.Print "List Error Code Descriptions"  
    For i = 0 To 65535  
        e = Error(i)  
        If e <> "Application-defined or object-defined error" Then Debug.Print i & ": " & e  
    Next i  
End Sub
```

イミディエイトウィンドウをするには、のをいます。

- VのIEWを|は、メニューバーからウィンドウを mmediate
- キーボードショートカット **Ctrl-G**をする

やめる

コマンドは、びされたときにをします。そこからプロセスをしたり、ステップごとにすることができます。

```
Sub Test()  
    Dim TestVar as String  
    TestVar = "Hello World"  
    Stop 'Sub will be executed to this point and then wait for the user  
    MsgBox TestVar  
End Sub
```

イミディエイトウィンドウ

サブコードをすることなくマクロコードのをテストするは、ウィンドウにコマンドをし、 ENTERをしてをすることができENTER。

のをテストするは、そののにけることができ?イミディエイトウィンドウにします。または、printコマンドをしてをすることもできます。

Visual Basic Editorで、イミディエイトウィンドウをくために、 CTRL + GしながらCTRL + Gします。されているシートのを "ExampleSheet"にするには、イミディエイトウィンドウでのようにし、ENTERしENTER

```
ActiveSheet.Name = "ExampleSheet"
```

イミディエイトウィンドウでされているシートのをするには

```
? ActiveSheet.Name  
ExampleSheet
```

このメソッドは、みみまたはユーザーのをコードにするにテストするのににです。のは、イミディエイトウィンドウをして、またはのをテストして、どおりのをするをしています。

```
'In this example, the Immediate Window was used to confirm that a series of Left and Right  
'string methods would return the desired string  
  
'expected output: "value"  
print Left(Right("1111value1111",9),5) ' <---- written code here, ENTER pressed  
value                                ' <---- output
```

イミディエイトウィンドウは、アプリケーション、ブック、またはそののなプロパティをまたはリセットするためにもできます。これは、`Application.EnableEvents = False`をサブルーチンでせずにスローし、を`True`リセットせずにさせるにちますこれは、イライラしてしないをきこすがあります。イミディエイトウィンドウにしてします。

```
? Application.EnableEvents      ' <---- Testing the current state of "EnableEvents"  
False                          ' <---- Output  
Application.EnableEvents = True ' <---- Resetting the property value to True  
? Application.EnableEvents      ' <---- Testing the current state of "EnableEvents"  
True                          ' <---- Output
```

よりなデバッグでは、コロン:をりとしてできます。これは、のではループなどののにできます。

```
x = Split("a,b,c",","): For i = LBound(x,1) to UBound(x,1): Debug.Print x(i): Next i ' <----  
Input this and press enter  
a ' <----Output  
b ' <----Output  
c ' <----Output
```

タイマーをしてパフォーマンスのボトルネックをつける

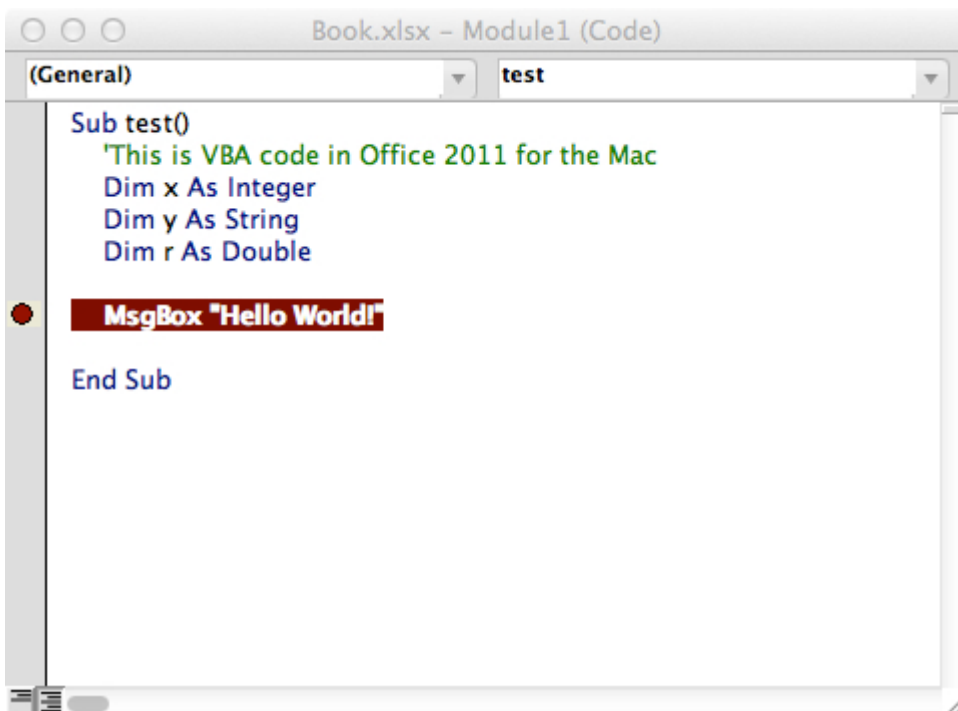
をするためののは、コードのものをつけることです。Timer VBAは、WindowsベースのPCで1/256 3.90625ミリののでからしたをします。VBAのNowとTimeは1でです。

```
Dim start As Double          ' Timer returns Single, but converting to Double to avoid  
start = Timer                ' scientific notation like 3.90625E-03 in the Immediate window  
' ... part of the code  
Debug.Print Timer - start; "seconds in part 1"  
  
start = Timer  
' ... another part of the code  
Debug.Print Timer - start; "seconds in part 2"
```

コードにブレークポイントをする

をするVBAコードののにあるのをクリックすることで、コードにブレークポイントをにできます。いがにされ、ブレークポイントコードもでハイライトされます。

コードにのブレークポイントをし、メニューバーの ""アイコンをすことでをできます。すべてのコードがとしてブレークポイントになることはできません。プロシージャのまたはのとコメントをブレークポイントとしてすることはできません。



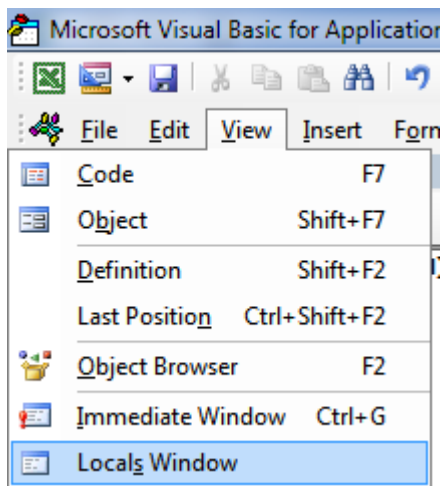
デバッグローカルウィンドウ

[Locals]ウィンドウでは、のまたはサブルーチンのスコープにあるとオブジェクトのののにアクセスできます。をつけるためにコードをデバッグし、をめるためにはなツールです。また、しなかったかもしれないプロパティをすることもできます。

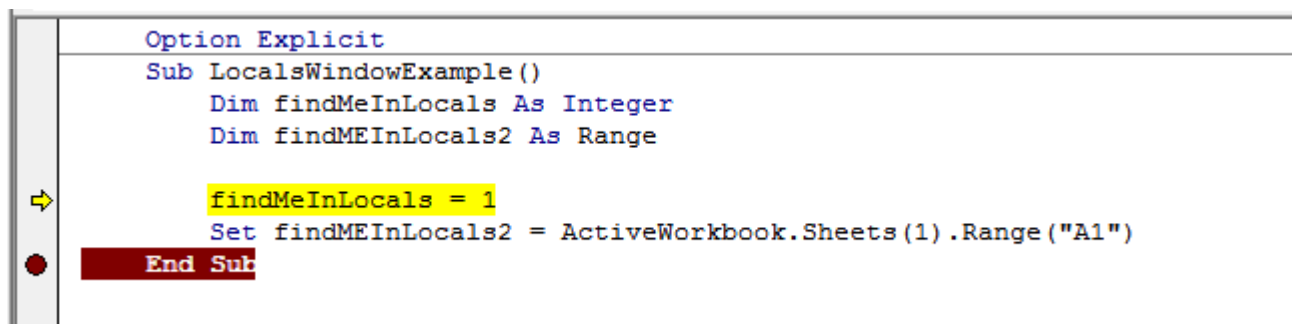
のをえてみましょう。

```
Option Explicit  
Sub LocalsWindowExample()  
    Dim findMeInLocals As Integer  
    Dim findMEInLocals2 As Range  
  
    findMeInLocals = 1  
    Set findMEInLocals2 = ActiveWorkbook.Sheets(1).Range("A1")  
End Sub
```

VBAエディタで、[] -> [ローカルウィンドウ]をクリックします。



に、サブルーチンのをクリックしたでF8をしてコードをステップすることによって、findMeInLocalsをりてるにしました。が0であることがわかります。これはしてをりてなかったにされます。オブジェクトは 'Nothing' です。

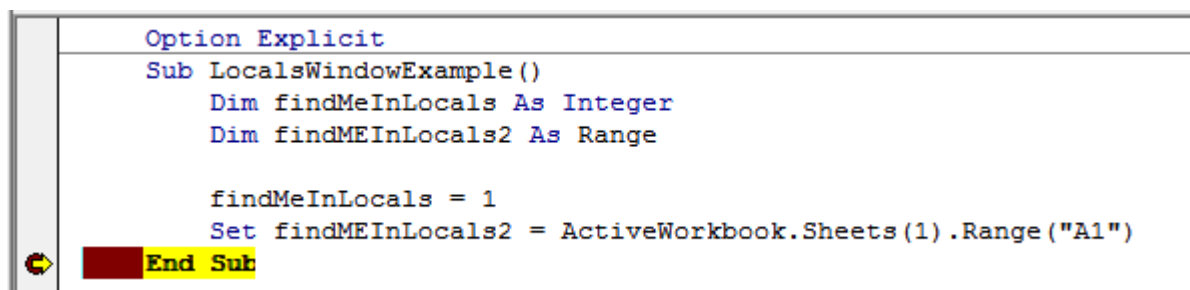


Locals

VBAProject.Sheet1.LocalsWindowExample

Expression	Value	Type
Me		Sheet
findMeInLocals	0	Integer
findMEInLocals2	Nothing	Range

サブルーチンがするにすれば、のをすることができます。



findMeInLocalsのは1、はInteger、FindMeInLocals2はRange / Rangeです。 +をクリックすると、オブジェクトがされ、カウントやなどのプロパティがされます。

Locals

VBAProject.Sheet1.LocalsWindowExample

Expression	Value	Type
Me		Sheet
findMeInLocals	1	Integer
findMEInLocals2		Range
AddIndent	False	Variant
AllowEdit	True	Boolean
Application		Application
Areas		Area
Borders		Border
Cells		Range
Column	1	Long
ColumnWidth	8.43	Variant
Comment	Nothing	Comment
Count	1	Long
CountLarge	1	Variant
Creator	xlCreatorCode	XICr
CurrentArray	<No cells were found.>	Range
CurrentRegion		Range
Dependents	<No cells were found.>	Range
DirectDependents	<No cells were found.>	Range
DirectPrecedents	<No cells were found.>	Range
DisplayFormat		Display

オンラインでデバッグとトラブルシューティングをむ <https://riptutorial.com/ja/excel-vba/topic/861/デバッグとトラブルシューティング>

18: バインディング

Examples

バインディングといバインディング

バインディングは、オブジェクトをまたはにりてるプロセスです。バインディングバインディングともばれますは、Excelでされたオブジェクトがワークシートやワークブックなどのオブジェクトであるです。オブジェクトおよびバリエーションなど、なオブジェクトのけがわれたときにがします。

バインディングは、レイトバインディングよりもいくつかのをします。

- バインディングは、ランタイムのバインディングよりもです。にバインディングをしてオブジェクトをすると、VBAプロジェクトがにロードされたときにバインディングがするまでにかかります。
- のバインディングは、キー/アイテムのペアをにすることによって、をします。
- コードによっては、バインディングがのレベルのチェックをし、エラーをらすことがあります。
- バインドされたオブジェクトのプロパティとメソッドをするときのVBEののは、バインディングではですが、レイトバインディングではできません。

バインディングをするには、VBEのTools→Referencesコマンドをして、VBAプロジェクトへのなをするがあります。

このライブラリは、プロジェクトとにばれます。VBAプロジェクトがされてのコンピュータでされるときにするはありません。

```
'Looping through a dictionary that was created with late binding1
Sub iterateDictionaryLate()
    Dim k As Variant, dict As Object

    Set dict = CreateObject("Scripting.Dictionary")
    dict.CompareMode = vbTextCompare          'non-case sensitive compare model

    'populate the dictionary
    dict.Add Key:="Red", Item:="Balloon"
    dict.Add Key:="Green", Item:="Balloon"
    dict.Add Key:="Blue", Item:="Balloon"

    'iterate through the keys
    For Each k In dict.Keys
        Debug.Print k & " - " & dict.Item(k)
    Next k

    dict.Remove "blue"          'remove individual key/item pair by key
    dict.RemoveAll              'remove all remaining key/item pairs

End Sub

'Looping through a dictionary that was created with early binding1
```

```

Sub iterateDictionaryEarly()
    Dim d As Long, k As Variant
    Dim dict As New Scripting.Dictionary

    dict.CompareMode = vbTextCompare          'non-case sensitive compare model

    'populate the dictionary
    dict.Add Key:="Red", Item:="Balloon"
    dict.Add Key:="Green", Item:="Balloon"
    dict.Add Key:="Blue", Item:="Balloon"
    dict.Add Key:="White", Item:="Balloon"

    'iterate through the keys
    For Each k In dict.Keys
        Debug.Print k & " - " & dict.Item(k)
    Next k

    'iterate through the keys by the count
    For d = 0 To dict.Count - 1
        Debug.Print dict.Keys(d) & " - " & dict.Items(d)
    Next d

    'iterate through the keys by the boundaries of the keys collection
    For d = LBound(dict.Keys) To UBound(dict.Keys)
        Debug.Print dict.Keys(d) & " - " & dict.Items(d)
    Next d

    dict.Remove "blue"                        'remove individual key/item pair by key
    dict.Remove dict.Keys(0)                  'remove first key/item by index position
    dict.Remove dict.Keys(UBound(dict.Keys)) 'remove last key/item by index position
    dict.RemoveAll                            'remove all remaining key/item pairs

End Sub

```

ただし、アーリーバインディングをしていて、しているライブラリの1つがしているシステムでドキュメントをすると、がします。われたライブラリをするルーチンはしくしないだけでなく、ドキュメントのすべてのコードのがになります。そのコンピュータでドキュメントのコードがしいがあります。

これは、レイトバインディングがなです。レイトバインディングをするは、[ツール]> []メニューにをするはありません。なライブラリをつマシンでは、コードはききします。ライブラリがないマシンでは、ライブラリをするコマンドはしませんが、ドキュメントののコードはすべてききします。

しているライブラリをしていないは、バインディングをしてコードをし、にレイトバインディングにりえるとです。これにより、にVBEのIntelliSenseとオブジェクトブラウザをすることができします。

オンラインでバインディングをむ <https://riptutorial.com/ja/excel-vba/topic/3811/バインディング>

19: ピボットテーブル

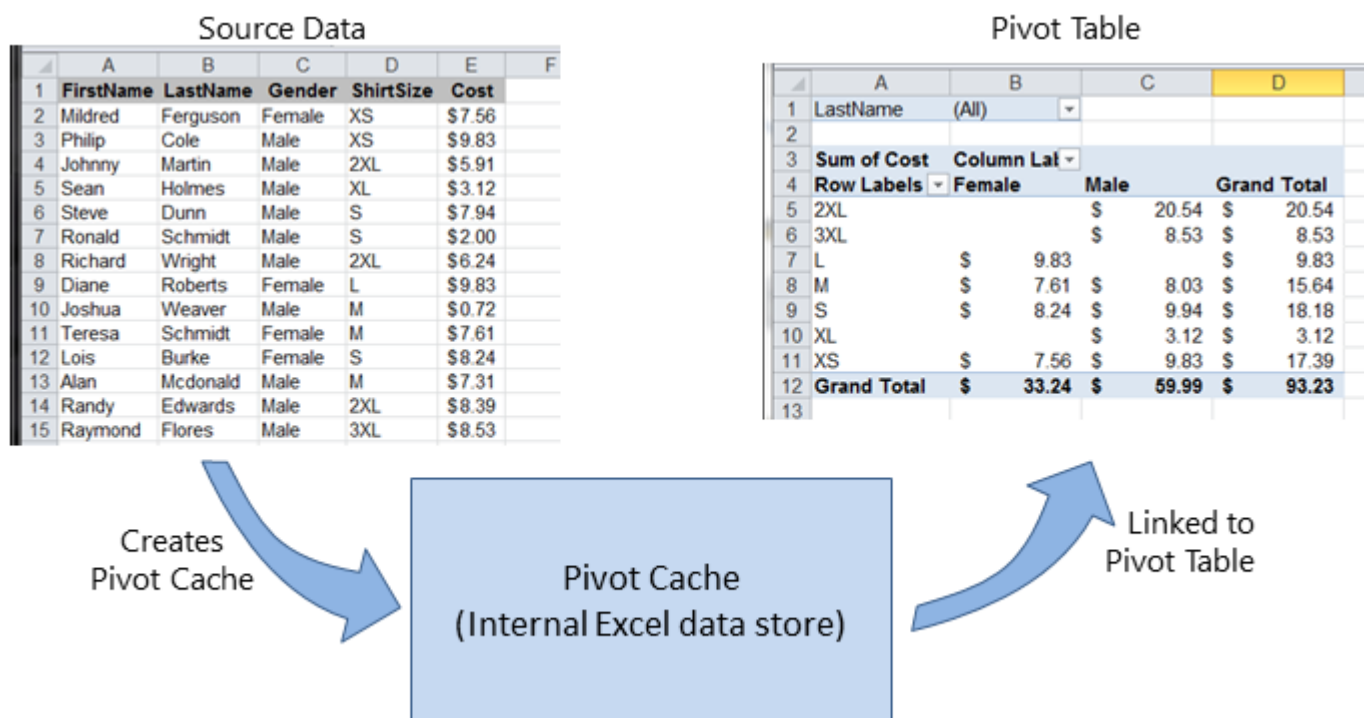
Webにはくのれたとサンプルソースがあります。のとは、ここでクイックアンサーのポイントとしてされています。よりなは、のオリジナルをコピーするのではなく、コンテンツにリンクされているがあります。

Examples

ピボットテーブルの

Excelでもなの1つは、ピボットテーブルをしてデータをべえてすることです。ピボット・テーブルとピボット・キャッシュのをし、テーブルのさまざまなをしてするをすると、VBAをしてピボットをおよびするがです。

もなソースデータは、WorksheetのデータのRangeです。このデータは、ののとしてヘッダーをつデータをしなければなりません。ピボットテーブルがされると、ユーザーはいつでもソースデータをおよびできます。ただし、ピボットテーブルにされているピボットキャッシュとばれるデータがするため、かにピボットテーブルにされることはありません。



のピボットテーブルがなは、じソースデータについてピボットキャッシュをピボットテーブルのデータストアとしてできます。これは、メモリをし、ストレージのExcelファイルのサイズをするため、いです。

Source Data

	A	B	C	D	E	F
1	FirstName	LastName	Gender	ShirtSize	Cost	
2	Mildred	Ferguson	Female	XS	\$7.56	
3	Philip	Cole	Male	XS	\$9.83	
4	Johnny	Martin	Male	2XL	\$5.91	
5	Sean	Holmes	Male	XL	\$3.12	
6	Steve	Dunn	Male	S	\$7.94	
7	Ronald	Schmidt	Male	S	\$2.00	
8	Richard	Wright	Male	2XL	\$6.24	
9	Diane	Roberts	Female	L	\$9.83	
10	Joshua	Weaver	Male	M	\$0.72	
11	Teresa	Schmidt	Female	M	\$7.61	
12	Lois	Burke	Female	S	\$8.24	
13	Alan	Mcdonald	Male	M	\$7.31	
14	Randy	Edwards	Male	2XL	\$8.39	
15	Raymond	Flores	Male	3XL	\$8.53	

Pivot Table

	A	B	C	D
1	LastName	(All)		
2				
3	Sum of Cost	Column Labels		
4	Row Labels	Female	Male	Grand Total
5	2XL		\$ 20.54	\$ 20.54
6	3XL		\$ 8.53	\$ 8.53
7	L	\$ 9.83		\$ 9.83
8	M	\$ 7.61	\$ 8.03	\$ 15.64
9	S	\$ 8.24	\$ 9.94	\$ 18.18
10	XL		\$ 3.12	\$ 3.12
11	XS	\$ 7.56	\$ 9.83	\$ 17.39
12	Grand Total	\$ 33.24	\$ 59.99	\$ 93.23
13				

Creates
Pivot CachePivot Cache
(Internal Excel data store)Linked to
Pivot Tables

	A	B	C	D
1	LastName	(All)		
2				
3	Sum of Cost	Column Labels		
4	Row Labels	Female	Male	Grand Total
5	2XL		\$ 20.54	\$ 20.54
6	3XL		\$ 8.53	\$ 8.53
7	L	\$ 9.83		\$ 9.83
8	M	\$ 7.61	\$ 8.03	\$ 15.64
9	S	\$ 8.24	\$ 9.94	\$ 18.18
10	XL		\$ 3.12	\$ 3.12
11	XS	\$ 7.56	\$ 9.83	\$ 17.39
12	Grand Total	\$ 33.24	\$ 59.99	\$ 93.23
13				

として、のにすソースデータについてピボットテーブルをするには

```

Sub test()
    Dim pt As PivotTable
    Set pt = CreatePivotTable(ThisWorkbook.Sheets("Sheet1").Range("A1:E15"))
End Sub

Function CreatePivotTable(ByRef srcData As Range) As PivotTable
    '--- creates a Pivot Table from the given source data and
    '    assumes that the first row contains valid header data
    '    for the columns
    Dim thisPivot As PivotTable
    Dim dataSheet As Worksheet
    Dim ptSheet As Worksheet
    Dim ptCache As PivotCache

    '--- the Pivot Cache must be created first...
    Set ptCache = ThisWorkbook.PivotCaches.Create(SourceType:=xlDatabase, _
                                                SourceData:=srcData)

    '--- ... then use the Pivot Cache to create the Table
    Set ptSheet = ThisWorkbook.Sheets.Add
    Set thisPivot = ptCache.CreatePivotTable(TableDestination:=ptSheet.Range("A3"))
    Set CreatePivotTable = thisPivot
End Function

```

MSDNピボットテーブルオブジェクト

ピボットテーブルの

これらののれたりファレンスソースは、ピボットテーブルのさまざまなのをします。

- [VBAのピボットテーブルの](#) - Jon PeltierのTech Blogから
- [VBAをしてExcelピボットテーブルのを](#)する - globalconnectからExcel VBA

ピボットテーブルへのフィールドの

ピボットテーブルにフィールドをするときにはすべきな2つの、とです。にはがフィールドのをすることがあるため、これらのパラメータをにすることがになります。これらのアクションは、ピボット・キャッシュではなく、したピボット・テーブルにのみします。

```
Dim thisPivot As PivotTable
Dim ptSheet As Worksheet
Dim ptField As PivotField

Set ptSheet = ThisWorkbook.Sheets("SheetNameWithPivotTable")
Set thisPivot = ptSheet.PivotTables(1)

With thisPivot
    Set ptField = .PivotFields("Gender")
    ptField.Orientation = xlRowField
    ptField.Position = 1
    Set ptField = .PivotFields("LastName")
    ptField.Orientation = xlRowField
    ptField.Position = 2
    Set ptField = .PivotFields("ShirtSize")
    ptField.Orientation = xlColumnField
    ptField.Position = 1
    Set ptField = .AddDataField(.PivotFields("Cost"), "Sum of Cost", xlSum)
    .InGridDropZones = True
    .RowAxisLayout xlTabularRow
End With
```

ピボットテーブルデータの

このでは、したピボットテーブルのデータ `DataBodyRange` にいくつかのをまたはします。 `Range` フォーマットなすべてのパラメータをできます。データのは、ピボット・テーブルにのみし、ピボット・キャッシュにはしません。

`TableStyle` プロパティが `PivotTable` のオブジェクトプロパティのメンバでないため、プロパティは `TableStyle` という `TableStyle2` 。

```
Dim thisPivot As PivotTable
Dim ptSheet As Worksheet
Dim ptField As PivotField

Set ptSheet = ThisWorkbook.Sheets("SheetNameWithPivotTable")
Set thisPivot = ptSheet.PivotTables(1)
```

```
With thisPivot
    .DataBodyRange.NumberFormat = "_($* #,##0.00_);_($* (#,##0.00);_($* \"-\"??_);_(@_)"
    .DataBodyRange.HorizontalAlignment = xlRight
    .ColumnRange.HorizontalAlignment = xlCenter
    .TableStyle2 = "PivotStyleMedium9"
End With
```

オンラインでピボットテーブルをむ <https://riptutorial.com/ja/excel-vba/topic/3797/ピボットテーブル>

20: ファイルシステムオブジェクト

Examples

ファイル、フォルダ、ドライブがする

ファイルがしています

```
Sub FileExists()  
    Dim fso as Scripting.FileSystemObject  
    Set fso = CreateObject("Scripting.FileSystemObject")  
    If fso.FileExists("D:\test.txt") = True Then  
        MsgBox "The file is exists."  
    Else  
        MsgBox "The file isn't exists."  
    End If  
End Sub
```

フォルダがする

```
Sub FolderExists()  
    Dim fso as Scripting.FileSystemObject  
    Set fso = CreateObject("Scripting.FileSystemObject")  
    If fso.FolderExists("D:\testFolder") = True Then  
        MsgBox "The folder is exists."  
    Else  
        MsgBox "The folder isn't exists."  
    End If  
End Sub
```

ドライブがする

```
Sub DriveExists()  
    Dim fso as Scripting.FileSystemObject  
    Set fso = CreateObject("Scripting.FileSystemObject")  
    If fso.DriveExists("D:\") = True Then  
        MsgBox "The drive is exists."  
    Else  
        MsgBox "The drive isn't exists."  
    End If  
End Sub
```

なファイル

コピー

```
Sub CopyFile()  
    Dim fso as Scripting.FileSystemObject  
    Set fso = CreateObject("Scripting.FileSystemObject")  
    fso.CopyFile "c:\Documents and Settings\Makro.txt", "c:\Documents and Settings\Macros\  
End Sub
```

```
Sub MoveFile()  
    Dim fso as Scripting.FileSystemObject  
    Set fso = CreateObject("Scripting.FileSystemObject")  
    fso.MoveFile "c:\*.txt", "c:\Documents and Settings\  
End Sub
```

```
Sub DeleteFile()  
    Dim fso  
    Set fso = CreateObject("Scripting.FileSystemObject")  
    fso.DeleteFile "c:\Documents and Settings\Macros\Makro.txt"  
End Sub
```

なフォルダ

```
Sub CreateFolder()  
    Dim fso as Scripting.FileSystemObject  
    Set fso = CreateObject("Scripting.FileSystemObject")  
    fso.CreateFolder "c:\Documents and Settings\NewFolder"  
End Sub
```

コピー

```
Sub CopyFolder()  
    Dim fso as Scripting.FileSystemObject  
    Set fso = CreateObject("Scripting.FileSystemObject")  
    fso.CopyFolder "C:\Documents and Settings\NewFolder", "C:\  
End Sub
```

```
Sub MoveFolder()  
    Dim fso as Scripting.FileSystemObject  
    Set fso = CreateObject("Scripting.FileSystemObject")  
    fso.MoveFolder "C:\Documents and Settings\NewFolder", "C:\  
End Sub
```

```
Sub DeleteFolder()  
    Dim fso as Scripting.FileSystemObject  
    Set fso = CreateObject("Scripting.FileSystemObject")  
    fso.DeleteFolder "C:\Documents and Settings\NewFolder"  
End Sub
```

そのの

ファイルをする

```
Sub GetFileName()  
    Dim fso as Scripting.FileSystemObject  
    Set fso = CreateObject("Scripting.FileSystemObject")  
    MsgBox fso.GetFileName("c:\Documents and Settings\Makro.txt")  
End Sub
```

Makro.txt

ベースをする

```
Sub GetBaseName()  
    Dim fso as Scripting.FileSystemObject  
    Set fso = CreateObject("Scripting.FileSystemObject")  
    MsgBox fso.GetBaseName("c:\Documents and Settings\Makro.txt")  
End Sub
```

Makro

をする

```
Sub GetExtensionName()  
    Dim fso as Scripting.FileSystemObject  
    Set fso = CreateObject("Scripting.FileSystemObject")  
    MsgBox fso.GetExtensionName("c:\Documents and Settings\Makro.txt")  
End Sub
```

txt

ドライブをする

```
Sub GetDriveName()  
    Dim fso as Scripting.FileSystemObject  
    Set fso = CreateObject("Scripting.FileSystemObject")  
    MsgBox fso.GetDriveName("c:\Documents and Settings\Makro.txt")  
End Sub
```

c

オンラインでファイルシステムオブジェクトをむ [https://riptutorial.com/ja/excel-vba/topic/9933/フ
ファイルシステムオブジェクト](https://riptutorial.com/ja/excel-vba/topic/9933/ファイルシステムオブジェクト)

21: マージされたセル/

Examples

マージされたセル/をするに2える

まず、マージされたセルは、シートのをするためにのみします。

シートとワークブックがにしたら、りにやるべきことです

Merged Rangeのデータはどこにありますか

をマージすると、1つのブロックのみがされます。

データはそののセルにあり、はのセルになります

それについての1つのいマージされたすべてのセルまたはをめるはなく、のセルをめるだけです；

このマージされたレンジののは、にです。

- のまたはをつけるをすると、いくつかのエラーがします
- をループしてみやすくするためにいくつかのをマージした、マージされたでされるではなくのセルがします

オンラインでマージされたセル/をむ <https://riptutorial.com/ja/excel-vba/topic/7308/マージされたセル->

22: マクロをする

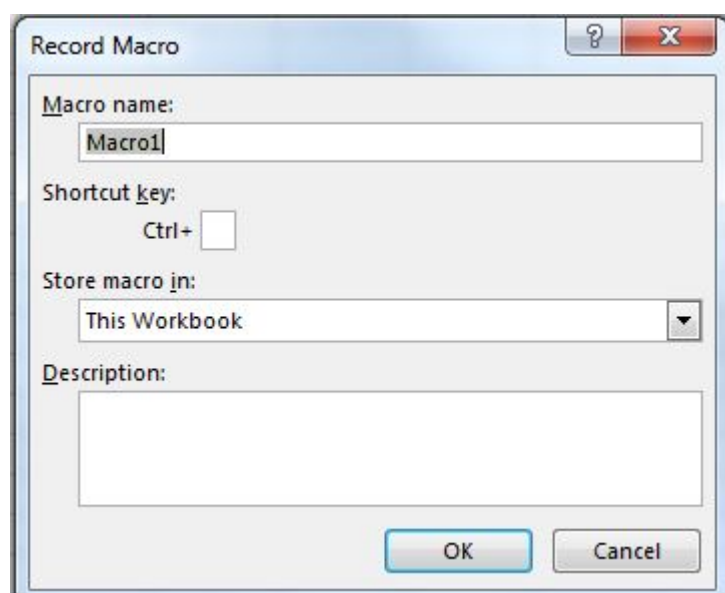
Examples

マクロをする



マクロをするものは、Excelのにあるボタンです。

これをクリックすると、マクロにをけて、ショートカットキーをするかどうかをねるポップアップがされます。また、マクロをするとをねます。のをにできます。スペースはできません。



マクロにりてられたショートカットをすばやくしたいは、えておくをして、マクロをももにりしできます。

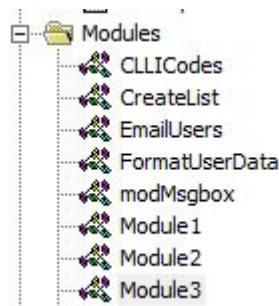
「このブック」、「しいブック」、または「マクロブック」にマクロをできます。しようとして
いるマクロをのワークブックでのみできるようにするには、「このワークブック」をします。しい
ブックにするは、「しいブック」をします。いてるブックにマクロをするは、「マクロブック」
をします。

このポップアップをした、「OK」をクリックします。

に、マクロをしてりしたいアクションをします。したらじボタンをクリックしてをします。これは
はのようになります。



これで、タブにき、Visual Basicをくことができます。 またはAlt + F11を



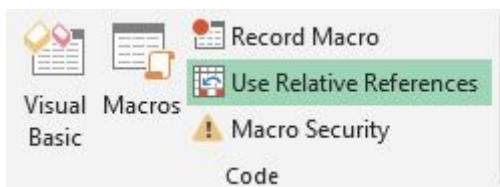
ModulesフォルダのにしいModuleがされます。

のモジュールには、したマクロがまれます。これをダブルクリックしてします。

はなコピーとりけをいました

```
Sub Macro1()  
'  
' Macro1 Macro  
'  
'  
  
    Selection.Copy  
    Range("A12").Select  
    ActiveSheet.Paste  
End Sub
```

に「A12」にペーストしたくないは、「」タブの「の」ボックスをチェックしてをすることがで



きます。

とじにって、マクロをのようにします。

```
Sub Macro2()  
'  
' Macro2 Macro  
'  
'  
  
    Selection.Copy  
    ActiveCell.Offset(11, 0).Range("A1").Select  
    ActiveSheet.Paste  
End Sub
```

まだ "A1"のをセル11にコピーしますが、はセルでじマクロをすることができ、そのセルからののは11のセルにコピーされます。

オンラインでマクロをするをむ <https://riptutorial.com/ja/excel-vba/topic/8204/マクロをする>

23: ユーザーUDF

1. **function functionName argumentVariable dataType**として、argumentVariable2は **dataType**として、オプションのargumentVariable3は**dataType**として **functionReturnDataType**として
の。すべてののはをとしますが、をとるはありません。は0か、えられたのをるかもしれません。をオプションとしてすることもできますをびすときにをしなくてもありません。にをし、にがすデータをすことをおめします。
2. **functionName = theVariableOrValueBeingReturned**
のプログラミングをしているは、Return キーワードにれているがあります。これはVBAではされません。わりに、をします。のまたはされるにできます。ファンクションのりのデータをした、したまたはデータは、そのデータであるがあります。
3.
 - 。Function コードブロックのわりをし、にあるがあります。VBEは、しいをするときにこれをにします。

ユーザーUDFは、ユーザーがしたタスクのをします。ワークシートex =SUM(...) としてびすことも、Sub プロシージャののプロセスにをすこともできます。UDFは、、1つのパラメーターとしてされたからをします。

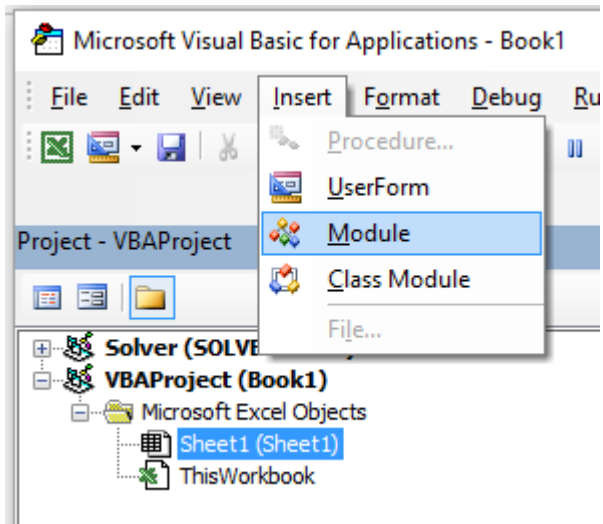
これはのでできます。

1. VBAをします。
2. Excel C APIをする - コンパイルされたをExcelにエクスポートするXLLをします。
3. COM インターフェイスをします。

Examples

UDF - Hello World

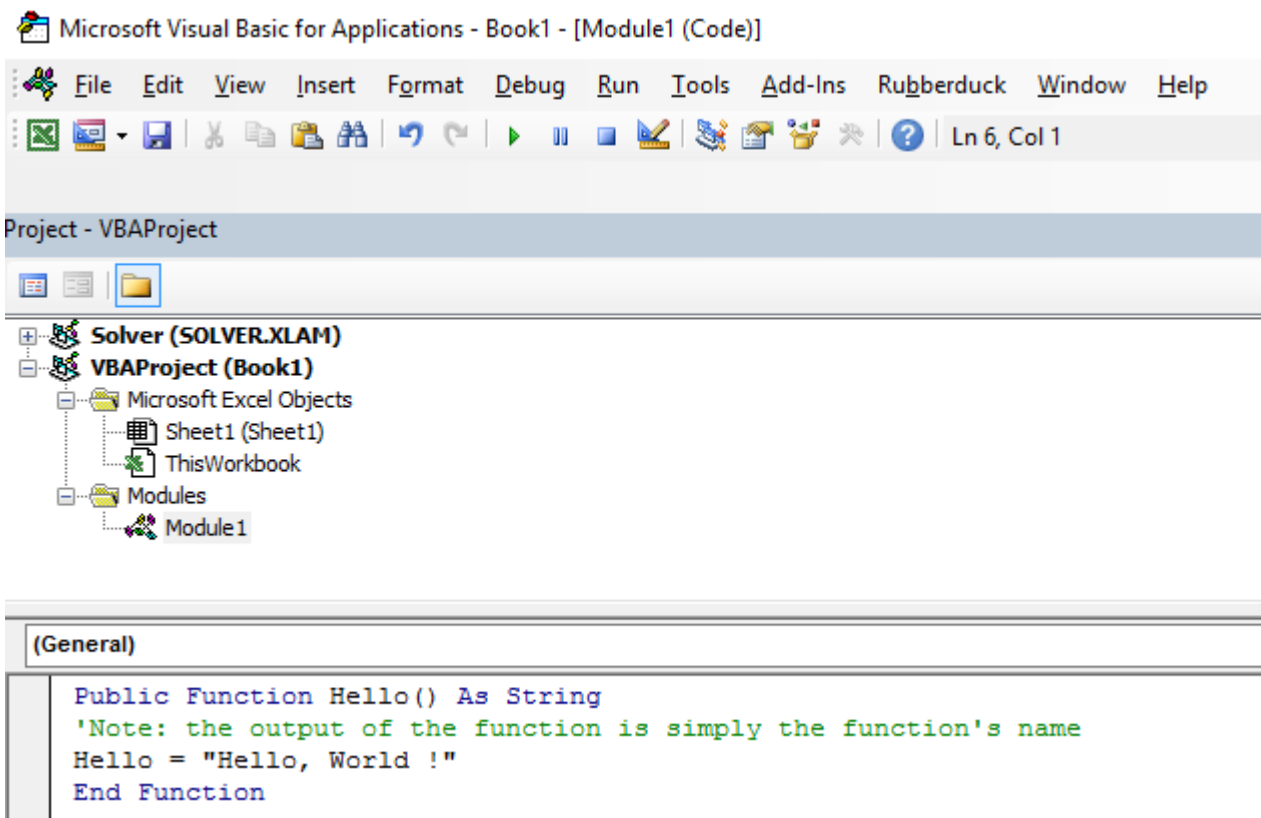
1. オープンExcel
2. Visual Basic EditorをきますVisual Basic Editorをくをしてください
3. [] -> [モジュール]をクリックしてしいモジュールをします。



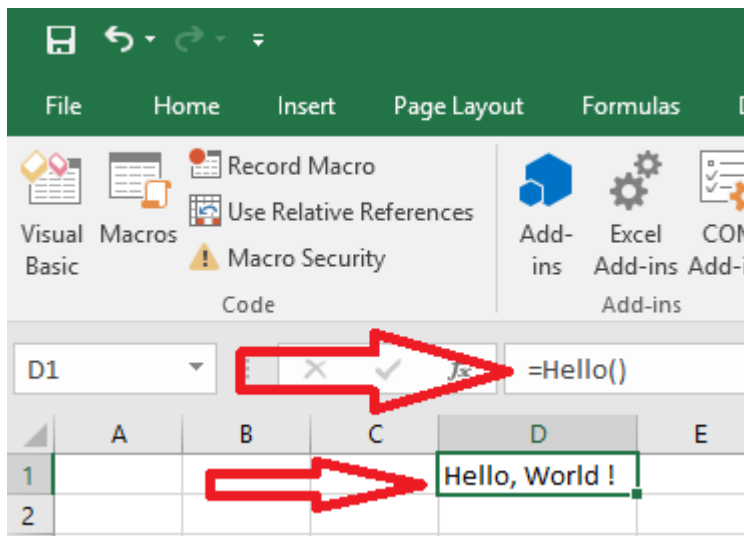
4. のコードをしいモジュールにコピーしてりけます。

```
Public Function Hello() As String
'Note: the output of the function is simply the function's name
Hello = "Hello, World !"
End Function
```

するには



5. ワークブックにつて、セルに "Hello" "Hello"とすると、 "Hello World"がされます。



ペナルティなしでなをする

なをパラメーターとしてすることができれば、ワークシートにUDFをするがです。しかし、コーディングのために、これらのにするループは、ものセルがにであることをしているがあります。これにより、VBAプロジェクトおよびワークブックがフリーズしてしまい、ながされなくなります。

ワークシートのセルをループすることは、タスクをするものの1つですが、ときどきやむをないこともあります。にとされているものにされたをカットすることは、ににかなっています。

このをするには、[Intersectメソッド](#)をして[Worksheet.UsedRangeプロパティ](#)へのなまたはをりてます。のサンプルでは、ワークシートのネイティブSUMIFをやかにするため、`sum_range`のに`criteria_range`のをするがあるため、`criteria_range`も`sum_range`にうようにサイズがされます。

ワークシートでされるUDFの[Application.Caller](#)は、それがするセルです。セルの[.Parent](#)プロパティはワークシートです。これは、`.UsedRange`をするためにされます。

モジュールコードシート

```
Option Explicit

Function udfMySumIf(rngA As Range, rngB As Range, _
    Optional crit As Variant = "yes")
    Dim c As Long, ttl As Double

    With Application.Caller.Parent
        Set rngA = Intersect(rngA, .UsedRange)
        Set rngB = rngB.Resize(rngA.Rows.Count, rngA.Columns.Count)
    End With

    For c = 1 To rngA.Cells.Count
        If IsNumeric(rngA.Cells(c).Value2) Then
            If LCase(rngB(c).Value2) = LCase(crit) Then
                ttl = ttl + rngA.Cells(c).Value2
            End If
        End If
    Next c
End Function
```

```
udfMySumIf = ttl
```

```
End Function
```

```
=udfMySumIf(*sum_range*, *criteria_range*, [*criteria*])
```

E3		fx =udfMySumIf(A:A,B:B, "YES")					
	A	B	C	D	E	F	G
1	numbers	include					
2	17	Yes					
3	L	Maybe			68		
4	17	Maybe					
5	15	Yes					
6	8	Maybe					
7	Y	No					
8	5	No					
9	18	Yes					
10	L	Maybe					
11	A	Yes					
12	J	Maybe					
13	18	Yes					
14	7	No					
15	16	Maybe					
16							
17							

これはになですが、な2のそれぞれ1,048,576をしますが、データとの15しかしないことをにしています。

々のメソッドとプロパティのなMSDNドキュメントをMicrosoft™のとリンクしました。

のユニークなをカウントする

```
Function countUnique(r As range) As Long
    'Application.Volatile False ' optional
    Set r = Intersect(r, r.Worksheet.UsedRange) ' optional if you pass entire rows or columns
    to the function
    Dim c As New Collection, v
    On Error Resume Next ' to ignore the Run-time error 457: "This key is already associated
    with an element of this collection".
    For Each v In r.Value ' remove .Value for ranges with more than one Areas
        c.Add 0, v & ""
    Next
    c.Remove "" ' optional to exclude blank values from the count
    countUnique = c.Count
End Function
```

コレクション

オンラインでユーザーUDFをむ <https://riptutorial.com/ja/excel-vba/topic/1070/ユーザー-udf->

24: よくあるい

Examples

worksheet、rangeまたは々のcellsをする、をにすることがです。

えば

```
ThisWorkbook.Worksheets("Sheet1").Range(Cells(1, 2), Cells(2, 3)).Copy
```

ではありません Cellsには、ワークブックとワークシートがけられていません。ながない、CellsはデフォルトでActiveSheetをします。Sheet1のワークシートがのActiveSheet、このコードはしますったをします。

これをするもなは、WithステートメントをのようにするWithです。

```
With ThisWorkbook.Worksheets("Sheet1")  
    .Range(.Cells(1, 2), .Cells(2, 3)).Copy  
End With
```

あるいは、ワークシートをすることもできます。これは、あるシートからのシートにデータをコピーするなど、コードがのワークシートをするがあるには、もましいです。

```
Dim ws1 As Worksheet  
Set ws1 = ThisWorkbook.Worksheets("Sheet1")  
ws1.Range(ws1.Cells(1, 2), ws1.Cells(2, 3)).Copy
```

のなは、ワークブックをすることなくワークシートコレクションをすることです。えば

```
Worksheets("Sheet1").Copy
```

ワークシートSheet1はされていないし、ブックがありません。コードでのワークブックがされている、これはするがあります。わりに、のいずれかをします。

```
ThisWorkbook.Worksheets("Sheet1")      '<--ThisWorkbook refers to the workbook containing  
                                         'the running VBA code  
Workbooks("Book1").Worksheets("Sheet1") '<--Where Book1 is the workbook containing Sheet1
```

ただし、のはけてください。

```
ActiveWorkbook.Worksheets("Sheet1")      '<--Valid, but if another workbook is activated  
                                         'the reference will be changed
```

にrangeオブジェクトの、にされていない、rangeはアクティブなシートをします


```
Range("A1")
```

じです

```
ActiveSheet.Range("A1")
```

ループのまたはの

ループのまたはをするは、にのわりからループし、すべてのステップにるがあります。コードをする

```
Dim i As Long
With Workbooks("Book1").Worksheets("Sheet1")
    For i = 1 To 4
        If IsEmpty(.Cells(i, 1)) Then .Rows(i).Delete
    Next i
End With
```

あなたはいくつかのをするでしょう。たとえば、コードが3をすると、4は3になります。ただし、`i`は4にされます。この、コードは1をき、にだったのをチェックします。

しいコードは

```
Dim i As Long
With Workbooks("Book1").Worksheets("Sheet1")
    For i = 4 To 1 Step -1
        If IsEmpty(.Cells(i, 1)) Then .Rows(i).Delete
    Next i
End With
```

ActiveWorkbookThisWorkbook

`ActiveWorkbook`と`ThisWorkbook`、オブジェクトがするものをにすることなく、VBAのしいユーザーによってにされることがあります。これは、にましくないをきこすがあります。これらのオブジェクトはどちらも[Applicationオブジェクト](#)にします

`ActiveWorkbook`オブジェクトは、にExcelアプリケーションオブジェクトののビューにあるブックをします。たとえば、このオブジェクトがされたでされ、できるブック

```
Sub ActiveWorkbookExample()

'// Let's assume that 'Other Workbook.xlsx' has "Bar" written in A1.

    ActiveWorkbook.ActiveSheet.Range("A1").Value = "Foo"
    Debug.Print ActiveWorkbook.ActiveSheet.Range("A1").Value '// Prints "Foo"

    Workbooks.Open("C:\Users\BloggsJ\Other Workbook.xlsx")
    Debug.Print ActiveWorkbook.ActiveSheet.Range("A1").Value '// Prints "Bar"
```

```
Workbooks.Add 1
Debug.Print ActiveWorkbook.ActiveSheet.Range("A1").Value '// Prints nothing

End Sub
```

`ThisWorkbook` オブジェクトは、コードがされているときにコードがするワークブックをします。

```
Sub ThisWorkbookExample()

'// Let's assume to begin that this code is in the same workbook that is currently active

ActiveWorkbook.Sheet1.Range("A1").Value = "Foo"
Workbooks.Add 1
ActiveWorkbook.ActiveSheet.Range("A1").Value = "Bar"

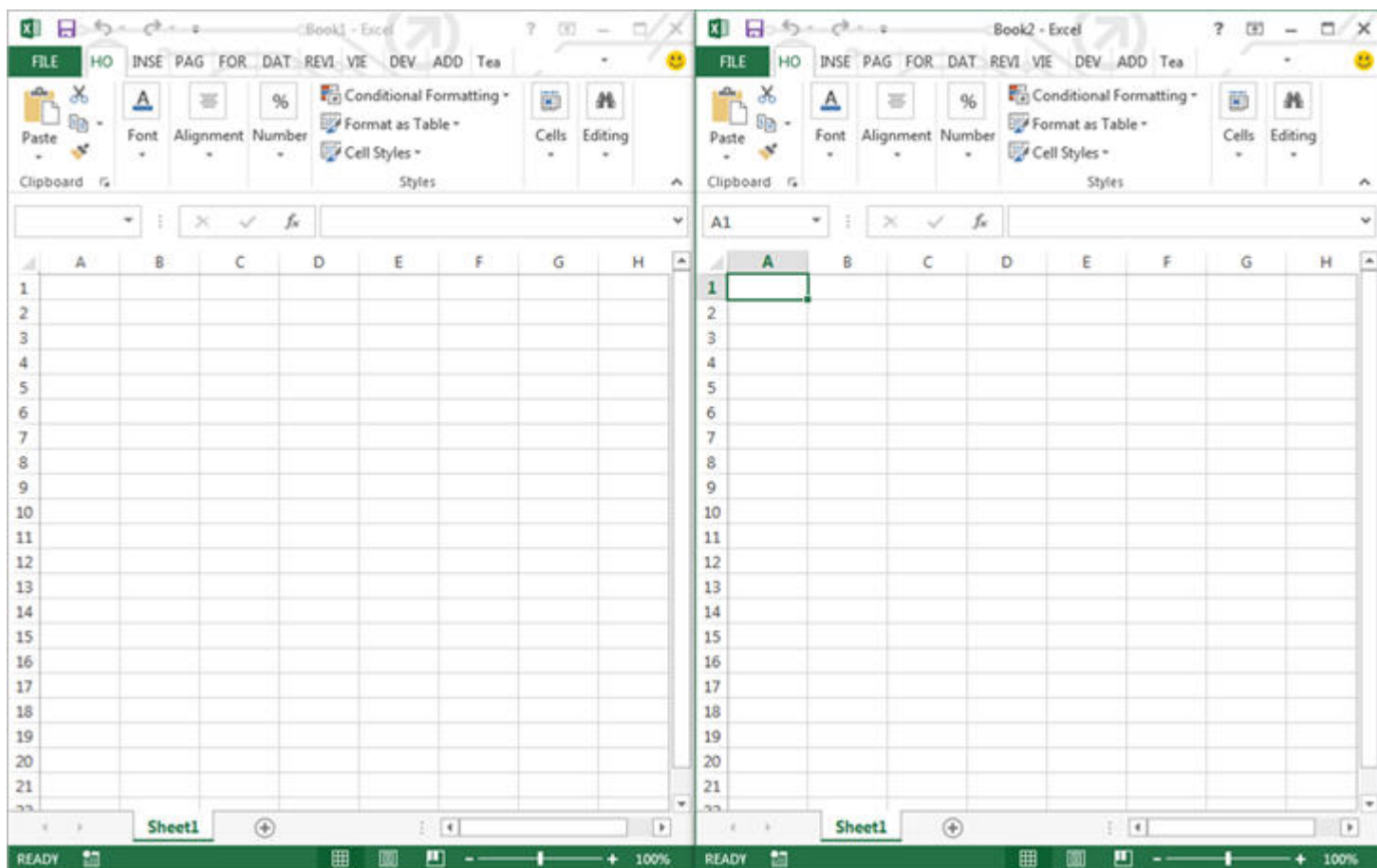
Debug.Print ActiveWorkbook.ActiveSheet.Range("A1").Value '// Prints "Bar"
Debug.Print ThisWorkbook.Sheet1.Range("A1").Value '// Prints "Foo"

End Sub
```

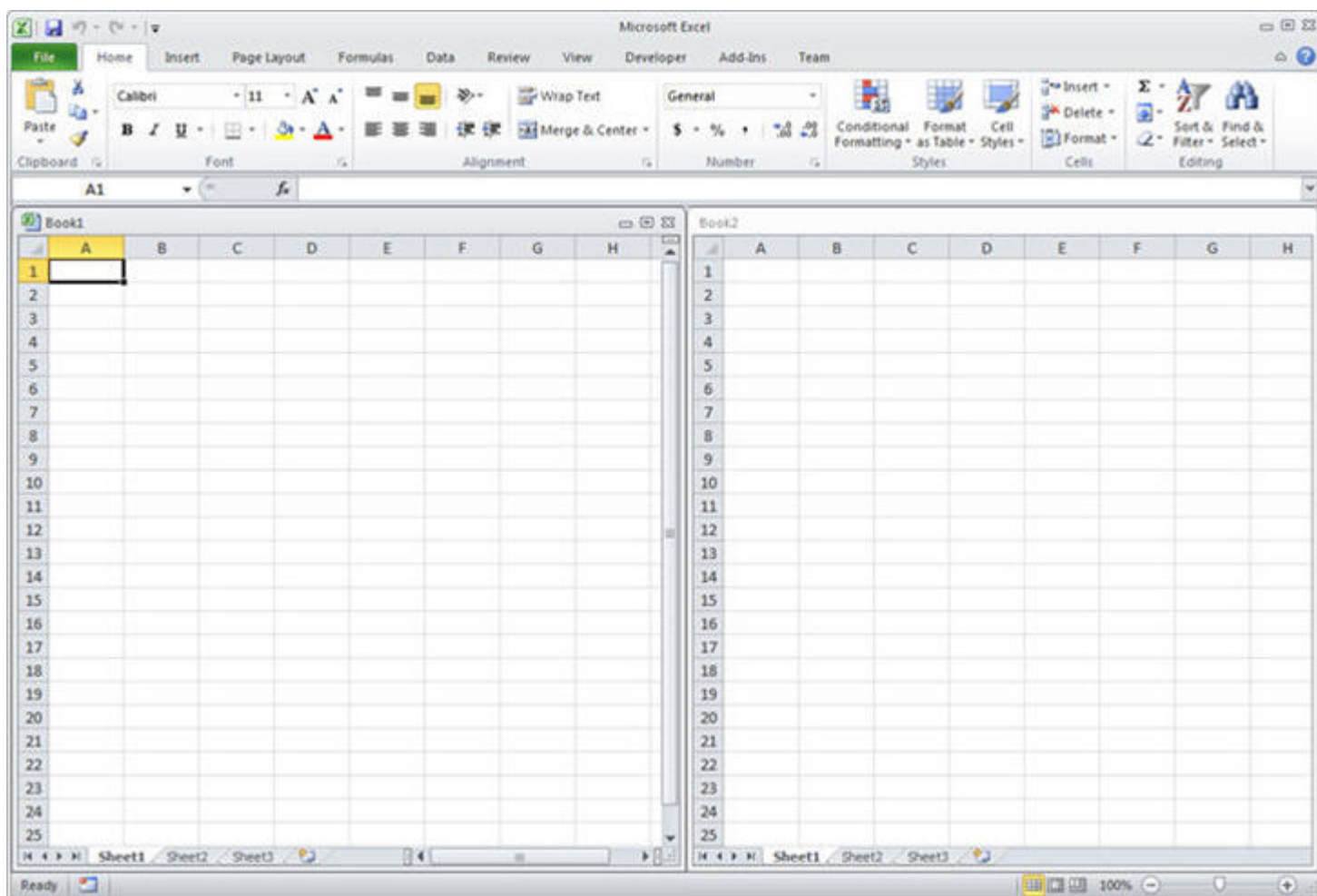
ドキュメント インタフェースドキュメント インタフェース

Microsoft Excel 2013はシングルドキュメントインターフェイスSDIをし、Excel 2010はドキュメントインターフェイスMDIをすることにしてください。

これは、Excel 2013SDIでは、1つのExcelインスタンスのブックにのりボンUIがまれていることをしています。



に、Excel 2010の、ExcelのインスタンスのワークブックはのリボンUIMDIをしていました。



リボンとやりりするVBAコード2010 <-> 2013をするは、いくつかのながします。

Excel 2013のすべてのブックで、じのリボンUIコントロールをするをするがあります。
。

ください

1. すべてのExcelアプリケーションレベルのウィンドウメソッド、イベント、およびプロパティはをけません。 `Application.ActiveWindow`、 `Application.Windows ...`
2. Excel 2013SDIでは、すべてのブックレベルのウィンドウメソッド、イベント、およびプロパティがトップレベルウィンドウでされるようになりました。 `Application.Hwnd`を
`Application.Hwnd`このウィンドウのハンドルをすることはです

については、こののソース[MSDN](#)をしてください。

これは、モードレスユーザーフォームにもいくつかのをきこします。は[こちら](#)をください。

オンラインでよくあるいをむ <https://riptutorial.com/ja/excel-vba/topic/1576/よくあるい>

25: ワークシートのにされたまたはをする

のメソッドがなぜされているのか/なのかについては、ここです <http://stackoverflow.com/a/11169920/4628637>

Examples

ののでないセルをする

ここでは、データセットのののでないをすメソッドをていきます。

このメソッドは、データセットののになくします。

Endメソッドがマージされたにして「」され、マージされたののセルがされるため、マージされたセルがまれるはがです。

さらに、ののでないセルはされません。

```
Sub FindingLastRow()  
    Dim wS As Worksheet, LastRow As Long  
    Set wS = ThisWorkbook.Worksheets("Sheet1")  
  
    'Here we look in Column A  
    LastRow = wS.Cells(wS.Rows.Count, "A").End(xlUp).Row  
    Debug.Print LastRow  
End Sub
```

のにするには、のをします。

```
LastRow = wS.Cells(wS.Rows.Count, "A").End(xlUp).Row
```

のものにきえることができます

1. "Sheet1"にされたの

```
LastRow = wS.UsedRange.Row - 1 + wS.UsedRange.Rows.Count
```

2. "Sheet1"の"A"ののでないセルの

```
Dim i As Long  
For i = LastRow To 1 Step -1  
    If Not (IsEmpty(Cells(i, 1))) Then Exit For  
Next i  
LastRow = i
```

きをしてのをする

シートにきがあり、そのきののをにしたいまた、きののからしないもとなります。

```
Sub FindingLastRow()  
    Dim wS As Worksheet, LastRow As Long  
    Set wS = ThisWorkbook.Worksheets("Sheet1")  
  
    'Here we look in Column A  
    LastRow = wS.Cells(wS.Rows.Count, "A").End(xlUp).Row  
    Debug.Print LastRow  
End Sub
```

```

Dim sht As Worksheet
Dim LastRow As Long
Dim FirstRow As Long

Set sht = ThisWorkbook.Worksheets("form")

'Using Named Range "MyNameRange"
FirstRow = sht.Range("MyNameRange").Row

' in case "MyNameRange" doesn't start at Row 1
LastRow = sht.Range("MyNameRange").Rows.count + FirstRow - 1

End Sub

```

のをするため、なをつきの@Jeepedによってなけがされました。そのにするために、コードはの
ようにされています。

Asumptionstarges sheet = form、き = MyNameRange

```

Sub FindingLastRow()
    Dim rw As Range, rwMax As Long
    For Each rw In Sheets("form").Range("MyNameRange").Rows
        If rw.Row > rwMax Then rwMax = rw.Row
    Next
    MsgBox "Last row of 'MyNameRange' under Sheets 'form': " & rwMax
End Sub

```

ののセルのをする

```

'if only one area (not multiple areas):
With Range("A3:D20")
    Debug.Print .Cells.CountLarge.Row
    Debug.Print .Item.Cells.CountLarge.Row 'using .item is also possible
End With 'Debug prints: 20

'with multiple areas (also works if only one area):
Dim rngArea As Range, LastRow As Long
With Range("A3:D20, E5:I50, H20:R35")
    For Each rngArea In .Areas
        If rngArea.Cells.CountLarge.Row > LastRow Then
            LastRow = rngArea.Cells.CountLarge.Row
        End If
    Next
    Debug.Print LastRow 'Debug prints: 50
End With

```

ワークシートのでないをする

```

Private Sub Get_Last_Used_Row_Index()
    Dim wS As Worksheet

    Set wS = ThisWorkbook.Sheets("Sheet1")
    Debug.Print LastCol_1(wS)
    Debug.Print LastCol_0(wS)

```

```
End Sub
```

ワークシートにデータがないかどうかをりたいたどうかにして、2つのからできます。

- **NOLastCol_1**を `wS.Cells(...,LastCol_1(wS))` でできます
- **YESUse LastCol_0**からられたが0かどうかをテストしてからするがあります

```
Public Function LastCol_1(wS As Worksheet) As Double
    With wS
        If Application.WorksheetFunction.CountA(.Cells) <> 0 Then
            LastCol_1 = .Cells.Find(What:="*", _
                                   After:=.Range("A1"), _
                                   Lookat:=xlPart, _
                                   LookIn:=xlFormulas, _
                                   SearchOrder:=xlByColumns, _
                                   SearchDirection:=xlPrevious, _
                                   MatchCase:=False).Column
        Else
            LastCol_1 = 1
        End If
    End With
End Function
```

Errオブジェクトのプロパティは、がするとにゼロにリセットされます。

```
Public Function LastCol_0(wS As Worksheet) As Double
    On Error Resume Next
    LastCol_0 = wS.Cells.Find(What:="*", _
                             After:=ws.Range("A1"), _
                             Lookat:=xlPart, _
                             LookIn:=xlFormulas, _
                             SearchOrder:=xlByColumns, _
                             SearchDirection:=xlPrevious, _
                             MatchCase:=False).Column
End Function
```

Range.CurrentRegionのセル

`Range.CurrentRegion`は、のセルでまれたのです。=""や=""などのをつのセルはとなされません
`ISBLANK` Excelによっても。

```
Dim rng As Range, lastCell As Range
Set rng = Range("C3").CurrentRegion ' or Set rng = Sheet1.UsedRange.CurrentRegion
Set lastCell = rng(rng.Rows.Count, rng.Columns.Count)
```

ワークシートのものでないをする

```
Private Sub Get_Last_Used_Row_Index()
    Dim wS As Worksheet

    Set wS = ThisWorkbook.Sheets("Sheet1")
    Debug.Print LastRow_1(wS)
    Debug.Print LastRow_0(wS)
```

```
End Sub
```

ワークシートにデータがないかどうかをりたいかどうかにして、2つのからできます。

- **NOLastRow_1**を `wS.Cells(LastRow_1(wS),...)` でできます `wS.Cells(LastRow_1(wS),...)`
- **YESUse LastRow_0**からられたが0かどうかをテストしてからするがあります

```
Public Function LastRow_1(wS As Worksheet) As Double
    With wS
        If Application.WorksheetFunction.CountA(.Cells) <> 0 Then
            LastRow_1 = .Cells.Find(What:="*", _
                                   After:=.Range("A1"), _
                                   Lookat:=xlPart, _
                                   LookIn:=xlFormulas, _
                                   SearchOrder:=xlByRows, _
                                   SearchDirection:=xlPrevious, _
                                   MatchCase:=False).Row
        Else
            LastRow_1 = 1
        End If
    End With
End Function

Public Function LastRow_0(wS As Worksheet) As Double
    On Error Resume Next
    LastRow_0 = wS.Cells.Find(What:="*", _
                              After:=ws.Range("A1"), _
                              Lookat:=xlPart, _
                              LookIn:=xlFormulas, _
                              SearchOrder:=xlByRows, _
                              SearchDirection:=xlPrevious, _
                              MatchCase:=False).Row
End Function
```

ののでないセルをする

このでは、ののでないをすメソッドをていきます。

このメソッドは、データセットののになくします。

Endメソッドがマージされたにして「」され、マージされたののセルがされるため、マージされたセルがまれるはがです。

さらに、のにあるでないセルはされません。

```
Sub FindingLastCol()
    Dim wS As Worksheet, LastCol As Long
    Set wS = ThisWorkbook.Worksheets("Sheet1")

    'Here we look in Row 1
    LastCol = wS.Cells(1, wS.Columns.Count).End(xlToLeft).Column
    Debug.Print LastCol
End Sub
```


ワークシートのものでないセルをつける - パフォーマンス

- をするのははるかにです
- optionalパラメータなしでびされた、デフォルトは.ThisWorkbook.ActiveSheetになります
.ThisWorkbook.ActiveSheet
- かの、Nothingではなく Cell(1, 1)がデフォルトとしてされます

GetMaxCell (Array): Duration: 0.0000790063 seconds

GetMaxCell (Find): Duration: 0.0002903480 seconds

- MicroTimerで

```
Public Function GetLastCell(Optional ByVal ws As Worksheet = Nothing) As Range
    Dim uRng As Range, uArr As Variant, r As Long, c As Long
    Dim ubR As Long, ubC As Long, lRow As Long

    If ws Is Nothing Then Set ws = Application.ThisWorkbook.ActiveSheet
    Set uRng = ws.UsedRange
    uArr = uRng
    If IsEmpty(uArr) Then
        Set GetLastCell = ws.Cells(1, 1): Exit Function
    End If
    If Not IsArray(uArr) Then
        Set GetLastCell = ws.Cells(uRng.Row, uRng.Column): Exit Function
    End If
    ubR = UBound(uArr, 1): ubC = UBound(uArr, 2)
    For r = ubR To 1 Step -1 '----- last row
        For c = ubC To 1 Step -1
            If Not IsError(uArr(r, c)) Then
                If Len(Trim$(uArr(r, c))) > 0 Then
                    lRow = r: Exit For
                End If
            End If
        Next
        If lRow > 0 Then Exit For
    Next
    If lRow = 0 Then lRow = ubR
    For c = ubC To 1 Step -1 '----- last col
        For r = lRow To 1 Step -1
            If Not IsError(uArr(r, c)) Then
                If Len(Trim$(uArr(r, c))) > 0 Then
                    Set GetLastCell = ws.Cells(lRow + uRng.Row - 1, c + uRng.Column - 1)
                    Exit Function
                End If
            End If
        Next
    Next
End Function
```

'Returns last cell (max row & max col) using Find

```
Public Function GetMaxCell2(Optional ByRef rng As Range = Nothing) As Range 'Using Find

    Const NONEMPTY As String = ""

    Dim lRow As Range, lCol As Range
```

```

If rng Is Nothing Then Set rng = Application.ThisWorkbook.ActiveSheet.UsedRange

If WorksheetFunction.CountA(rng) = 0 Then
    Set GetMaxCell12 = rng.Parent.Cells(1, 1)
Else
    With rng
        Set lRow = .Cells.Find(What:=NONEMPTY, LookIn:=xlFormulas, _
                                After:=.Cells(1, 1), _
                                SearchDirection:=xlPrevious, _
                                SearchOrder:=xlByRows)

        If Not lRow Is Nothing Then
            Set lCol = .Cells.Find(What:=NONEMPTY, LookIn:=xlFormulas, _
                                    After:=.Cells(1, 1), _
                                    SearchDirection:=xlPrevious, _
                                    SearchOrder:=xlByColumns)

            Set GetMaxCell12 = .Parent.Cells(lRow.Row, lCol.Column)
        End If
    End With
End If
End Function

```

。

MicroTimer

```

Private Declare PtrSafe Function getFrequency Lib "Kernel32" Alias "QueryPerformanceFrequency"
(cyFrequency As Currency) As Long
Private Declare PtrSafe Function getTickCount Lib "Kernel32" Alias "QueryPerformanceCounter"
(cyTickCount As Currency) As Long

Function MicroTimer() As Double
    Dim cyTicks1 As Currency
    Static cyFrequency As Currency

    MicroTimer = 0
    If cyFrequency = 0 Then getFrequency cyFrequency          'Get frequency
    getTickCount cyTicks1                                     'Get ticks
    If cyFrequency Then MicroTimer = cyTicks1 / cyFrequency 'Returns Seconds
End Function

```

オンラインでワークシートのにされたまたはをするをむ <https://riptutorial.com/ja/excel-vba/topic/918/ワークシートのにされたまたはをする>

26: ワークブック

Examples

アプリケーションワークブック

多くのExcelアプリケーションでは、VBAコードにまれているブックにけられたがわれます。そのブックを ".xlsm" でし、VBAマクロはワークシートのワークシートとデータのみフォーカスします。ただし、のワークブックのデータをまたはマージするがあるや、のワークブックにデータのをきむがあるがあります。くのVBAアプリケーションでは、のブックをいたりじたり、したり、したり、したりすることがなです。

VBAエディタでいつでも、`Application` オブジェクトの `Workbooks` プロパティをして、Excelのそのインスタンスでいてるすべてのワークブックをおよびアクセスできます。 [MSDNのドキュメント](#)では、リファレンスについてしています。

ActiveWorkbookとThisWorkbookをするタイミング

VBAコードがするワークブックをにするためのVBAベストプラクティスです。このをすると、VBAはコードがアクティブなブック `ActiveWorkbook` にけられていると `ActiveWorkbook` ます。

```
'--- the currently active workbook (and worksheet) is implied
Range("A1").value = 3.1415
Cells(1, 1).value = 3.1415
```

ただし、にのワークブックがにいてる、にVBAコードがExcelアドインからされているは、`ActiveWorkbook` へのがしたりったりするがあります。たとえば、をチェックし、アドインのワークシートの1つにされているとするUDFをつアドインはユーザーがにはえないは、どのブックがあるかをにするがありますされています。このでは、いたアクティブなワークブックに `A1` `=EarlyOrLate()` セルのがあり、そのアクティブブックにきまれたVBAはありません。アドインには、のユーザーUDFがあります。

```
Public Function EarlyOrLate() As String
    If Hour(Now) > ThisWorkbook.Sheets("WatchTime").Range("A1") Then
        EarlyOrLate = "It's Late!"
    Else
        EarlyOrLate = "It's Early!"
    End If
End Function
```

UDFのコードは、インストールされたExcelアドインにきまれ、されます。これは、"WatchTime" とばれるアドインのワークシートにされたデータをします。 `ThisWorkbook` が `ThisWorkbook` わりに `ActiveWorkbook` をしていた、どのワークブックがされているかをすることはできません。

にされているでもAワークブックをく

にいているワークブックにアクセスするは、 `Workbooks` コレクションからりてをするのはです。

```
dim myWB as Workbook
Set myWB = Workbooks("UsuallyFullPathnameOfWorkbook.xlsx")
```

しいブックをするは、 `Workbooks` コレクションオブジェクトをしてしいエントリを `Add` します。

```
Dim myNewWB as Workbook
Set myNewWB = Workbooks.Add
```

あなたがとするワークブックがにいているかどうか、またはしないがあるや、またはにしないがあります。このサンプルは、なブックオブジェクトをにすをしています。

```
Option Explicit
Function GetWorkbook(ByVal wbFilename As String) As Workbook
    '--- returns a workbook object for the given filename, including checks
    '    for when the workbook is already open, exists but not open, or
    '    does not yet exist (and must be created)
    '    *** wbFilename must be a fully specified pathname
    Dim folderFile As String
    Dim returnedWB As Workbook

    '--- check if the file exists in the directory location
    folderFile = File(wbFilename)
    If folderFile = "" Then
        '--- the workbook doesn't exist, so create it
        Dim pos1 As Integer
        Dim fileExt As String
        Dim fileFormatNum As Long
        '--- in order to save the workbook correctly, we need to infer which workbook
        '    type the user intended from the file extension
        pos1 = InStrRev(sFullName, ".", , vbTextCompare)
        fileExt = Right(sFullName, Len(sFullName) - pos1)
        Select Case fileExt
            Case "xlsx"
                fileFormatNum = 51
            Case "xlsm"
                fileFormatNum = 52
            Case "xls"
                fileFormatNum = 56
            Case "xlsb"
                fileFormatNum = 50
            Case Else
                Err.Raise vbObjectError + 1000, "GetWorkbook function", _
                    "The file type you've requested (file extension) is not recognized. "
& _
                    "Please use a known extension: xlsx, xlsm, xls, or xlsb."
        End Select
        Set returnedWB = Workbooks.Add
        Application.DisplayAlerts = False
        returnedWB.SaveAs filename:=wbFilename, FileFormat:=fileFormatNum
        Application.DisplayAlerts = True
        Set GetWorkbook = returnedWB
    Else
```

```

'--- the workbook exists in the directory, so check to see if
'    it's already open or not
On Error Resume Next
Set returnedWB = Workbooks(sFile)
If returnedWB Is Nothing Then
    Set returnedWB = Workbooks.Open(sFullName)
End If
End If
End Function

```

ユーザーにせずにワークブックをする

くの、VBAをしてのブックにしいデータをすると、ファイルがにすることをすポップアップがされます。

このポップアップをするには、これらののアラートをするがあります。

```

Application.DisplayAlerts = False      'disable user prompt to overwrite file
myWB.SaveAs FileName:="NewOrExistingFilename.xlsx"
Application.DisplayAlerts = True        're-enable user prompt to overwrite file

```

しいワークブックでのワークシートのデフォルトの

しいExcelワークブックでされるワークシートの「のデフォルト」は、3にされます。あなたのVBAコードは、しいワークブックのワークシートののをにできます。

```

'--- save the current Excel global setting
With Application
    Dim oldSheetsCount As Integer
    oldSheetsCount = .SheetsInNewWorkbook
    Dim myNewWB As Workbook
    .SheetsInNewWorkbook = 1
    Set myNewWB = .Workbooks.Add
    '--- restore the previous setting
    .SheetsInNewWorkbook = oldsheetcount
End With

```

オンラインでワークブックをむ <https://riptutorial.com/ja/excel-vba/topic/2969/ワークブック>

27: き

き

トピックには、されたきを、、、およびアクセスするためのメソッドなど、Excelのきにのがまれているがあります。

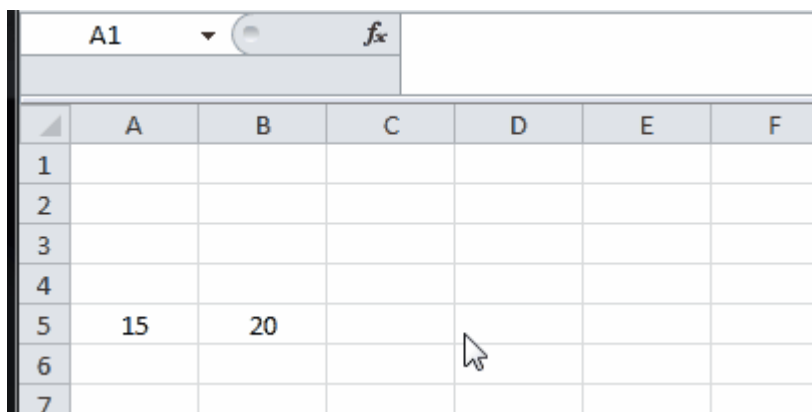
Examples

きをする

きをすると、セルののをし、のセルアドレスのわりにこのみのをできます。

たとえば、`=A5*B5`は、`=Width*Height`にきえて、をみやすくしやすくすることができます。

しいきをするには、をける1つまたはのセルをし、バーののボックスにしいをします。



	A	B	C	D	E	F
1						
2						
3						
4						
5	15	20				
6						
7						

きのデフォルトはグローバルスコープです。つまり、ワークブックのどこからでもアクセスできます。いバージョンのExcelではがしていますので、グローバルスコープのがしないようにするがあります。そうしないとができなくなります。[]タブの[マネージャ]をしてをします。

VBAでのきの

セル_{A1}りてられた 'MyRange' というしいきをする

```
ThisWorkbook.Names.Add Name:="MyRange", _  
    RefersTo:=Worksheets("Sheet1").Range("A1")
```

されたきをでする

```
ThisWorkbook.Names("MyRange").Delete
```

できにアクセスする

```
Dim rng As Range
Set rng = ThisWorkbook.Worksheets("Sheet1").Range("MyRange")
Call MsgBox("Width = " & rng.Value)
```

ショートカットできにアクセスする

の**とに**、**き**には、`Range`オブジェクトをするの**ない**ショートカットをしてアクセスできます。のコードの**3**を**1**に**き**えることができます

```
Call MsgBox("Width = " & [MyRange])
```

`Range`のデフォルトプロパティは**Value**であるため、`[MyRange]`は `[MyRange].Value` とじで**す**

の**メソッド**を**び**すこともできます。は `MyRange` をし `MyRange`

```
[MyRange].Select
```

の**1**つは、ショートカットがVBAライブラリのものでされているではないことです。たとえば、`Width`というのは `[Width]` としてアクセスできませんが、

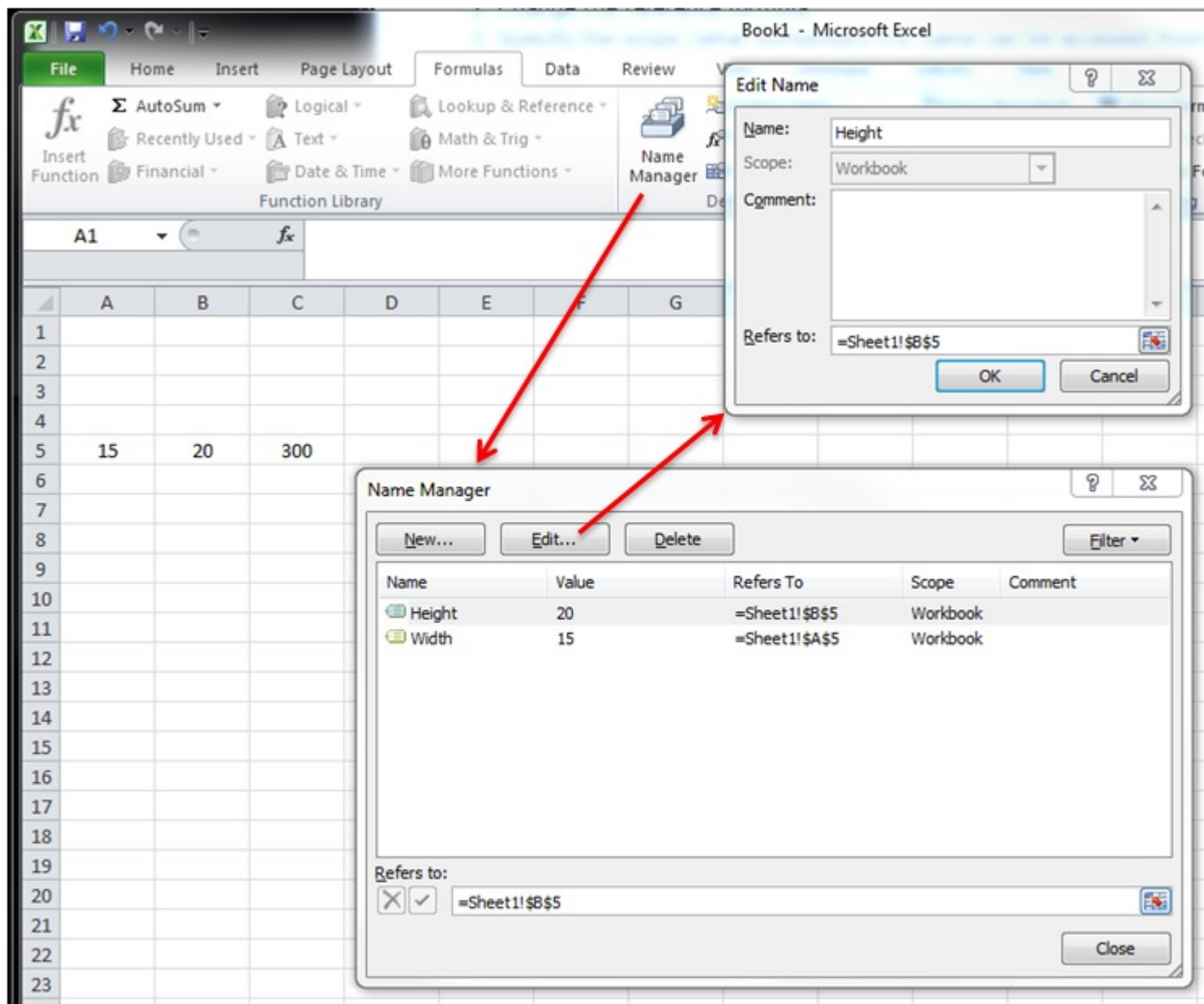
```
ThisWorkbook.Worksheets("Sheet1").Range("Width") からアクセスするとどおりにします
ThisWorkbook.Worksheets("Sheet1").Range("Width")
```

マネージャをしてきをする

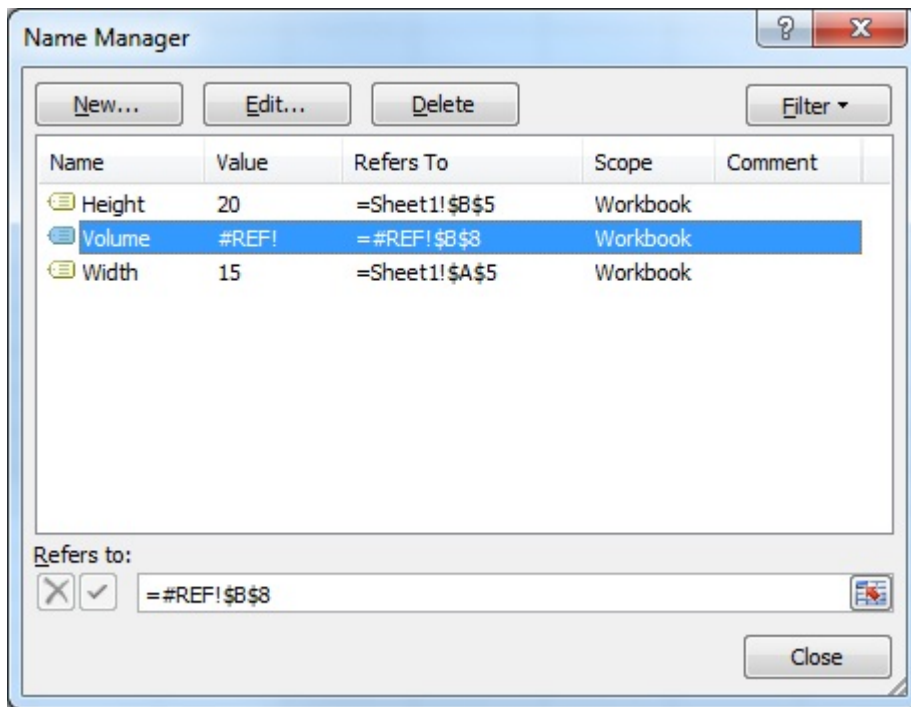
タブ>みのグループ>マネージャボタン

きマネージャーをすると、

1. のまたは
2. セルをまたはする
3. スコープをまたはする
4. のきをする

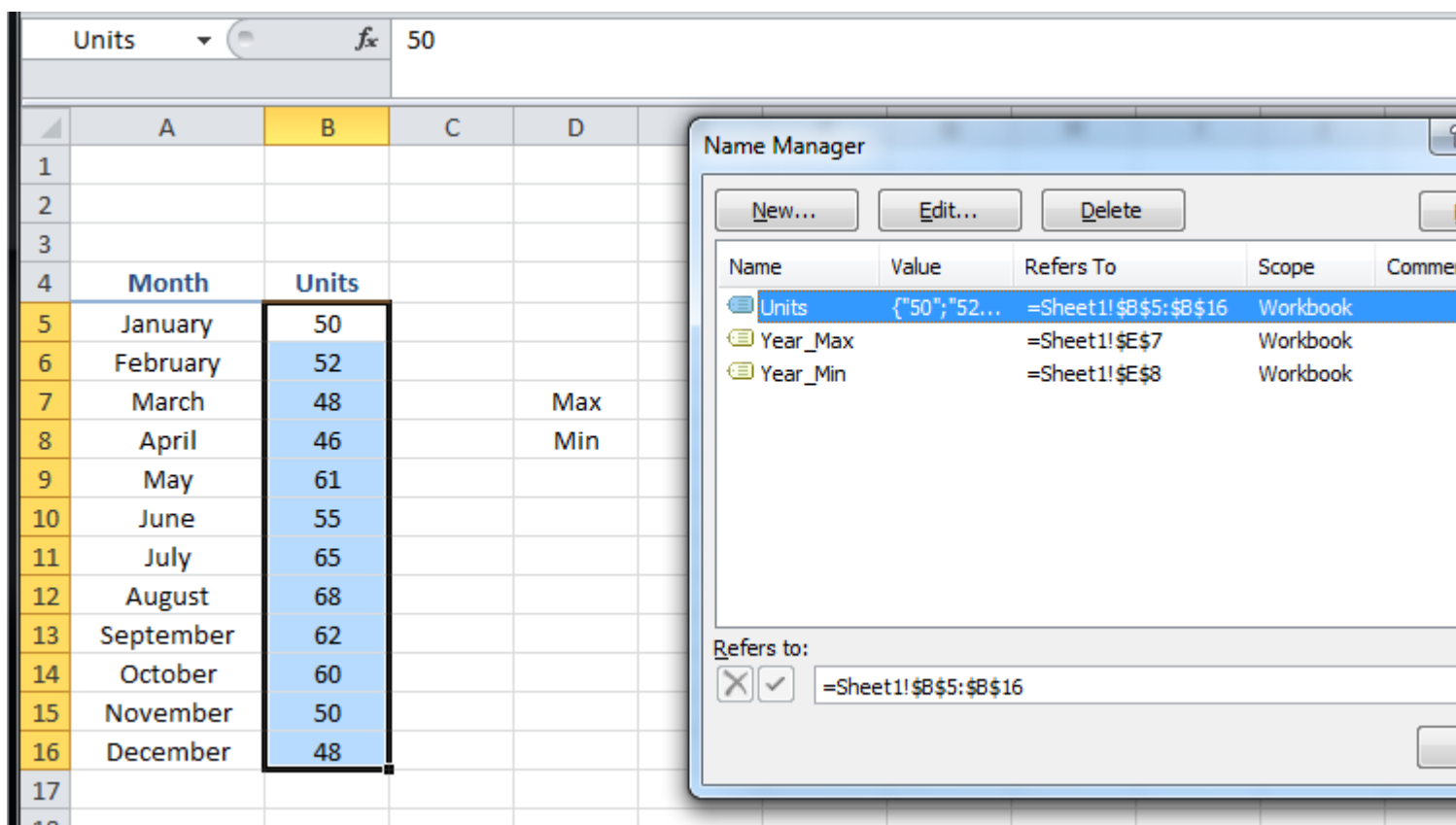


きマネージャは、れたリンクをすばやくにすことができます。



き

サンプルシート



コード

```
Sub Example()
    Dim wks As Worksheet
```

```

Set wks = ThisWorkbook.Worksheets("Sheet1")

Dim units As Range
Set units = ThisWorkbook.Names("Units").RefersToRange

Worksheets("Sheet1").Range("Year_Max").Value = WorksheetFunction.Max(units)
Worksheets("Sheet1").Range("Year_Min").Value = WorksheetFunction.Min(units)
End Sub

```

Month	Units			
January	50			
February	52			
March	48		Max	68
April	46		Min	46
May	61			
June	55			
July	65			
August	68			
September	62			
October	60			
November	50			
December	48			

オンラインできをむ <https://riptutorial.com/ja/excel-vba/topic/8360/>き

28: のCustomDocumentProperties

き

CustomDocumentPropertiesCDPをするのは、じワークブックでなでユーザーのをするのにして
いますが、されていないワークシートにするセルをにすることはしてください*。

CDPはBuiltInDocumentPropertiesにするのコレクションをしますが、コレクションのわりにのユ
ーザープロパティをすることができます。

*または、れたまたは「にれた」ブックにをすることもできます。

Examples

しいをする

をやしてをすることはにわれます。 CustomDocumentPropertiesCDPをすると、じブックになで
そのようなをすることはできますが、されていないシートにするセルをにすることはしてくださ
い。

のヒント

また、しワークシートやいわゆる「にれた」ワークシートにをすることもできます [xlVeryHidden
シートのを](#)してください。もちろん、ファイルiniファイル、csvまたはのタイプまたはレジストリ
。

コンテンツの

のは、

- のをしてすNextInvoiceNo、
- InvoiceのCDPをにするDeleteInvoiceNoプロセス、
- すべてのをむなCDPコレクションをリストするshowAllCDP。 VBAをしていないは、ワーク
ブックのをしてリストすることもできます。プロパティ[DropDown] |なプロパティ|カスタム

プレフィックスのをにするために、のをびすだけでのをしてすることができます。「InvoiceNo
」は、すべてのでCDPとしてにされます。

```
Dim sNumber As String  
sNumber = NextInvoiceNo ()
```

コード

```
Option Explicit
```

```

Sub Test()
    Dim sNumber As String
    sNumber = NextInvoiceNo()
    MsgBox "New Invoice No: " & sNumber, vbInformation, "New Invoice Number"
End Sub

Function NextInvoiceNo() As String
    ' Purpose: a) Set Custom Document Property (CDP) "InvoiceNo" if not yet existing
    '           b) Increment CDP value and return new value as string
    ' Declarations
    Dim prop As Object
    Dim ret As String
    Dim wb As Workbook
    ' Set workbook and CDPs
    Set wb = ThisWorkbook
    Set prop = wb.CustomDocumentProperties

    ' -----
    ' Generate new CDP "InvoiceNo" if not yet existing
    ' -----
    If Not CDPEExists("InvoiceNo") Then
        ' set temporary starting value "0"
        prop.Add "InvoiceNo", False, msoPropertyTypeString, "0"
    End If

    ' -----
    ' Increment invoice no and return function value as string
    ' -----
    ret = Format(Val(prop("InvoiceNo")) + 1, "0")
    ' a) Set CDP "InvoiceNo" = ret
    prop("InvoiceNo").value = ret
    ' b) Return function value
    NextInvoiceNo = ret
End Function

Private Function CDPEExists(sCDPName As String) As Boolean
    ' Purpose: return True if custom document property (CDP) exists
    ' Method: loop thru CustomDocumentProperties collection and check if name parameter exists
    ' Site: cf. http://stackoverflow.com/questions/23917977/alternatives-to-public-variables-in-vba/23918236#23918236
    ' vgl.: https://answers.microsoft.com/en-us/msoffice/forum/msoffice\_word-mso\_other/using-customdocumentproperties-with-vba/91ef15eb-b089-4c9b-a8a7-1685d073fb9f
    ' Declarations
    Dim cdp As Variant ' element of CustomDocumentProperties Collection
    Dim boo As Boolean ' boolean value showing element exists
    For Each cdp In ThisWorkbook.CustomDocumentProperties
        If LCase(cdp.Name) = LCase(sCDPName) Then
            boo = True ' heureka
            Exit For ' exit loop
        End If
    Next
    CDPEExists = boo ' return value to function
End Function

Sub DeleteInvoiceNo()
    ' Declarations
    Dim wb As Workbook
    Dim prop As Object
    ' Set workbook and CDPs

```

```

Set wb = ThisWorkbook
Set prop = wb.CustomDocumentProperties

' -----
' Delete CDP "InvoiceNo"
' -----
If CDPEExists("InvoiceNo") Then
    prop("InvoiceNo").Delete
End If

```

エンドサブ

```

Sub showAllCDPs()
' Purpose: Show all CustomDocumentProperties (CDP) and values (if set)
' Declarations
Dim wb      As Workbook
Dim cdp     As Object

Dim i       As Integer
Dim maxi    As Integer
Dim s       As String
' Set workbook and CDPs
Set wb = ThisWorkbook
Set cdp = wb.CustomDocumentProperties
' Loop thru CDP getting name and value
maxi = cdp.Count
For i = 1 To maxi
    On Error Resume Next ' necessary in case of unset value
    s = s & Chr(i + 96) & ") " & _
        cdp(i).Name & "=" & cdp(i).value & vbCr
Next i
' Show result string
Debug.Print s
End Sub

```

オンラインでのCustomDocumentPropertiesをむ <https://riptutorial.com/ja/excel-vba/topic/10932/のcustomdocumentproperties>

Examples

If ステートメント

If コントロールステートメントでは、きブールステートメントのにじてなるコードをできます。
False、True または False いずれかをするものです $x > 2$ 。

にする If をするにできるパターンは3つあります。If きののに Then がきます。

1. 1つの If をし、それが True あればかをする

If

これは、If をするためであり、True に1つのステートメントだけをするがあるにです。このをするは、すべてのコードを1にするがあります。に End If をめないでください。

```
If [Some condition is True] Then [Do something]
```

If ブロック

True にのコードをするがある If は、If ブロックをできます。

```
If [Some condition is True] Then
  [Do some things]
End If
```

の If ブロックがされている、する End If がであることにしてください。

2. 1つのき If をし、True は1つのをい、False はのをいます

If、Else ステートメント

これは、True のに1つのステートメントがされ、False でなるステートメントがされるにされます。このは、Else ステートメントがあることをにはあまりではないので、してください。このをするは、すべてのコードを1にするがあります。に End If をめないでください。

```
If [Some condition is True] Then [Do something] Else [Do something else]
```

If、Else ブロック

If、Else ブロックをしてコードにをするか、True または False いずれかのでのコードをするがある。

```
If [Some condition is True] Then
```

```
[Do some things]
Else
  [Do some other things]
End If
```

の `If` ブロックがされている、する `End If` がであることをしてください。

3. のステートメントがすべて `False` である、くのステートメントをし、ステートメントごとになるをう

このパターンは、`If` もなであり、なるをとするしないがいにされます。の2つのパターンとはなり、このケースでは、ごとに1のコードしかされないでも、`If` ブロックをするがあります。

`If`、`ElseIf`、...、`Else` ブロック

くの `If` ブロックをのブロックのに1つするわりに、`ElseIf` をしてなをすることが `ElseIf`。 `ElseIf` は、の `If` が `False` `If` のみされ `ElseIf`。

```
If [Some condition is True] Then
  [Do some thing(s)]
ElseIf [Some other condition is True] Then
  [Do some different thing(s)]
Else 'Everything above has evaluated to False
  [Do some other thing(s)]
End If
```

にじて、`If` と `End If` にくの `ElseIf` ステートメントをめることが `ElseIf`。 `ElseIf` をするは `Else` はありませんがされていますが、まれているは、`End If` よりののにするがあります。

オンラインでをむ <https://riptutorial.com/ja/excel-vba/topic/9632/>

30: と

- **Set - Range**などのオブジェクトへのをするためにされる
- **For Each** - コレクションのすべてのアイテムをループするためにされた

`r`、`cell`などはをけてもいませんが、なをけることで、コードがわかりやすくなります。

Examples

をする

とじてRangeをしたり、したりすることはできません。

```
Sub RangeTest()  
    Dim s As String  
    Dim r As Range 'Specific Type of Object, with members like Address, WrapText, AutoFill,  
    etc.  
  
    ' This is how we fill a String:  
    s = "Hello World!"  
  
    ' But we cannot do this for a Range:  
    r = Range("A1") '//Run. Err.: 91 Object variable or With block variable not set//  
  
    ' We have to use the Object approach, using keyword Set:  
    Set r = Range("A1")  
End Sub
```

をするのがベストプラクティスとえられますので、ここからはじアプローチをします。

MSDNのオブジェクトRangeのMSDNのSetステートメントの

じをするさまざまながあります

```
Sub SetRangeVariable()  
    Dim ws As Worksheet  
    Dim r As Range  
  
    Set ws = ThisWorkbook.Worksheets(1) ' The first Worksheet in Workbook with this code in it  
  
    ' These are all equivalent:  
    Set r = ws.Range("A2")  
    Set r = ws.Range("A" & 2)  
    Set r = ws.Cells(2, 1) ' The cell in row number 2, column number 1  
    Set r = ws.[A2] ' Shorthand notation of Range.  
    Set r = Range("NamedRangeInA2") ' If the cell A2 is named NamedRangeInA2. Note, that this  
    is Sheet independent.  
    Set r = ws.Range("A1").Offset(1, 0) ' The cell that is 1 row and 0 columns away from A1  
    Set r = ws.Range("A1").Cells(2,1) ' Similar to Offset. You can "go outside" the original  
    Range.  
  
    Set r = ws.Range("A1:A5").Cells(2) ' Second cell in bigger Range.  
    Set r = ws.Range("A1:A5").Item(2) ' Second cell in bigger Range.
```



```
Set r = ws.Range("A1:A5") (2) 'Second cell in bigger Range.  
End Sub
```

Cell2,1がRange "A2"とであることをしてください。これは、CellがRangeオブジェクトをすためです。

いくつかの [のチップピアソン](#) - ; [MSDN-Rangeオブジェクト](#) 。 [John Walkenback - あなたのVBAコードのをしてください](#) 。

また、のでがされ、がRange "A"2のようにでまれていないは、そのを/ログ。えは

```
Sub RangeIteration()  
    Dim wb As Workbook, ws As Worksheet  
    Dim r As Range  
  
    Set wb = ThisWorkbook  
    Set ws = wb.Worksheets(1)  
  
    For i = 1 To 10  
        Set r = ws.Range("A" & i)  
        ' When i = 1, the result will be Range("A1")  
        ' When i = 2, the result will be Range("A2")  
        ' etc.  
        ' Proof:  
        Debug.Print r.Address  
    Next i  
End Sub
```

ダブルループをしているは、セルがれています。

```
Sub RangeIteration2()  
    Dim wb As Workbook, ws As Worksheet  
    Dim r As Range  
  
    Set wb = ThisWorkbook  
    Set ws = wb.Worksheets(1)  
  
    For i = 1 To 10  
        For j = 1 To 10  
            Set r = ws.Cells(i, j)  
            ' When i = 1 and j = 1, the result will be Range("A1")  
            ' When i = 2 and j = 1, the result will be Range("A2")  
            ' When i = 1 and j = 2, the result will be Range("B1")  
            ' etc.  
            ' Proof:  
            Debug.Print r.Address  
        Next j  
    Next i  
End Sub
```

のセルをする

のExcelワークシートのセルをするもなは、のA1をでむだけです。

```
[a3] = "Hello!"
```

は、Application オブジェクトの Evaluate メソッドの **な**にぎないことにしてください。には、これはのコードとじです。

```
Application.Evaluate("a3") = "Hello!"
```

とをり、セルをす Cells メソッドをびすこともできます。

```
Cells(3, 1).Formula = "=A1+A2"
```

VBA からとを Excel にすたびに、そのがににされ、がにされます。これは、がにされるの A1 のであるためします。

これらのでは、ワークシートをしていないため、Excel はアクティブシートユーザーインターフェイスのにあるシートをします。アクティブなシートをにすることができます

```
ActiveSheet.Cells(3, 1).Formula = "=SUM(A1:A2)"
```

または、のシートのをすることもできます。

```
Sheets("Sheet2").Cells(3, 1).Formula = "=SUM(A1:A2)"
```

あるからのにするためにできるさまざまがあります。たとえば、Rows メソッドをしてのの々のをし、Cells メソッドをしてまたはの々のセルにすることができます。したがって、のコードはセル C1 をします。

```
ActiveSheet.Rows(1).Cells(3).Formula = "hi!"
```

へのセルへのの

のセルへのをするには、Set をするがあります。にをします。

```
Dim R as Range  
Set R = ActiveSheet.Cells(3, 1)
```

...

```
R.Font.Color = RGB(255, 0, 0)
```

Set キーワードがな Set は、_ のがオブジェクトであることを Visual Basic にします。

オフセットプロパティ

- **OffsetRows**、**Columns** - は、のセルからののをにするためにされました。くの、ループでされます。セクションののはにし、のにはにします。のではポジティブがにし、ネガはにします。

すなわち、

```
Private Sub this()  
    ThisWorkbook.Sheets("Sheet1").Range("A1").Offset(1, 1).Select  
    ThisWorkbook.Sheets("Sheet1").Range("A1").Offset(1, 1).Value = "New Value"  
    ActiveCell.Offset(-1, -1).Value = ActiveCell.Value  
    ActiveCell.Value = vbNullString  
End Sub
```

このコードはB2をし、そこにしいをき、そのB2をクリアしたにそのをA1にします。

をトランスポーズすると、またはその

```
Sub TransposeRangeValues()  
    Dim TmpArray() As Variant, FromRange as Range, ToRange as Range  
  
    set FromRange = Sheets("Sheet1").Range("a1:a12")           'Worksheets(1).Range("a1:p1")  
    set ToRange = ThisWorkbook.Sheets("Sheet1").Range("a1")  
    'ThisWorkbook.Sheets("Sheet1").Range("a1")  
  
    TmpArray = Application.Transpose(FromRange.Value)  
    FromRange.Clear  
    ToRange.Resize(FromRange.Columns.Count, FromRange.Rows.Count).Value2 = TmpArray  
End Sub
```

Copy / PasteSpecialには、セルのもにする「りけ」オプションがあります。

オンラインでとをむ <https://riptutorial.com/ja/excel-vba/topic/1503/>と

31:

Examples

をりむをする

をするはあります。

```
'one-dimensional
Dim arrayDirect1D(2) As String
arrayDirect(0) = "A"
arrayDirect(1) = "B"
arrayDirect(2) = "C"

'multi-dimensional (in this case 3D)
Dim arrayDirectMulti(1, 1, 2)
arrayDirectMulti(0, 0, 0) = "A"
arrayDirectMulti(0, 0, 1) = "B"
arrayDirectMulti(0, 0, 2) = "C"
arrayDirectMulti(0, 1, 0) = "D"
'...
```

Arrayの

```
'one-dimensional only
Dim array1D As Variant 'has to be type variant
array1D = Array(1, 2, "A")
'-> array1D(0) = 1, array1D(1) = 2, array1D(2) = "A"
```

から

```
Dim arrayRange As Variant 'has to be type variant

'putting ranges in an array always creates a 2D array (even if only 1 row or column)
'starting at 1 and not 0, first dimension is the row and the second the column
arrayRange = Range("A1:C10").Value
'-> arrayRange(1,1) = value in A1
'-> arrayRange(1,2) = value in B1
'-> arrayRange(5,3) = value in C5
'...

'You can get an one-dimensional array from a range (row or column)
'by using the worksheet functions index and transpose:

'one row from range into 1D-Array:
arrayRange = Application.WorksheetFunction.Index(Range("A1:C10").Value, 3, 0)
'-> row 3 of range into 1D-Array
```

```
'-> arrayRange(1) = value in A3, arrayRange(2) = value in B3, arrayRange(3) = value in C3

'one column into 1D-Array:
'limited to 65536 rows in the column, reason: limit of .Transpose
arrayRange = Application.WorksheetFunction.Index( _
Application.WorksheetFunction.Transpose(Range("A1:C10").Value), 2, 0)
'-> column 2 of range into 1D-Array
'-> arrayRange(1) = value in B1, arrayRange(2) = value in B2, arrayRange(3) = value in B3
'...

'By using Evaluate() - shorthand [] - you can transfer the
'range to an array and change the values at the same time.
'This is equivalent to an array formula in the sheet:
arrayRange = [(A1:C10*3)]
arrayRange = [(A1:C10&"_test")]
arrayRange = [(A1:B10*C1:C10)]
'...
```

Evaluateをした2D

```
Dim array2D As Variant
'[] ist a shorthand for evaluate()
'Arrays defined with evaluate start at 1 not 0
array2D = [{"1A","1B","1C";"2A","2B","3B"}]
'-> array2D(1,1) = "1A", array2D(1,2) = "1B", array2D(2,1) = "2A" ...

'if you want to use a string to fill the 2D-Array:
Dim strValues As String
strValues = "{""1A"", ""1B"", ""1C""; ""2A"", ""2B"", ""2C""}"
array2D = Evaluate(strValues)
```

Splitをする

```
Dim arraySplit As Variant 'has to be type variant
arraySplit = Split("a,b,c", ",")
'-> arraySplit(0) = "a", arraySplit(1) = "b", arraySplit(2) = "c"
```

のサイズと

Excel-VBAのコンテンツではないため、これはVBAドキュメントにされました。

リンク [ダイナミックアレイのサイズとダイナミックハンドリング](#)

ジグザグの

Excel-VBAのコンテンツではないため、これはVBAドキュメントにされました。

リンク [ぎざぎざの](#)

がされているかどうかをしますがまれているかどうか。

なは、をたないArrayをしようとしているがあります。えは

```
Dim myArray() As Integer
For i = 0 To UBound(myArray) 'Will result in a "Subscript Out of Range" error
```

このをし、にがまれているかどうかをするには、のonelinerをします。

```
If Not Not myArray Then MsgBox UBound(myArray) Else MsgBox "myArray not initialised"
```

[、サイズ]

```
Sub Array_clarity()

Dim arr() As Variant 'creates an empty array
Dim x As Long
Dim y As Long

x = Range("A1", Range("A1").End(xlDown)).Cells.Count
y = Range("A1", Range("A1").End(xlToRight)).Cells.Count

ReDim arr(0 To x, 0 To y) 'fixing the size of the array

For x = LBound(arr, 1) To UBound(arr, 1)
    For y = LBound(arr, 2) To UBound(arr, 2)
        arr(x, y) = Range("A1").Offset(x, y) 'storing the value of Range("A1:E10") from
        activesheet in x and y variables
    Next
Next

'Put it on the same sheet according to the declaration:
Range("A14").Resize(UBound(arr, 1), UBound(arr, 2)).Value = arr

End Sub
```

オンラインでをむ <https://riptutorial.com/ja/excel-vba/topic/2027/>

クレジット

S. No		Contributors
1	excel-vbaをいめる	Branislav Kollár , chris neilsen , Cody G. , Comintern , Community , Doug Coats , EEM , Gordon Bell , Jeeped , Joel Spolsky , Kaz , Laurel , LucyMarieJ , Macro Man , Malick , Maxime Porté , Regis , RGA , Ron McMahon , SandPiper , Shai Rado , Taylor Ostberg , whytheq
2	Excel VBAのSQL - ベストプラクティス	Zsmaster
3	Excel VBAのヒントとテクニック	Andre Terra , Cody G. , Jeeped , Kumar Sourav , Macro Man , RGA
4	Excel-VBAの	Masoud , paul bica , T.M.
5	Sheetオブジェクトではなく、Worksheetオブジェクトをする	Vityata
6	VBAでのExcelテーブルの	Excel Developers
7	VBAによるPowerPointの	mnoronha , RGA
8	VBAのセキュリティ	Chel , TheGuyThatDoesn'tKnowMuch
9	VBAベストプラクティス	Alexis Olson , Branislav Kollár , Chel , Cody G. , Comintern , EEM , FreeMan , genespos , Hubisan , Huzaifa Essajee , Jeeped , JKAbrams , Kumar Sourav , Kyle , Macro Man , Malick , Máté Juhász , Munkeeface , paul bica , Peh , PeterT , Portland Runner , RGA , Shai Rado , Stefan Pinnow , Steven Schroeder , Taylor Ostberg , ThunderFrame , Verzweifler , Vityata
10	VBAをしたき	Zsmaster
11	アクティブブックのすべてのシートをループする	Doug Coats , Shai Rado
12	アプリケーションオブジェクト	Captain Grumpy , Joel Spolsky
13	あるのするの	quadrature , T.M.

14	オートフィルター;とベスト プラクティス	Sgdva
15	コンボボックスでアクティ ブワークシートのドロップ ダウンメニューをする	Macro Man , quadrature , R3uK
16	チャートとチャート	Byron Wall
17	デバッグとトラブルシュー ティング	Cody G. , Etheur , Gregor y , Julian Kuchlbauer , Kyle , Malick , Michael Russo , RGA , Ron McMahon , Slai , Steven Schroeder , Taylor Ostberg
18	バインディング	Captain Grumpy , EEM , Jeeped , jlookup , Malick , Raystafarian
19	ピボットテーブル	PeterT
20	ファイルシステムオブジェ クト	Zsmaster
21	マージされたセル/	R3uK
22	マクロをする	Mike , Robby
23	ユーザーUDF	Jeeped , Malick , Slai , user3561813 , Vegard
24	よくあるい	Egan Wolf , Gordon Bell , Macro Man , Malick , Peh , SWa , Taylor Ostberg
25	ワークシートのにされたま たはをする	curious , Hubisan , Máté Juhász , Michael Russo , Miqi180 , paul bica , R3uK , Raystafarian , RGA , Shai Rado , Slai , Thomas Inzina , YowE3K
26	ワークブック	PeterT
27	き	Andre Terra , Portland Runner
28	の CustomDocumentProperties	T.M.
29		SteveES
30	と	Adam , Branislav Kollár , Doug Coats , Gregor y , Jbjstam , Joel Spolsky , Julian Kuchlbauer , Máté Juhász , Miguel_Ryu , Patrick Wynne , Vegard
31		Alon Adler , Hubisan , Miguel_Ryu , Shahin