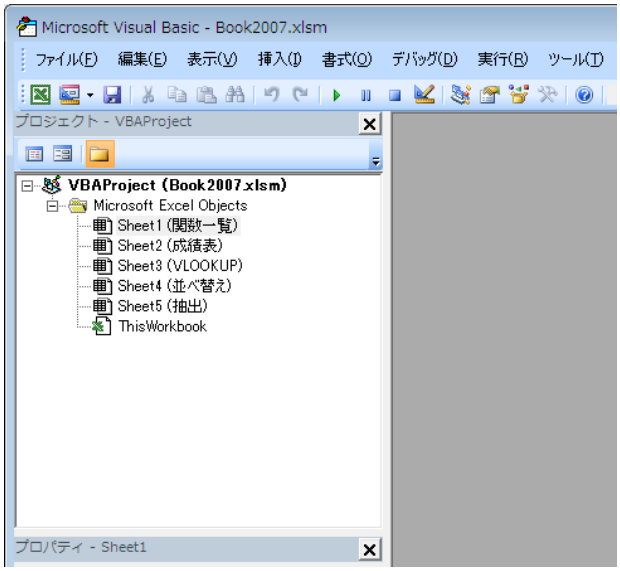
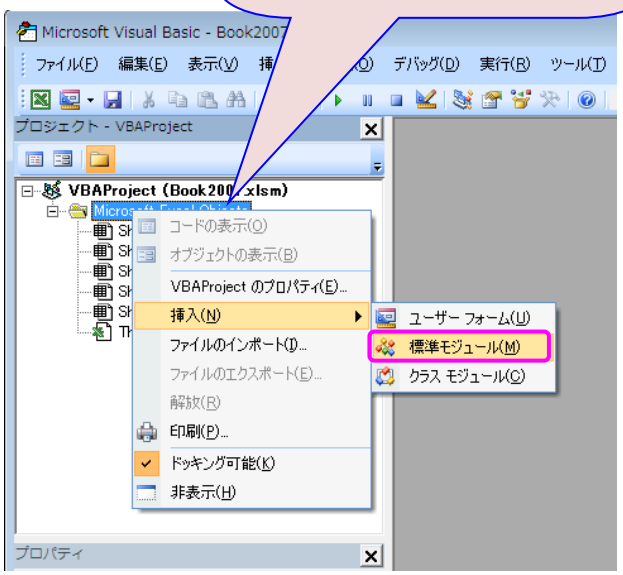
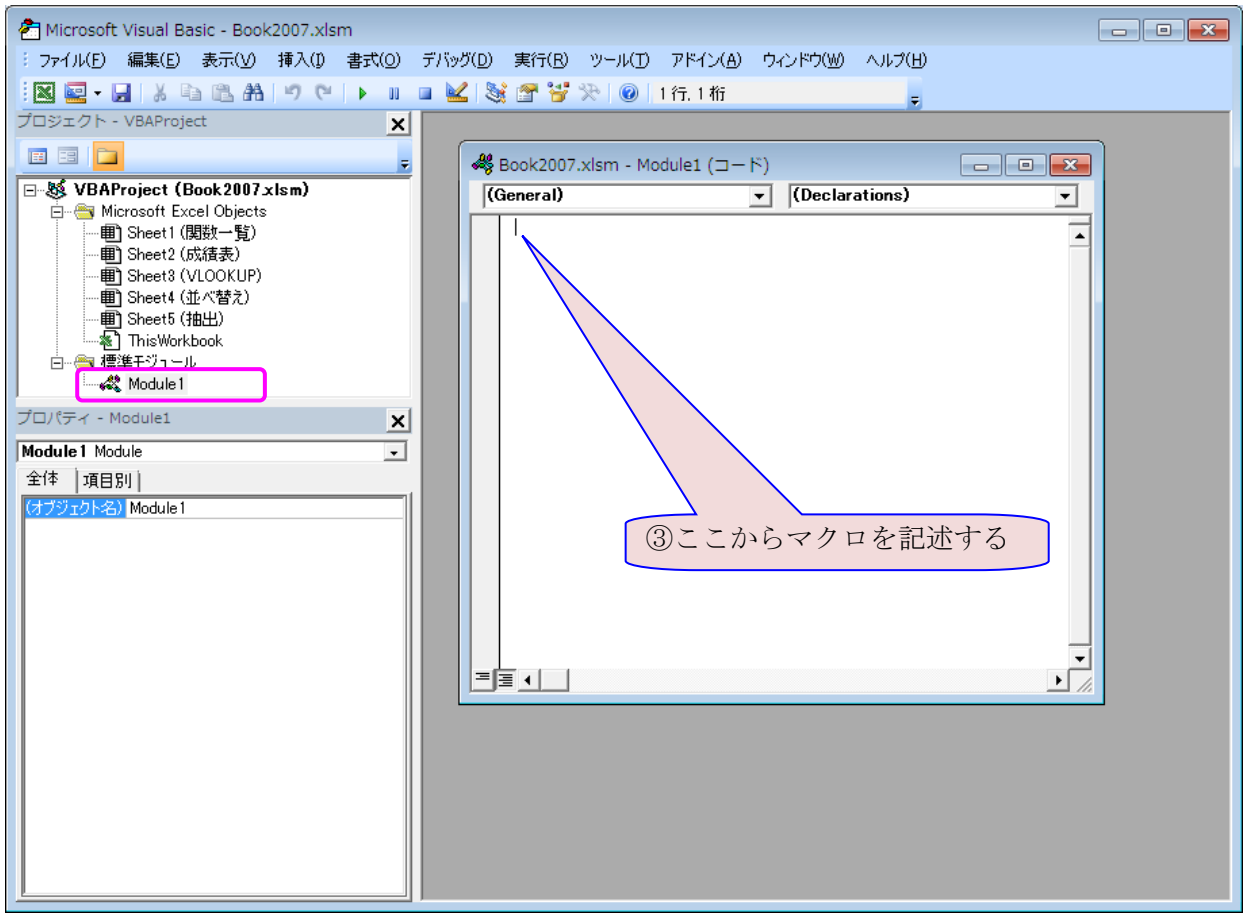
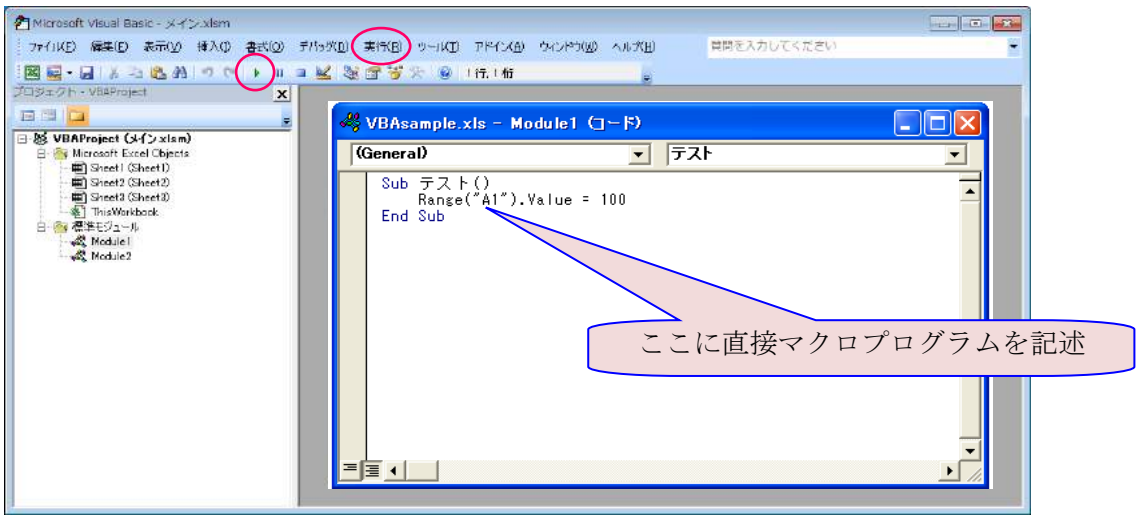
3．Visual Basic を起動して自分でマクロ作成する方法









（例１）

```
Sub テスト()

    Range("A1").Value = 100

End Sub
```

このマクロは、
EXCEL のセル A1 に 100 という数字を入力する

（例２）

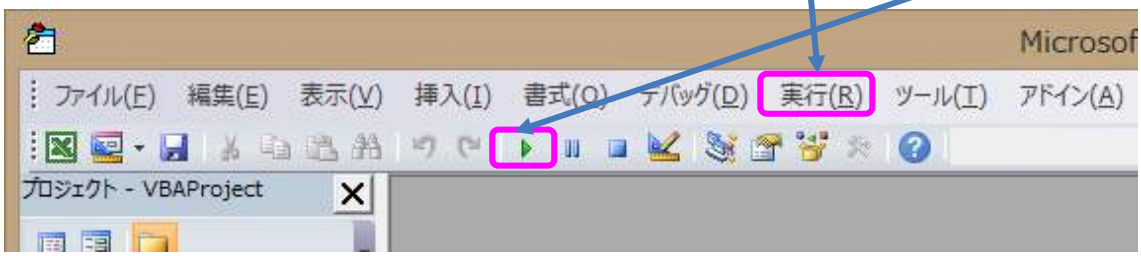
```
Sub テスト()

    MsgBox ("ようこそ VisualBASIC へ！！")

End Sub
```

このマクロは、
「ようこそ VisualBASIC へ！！」という文字列を表示する

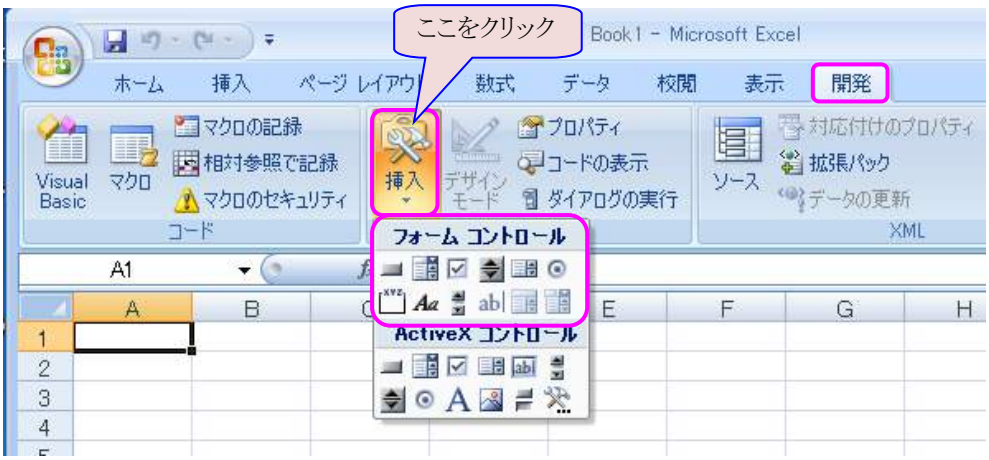
注意）ここでそれぞれのマクロを試しに実行するには、メニューの実行や実行ボタンを押す



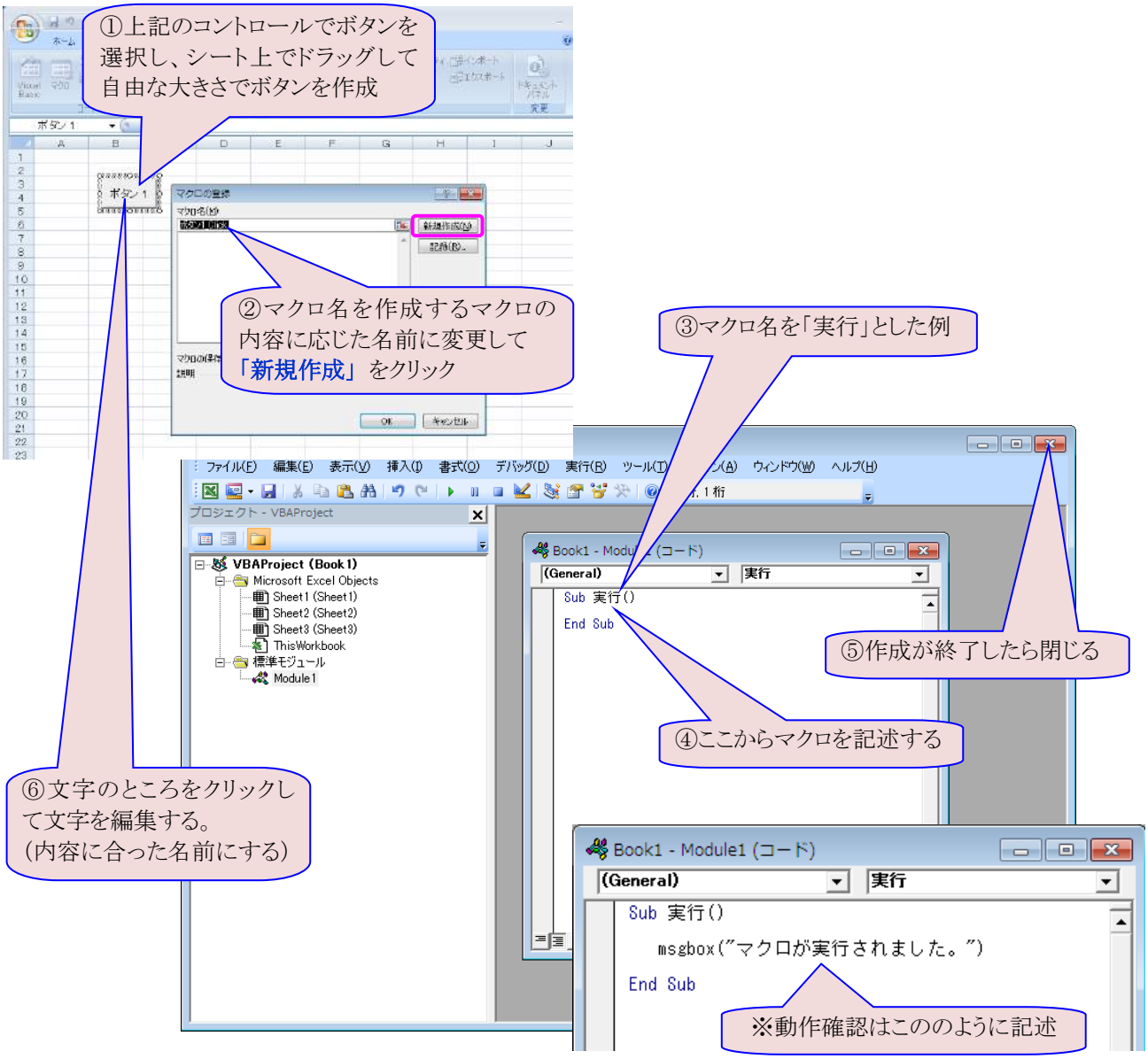
４．コントロールツールボックスからボタンなどを配置してマクロを作成する

（2003 以前：参考）　メニューの　［表示］－［ツールバー］－［コントロールツールボックス］

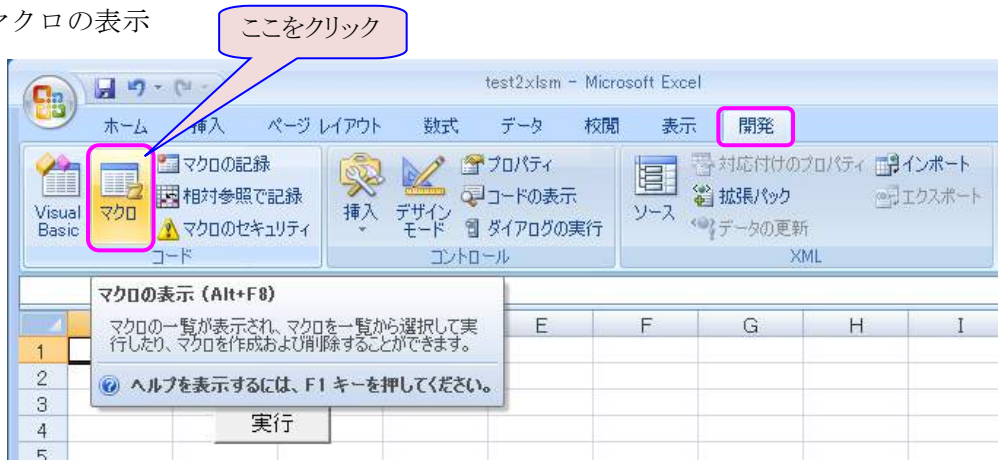
（2007 以降）　**「挿入」**をクリック→「フォームコントロール」

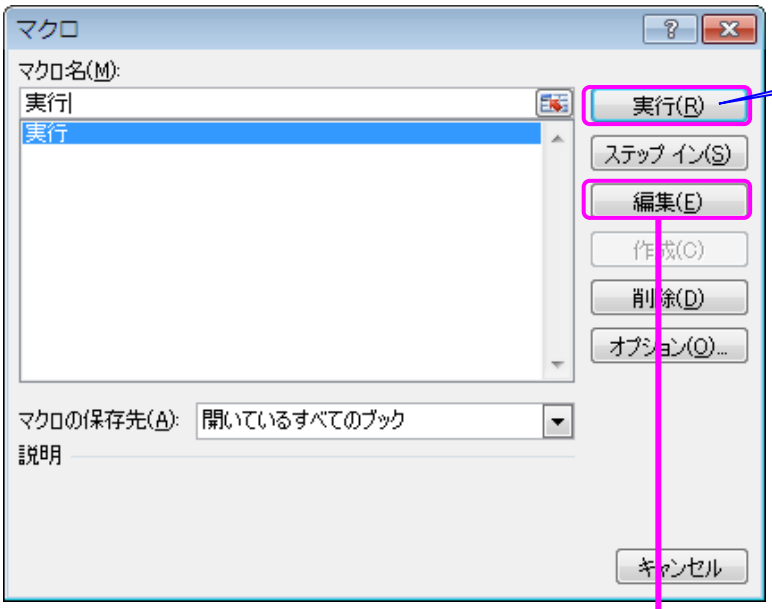


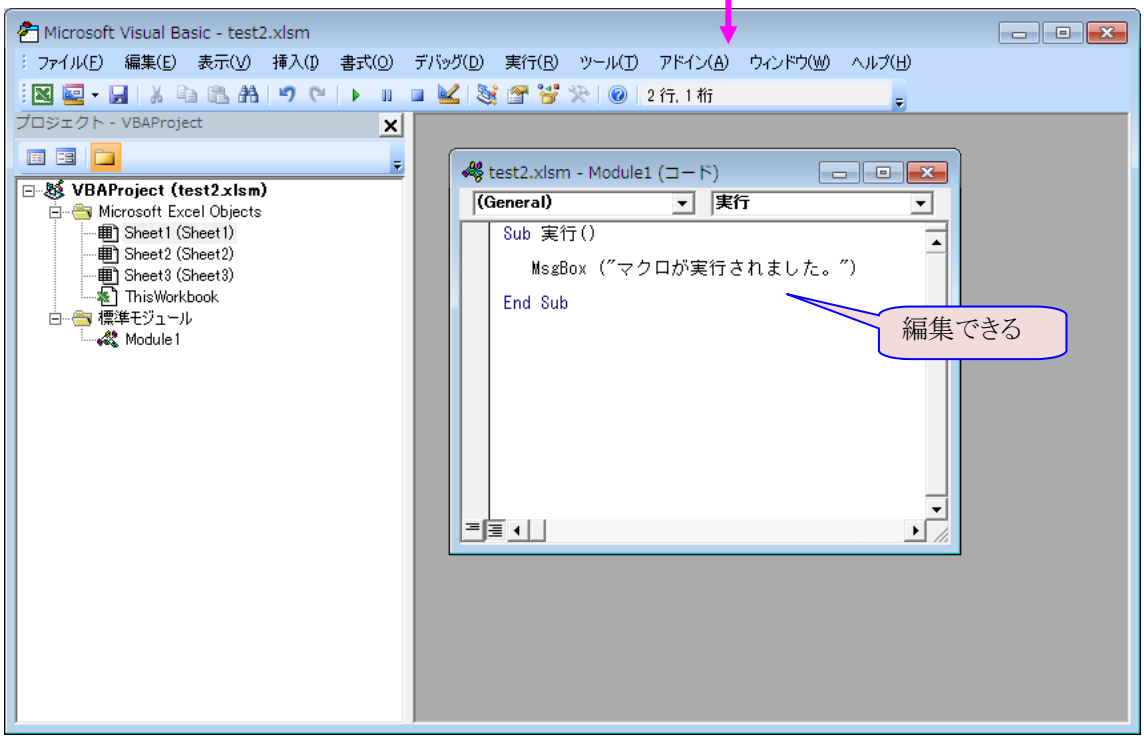
●ボタンのセットの例



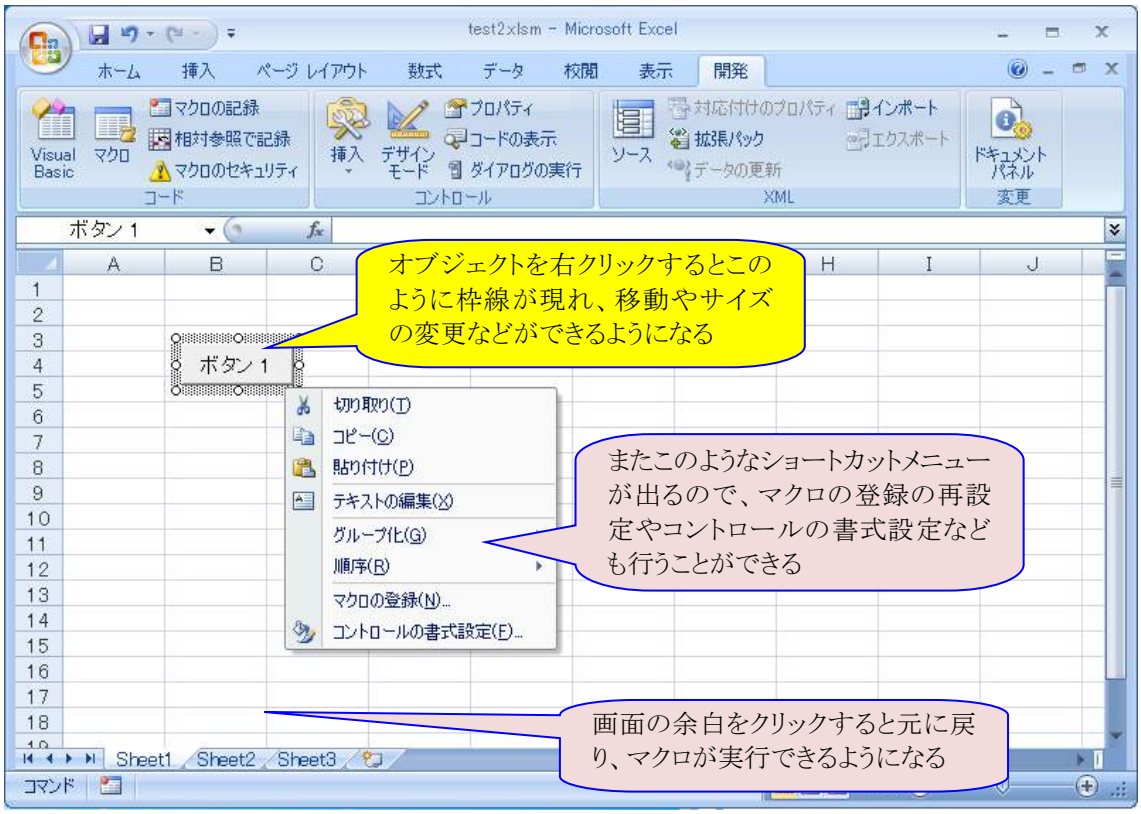
●マクロの表示


ここをクリック
test2.xlsm - Microsoft Excel
ホーム
挿入
ページ レイアウト
数式
データ
校閲
表示
開発
Visual Basic
マクロ
マクロの記録
相対参照で記録
マクロのセキュリティ
コード
挿入
デザイン モード
プロパティ
コードの表示
ダイアログの実行
コントロール
ソース
対応付けのプロパティ
拡張パック
データの更新
インポート
エクスポート
XML
マクロの表示 (Alt+F8)
マクロの一覧が表示され、マクロを一覧から選択して実行したり、マクロを作成および削除することができます。
ヘルプを表示するには、F1 キーを押してください。
実行

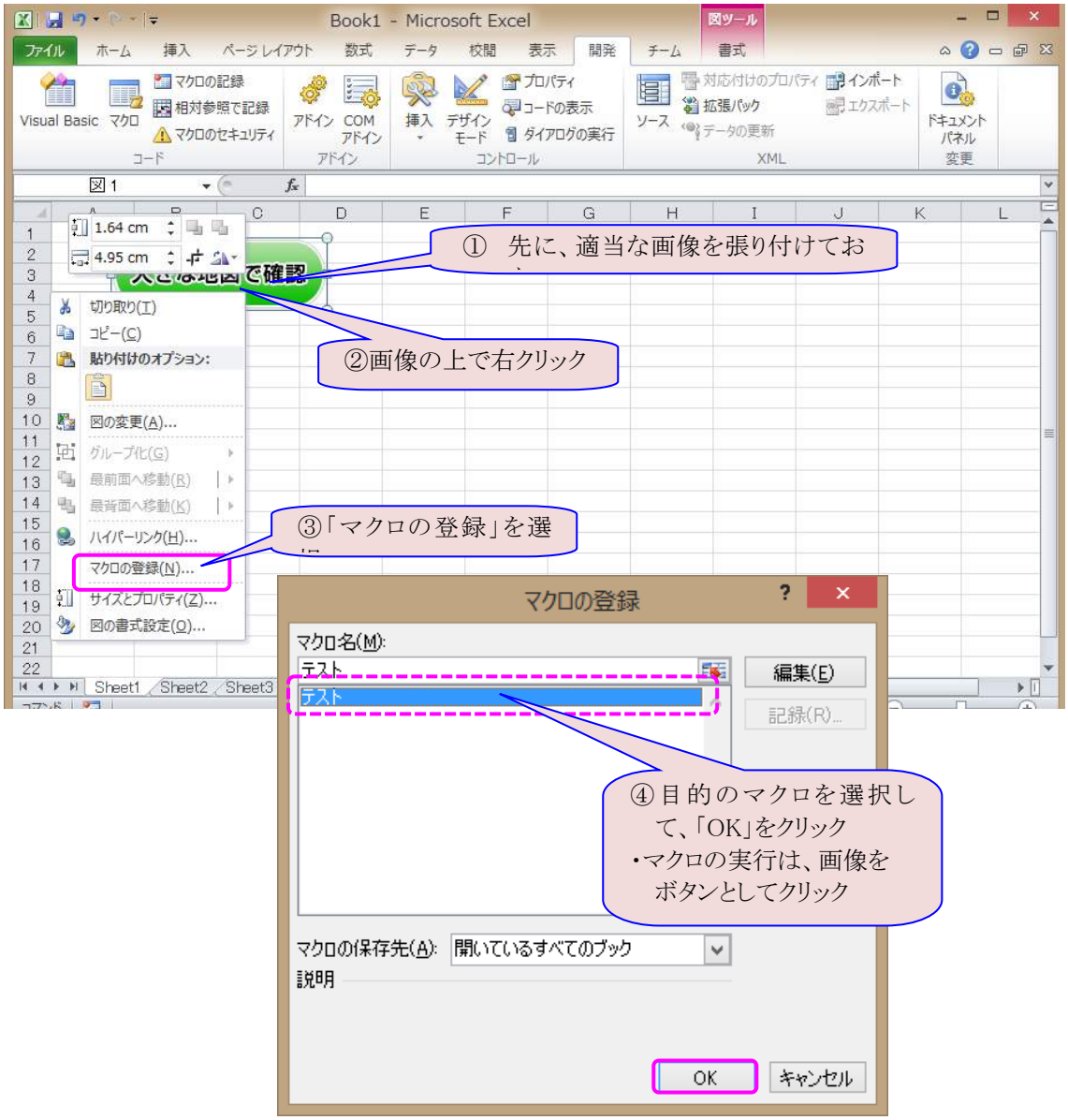

マクロ
マクロ名(M):
実行
実行
実行(R)
ステップ イン(S)
編集(E)
作成(C)
削除(D)
オプション(O)...
マクロの保存先(A):
開いているすべてのブック
説明
キャンセル
実行もできる


Microsoft Visual Basic - test2.xlsm
ファイル(F)
編集(E)
表示(V)
挿入(I)
書式(O)
デバッグ(D)
実行(R)
ツール(T)
アドイン(A)
ウィンドウ(W)
ヘルプ(H)
2 行, 1 桁
プロジェクト - VBAProject
VBAProject (test2.xlsm)
Microsoft Excel Objects
Sheet1 (Sheet1)
Sheet2 (Sheet2)
Sheet3 (Sheet3)
ThisWorkbook
標準モジュール
Module1
test2.xlsm - Module1 (コード)
(General)
実行
Sub 実行()
MsgBox ("マクロが実行されました。")
End Sub
編集できる

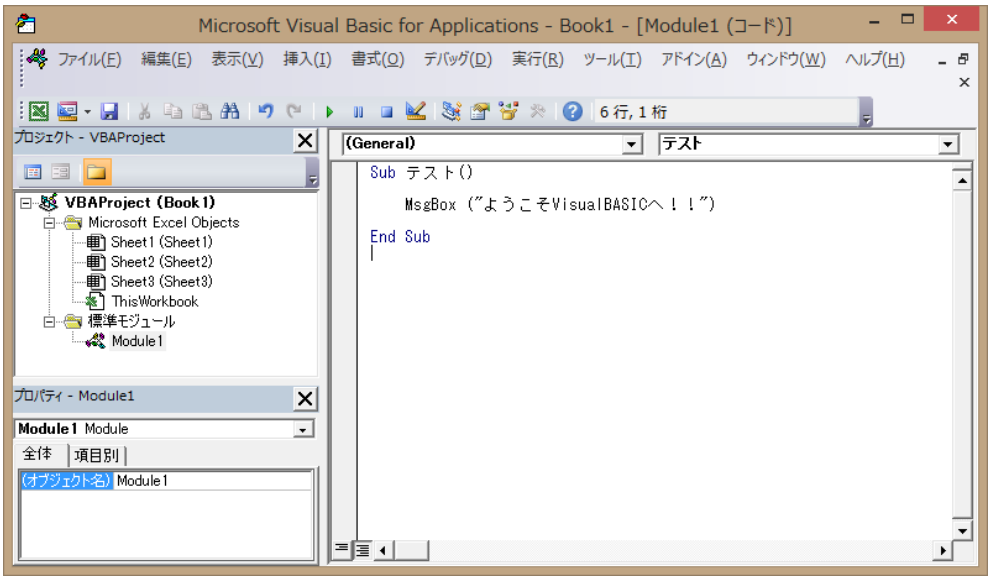
●オブジェクトの編集


test2xlsm - Microsoft Excel
ホーム
挿入
ページ レイアウト
数式
データ
校閲
表示
開発
Visual Basic
マクロ
マクロの記録
相対参照で記録
マクロのセキュリティ
コード
挿入
デザイン モード
プロパティ
コードの表示
ダイアログの実行
コントロール
ソース
対応付けのプロパティ
拡張パック
データの更新
インポート
エクスポート
XML
ドキュメント パネル
変更
ボタン 1
オブジェクトを右クリックするとこのように枠線が現れ、移動やサイズの変更などができるようになる
ボタン 1
切り取り(T)
コピー(C)
貼り付け(P)
テキストの編集(X)
グループ化(G)
順序(R)
マクロの登録(N)...
コントロールの書式設定(F)...
またこのようなショートカットメニューが出るので、マクロの登録の再設定やコントロールの書式設定なども行うことができる
画面の余白をクリックすると元に戻り、マクロが実行できるようになる
Sheet1
Sheet2
Sheet3
コマンド

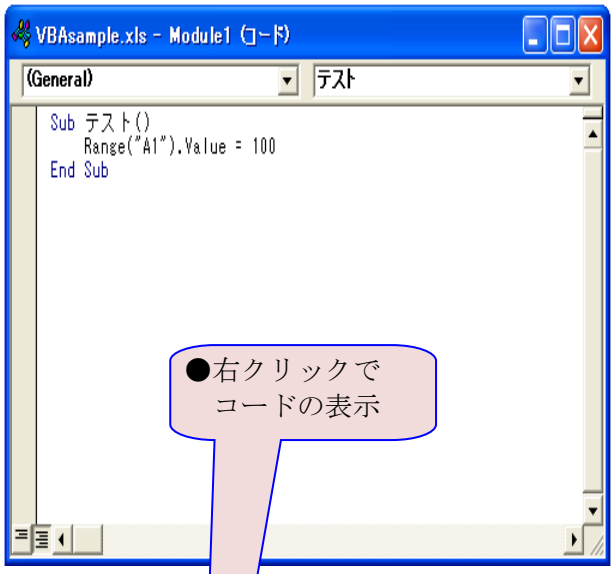
５．画像を配置してボタンにする



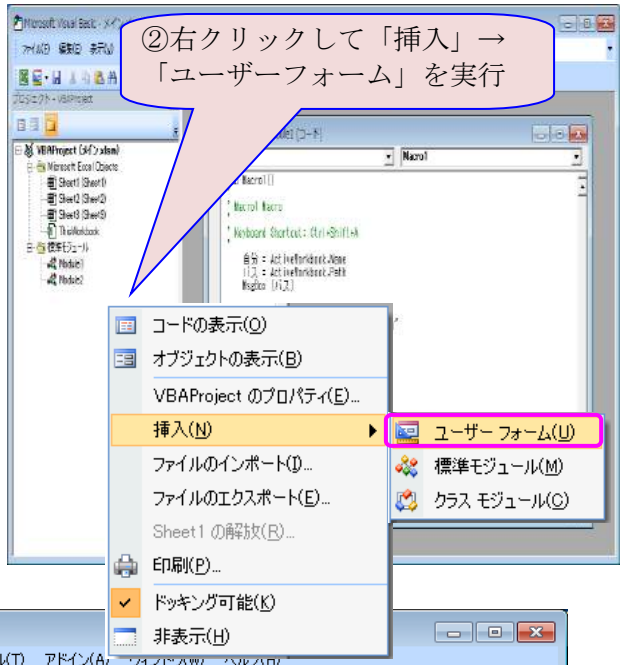
・以下のように、必要なプログラムを前もって作成してあるものとする

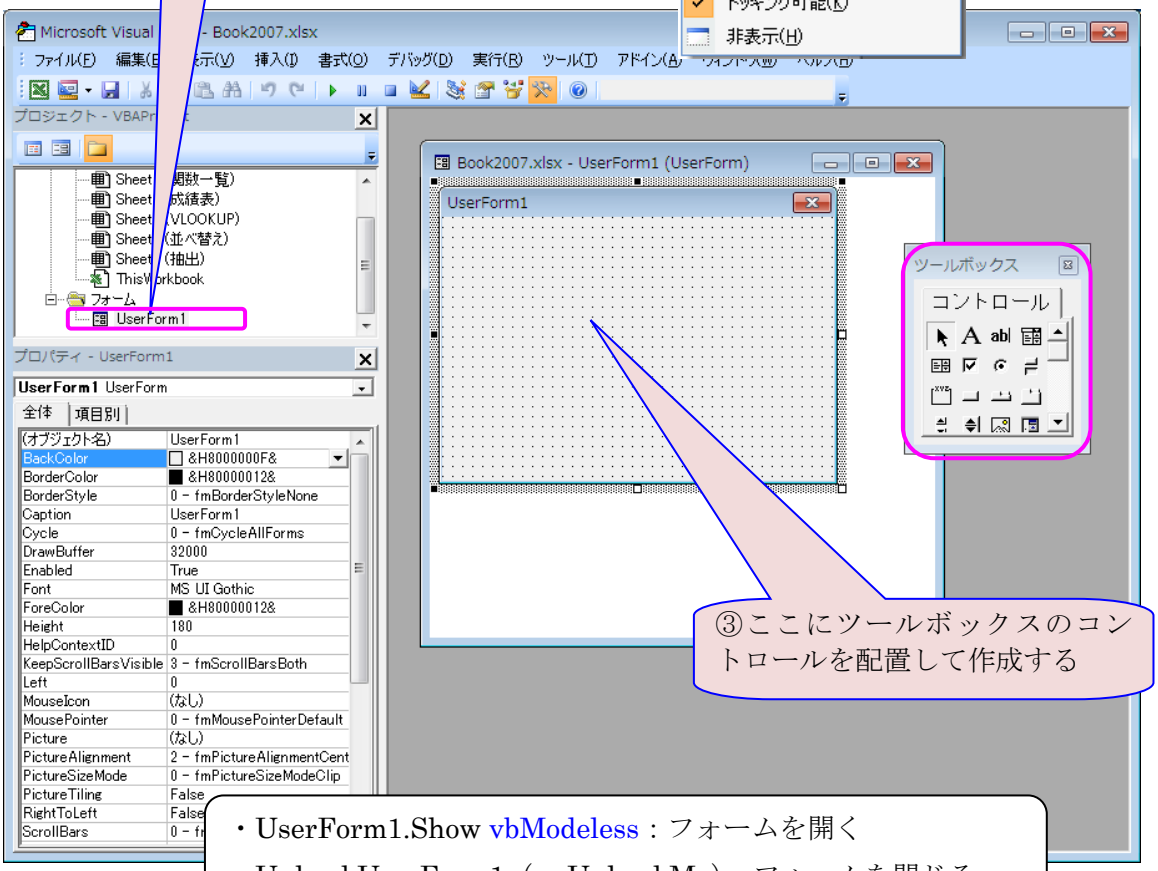


６．ユーザーフォームを作成して、そこにマクロを割り付ける



●右クリックで
コードの表示





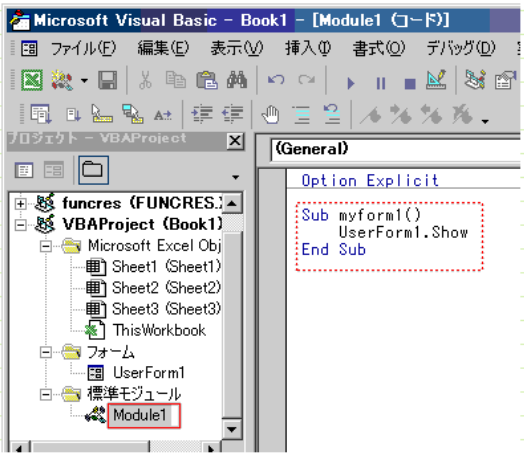
③ここにツールボックスのコントロールを配置して作成する

・UserForm1.Show vbModeless：フォームを開く
・Unload UserForm1（or Unload Me）：フォームを閉じる

●ユーザフォームの実行方法

シートからユーザーフォームを表示する　topへ

1. 標準モジュールを挿入し、Module1に以下のコードを書きます。
   - Sub myform1()
     UserForm1.Show
     End Sub



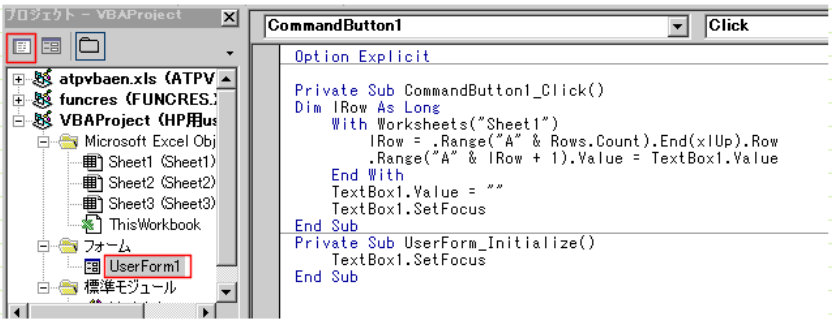
   - ユーザーフォームを表示したまま選択しているセルの範囲や内容を書き換えるなど、他の操作をしたいときはモードレスで表示します。
     Sub myform1()
     UserForm1.Show vbModeless
     End Sub

2. シートにフォームのボタンを描画し、マクロ「myform1」を登録します。
3. ボタンをクリックするとUserForm1が表示されます。
   (注)UserForm1はタイトルバーの閉じるボタン[X]で閉じることができます。

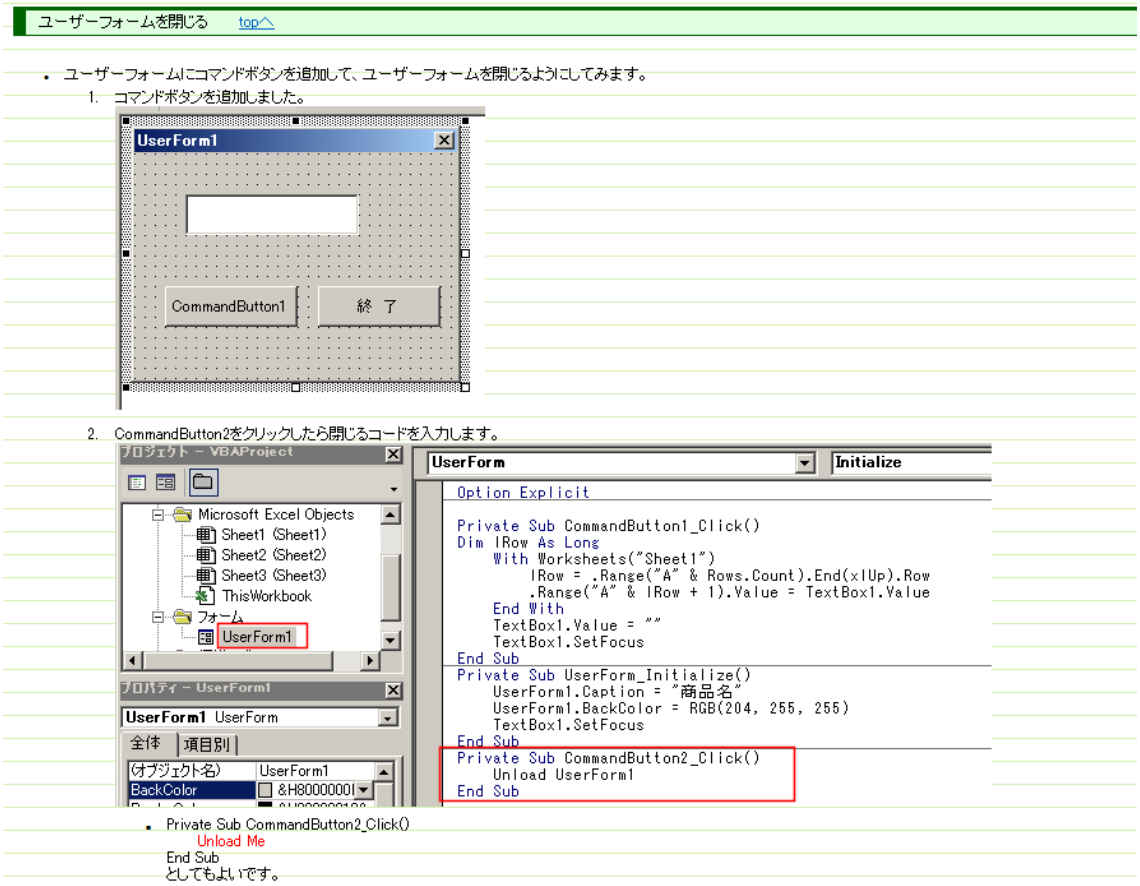
●使用例

コントロールを使う　topへ

- テキストボックスに文字を入力して、コマンドボタンをクリックしたらシートに値を入力するコードを作成してみます。
  1. プロジェクトウィンドウでUserform1を選択し、コードの表示ボタンをクリックします。
  2. 表示されたコードウィンドウに下図のようにコードを入力します。



- テキストボックスに文字を入力してコマンドボタンをクリックします。
  A列の最終行に入力した文字列が入力されます。
  終了はコードを作成していないので、Userform1の閉じるボタンを使ってください。



○ファイル選択ダイアログ

ダイアログの「開く」ボタンを押せば**選んだファイルのフルパス**が実行結果として返り、「キャンセル」ボタンを押したときは **False** が返ります。

```
Sub prcApplicationGetOpenFilename()

  Dim vntFileName As Variant

  'ファイルを開くダイアログを開きます
  vntFileName = _
    Application.GetOpenFilename( _
      FileFilter:="エクセルファイル(*.xls),*.xls" & _
            ",CSV ファイル(*.csv),*.csv" _
     , FilterIndex:=1 _
     , Title:="  ファイルオープンダイアログ" _
     , MultiSelect:=False _
      )

  'ファイルが選択されているときは
  '選択したファイルを Workbooks.Open メソッドで開きます
  If vntFileName <> False Then
    Workbooks.Open Filename:=vntFileName
  End If

End Sub
```

７．Excel を開いたときにマクロを実行させる方法

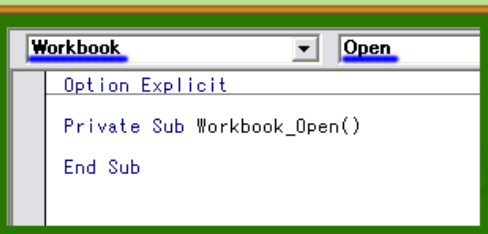
ワークブックを開いたときに自動実行したい場合は、
①ワークブックが持つマクロのイベント **Workbook/Open** という所にコードを記述するか、
②標準モジュールの中に **Auto_Open** という名前のプロシジャを作っておく 2 種類の方法があります。

①

ワークブックのOpenイベントに書く場合

ワークブックの**Open**イベントに記述する場合は、**ThisWorkbook**にある**Workbook**オブジェクトの**Open**というイベントを選択(右図参照)し、その中へコードを記述してください。

例では、ワークブックを開いたときに「いらっしゃいませ」というメッセージを表示しています。




```
Private Sub Workbook_Open()
    MsgBox "いらっしゃいませ！"
End Sub
```

※このエクセルマクロは ThisWorkbookへ記述するマクロです

②

Auto_Openを使う場合

Auto_Openを使う場合は、標準モジュールを追加し**Auto_Open**という名前のプロシジャを作り、その中へマクロを記述してください。

例では、ワークブックを開いたときに「あとは運次第…」という怪しげな(笑)メッセージを表示しています。

```
Sub Auto_Open()
    MsgBox "あとは運次第…"
End Sub
```

※このエクセルマクロは 標準モジュールへ記述するマクロです

③その他

どちらが先に動くか

ワークブックの**Open**イベントと、**Auto_Open**へ書いたマクロのうち、先に動くのはワークブックの**Open**イベントの方です。

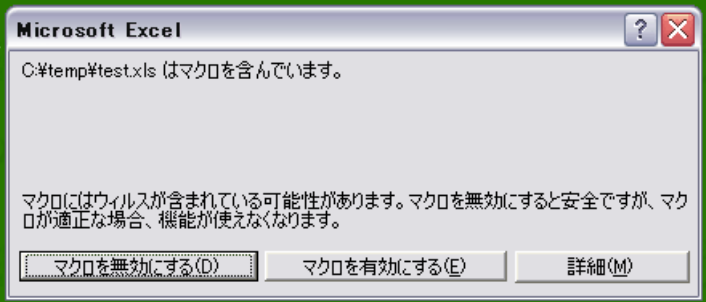
ワークブックの**Open**イベントのマクロが動いた後、**Auto_Open**のマクロが動作します。

Auto_Openを回避したいとき

マクロの自動実行を仕込んだ後は、**マクロを有効にする**ボタンを押した瞬間マクロが動き出します。しかし、いざ何かの都合でマクロを直したいときなどに困ってしまいます。かと言って**マクロを無効にする**を押すと、テストするときに不便だし…なんてときの自動実行回避策です。

「マクロを有効にするボタンを押すときに**シフトキーを押したまま**」にしておきます。**Auto_Open**などを回避しワークブックを開くことが出来ます。このあたり(シフトキーで回避)のやりかたはAccessと同じですね！

もし、**マクロのセキュリティを低**(なかなかいないと思いますが…詳しくはエクセルスマイル「5.マクロのセキュリティについて」)にしている強者の方だとこのダイアログが出ないので、**ワークブックを開くときにシフトキーを押しっぱなし**にして開いてください。



●ファイルオープン・ダイアログ

開くブックをユーザーに選択してもらう場合は、[ファイルを開く]ダイアログボックスを表示する

```
Sub Sample1()
    Dim OpenFileName As String
    OpenFileName = Application.GetOpenFilename("Microsoft Excel ブック,*.xls?")
    Workbooks.Open OpenFileName
End Sub
```

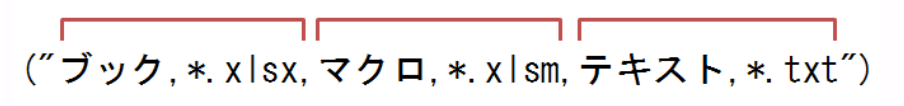
書式

GetOpenFilename FileFilter, FilterIndex, Title, ButtonText, MultiSelect

どんな種類(拡張子)のファイルを表示するかは、**引数 FileFilter** で指定します。

例）**"Microsoft Excel ブック,*.xls?"**

複数の拡張子を表示したいときは、表示したい拡張子を**セミコロン(;)**で区切ります。



ユーザーが[キャンセル]ボタンをクリックしたかどうかは、次のように判定します。

```
Sub Sample2()
 Dim OpenFileName As Variant
    OpenFileName = Application.GetOpenFilename("Microsoft Excel ブック,*.xls?")
    If OpenFileName <> False Then
        Workbooks.Open OpenFileName
    Else
        MsgBox "キャンセルされました"
    End If
End Sub
```

８．Function を作成して利用する

・祭日判定関数（１つの例）

IsHoliday（ｾﾙ：日付）　：　祭日の時は、TRUE
祭日ではない時は、FALSE
を返します。

●祭日判定関数用マクロ（標準モジュールに記述）

注意）祭日は、この関数の中に具体的に記述しています。
現在は、2010/1/1 から 2017/12/31 までの範囲で記述しています。
途中で、法律が改正され、祭日が増えたり、減ったり、移動したりした場合には変更して下さい。また、2017/12/31 以降は、必要となった時点で追記して下さい。
ただし、この部分の記述文字数には限度がありますので、古い年のものは削除する必要が出る場合もあります。

```
Function IsHoliday(d As Date) As Boolean
    Dim expr As String
    expr = Format(d, "yyyy/mm/dd")

    Const Holidays = _
     "2010/01/01 2010/01/11 2010/02/11 2010/03/21 2010/03/22 2010/04/29 2010/05/03 2010/05/04 2010/05/05
      2010/07/19 2010/09/20 2010/09/23 2010/10/11 2010/11/03 2010/11/23 2010/12/23 " & _
     "2011/01/01 2011/01/10 2011/02/11 2011/03/21 2011/04/29 2011/05/03 2011/05/04 2011/05/05 2011/07/18
      2011/09/19 2011/09/23 2011/10/10 2011/11/03 2011/11/23 2011/12/23 " & _
     "2012/01/01 2012/01/02 2012/01/09 2012/02/11 2012/03/20 2012/04/29 2012/04/30 2012/05/03 2012/05/04
      2012/05/05 2012/07/16 2012/09/17 2012/09/22 2012/10/08 2012/11/03 2012/11/23 2012/12/23 2012/12/24 " &
      _
     "2013/01/01 2013/01/14 2013/02/11 2013/03/20 2013/04/29 2013/05/03 2013/05/04 2013/05/05 2013/05/06
      2013/07/15 2013/09/16 2013/09/23 2013/10/14 2013/11/03 2013/11/04 2013/11/23 2013/12/23 " & _
     "2014/01/01 2014/01/13 2014/02/11 2014/03/21 2014/04/29 2014/05/03 2014/05/04 2014/05/05 2014/05/06
      2014/07/21 2014/09/15 2014/09/23 2014/10/13 2014/11/03 2014/11/23 2014/11/24 2014/12/23 " & _
     "2015/01/01 2015/01/12 2015/02/11 2015/03/21 2015/04/29 2015/05/03 2015/05/04 2015/05/05 2015/05/06
      2015/07/20 2015/09/21 2015/09/22 2015/09/23 2015/10/12 2015/11/03 2015/11/23 2015/12/23 " & _
     "2016/01/01 2016/01/11 2016/02/11 2016/03/20 2016/03/21 2016/04/29 2016/05/03 2016/05/04 2016/05/05
      2016/07/18 2016/09/19 2016/09/22 2016/10/10 2016/11/03 2016/11/23 2016/12/23 " & _
     "2017/01/01 2017/01/02 2017/01/09 2017/02/11 2017/03/20 2017/04/29 2017/05/03 2017/05/04 2017/05/05
      2017/07/17 2017/09/18 2017/09/23 2017/10/09 2017/11/03 2017/11/23 2017/12/23 "

    IsHoliday = InStr(1, Holidays, expr) > 0

End Function
```

９．イベントプロシージャの作成

イベントプロシージャは、大きく①**ワークブック**のイベントと②**ワークシート**のイベントの２つに分類することができます。

①**ワークブック**のイベント

・ブックの主なイベント

イベントの種類	発生するタイミング
Activate	ブックがアクティブになったとき
AddinInstall	ブックがアドインとして組み込まれたとき
AddinUninstall	ブックがアドインから削除されたとき
BeforeClose	ブックを閉じるとき
BeforePrint	ブックを印刷するとき
BeforeSave	ブックを保存するとき
Deactivate	ブックがアクティブでなくなったとき
NewSheet	ブックに新しいシートを作成したとき
Open	ブックを開いたとき
SheetActivate	ブック内のシートがアクティブになったとき
SheetBeforeDoubleClick	ブック内のシートがダブルクリックされたとき
SheetBeforeRightClick	ブック内のシートが右クリックされたとき
SheetCalculate	ブック内のシートが再計算されたとき
SheetChange	ブック内のシートが変更されたとき
SheetDeactivate	ブック内のシートがアクティブでなくなったとき
SheetFollowHyperlink	ブック内のハイパーリンクをクリックしたとき
SheetSelectionChange	ブック内のシートの選択範囲を変更したとき
WindowActivate	ブックウィンドウがアクティブになったとき
WindowDeactivate	ブックウィンドウがアクティブでなくなったとき
WindowResize	ブックウィンドウの大きさを変更したとき

例）
```
Private Sub Workbook_Open()
    'エクセルのウィンドウを最大化します
     Application.WindowState = xlMaximized
End Sub
```

②ワークシートのイベント

・ワークシートの主なイベント

イベントの種類	発生するタイミング
Activate	ワークシートがアクティブになったとき
BeforeDoubleClick	ワークシートをダブルクリックしたとき
BeforeRightClick	ワークシートを右クリックしたとき
Calculate	ワークシートが再計算されたとき
Change	ワークシートのセルが変更されたとき
Deactivate	ワークシートがアクティブでなくなったとき
FollowHyperlink	ワークシートのハイパーリンクをクリックしたとき
SelectionChange	ワークシートの選択範囲を変更したとき

例）
```
Private Sub Worksheet_Activate()
    ActiveSheet.Range("A1").Select
End Sub
```

Change イベントと Calculate イベントについて

ちょっとわかりづらいのが「Change イベント」と「Calculate イベント」です。

Change イベントは一般的に「値が変更された時に発生する」とありますので、セルの値が「1」→「2」に変更された時に発生ものと思っていました。

しかし数式で計算されて値が変更した場合、Calculate イベントのみ発生し、Change イベントは発生しません。

考えてみればそれも当然で、見ための値は変更してもセルの中に入っている数式（仮に=SUM(A1:A5)）は変更されてないからなんですね。

だけど値を変更してなくてもセルを編集モードにすると Change イベントが発生します。

■イベントプロシージャの引数

イベントプロシージャの種類によっては、引数を持つものがあります。これらの引数はイベントプロシージャ内で利用することができます。主なイベントプロシージャの引数について解説していきます。

・イベントプロシージャの引数

引数	内容
Sh	SheetActivateイベントなどで対象となるシートオブジェクトが格納されます
Source	SheetChangeイベントなどで対象となるセル範囲が格納されます
Wb	WindowActivateイベントなどで対象となるブックオブジェクトが格納されます
Wn	WindowActivateイベントなどで対象となるウィンドウオブジェクトが格納されます
Target	SelectionChangeイベントなどで対象となるセル範囲が格納されます
Cancel	イベントで操作をCancelするときはTrueを指定します。既定値はFalseです

■自動実行マクロ

「Workbook_Open」と「Workbook_BeforeClose」イベントプロシージャには、ブックを開くときと閉じるときに実行したいプロシージャを記述します。これと似た働きをするマクロに「Auto_Open」と「Auto_Close」マクロがあります。どちらを使用してもブックを開くときと閉じるときに、プロシージャを自動実行することができますが、以下の違いがあります。

マクロの種類	手動でブックを開く	VBAからブックを開く
Workbook_Open	実行される	実行される
Auto_Open	実行される	実行されない
マクロの種類	**手動でブックを開く**	**VBAからブックを開く**
Workbook_BeforeClose	実行される	実行される
Auto_Close	実行される	実行されない

VBAのコードからブックを開いたり閉じたりするときに、自動実行させたくないプロシージャは「Auto_Open」と「Auto_Close」マクロを使用して記述します。この場合、手動でブックを開いたり閉じたりしたときに自動実行されます。

■イベントを発生させないようにする

次のコードを実行することで、イベントの発生を抑制することができます。

[構文]
Application.EnableEvents = True または False

Trueを指定した場合、イベントが発生します。Falseを指定するとイベントの発生が抑制されます。

※ コードの中で、一時的にイベントの発生を停止した場合、あとのコードでTrueを指定して元に戻すのを忘れないようにしてください。イベントの発生停止を解除しない限り、そのあとのイベントがすべて停止されるので注意が必要です。