目　　次

Lesson１　本講座で学ぶこと

（１）　はじめに

この「業務カレンダー」は、ＶＢＡを使いこなせず悩んでいる方へ、少しでも覚えるきっかけにしてほしいと考え、作りました。「業務カレンダー」を使うことが目的ではなく、ＶＢＡを学びコードを書くことにより実現できる「繰り返しの処理」を習得していただけると非常にうれしく思います。受講生の中には、コードを書くのはハードルが高いと思われる方、また、コードを書いたことはないが、「マクロの記録」なら使ったことがあるという方、様々いらっしゃると思います。マクロの記録だけではできないこと、そして「マクロの記録」の正しい使い方を学んでいただきたいと思います。

しかし、ＶＢＡは魔法の杖ではありません。ＶＢＡを使うには、基礎を学習する必要があります。決まり文句、任意で書く箇所、それぞれを理解し、応用できるＶＢＡを心がけたいと思いますのでよろしくお願い致します。

先にポイントをお伝えします。今はわからなくてもいいです。

【この講座というよりも、VBA を扱う上でのポイントといっても過言ではありません。】

ポイント	講座を通して覚えていただきたいこと	コード
その１　範囲はどこか？	範囲の指定の方法	Range　や　Cells
その２　条件は何か？	条件によって処理を変える	If～Then～ Else～ End　If
その３　処理は何か？	変数の扱い方 繰り返し セルのクリア セルに色をつける（ぬりつぶし） シートのコピー シートの名前を変える	Dim For　～　Next ClearContents Interior．Color Copy Name
※VBA を使うと、セルを選択せずに、特定のセルを操作することができます。 ワークシート上の作業では、「セルを選択」→「そのセルに処理を行う」ことが多いと思います。 VBA は選択しなくても処理ができるということがもう１つのポイントでもあります。		

（２）　　題材の紹介

題材	業務カレンダー	
機能・特徴	縦方向に年月日、横方向に項目を入れるカレンダーです。 ＶＢＡを使い、下記をボタンクリックで実現します。 ・シートのコピー　　・シート名の変更 ・月の更新　　・土日判定　　・シートの削除	
対象者	ＶＢＡを使ったことがない方 若しくはマクロの記録しか使ったことがない方 または、上記以外の方	
覚えていただきたいこと	ＶＢＡ	機能・関数
	コードを書くこと 繰り返し処理 条件処理	シリアル値 表示形式 日付に関する関数
目標	上記のことを実現するためには、ＶＢＡの基礎を学習することが必要になります。ＶＢＡの基礎とマクロの記録の正しい使い方を学び、ＶＢＡに対する苦手意識をなくしていただけるような講座にしていきたいと思います。	

（３）　　日程

年月日	内容
2021 年 10 月 15 日（金）	・「題材」業務カレンダーの機能紹介及び機能や関数の紹介 ・VBA の概念
10 月 22 日（金）	・マクロ記録 ・VBA の構文
11 月 5 日（金）	・VBA の変数 ・セルの操作
11 月 12 日（金）	・ステートメント ・VBA の実行
11 月 19 日（金）	・VBA 以外の機能や関数 ・題材のコード作成と実行
11 月 26 日（金）	・まとめ

（講座の進行具合によって上記の通りにならない場合がありますので、ご了承ください。）

（４）　　題材でのコード提示

もちろん、いまは何が書いてあるかわからなくても結構です。

‘緑色は「コメント」といい、メモのようなものです。

‘コメントを書く位置は自由ですが、コードの行の前に書くことはできません。

‘本講座では、コードの１行手前に書きました。

１．題材コード　その１　シートをコピーします

```
Sub シートコピー()

    '変数を定義します　ms　[メッセージ]
    Dim ms As Long

    'メッセージボックスを表示します
    ms = MsgBox("コピーしますか？", vbYesNo + vbQuestion, "確認")

    'はい(Y)なら
    If ms = vbYes Then

        'アクティブシートをシート名：「マスター」の後ろにコピーします
        ActiveSheet.Copy After:=Sheets("マスター")

        'コピー後にメッセージ表示します
        MsgBox "コピーしました"

    'いいえ(N)なら
    Else

        'なにもせず、メッセージ表示するのみです
        MsgBox "コピーしませんでした"

    'ifの終わり
    End If

End Sub
```

覚えること	覚えると・・・
変数	変数の使い方が分かるようになります
Msgbox	メッセージボックスが使えるようになります
If　Else　End　If	条件によって処理を変えることができるようになります

２．題材コード　その２　月を変更します

```
Sub 月の変更()

    'メッセージボックスを表示します
    MsgBox "セルC3を選択して、月の変更をしてください"

    'セルC3を選択し,月を変更します
    Range("C3").Select

End Sub
```

覚えること	覚えると・・・
Select	セルの選択ができるようになります

3．題材コード　その３　シート名を変更します

```
Sub シート名変更()

    'セルC2とセルC3の値を使い,アクティブシートの名前を変更します
    ActiveSheet.Name = Range("C2").Value & "年" & Range("C3").Value & "月"

End Sub
```

覚えること	覚えると・・・
ActiveSheet	特定のシートではなく現在アクティブにしているシートの操作ができるようになります
Name	シートの名前の変更ができるようになります
＆	違うセルのデータ（値）を１つのセルで表示できるようになります

4．題材コード　その４　カレンダーを翌月に更新します

```
Sub 月の更新()

    '変数を定義します　irow　[i カウント　row　行]
    Dim irow As Long

    '変数を定義します　ofs　[基準となる位置からの差　Offset]
    Dim ofs As Long

    '変数を定義します　gm　[月末]
    Dim gm As Long

    '変数を定義します　mst　[ monthstart　]
    Dim mst As Long

    '入力されている内容をクリアします
    Range("E3:H33").ClearContents

    '基準となる位置からの差を２とします
    ofs = 2

    '変数mstに、セルC6の値を代入します
    mst = Range("C6").Value

    '変数gmにC9を代入します
    gm = Range("C9").Value

    'irowを１から月末まで繰り返します
    For irow = 1 To gm

        'E列の該当行に、日付を代入していきます。
        Cells(irow + ofs, "E").Value = mst + irow - 1

    'Forの終わり
    Next irow

End Sub
```

覚えること	覚えると・・・
ClearContents	セルのクリアができるようになります
Range	セルの操作ができるようになります
Cells	セルの操作ができるようになります
For　Next	繰り返しの処理ができるようになります

５．題材コード　その５　土日を判定し、色を付けます

```
Sub 土日判定()

    '変数を定義します　irow2　[i カウント　row　行]
    Dim irow2 As Long

        'セルE3からセルI33の色を塗りつぶしなしにします
        Range("E3:I33").Interior.ColorIndex = xlNone

    '３行目から３３行目まで繰り返します
    For irow2 = 3 To 33

        'もし、I列が,"日"だったら
        If Cells(irow2, "I").Value = "日" Then

            'EからIまでの塗りつぶし色をうすい赤に設定します
            Range(Cells(irow2, "E"), Cells(irow2, "I")).Interior.Color = RGB(255, 100, 100)

        'ifの終わり
        End If

        'もし、I列が、"土"だったら
        If Cells(irow2, "I").Value = "土" Then

            'EからIまでの塗りつぶし色をうすい青に設定します
            Range(Cells(irow2, "E"), Cells(irow2, "I")).Interior.Color = RGB(100, 100, 255)

        'ifの終わり
        End If

    'Forの終わり
    Next irow2

End Sub
```

覚えること	覚えると・・・
Interior.ColorIndex = xlNone Interior.Color = RGB(255, 0, 0)	塗りつぶしなしを設定することができます 数値を変えることでいろいろな色の設定ができるようになります

６．題材コード　その６　シートを削除します

```
Sub シート削除()
    '変数を定義します　ms2　[メッセージ]
    Dim ms2 As Long

    'メッセージボックスを表示します
    ms2 = MsgBox("このシートを削除しますか？", vbYesNo + vbQuestion, "確認")

    'はい(Y)なら
    If ms2 = vbYes Then

        'アクティブシートを削除します
        ActiveSheet.Delete

        'コピー後にメッセージ表示します
        MsgBox "削除しました"

    'いいえ(N)なら
    Else

        'なにもせず、メッセージ表示するのみです
        MsgBox "削除しませんでした"

    'Ifの終わり
    End If

End Sub
```

覚えること	覚えると・・・
ActiveSheet.Delete	シートの削除ができるようになります

これらのコードは、マクロの記録だけでは無理です。
１行ずつ学習が必要なので、基本から学習していきます。
また、上記のコードの説明だけでは、応用が利かないと思いますので、基本から押さえていきます。

（５）　　コードを書く練習

ユーザーと対話する関数

　Msgbox 関数は、引数に指定した文字列を画面に表示します。ユーザーが操作できるボタンやアイコンを指定することができ、どのボタンをクリックしたかを返します。

【書式】Msgbox（文字列，ボタンとアイコン，タイトル）

引数　ボタンとアイコンに指定できる主な定数は次の通りです。

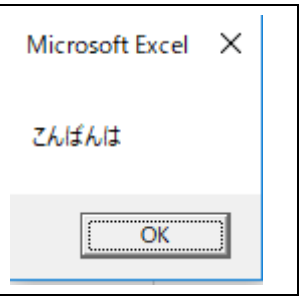
定数	説明
vbOKOnly	〔OK〕のみ表示
vbOKCancel	〔OK〕と〔キャンセル〕を表示
vbYesNo	〔はい〕と〔いいえ〕を表示
vbYesNoCancel	〔はい〕、〔いいえ〕および〔キャンセル〕を表示

アイコンに関する定数

定数	説明
v bCritical	警告
vbQuestion	問い合わせ
vbExclamation	注意
vbInformation	情報

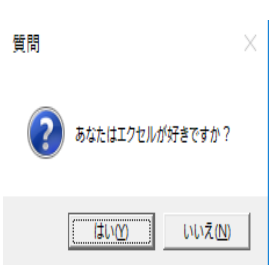
使用例１

```
Sub メッセージ1()
    MsgBox "こんばんは"
End Sub
```



使用例２

```
Sub メッセージ2()
    Dim ms As Long
    ms = MsgBox("あなたはエクセルが好きですか？", vbYesNo + vbQuestion, "質問")
End Sub
```

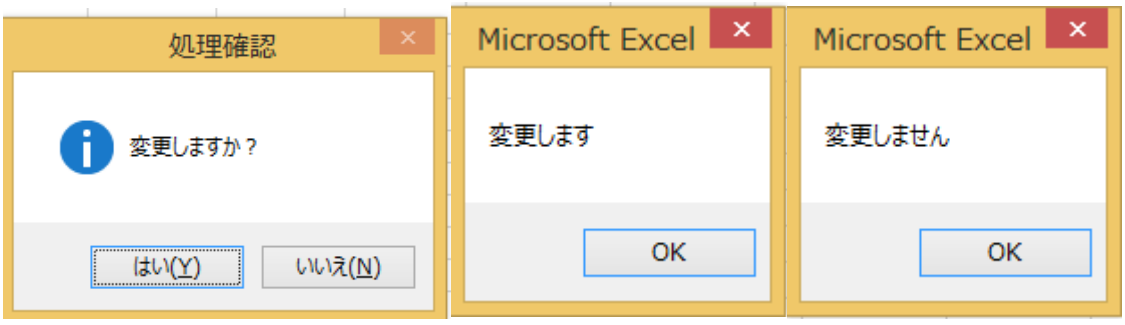


この Msgbox 関数は、ユーザーがどのボタンをクリックしたかの結果を返します。
返す定数は次の通りです。

定数	説明
vbOK	〔OK〕ボタンがクリックされた
vbCancel	〔キャンセル〕ボタンがクリックされた
vbYes	〔はい〕ボタンがクリックされた
vbNo	〔いいえ〕ボタンがクリックされた

使用例 3

```
Sub メッセージ3()
    '変数定義　ms
    Dim ms As Long
    'メッセージボックス　ボタンとアイコン　タイトルを表示させます
    ms = MsgBox("変更しますか？", vbYesNo + vbInformation, "処理確認")
    'もし、はいが押されたら、
    If ms = vbYes Then
        'はいの時の処理→変更しますと表示し、
        MsgBox "変更します"
    'Ifの区切り
    Else
        'いいえの時の処理→変更しませんと表示します
        MsgBox "変更しません"
    'Ifの終わり
    End If
End Sub
```



Msgbox 関数の結果を利用するときは、引数全体を（）で囲います。文字列を表示するだけで、結果を利用しないときは、()で囲む必要はありません。
関数の結果を受け取る変数　→　実態は整数です。
Msgbox 関数の結果を受け取る変数は、Long で宣言します。

Lesson２　VBA の基礎

（１）　マクロと VBA の概念

マクロは Excel が持つ機能の１つです。このマクロの命令を記述するときに使うのが VBA（Visual Basic for Applications）というプログラミング言語です。

１．マクロとは何か

Excel には、さまざまな機能があります。計算をしたり、グラフを作ったり。そのうちの１つにマクロ機能があります。マクロを「実際に行った操作を記録し、そのまま繰り返し実行できる機能」と認識している方も多いのですが、マクロの機能はそれだけではありません。操作を記録するだけではなく、条件分岐や繰り返し処理といった一般機能だけではできないことを実行することができます。

２．VBA とは何か

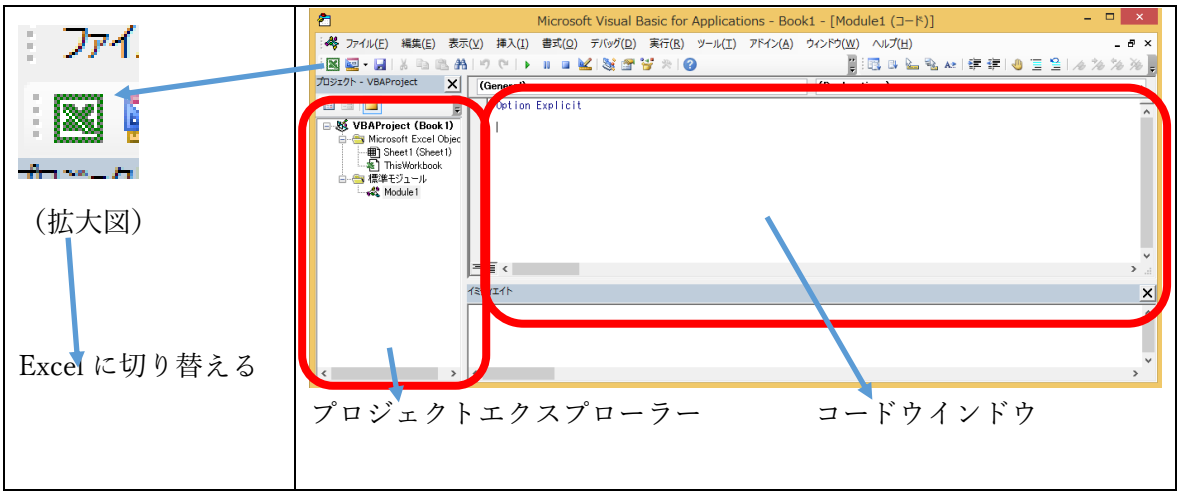
マクロは一種の命令書です。「ボタンをクリックするとそのシートの印刷を行う」や「セルのデータが変化したら計算を行う」など、その命令書に従って、Excel が自動操作されます。その命令書を記述するときに使うプログラミング言語が VBA です。

マクロ≒VBA ですが、マクロの記録は全く別物です。

３．Visual Basic Editor の操作

VBA の作成・編集をするには、VBE という特別な画面を使います。

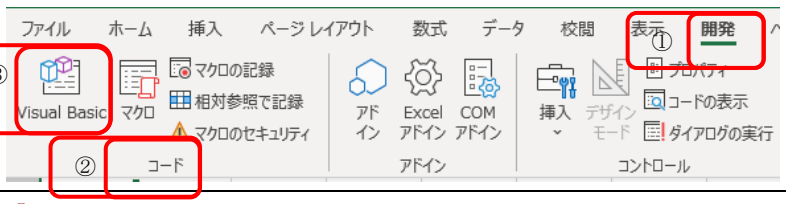
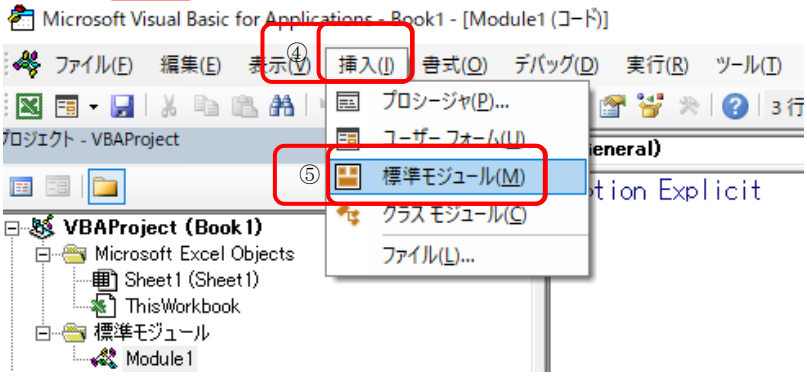
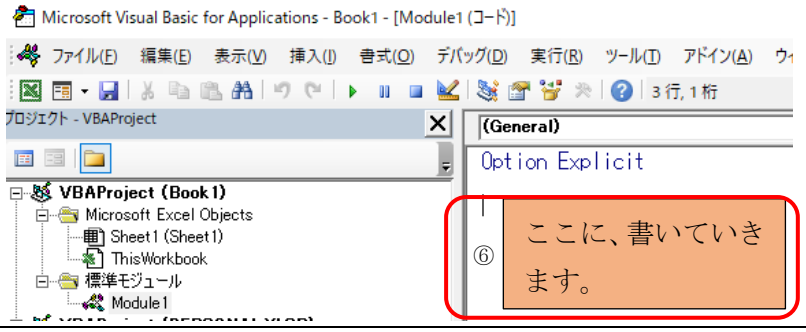
VBE の起動　【開発】タブ→【Visual Basic】をクリック



4．コードを書く場所

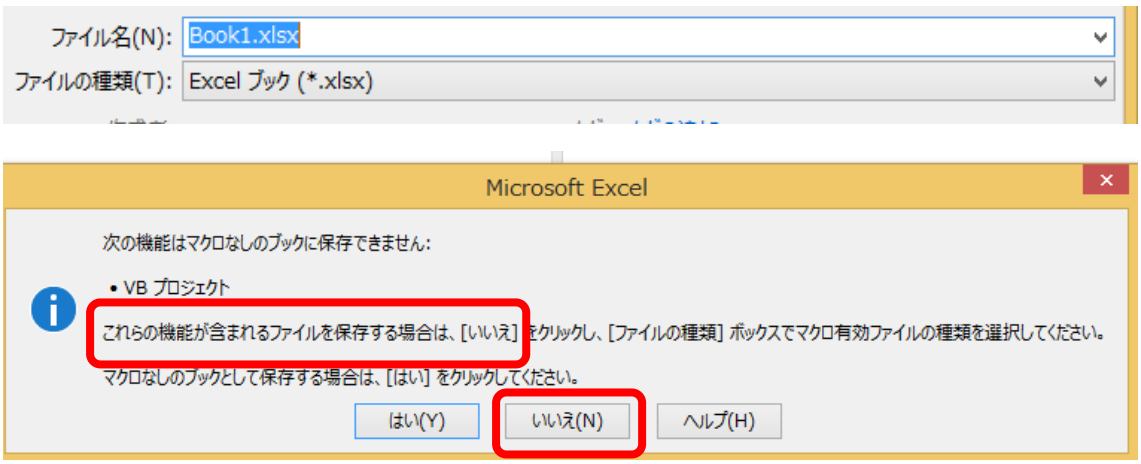
標準モジュールを挿入して書いていきます。

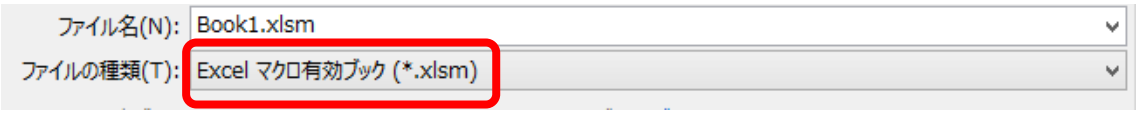
挿入方法は下記に示す通りです。　【開発】→【Visual Basic】

①［開発］タブ ②［コード］グループ ③［Visual Basic］ボタン	
④[挿入］タブ ⑤［標準モジュール]ボタン	
⑥書く場所	

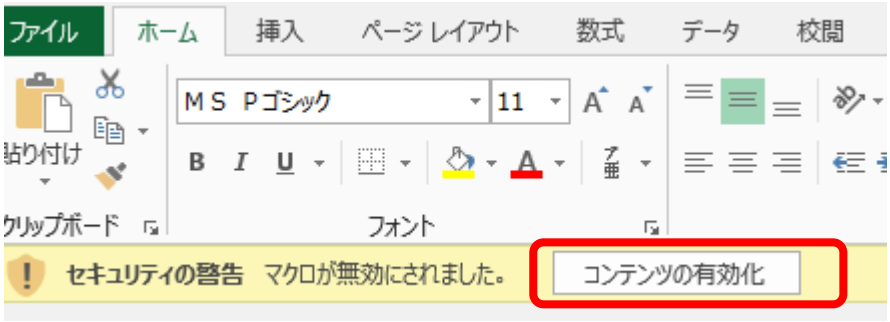
5．ファイルの保存

保存の画面

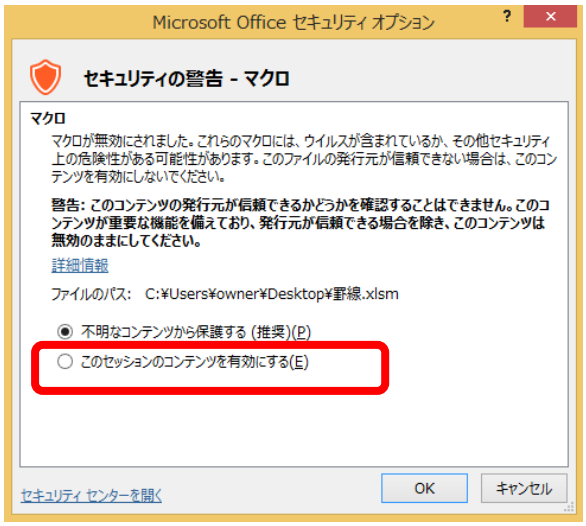




6．セキュリティの警告



また、ブックによっては次のような警告画面が表示される場合があります。



「このセッションのコンテンツを有効にする」として進んでください。

（2）　マクロの記録

1．マクロの記録とは

Excel に対して実際に行った操作を VBA のコードとして記録する機能です。

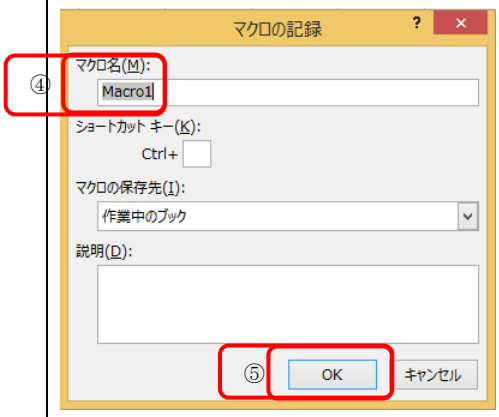
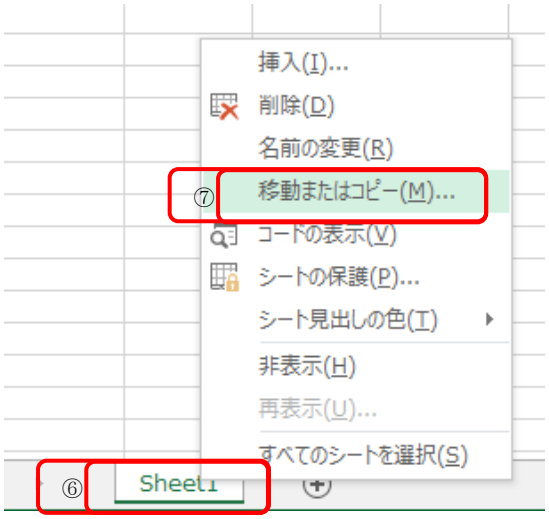
2．マクロの記録の限界

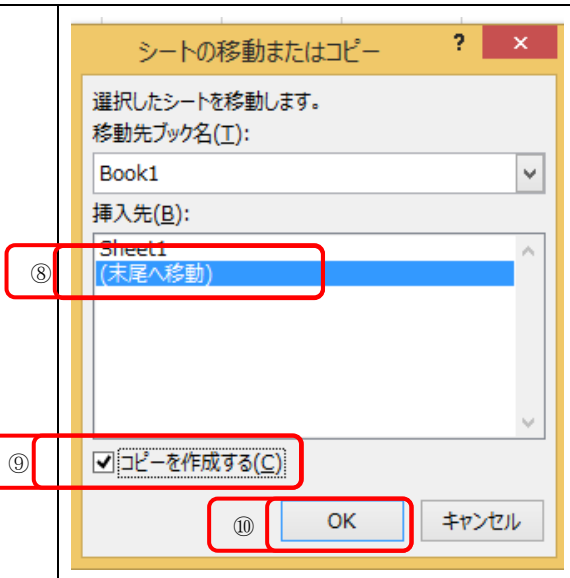
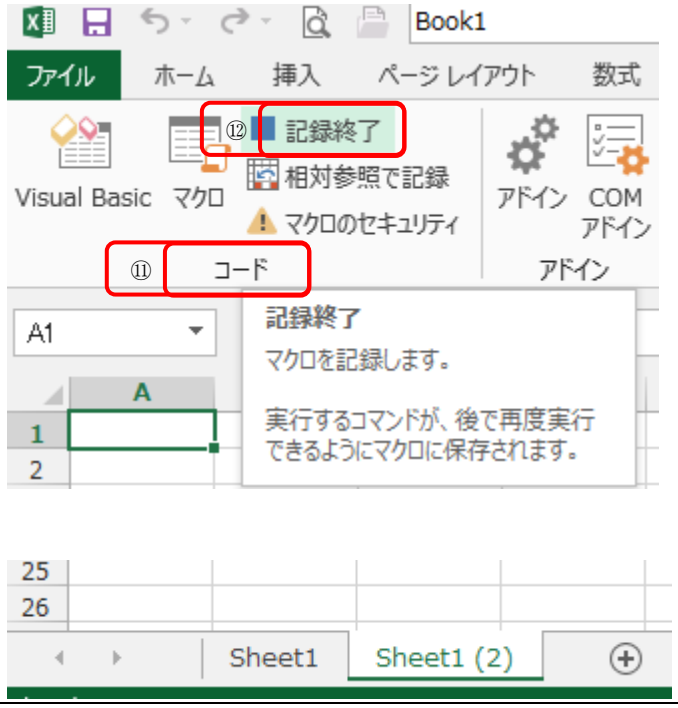
実際に行った操作を VBA のコードとして記録する機能だけでも十分素晴らしい機能ですが、残念ながら限界があります。

それは「ステートメントや関数は記録されないこと」と「不要な設定まで記録されること」です。

記録マクロの操作方法	マクロの記録　→　操作　→　記録終了

例）　シートのコピー

手順１ 新しいブックを開いて、 ①［開発］タブ ②［コード］グループ ③［マクロの記録］ボタン	
手順２ ④マクロ名はそのままで （番号は忘れずに） ⑤　OK	
手順３ ⑥シート名を右クリック、 ⑦移動またはコピー	

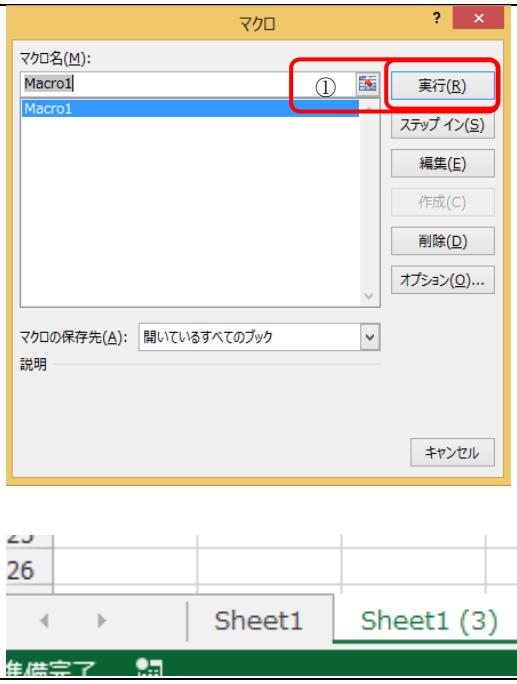
手順４ ⑧（末尾へ移動） ⑨コピーを作成する ⑩　OK	
手順５ ［開発］タブ ⑪［コード］グループ ⑫［記録終了］ボタン ※注意 ［マクロの記録］ボタンが ［記録終了］ボタンに変わっています）	
操作は以上です。	

記録したマクロの実行方法

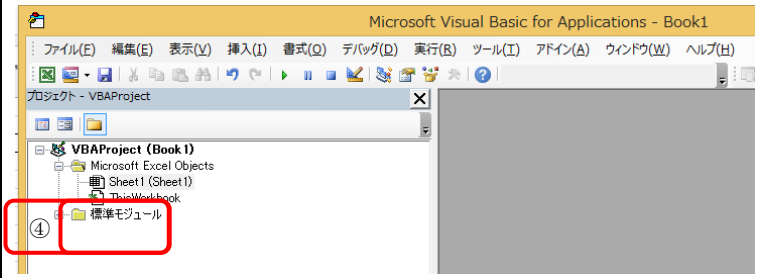
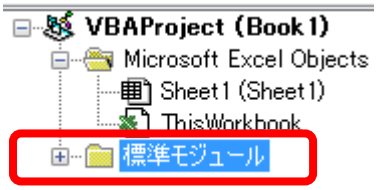
マクロ　→　マクロ名を選んで　→　実行

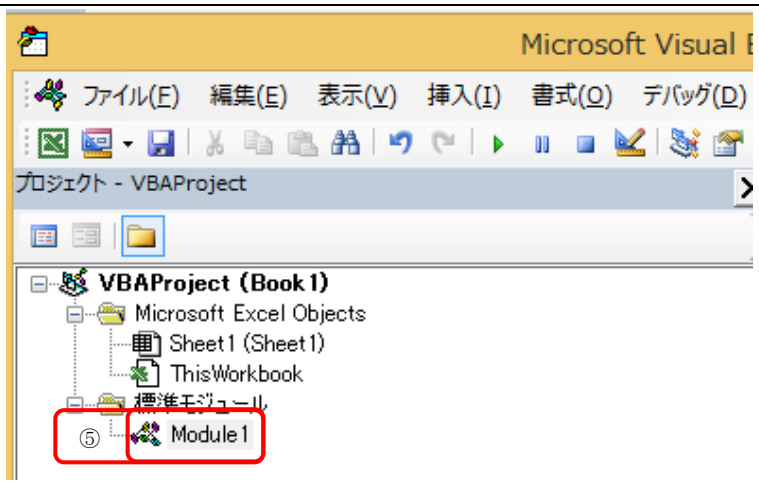
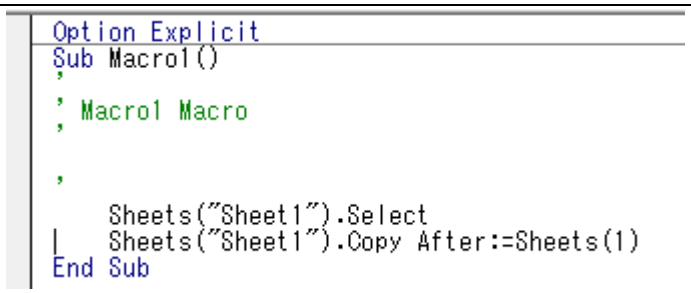
Sheets1（２）が選択された状態のまま、先ほどのマクロを実行します。

手順１ ①［開発］タブ ②［コード］グループ ③［マクロ］ボタン	

手順２ 該当のマクロ（Macro1）を選択し、 ①実行	
操作は以上です。	

記録したマクロの確認

手順１ ①［開発］タブ ②［コード］グループ ③［Visual　Basic］ボタン	
手順２ ④標準モジュールをダブルクリックし、	 （拡大図） 

手順３ ⑤Module１をクリックすると、	
右記のコードが表示されます。	

マクロの記録で作られたコードの解説

Sub Macro1() ' ' Macro1 Macro ' ' 　　Sheets("Sheet1").Select 　　Sheets("Sheet1").Copy after:=Sheets(1) End Sub	←決まり文句 ←自動で書かれるコメント ←　〃 ←　〃 ←　〃 ←　〃 ←選択しているだけなので不要 ←必要なのはこの行だけ ←決まり文句

Sheets("Sheet1").Copy after:=Sheets (1)

シート名（Sheet1）をコピーする　１番目のシートのあとに

しかし、このままでは常に、Sheet1　を　１番目のシートの後にコピーしてしまいます。扱っているブックによってシートの枚数は違うことでしょう。シートの特定がポイントになります。

それでは、現在表示されているシートを、末尾にコピーするには？

現在表示されているシート　→　ActiveSheet　と書きます。

末尾へコピー　→　シートの枚数を数えると考えます　→　Sheets.Count

```
Sub シートの末尾へコピー()
'アクティブシートをコピーする　シートの枚数　番目のシートのあとに
    ActiveSheet.Copy after:=Sheets(Sheets.Count)
End Sub
```

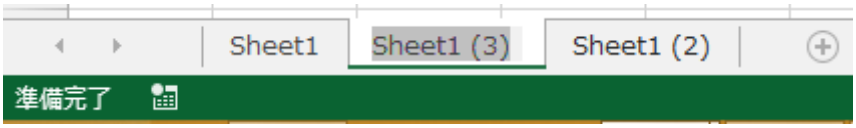
【練習】マクロの記録を使って、シート名の変更を行ってください。
下記に操作方法を示しますが、できそうな方は見ずにやってみてください。

マクロの記録　→　操作　→　記録終了

手順１　［開発］タブ　→　［コード］グループ　→　［マクロの記録］ボタン

手順２　マクロ名はそのままで（番号は忘れずに）　→　OK

手順３　シート名をクリックし、任意の名前へ変更（ここでは〈練習〉としました）



手順４　［開発］タブ　→　［コード］グループ　→　［記録終了］ボタン

記録されたコード

```
Sub Macro8()
'
' Macro8 Macro
'

'
    Sheets("Sheet1 (3)").Select
    Sheets("Sheet1 (3)").Name = "練習"
End Sub
```

もしくは

```
Sub Macro9()
'
' Macro9 Macro
'

'
    Sheets("Sheet1 (3)").Select
    Sheets("Sheet1 (3)").Name = "練習"
    Range("A1").Select
End Sub
```

３．マクロの記録の活用方法

マクロ記録とは、Excel に対して実際に行った操作を VBA のコードとして記録しますが、「わからないことを記録する」（調べる）ことに活用していくと知識の幅が広がります。

先ほど、マクロ記録の説明で、『不要な設定まで記録される』ことを申し上げました。

上記の例ですと

```
Sub Macro9()
    Sheets("Sheet1 (3)").Select                ←  この行はなくても動きます。
    Sheets("Sheet1 (3)").Name = "練習"
    Range("A1").Select                         ←  この行はなくても動きます。
End Sub
```

```
Sub Macro9()
    Sheets("Sheet1 (3)").Name = "練習"
End Sub
```

} この３行だけで実現できます。

Sheets("Sheet1 (3)").Name = "練習"　　は

シート名“Sheet1 (3)”　の名前を　練習　にする

という意味ですが、ここでは、シートの名前は　．Name　の部分だと推測できます。

ではこのまま再利用できるかというと、そうはいきません。

シート名“Sheet1 (3)”　が、存在しなかったら？？

既に、練習という名前のシートが存在していたら？？

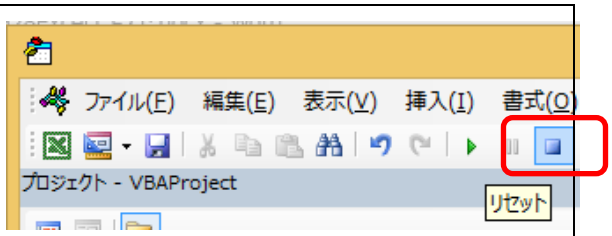
はい、エラーになります。

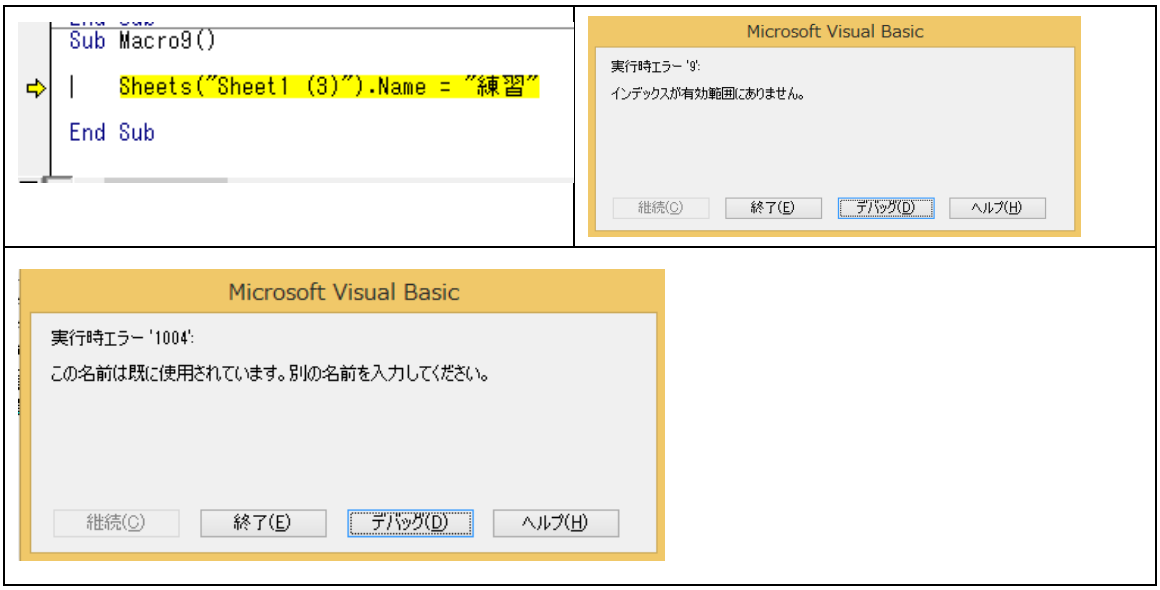
エラーになったとき、エラーの状態を

解除するには[リセット]　で　エラー状態を

回避することができます。

エラーになっても、壊れるわけではありませんので、慌てずに操作してください。





マクロの上達の方法に、

・やりたいことを紙に書く
・わからないことをマクロ記録する
・そのコードを直す

の繰り返しをしていくと書かれている書籍もあります。

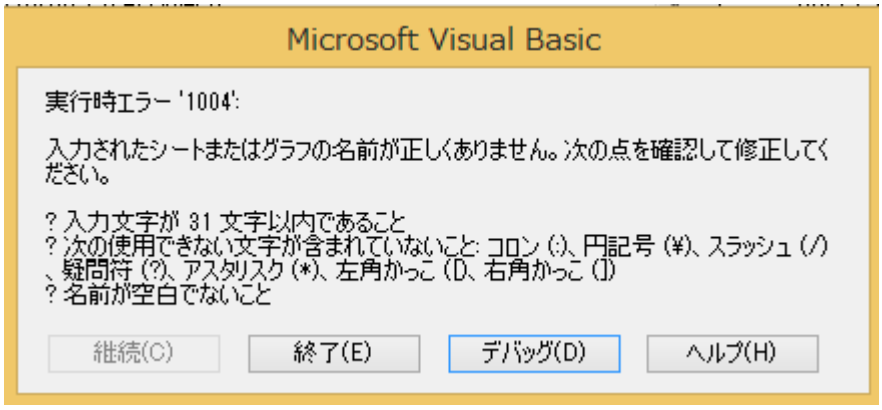
マクロの記録を否定しているわけではなく、記録マクロのコードをそのまま使うと、エラーになったとき基礎がわかっていないと直すこともできないということを申し上げたいのです。

やりたいこと	シートの名前を変えたい シート名は、セル A1 の文字列を使いたい
記録マクロしてみる	シートの名前は、.Name　とわかる
現在、選択しているシートは？	ActiveSheet　と書きます（基礎学習が必要）
セルの指定方法は？	Range（“A1”）　と書きます（基礎学習が必要）

上記のやりたいことをコードにすると次のようになります。

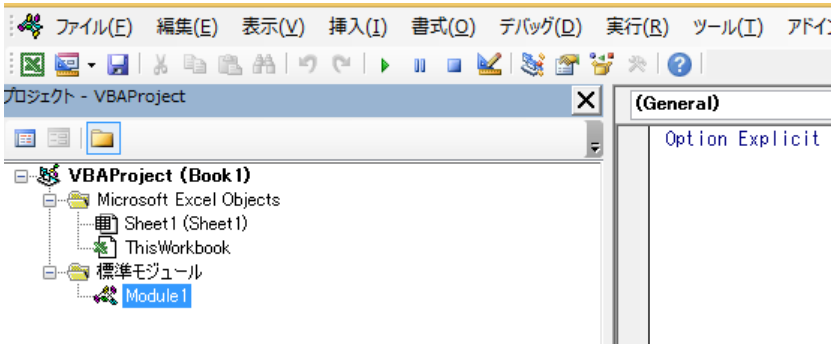
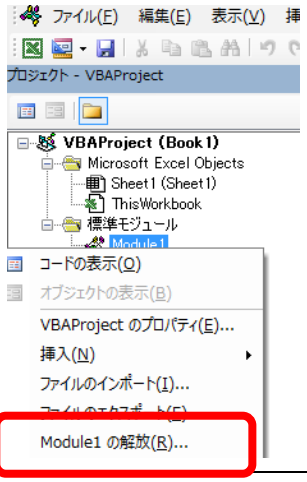
```
Sub セルA1をシート名に設定()
    'セルA1をシート名に設定
    ActiveSheet.Name = Range("A1").Value
End Sub
```

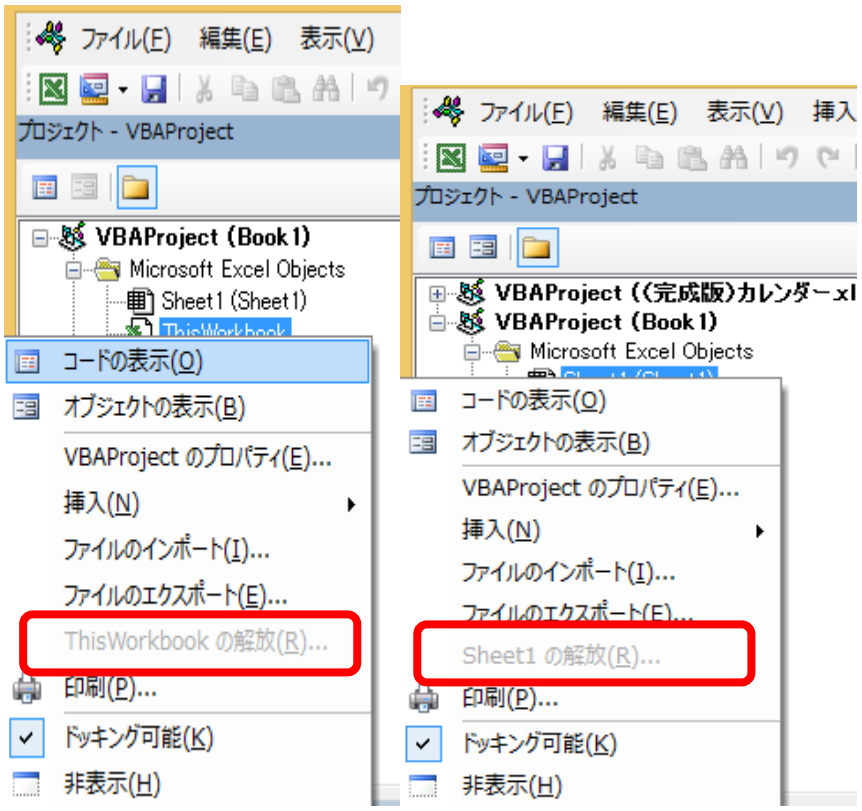
しかし、これももちろん完璧ではありません。同じブック内で、既に同じ名前があればエラーになりますし、シート名に使えない文字列が含まれているとエラーになります。



基礎の学習　＋　記録マクロの活用　＝　VBA の上達

　また、記録マクロで作られたコードは、ブック内の標準モジュールに記録されます。
本講座で書いていくコードもすべて、標準モジュールに書いてください。
標準モジュールは、特定のシートに関連付けられていないモジュールです。

標準モジュール	
解放できる （モジュールごとの削除）	

解放できない（グレーアウトになっています。）	

（３）　VBA の構文

VBA の構文は、次の 3 つに分類できます。

1．オブジェクト式	2．ステートメント	3．関数

1．オブジェクト式

「オブジェクト式」とは、Excel を操作する構文です。たとえば、セルに値を代入したり、ワークシートを挿入したり、ブックを開いたり・・・などなど。

オブジェクト式には次の 2 種類があります。

A)　対象．様子　＝　値	B)　対象．命令　：＝オプション
Sub セルA1をシート名に設定() 'セルA1をシート名に設定 ActiveSheet.Name = Range("A1").Value End Sub	Sub セルを削除左へシフト() 'セルを削除左へシフト Selection.Delete Shift:=xlToLeft End Sub
操作対象の様子や状況を設定するものです。	操作対象に何かのアクションを起こさせるものです。

２．　ステートメント

「ステートメント」とは、マクロの実行を制御する構文です。条件によって処理を分岐する If Then Else ステートメントや、繰り返しの For Next ステートメント、変数を宣言する Dim ステートメントなどなど。

オブジェクトに対し何かを働きかけるのではなく動きを制御します。

Dim	If　～　Else　～End	For　～　Next

マクロの記録では無理なところです。

３．関数

「関数」は、ワークシート上で使う SUM 関数や VLOOKUP 関数のように、決められた引数を与えると、いつも決められた結果を返すものです。

（４）　変数

１．変数の意味

変数とはデータなどを格納する一時的な記憶領域です。変数を使うときは必ず宣言をします。宣言時に適切な型が分からないときは型の指定を省略してもかまいません。

Dim　変数名　As　型

２．変数の宣言

宣言とは、「こういう変数を使いますよ」と記述することです。

型とは、種類はたくさんありますがすべて覚える必要はありません。

長整数型　Long　と　文字列　　String　を最初に覚えてください。

型の主な種類

長整数型	Long
文字列	String
日付型	Date
オブジェクト型	Object
バリアント型	Variant

変数の名前には次のような制限があります。

・文字（英数字、漢字、ひらがな、カタカナ）と「_　（アンダーバー）」が使えます。スペースや記号は使えません。

・文字の先頭は、英字、漢字、ひらがな、カタカナのいずれかです。

・変数名の長さは、255 文字以内です。

・VBA が使用する特別な語と同じ名前の変数は使用できません。
たとえば、「Next」はステートメントの一部なので使用できません。
・大文字・小文字の区別については、変数で定義したものに変わります。

例

```
Sub 変数()
    Dim i As Long
    Dim J As Long

    'Iと入力してもiに変わります
    i = 100
    'jと入力してもJに変わります
    J = 200

End Sub
```

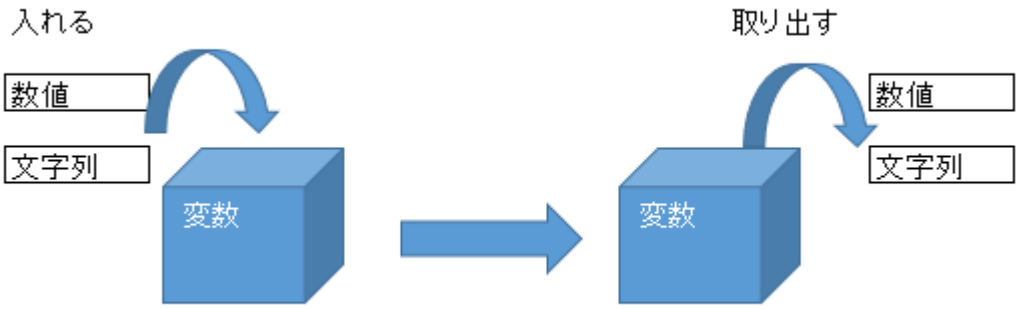
※変数名の付け方ポイント
　上記の制限を理解しつつ、自分なりのルールでつけてください。
日本語が、わかりやすいと感じれば、日本語で、
一文字目、大文字がわかりやすいと思えば、そのようにしていただければと思います。

よく使われている変数名

変数名	用途	語源
buf	中間処理などで一時的に格納するときなど	Buffer「緩衝器」
tmp	一時的に使用する変数など	Temporary「仮の」
i , j , k	For Next ステートメントのカウンター変数	Iteration「繰り返し」
cnt	件数などを数えるときの変数	Counter「カウンター」
msg、ms	ユーザーに対して表示する文字列を格納する変数	Message「メッセージ」
n	一時的な数値を格納する変数	Number「数」

３．変数の代入と取得
変数にいれる文字列や数値のことを値と呼びます。また、なんらかの値を変数にいれることを代入するまたは格納すると呼びます。

宣言　→　値を入れる　→　取り出す



使用例　1

```
Sub 変数練習()
    '文字列型の変数を宣言します
    Dim tmp As String
    '変数に文字列を格納します
    tmp = "エクセル"
    '変数の値をセルに代入します
    Range("A1").Value = tmp
End Sub
```

使用例　2

```
Sub 変数練習2()
    'staxとは、消費税率の意味です
    Dim stax As Long
    '消費税率8%に設定します
    stax = 8
    'セルA2へセルB2×1.08の数値を代入します
    Range("A2").Value = Range("B2").Value * (1 + stax / 100)
    'セルA3へセルB3×1.08の数値を代入します
    Range("A3").Value = Range("B3").Value * (1 + stax / 100)
End Sub
```

(1 + stax / 100)　の stax の部分は、変数を使わずに、

(1 +　8 / 100)でも同じ結果になります。

しかし、消費税率が１０％になったときを想像してみてください。

変数を使っていると、変更箇所は１か所で済みます。(stax　＝　8)

変数とは、値を一時的に保存しておき、必要なときに取り出せる一時記憶領域です。

使用例　3

やりたいこと	Msgbox へ表示	セルへ表示
処理のコード	Msgbox　“Hello“　&　i	Cells（i　，“A“）．Value
解説	メッセージボックスへ表示 i には、１～３が順番に代入されます。 数字が入るので、「Long」です。	セルへ表示 i には、１～３が順番に代入されます。 数字が入るので、「Long」です。
全体のコード	Sub 繰り返し() '変数定義　i 　Dim i As Long '1から3まで繰り返します 　For i = 1 To 3 'メッセージボックスへ表示します 　　MsgBox "Hello" & i 　Next i End Sub	Sub 繰り返し2() '変数定義　i 　Dim i As Long '1から3まで繰り返します 　For i = 1 To 3 'セルA1から繰り返します 　　Cells(i, "A").Value = i 　Next i End Sub

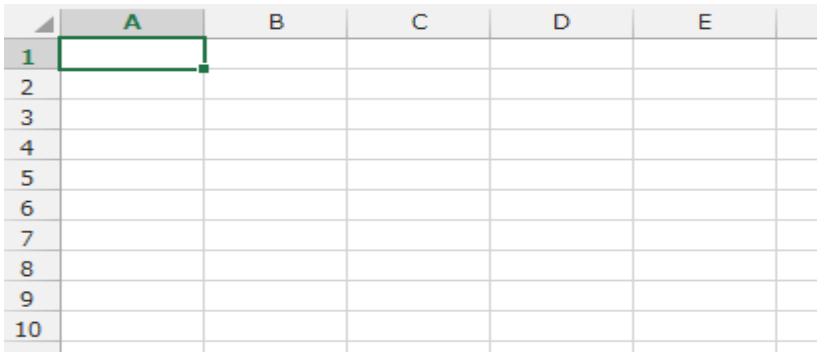
「　i　」だけで、セルを指定することはできません。
Cells を使うことにより、セルを指定することができます。

（５）　セルの操作

１．指定の方法

ExcelVBA で最も頻繁に行う操作の一つがセルの操作です。操作対象のセルを的確に指定するのがポイントです。

Range と Cells



ワークシート上	コード
セル A1	**Range（“A1”）** または Cells（１，１） または **Cells（１,“A”）**

セル A1 からセル B3	Range（“A1:B3”） または **Range（Cells（１,“A”），Cells（３，“ B”））**
行全体の指定	Rows
列全体の指定	Columns
アクティブセル	ActiveCell

Range と Cells　使い分けについて

コード	説明	例
Range（）	カッコ内のアドレスは、文字列形式で指定します。「“　ダブルクォーテーション）で囲む必要があります。	Range（”A1”）→セル A1 Range（“A1:B3”）→セル範囲 A1 から B3 Range（”A1,C5”）→セル A1 とセル C5
Cells（）	カッコ内は、行、列を指定します。 Cells（行，列）	Cells（1,1）→セル A1 若しくは Cells（1,”A”）　→セル A1

※変数を使ってセルを指定する場合は、Cells を使ってください。

行に変数を使った場合，　Cells（変数，”A”）がよく使われます。

Range（”A”&変数）でも動かないことはないですが、可読性の理由から、
おすすめしません。

２．プロパティ

Value プロパティは、セルの代入されている値を表します。
セルに数値や文字列を代入するときは、Value プロパティを代入します。
（プロパティとは、様子や状態のことです）

```
Sub 入力()
    Range("A1").Value = 100
End Sub


Sub 入力()
    Range("A1") = 100
End Sub
```

省略可能です。省略すると Value を指定したものとみなされます。

Value の省略について

よくある質問：Value は省略してもいいと説明がありましたが、どちらがいいのですか？

回答：省略した書き方の場合、省略していると認識していることが大事です。

セルに値を代入するときは、下記のように書きます。 例）セル A1 に、100 を代入する	
.Value　をつけた書き方	.Value　を省略した書き方
Sub テスト() 　Range("A1").Value = 100 End Sub	Sub テスト() 　Range("A1") = 100 End Sub

Value の省略は、意見が分かれるところです。また、明確な正解もありません。Value をつけたときと、省略したときで、若干マクロの実行速度が変わるそうですが、現在のパソコンでは誤差の範囲だそうです。値とは Value なのだから明記すべきだという意見はもっともですし、いや、省略した方が簡単なのだから省略すべきだという意見にも一理あります。

大事なことは **Value を省略している**と認識することです。
Value 省略した書き方は VBA のルールで Value を指定したことになると。
この講座では、Value　を省略しない書き方で進めていきます。

3．メソッド
（メソッドとは、動作を伴う命令のことです。）
覚えておきたいメソッド（選択・クリア・削除・コピー）

メソッド	使用例
Select	Sub メソッド1() 'セルA1を選択する Range("A1").Select End Sub
ClearContents	Sub メソッド2() 'セルA1をクリアする Range("A1").ClearContents End Sub
Delete	Sub メソッド3() 'セルA1を削除し、左方向へシフトします Range("A1").Delete Shift:=xlToLeft End Sub
Copy	Sub メソッド4() 'セルA1をセルB1へコピーします Range("A1").Copy Range("B1") End Sub

（6）　ステートメント

ステートメントとは、条件によって処理を変えたり、同じ処理を繰り返したり、マクロの動きを制御する命令です。

1．If ステートメント

特定の条件によって処理を変える

【書式１】	If　条件　Then　処理１
【書式２】	If　条件　Then 　　処理１ End　If
【書式３】	If　条件　Then 　　処理１ Else 　　処理２ End　If

【書式１】と【書式２】は実行する処理が１つしかないときの書き方です。条件が正しくなかったときは何もしません。実行する処理１の命令が１行で済むときは【書式１】で、処理１が複数行の命令の時は【書式２】を使います。

【書式３】は、条件が正しいとき処理１、正しくないとき処理２を実行します。

条件には「正しい」か「正しくないか」を指定します。

2．For　Next　ステートメント

同じ処理を指定の回数繰り返す

For　Next	For　変数名　＝　初期値　To　終了値 　　処理 Next　変数名
説明	For　Next ステートメントでは、変数を使います。この変数には繰り返されている回数が、数値で格納されます。
基本的な動作	Dim　i　As　Long For　i　＝　1　To　3 　　処理 Next　i
MsgBox へ表示	Dim　i　As　Long For　i　＝　1　To　3 　　MsgBox　“Hello”　&　i Next　i

セルへ表示	Dim　i　As　Long For　i　＝　1　To　3 　　Cells（i, “A”）.value　＝“Hello” Next　i
考え方	セルＡ1　→　Cells（1，“A”） セルＡ2　→　Cells（2，“A”） セルＡ3　→　Cells（3，“A”）
	1回目　→　Cells（1，“A”）→　Cells（i, “A”） 2回目　→　Cells（2，“A”）→　Cells（i, “A”） 3回目　→　Cells（3，“A”）→　Cells（i, “A”）
ポイント	If　と　For　Next を組み合わせることで より実用的なマクロを作り上げることができます。

ここは、非常に大事ですが難しい箇所です。
練習用のファイルを準備していますので、そちらでじっくり理解を深めていただきたいと思います。

ファイル名：講座練習用ファイル.xlsm

```
Sub カウンタ練習1()
    '変数iを定義
    Dim i As Long
    'iを1から10まで繰り返す
    For i = 1 To 10
        'セルA1に、 1ずつ加算する
        Range("A1").Value = i
    'Forの終わり
    Next i
End Sub
```

```
Sub カウンタ練習2()
    Dim i As Long
    For i = 1 To 10
        Cells(1, "A").Value = i  'RangeをCellsに
    Next i
End Sub
```

```
Sub カウンタ練習3()
    Dim i As Long
    For i = 1 To 10
        Cells(i, "A").Value = i  'cells(1 → cells(i ヘ
    Next i
End Sub
```

```
Sub カウンタ練習4()
    Dim i As Long
    For i = 1 To 10
        Cells(i, "B").Value = i  'Bの列で
    Next i
End Sub
```

```
Sub カウンタ練習5()
    Dim i As Long
    For i = 1 To 10
        Cells(i + 1, "C").Value = i 'Cの列で、2行目から入力
    Next i
End Sub
```

```
Sub カウンタ練習6()
    Dim i As Long
    Dim ofs As Long  '基準となる位置からの差  Offset

    ofs = 1  '1行ずらすの意味

    For i = 1 To 10
        Cells(i + ofs, "D").Value = i 'Dの列で、2行目から入力
    Next i
End Sub
```

```
Sub カウンタ練習7()
    Dim i As Long
    Dim ofs As Long  '基準となる位置からの差  Offset

    ofs = 2  '2行ずらすの意味

    For i = 1 To 10
        Cells(i + ofs, "E").Value = i  'Eの列で、3行目から入力
    Next i
End Sub
```

```
Sub カウンタ練習8()
    Dim i As Long   'カウンタ
    Dim ofs As Long '基準となる位置からの差  Offset
    Dim gm As Long  '月末

    ofs = 2
    gm = 31

    For i = 1 To gm
        Cells(i + ofs, "F").Value = i   'Fの列で、3行目から入力
    Next i
End Sub
```

```
Sub カウンタ練習9()
    Dim i As Long   'カウンタ
    Dim ofs As Long '基準となる位置からの差  Offset
    Dim gm As Long  '月末
    Dim mst As Long  '月初 monthstart

    ofs = 2
    gm = 31
    mst = Range("G1").Value

    For i = 1 To gm
        Cells(i + ofs, "G").Value = mst + i  'Gの列で、3行目から入力
    Next i
End Sub
```

```
Sub カウンタ練習10()
    Dim i As Long   'カウンタ
    Dim ofs As Long '基準となる位置からの差  Offset
    Dim gm As Long  '月末
    Dim mst As Long  '月初 monthstart

    ofs = 2
    gm = 31
    mst = Range("G1").Value

    For i = 1 To gm                         '↓の-1がポイントです
        Cells(i + ofs, "H").Value = mst + i - 1 'Hの列で、3行目から入力
    Next i
End Sub

'セルG1の値をH3に使うための－１
```

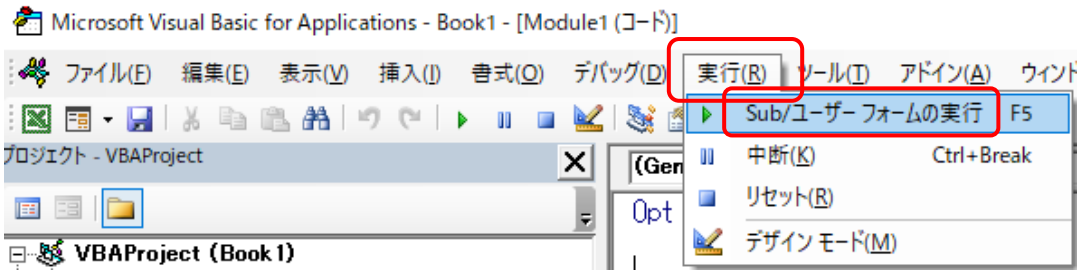
（７）　　マクロの実行

マクロの実行方法には主に下記の４つがあります。

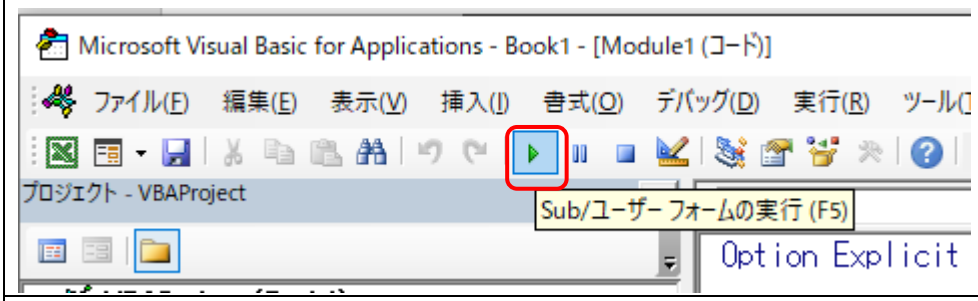
１．VBE から実行する
２．マクロ　ダイアログボックスから実行する
３．シート上のボタンから実行する
４．クイックアクセスツールバーから実行する

１．VBE から実行する

［実行］メニュー　　→　〔Sub/ユーザーフォームの実行〕をクリック



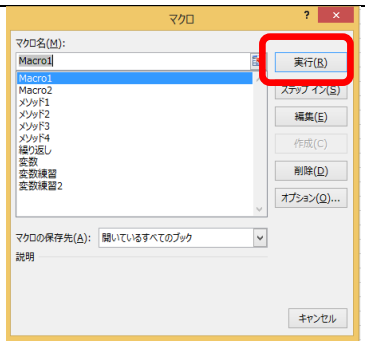
ツールバーの［Sub/ユーザーフォームの実行］ボタン



〔F５〕キーを押す

〔F８〕キーを押す（１行ずつ実行します。）

２．マクロ　ダイアログボックスから実行する

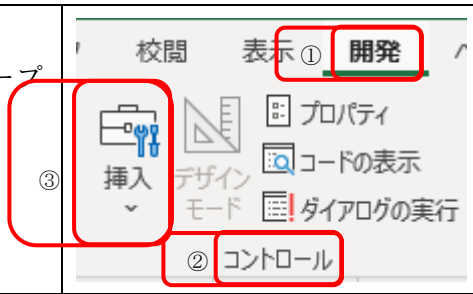
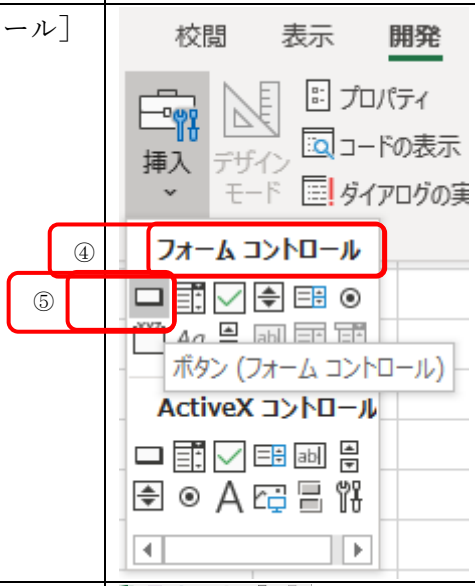
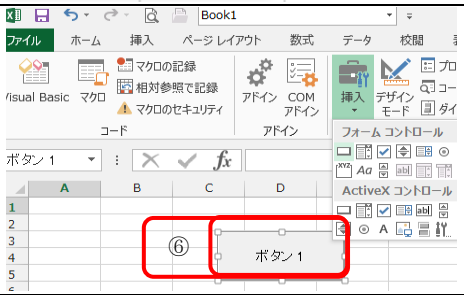
［表示］タブ →［マクロ］グループ →［マクロ］ボタン または ［開発］タブ →［コード］グループ →［マクロ］ボタン	［実行］ボタン	

参考：マクロ名に使える文字

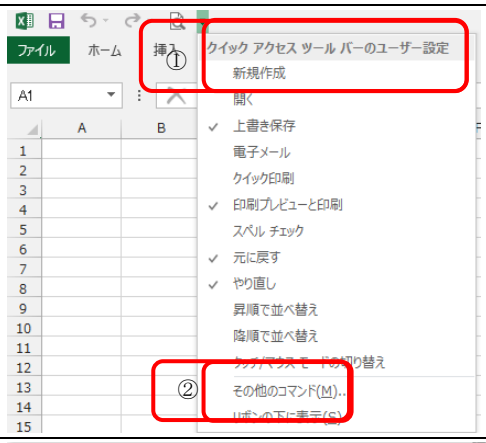
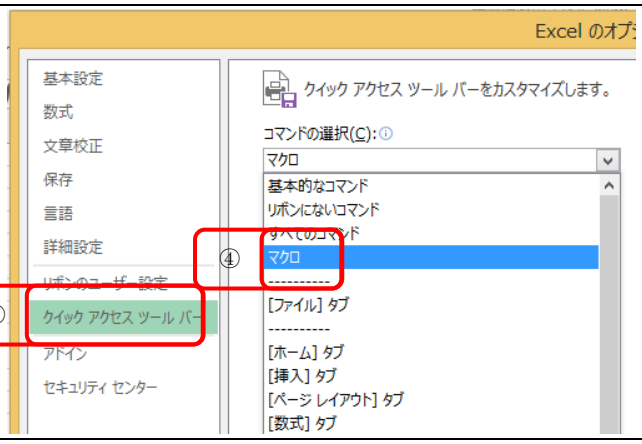
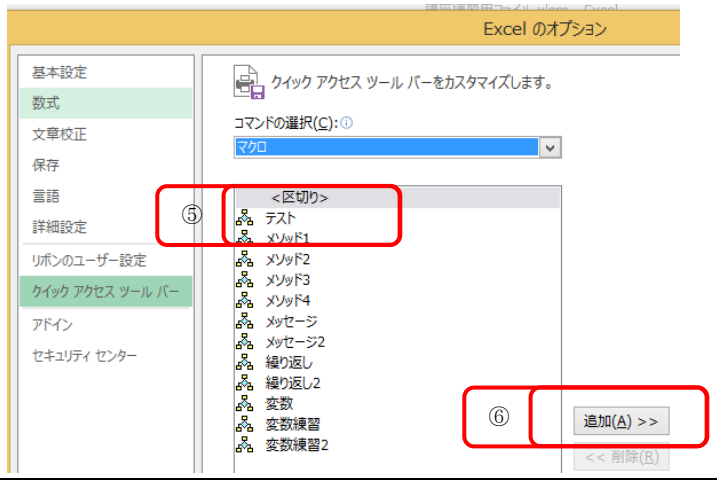
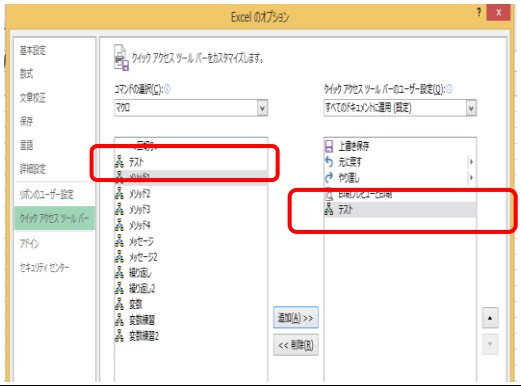
名前の先頭は、英文字、ひらがな、カタカナ、漢字です。

名前に、空白や「？」「＊」など記号が含まれてはいけません。

3．シート上のボタンから実行する

①［開発］タブ ②［コントロール］グループ ③［挿入］ボタン	
④［フォームコントロール］ ⑤［ボタン］ボタン	
⑥ボタンを配置します	

４．クイックアクセスツールバーから実行する

手順１ ①クイックアクセスツールバーの設定 ↓ ②その他のコマンド	
手順２ ③クイックアクセスツールバー ↓ ④コマンドの選択の下でマクロを選ぶと	
手順３ ⑤いままで書いたマクロがでてきますので、 追加したいマクロを選択して、 ⑥追加をクリック	
設定完了です。	

Lesson３　題材で使う VBA 以外の機能・関数

（１）　シリアル値

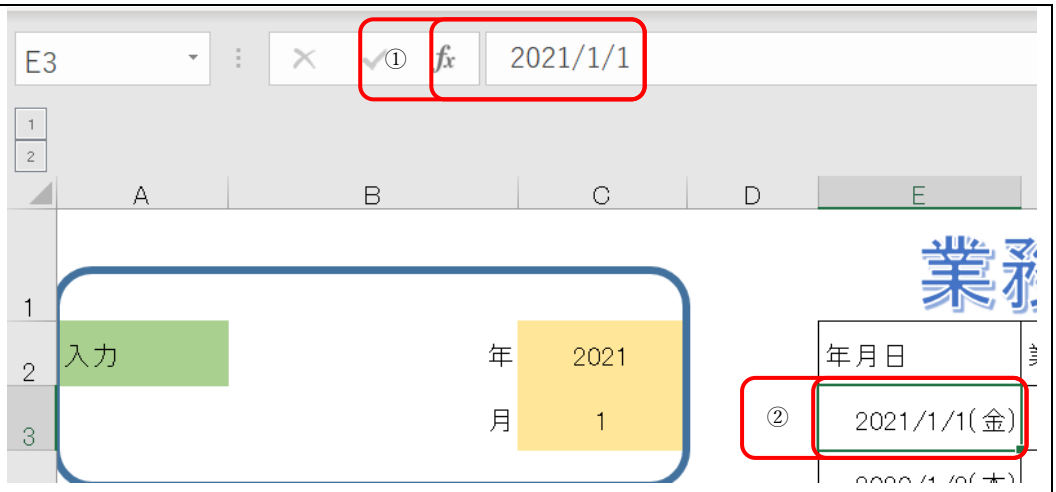
シリアル値とは、Excel において、日時を計算処理するために格納されている数値のことです。1900 年 1 月 1 日の午前 0 時を起点として、経過日数と時間を通算した値を示します。

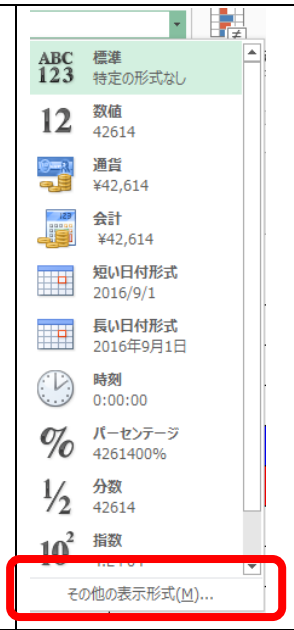
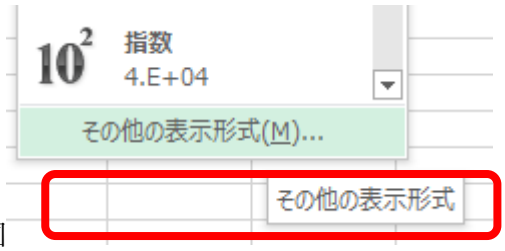
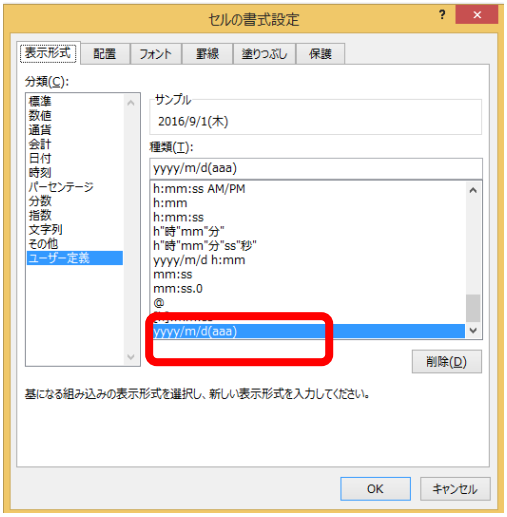
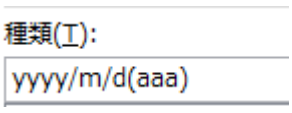
シリアル値は、1 日（24 時間）を 1.0 とし、1 日ごとに 1 ずつ増えます。整数部分は日付を、小数部分は時刻を表します。時刻は、1 日の 1 部分として、1 を 24 で割った数値となります。セルに日付を入力するとセルの表示形式が自動的に日付の形式となりますが、セルにはシリアル値が入力されています。シリアル値の数値を表示するには、セルの書式設定で表示形式を「標準」にします。

（２）　表示形式

表示形式 とは、セルに入力した数値データを、値を変えずに見た目のみを変えることができる形式のことです。

<table>
<tr><td>①数式バー

②セル E3</td><td> 
</td></tr>
<tr><td>セル E3 の
入力値とみため</td><td>入力されている値　→　2021/1/1
みため　→　2021/1/1(金)</td></tr>
</table>

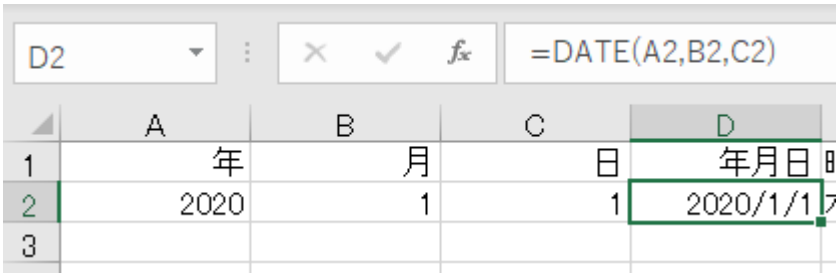
表示形式	 拡大図 
表示方法	 拡大図 

（３）　関数

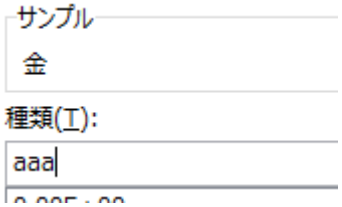
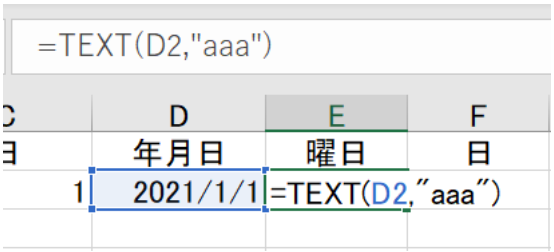
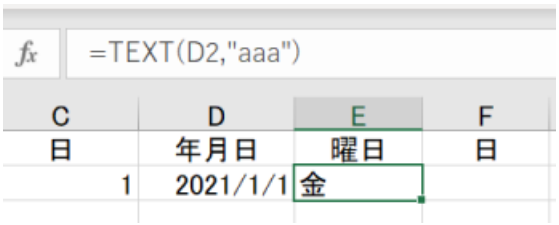
今回使用した関数は下記の４つです。

１．DATE	２．TEXT	３．DAY	４．IF

１．DATE

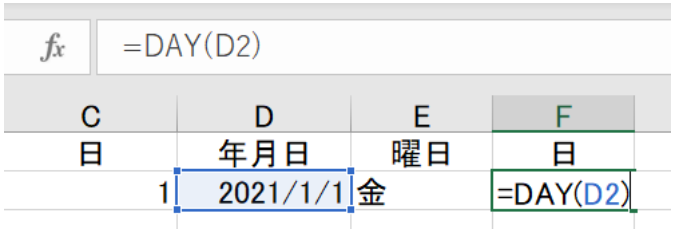
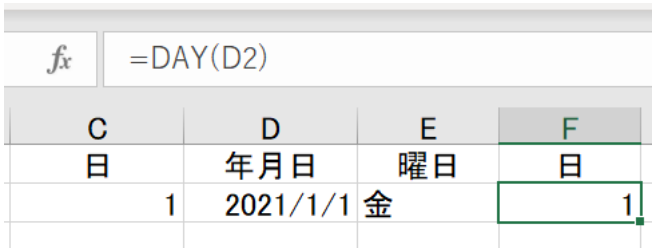
関数名	DATE
書式	DATE(年,月,日)
説明	特定の日付を表す連続したシリアル値を返します。
引数	年 必ず指定します。年引数には、1 ～ 4 桁で年を指定します。標準では 1900 年日付システムが使われます。最初の日付は 1900 年 1 月 1 日です。 ヒント: 不適切な結果が生成されるのを防ぐため、年引数には 4 桁の数値を使用してください。たとえば、"07" を使用すると、年の値として "1907" または "2007" が返されます。4 桁の年を使用すれば混乱が防げます。
	月 月を表す正または負の整数を指定します。ただし、返される値の範囲は 1 ～ 12（1 月から 12 月）になります。
	日 日を表す正または負の整数を指定します。
	上記、引数は Excel ヘルプから引用しました。詳しくはヘルプをご覧ください。
使用例	

2．TEXT

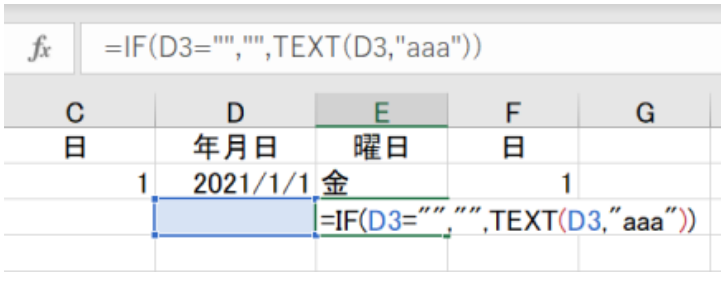
関数名	TEXT
書式	TEXT（値，表示形式）
引数	**値**　必ず指定します。 数値、数値に評価される数式、または数値を含むセルへの参照を指定します。 **表示形式**　必ず指定します。 数値書式を、"m/d/yyyy" や "#,##0.00" など、引用符で囲んだテキスト文字列として指定します。
表示形式のサンプル	
入力例	
結果	

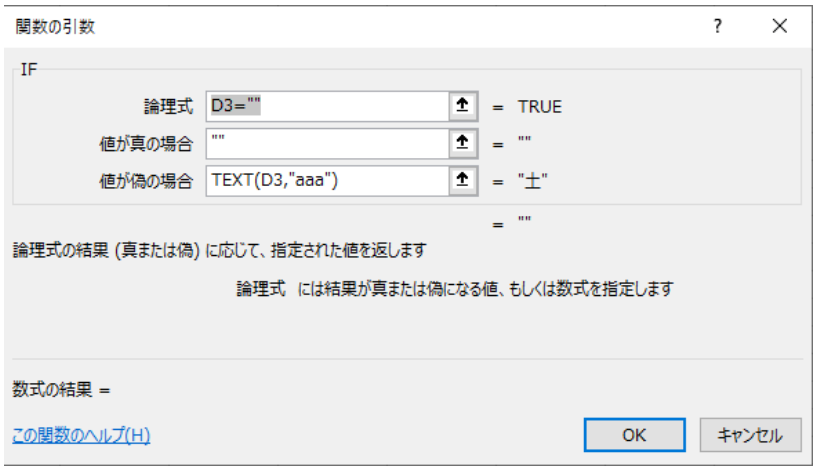
3．DAY

関数名	DAY
書式	DAY(シリアル値)

説明	シリアル番号で表された、日付の日情報を返します。 日情報は 1 ～ 31 の範囲内の整数で示されます。
引数	シリアル値　　検索する日付を指定します。
入力例	
結果	

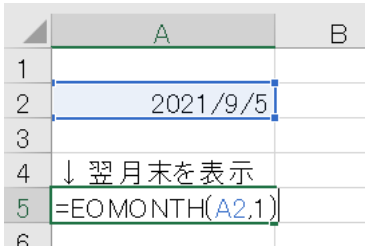
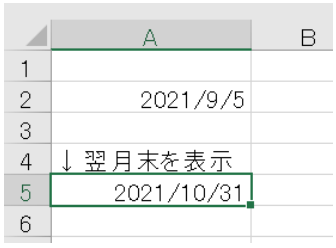
4．IF

関数名	IF
書式	IF(論理式, [真の場合], [偽の場合])
説明	IF 関数は、指定された条件を評価した結果が TRUE の場合はある値を返し、評価した結果が FALSE の場合は別の値を返します。
引数	**論理式** 真または偽のどちらかに評価できる値または式を指定します。
	真の場合 "***論理式***" 引数が TRUE に評価された場合に返す値を指定します。
	偽の場合 "***論理式***" 引数が FALSE に評価された場合に返す値を指定します。
使用例	

ダイアログボックス	

5．EOMONTH

題材では使用していませんが、月末を求めることのできる関数です

関数名	EOMONTH（エンドオブ・マンス）
書式	EOMONTH（開始日，月）
説明	開始日から起算して、指定された月数だけ前または後の月の最終日に対応するシリアル値を返します。
引数	開始日　必ず指定します。 起算日を表す日付を指定します。 日付は、DATE 関数を使って入力するか、他の数式または他の関数の結果を指定します。 たとえば、2008 年 5 月 23 日を入力する場合は、DATE(2008,5,23) を使用します。 月　必ず指定します。 開始日から起算した月数を指定します。 "月" に正の数を指定すると起算日より後の日付を返し、負の数を指定すると起算日より前の日付を返します。
使用例及び結果	 

Lesson４　題材コードの解説

（１）　シートコピー

行	コード
1	Sub シートコピー()
2	Dim ms As Long
3	ms = MsgBox("コピーしますか？", vbYesNo + vbQuestion, "確認")
4	If ms = vbYes Then
5	ActiveSheet.Copy After:=Sheets("マスター")
6	MsgBox "コピーしました"
7	Else
8	MsgBox "コピーしませんでした"
9	End If
１０	End Sub
行	解説
1	マクロ　シートコピー　の始まりです。
2	変数を定義します。　ms ［メッセージ］
3	メッセージボックスを表示します。
4	はい（Y）なら、
5	アクティブシートをシート名：「マスター」の後ろにコピーします。
6	コピー後にメッセージ表示します。
7	処理が２つある時の、If　と　End　If　をつなぐ決まり文句です。
8	いいえ(N)なら、なにもせず、メッセージ表示するのみです。
9	If の終わりの決まり文句です。
１０	マクロの終わりです。

（２）　月の変更

行	コード
1	Sub 月の変更()
2	MsgBox "セル C3 を選択して、月の変更をしてください"
3	Range("C3").Select
4	End Sub
行	説明
1	マクロ　月の変更　の始まりです。
2	メッセージボックスを表示します。
3	セル C3 を選択し,月を変更します。
4	マクロの終わりです。

（３）　　シート名変更

行	コード
1	Sub シート名変更()
2	ActiveSheet.Name = Range("C2").Value & "年" & Range("C3").Value & "月"
3	End Sub
行	説明
1	マクロ　シート名変更　の始まりです。
2	セル C2 とセル C3 の値を使い,アクティブシートの名前を変更します。
3	マクロの終わりです。

（４）　　月の更新

行	コード
1	Sub 月の更新()
2	Dim irow As Long
3	Dim ofs As Long
4	Dim gm As Long
5	Dim mst As Long
6	Range("E3:H33").ClearContents
7	ofs = 2
8	mst = Range("C6").Value
9	gm = Range("C9").Value
10	For irow = 1 To gm
11	Cells(irow + ofs, "E").Value = mst + irow – 1
12	Next irow
13	End Sub
行	説明
1	マクロ　月の更新　の始まりです。
2	変数を定義します。　irow　［i カウント　row　行］
3	変数を定義します。　ofs　［基準となる位置からの差　Offset］
4	変数を定義します。　gm　［月末］
5	変数を定義します。　mst　［ monthstart　］
6	入力されている内容をクリアします。
7	基準となる位置からの差を２とします。
8	変数 mst に、セル C6 の値を代入します。
9	変数 gm に C9 を代入します。
10	irow を 1 から月末まで繰り返します。
11	E 列の該当行に、日付を代入していきます。

１２	For の終わり
１３	マクロの終わりです。

（５）　　土日判定

行	コード
1	Sub 土日判定()
2	Dim irow2 As Long
3	Range("E3:I33").Interior.ColorIndex = xlNone
4	For irow2 = 3 To 33
5	If Cells(irow2, "I").Value = "日" Then
6	Range(Cells(irow2, "E"), Cells(irow2, "I")).Interior.Color = RGB(255, 100, 100)
7	End If
8	If Cells(irow2, "I").Value = "土" Then
9	Range(Cells(irow2, "E"), Cells(irow2, "I")).Interior.Color = RGB(100, 100, 255)
１０	End If
１１	Next irow2
１２	End Sub
行	説明
1	マクロ　土日判定　の始まりです。
2	変数を定義します。　irow2　［i カウント　row　行］
3	セル E3 からセル I33 の色を塗りつぶしなしにします。
4	３行目から３３行目まで繰り返します。
5	もし、I 列が,"日"だったら、
6	E から I までの塗りつぶし色をうすい赤に設定します。
7	if の終わりです。
8	もし、I 列が、"土"だったら、
9	E から I までの塗りつぶし色をうすい青に設定します。
１０	if の終わりです。
１１	For の終わりです。
１２	マクロの終わりです。

（６）　　シート削除

行	コード
1	Sub シート削除()
2	Dim ms2 As Long
3	ms2 = MsgBox("このシートを削除しますか？", vbYesNo + vbQuestion, "確認")
4	If ms2 = vbYes Then
5	ActiveSheet.Delete
6	MsgBox "削除しました"
7	Else
8	MsgBox "削除しませんでした"
9	End If
１０	End Sub
業	説明
1	マクロ　シート削除　の始まりです。
2	変数を定義します。　ms2 ［メッセージ］
3	メッセージボックスを表示します。
4	はい(Y)なら、
5	アクティブシートを削除します。
6	コピー後にメッセージ表示します。
7	いいえ(N)なら、
8	なにもせず、メッセージ表示するのみです。
9	If の終わりです。
１０	メッセージボックスを表示します。

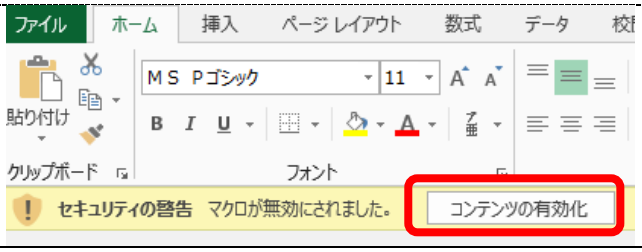
Lesson５　注意点・その他

（１）　VBA　共通編

１．ブックを開くとき

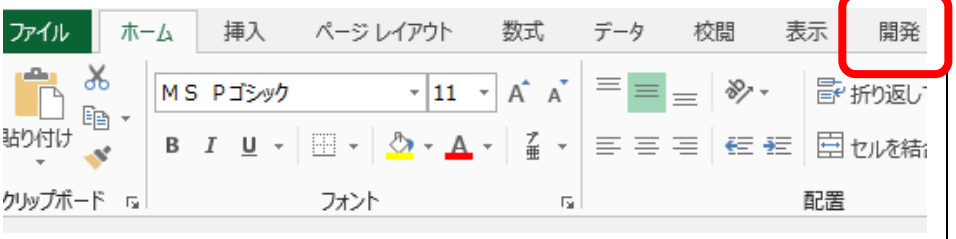
マクロを含むブックを開くと、下記のメッセージが表示されることがあります。

コンテンツの有効化 にして進んでください。



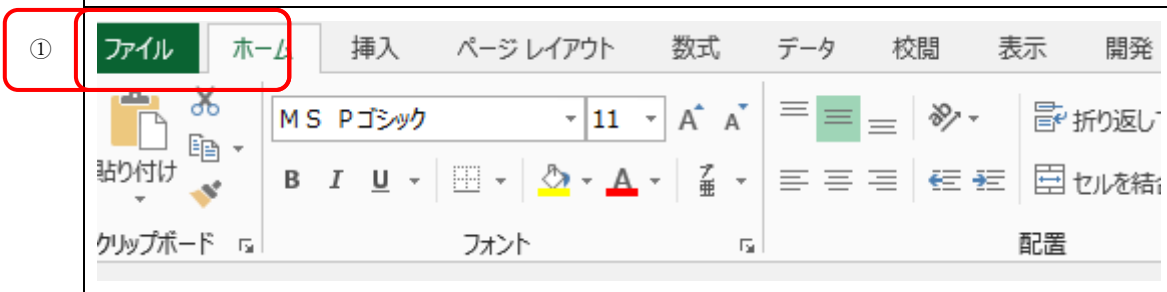
２．毎回の作業　（その１）

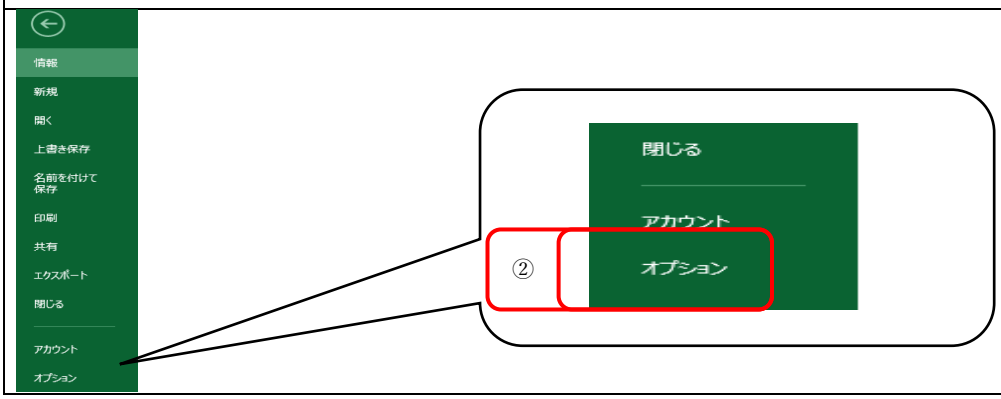
開発タブの表示（講座で使用するパソコンに関して毎回この操作が必要です。）

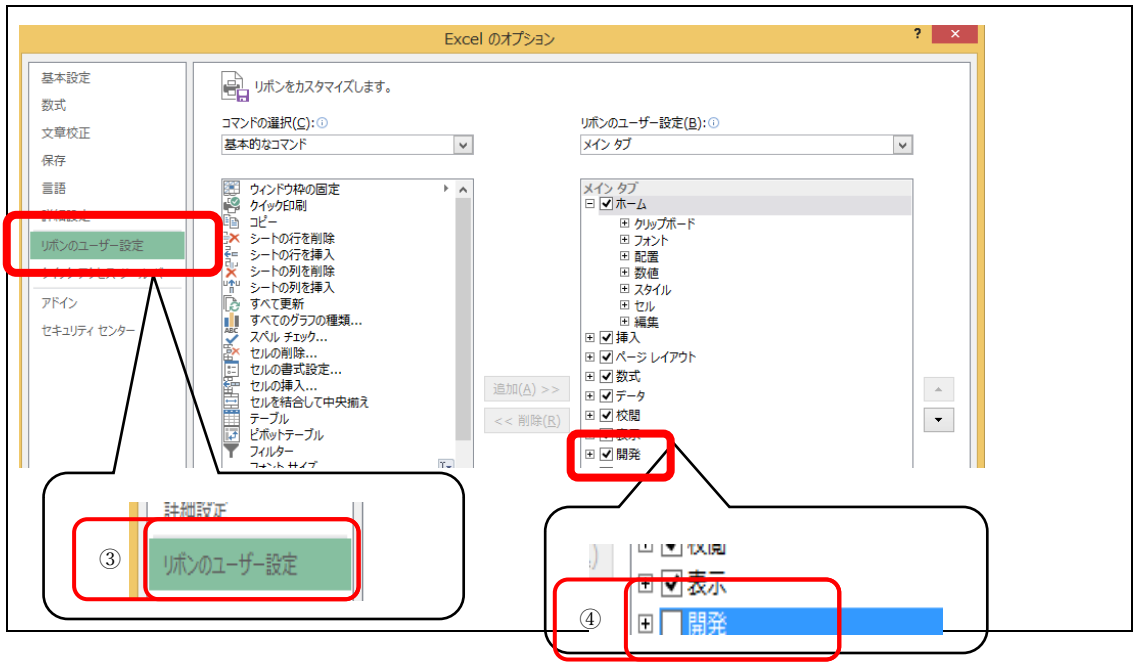


手順

①［ファイル］タブ　②オプション　③リボンのユーザー設定　④開発タブにチェック





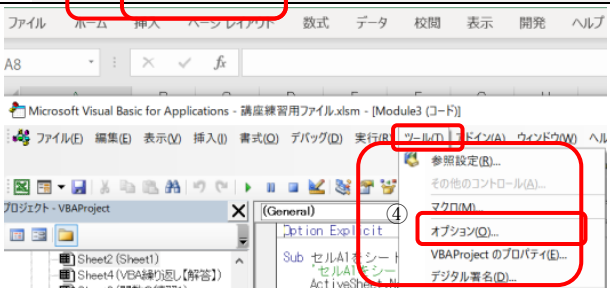


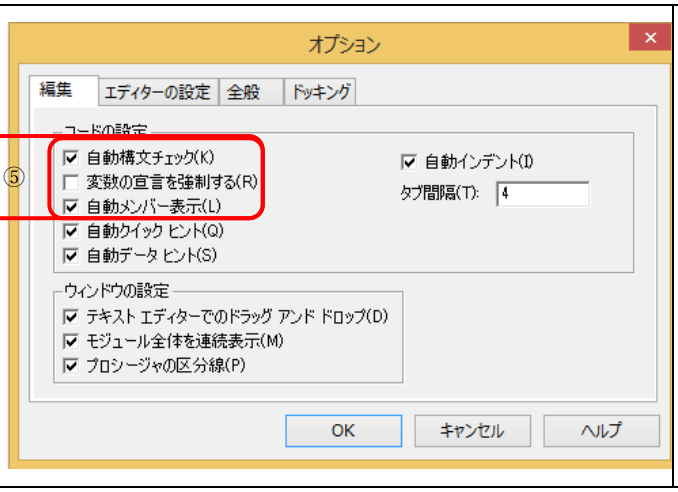
3．毎回の作業　（その２）

変数の宣言を強制する

①［開発］タブ　②［コード］グループ　③Visual Basic　④ツール　オプション

⑤　変数の宣言を強制する





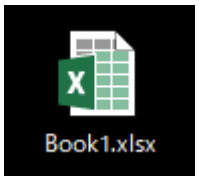
自動で追加されます。

```
Option Explicit
```

4．拡張子の表示方法

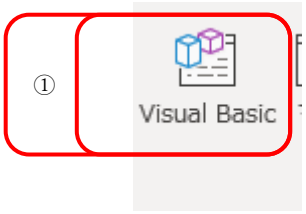
ファイル名の『.』以降を、拡張子と呼びます。 右の例では、『xlsx』です。	
表示方法 タスクバーの『エクスプローラー』をクリックし、	
【表示】タブから、	
ファイル名拡張子のチェックを入れると表示されます。	

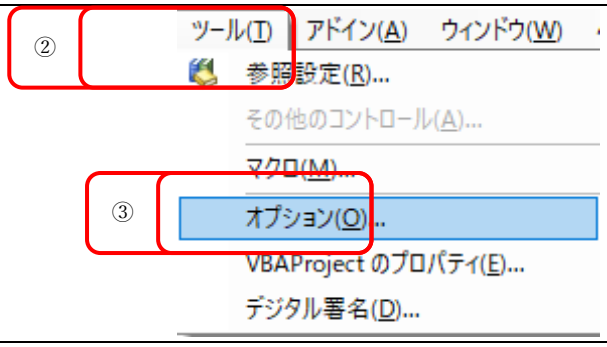
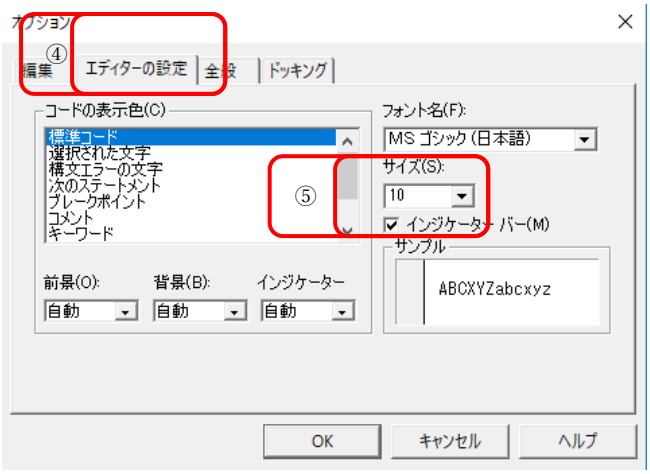
5．　ファイルのアイコン

マクロ入りのファイルは、 感嘆符　《！》がついています。 拡張子は、『xlsm』です。	
マクロが入っていないと、 感嘆符　《！》がつかず、 拡張子は、『xlsx』です。	

（2）　　VBA　コード編

1．コード内のフォント大きさ

コードウインドウの文字サイズの変更方法です ①［Visual　Basic］ボタン　から	

②［ツール］タブ ③［オプション］ と進み	
④［エディターの設定］タブ ⑤サイズ です。	

2．　コード内のスペースについて

VBA で、マクロのコードとして認識される語は、次のように分類されます。

1．VBA で定められている単語	（例）　Range ，　Dim ，　If
2．変数名	（例）　ｍｓ ，　irow
3．一部の記号	（例）　（）カッコ　，　．ピリオド ，ｶﾝﾏ　，　＊ ｱｽﾀﾘｽｸ
4．数値	（例）　１，　２，　３，　４

「1.VBA で定められている単語」は、オブジェクトやプロパティやメソッドや、ステートメントや関数などです。「2.変数名や定数名」は、上記のように、あらかじめ VBA で定義されている定数だけでなく、ユーザーが自由に定義できる変数名や定数名もあります。文字列の"Sheet1"とか"A1"などは、「3.一部の記号」で使用できる""で囲まれた任意の文字列です。スペースも、この「3.一部の記号」に含まれます。

「1.VBA で定められている単語」と「2.変数名や定数名」を合わせて「単語」と考えれば、スペースが使われるのは「単語」「記号」「数値」を区切るところです。

つまり、

単語と単語の間	単語と記号の間	単語と数値の間	記号と数値の間	記号と記号の間

ということです。基本的に、スペースは**VBE が自動的に調整してくれます。**スペースが足りないときは自動的に挿入してくれますし、スペースが多すぎるときは 1 つにまとめてくれます。ですから、それほど意識をする必要はないのですが、いくつかエラーになるケースがありますので、ご紹介します。

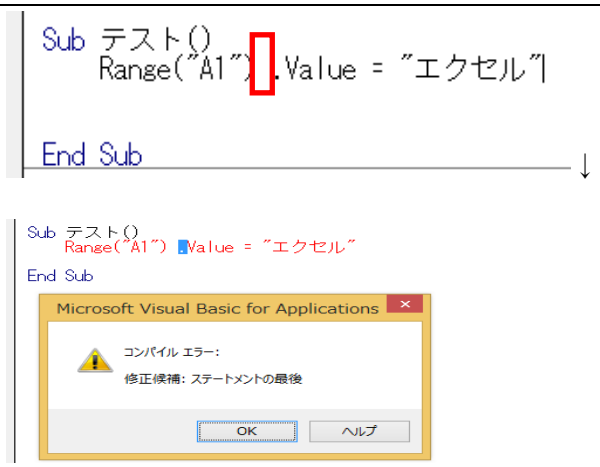
オブジェクト式はスペースを使わない

オブジェクト式とは、Excel を操作するような命令です。セルに値を代入するとか、新しいワークシートを挿入するとか、ブックを開くとか、グラフを作るとか・・・このようなとき、オブジェクトやプロパティやメソッドを、ピリオドで区切ります。このとき、**単語とピリオドの間**には、スペースを入れません。

正しい：○○.△△.××　　誤　り：○○.□△△.□××

ちなみに、ピリオドの**右**に入れたスペースは、行末で Enter キーを押したときに、VBE が自動的に削除してくれますが、ピリオドの**左**にスペースが入っていると、行末で Enter キーを押したとき、エラーになります。

ピリオドの右にスペース Enter キーを押すと エラーになりません	Sub テスト() 　Range("A1"). Value = "エクセル" End Sub Sub テスト() 　Range("A1").Value = "エクセル" End Sub

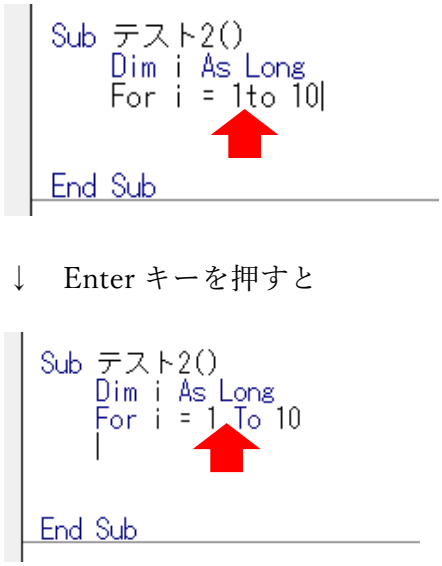
ピリオドの左にスペース Enter キーを押すと エラーになります	

For Next は To の右にスペースを入れる

繰り返しのステートメント For Next は、次のように書きます。

For 変数 ＝ 初期値 To 終了値

それぞれ、単語や記号、数値はスペースで区切られています。このとき、To の左にスペースを入れ忘れたときは、行末で Enter キーを押したとき、VBE が自動的に補完してくれます。しかし、To の右にスペースがないと、行末で Enter キーを押したとき、エラーになります。

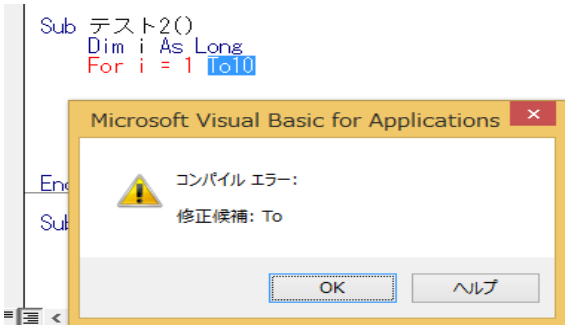
For に続く To の**前**スペースなしで、10 の後に、Enter を押すと エラーになりません	

For　に続く　To の後スペースなしで、10 の後に、Enter を押すと

エラーになります

```
Sub テスト2()
    Dim i As Long
    For i = 1 To10
End Sub
```

↓　Enter キーを押すと



演算子の両側にはスペースを入れる

数値を計算する＋や＊などの記号を演算子と呼びます。演算子の両側にはスペースを入れます。もしスペースを入れ忘れたときは、行末で Enter キーを押したとき、VBE が自動的に補完してくれます。

```
Sub 変数練習2()
    Dim stax As Long
    stax = 8

    Range("A2").Value = Range("B2").Value * (1+stax/100)
End Sub
```

↓　Enter キーを押すと

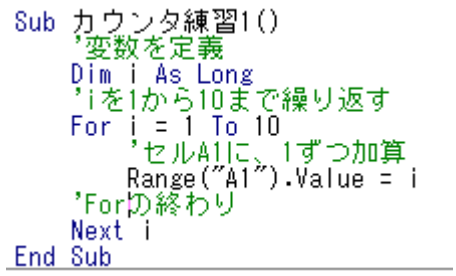
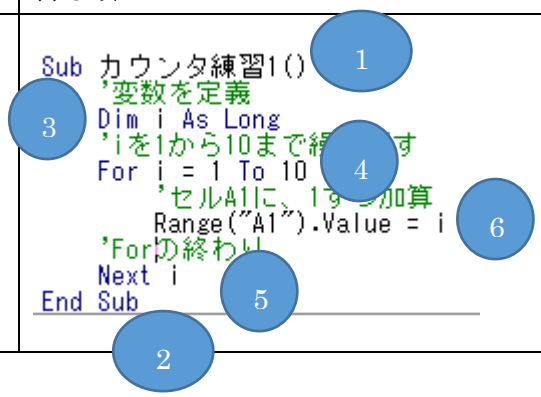
```
Sub 変数練習2()
    Dim stax As Long
    stax = 8

    Range("A2").Value = Range("B2").Value * (1 + stax / 100)

End Sub
```

3．コードの書き方のコツ

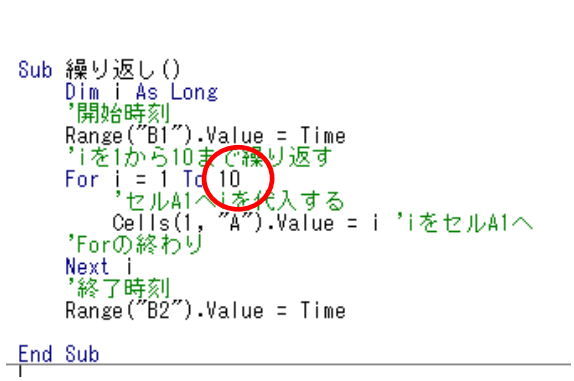
わかっているところから書いていくことです

実際のコード	書き順
	

```
Sub カウンタ練習1()
    '変数を定義
    Dim i As Long
    'iを1から10まで繰り返す
    For i = 1 To 10
        'セルA1に、1ずつ加算
        Range("A1").Value = i
    'Forの終わり
    Next i
End Sub
```

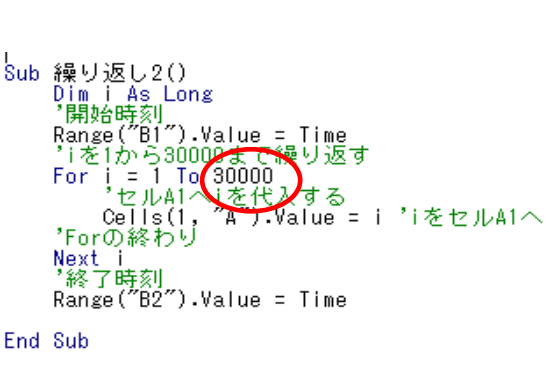

上記のコードは、10 なので、F５で実行すると、高速で『1，２，３・・・』『10』となります。

10 を 30000 に変えて実行すると・・・



```
Sub 繰り返し()
    Dim i As Long
    '開始時刻
    Range("B1").Value = Time
    'iを1から10まで繰り返す
    For i = 1 To 10
        'セルA1へiを代入する
        Cells(1, "A").Value = i 'iをセルA1へ
    'Forの終わり
    Next i
    '終了時刻
    Range("B2").Value = Time

End Sub
```



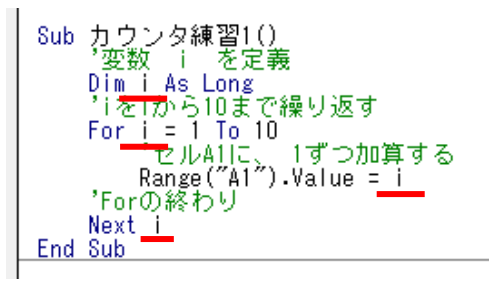
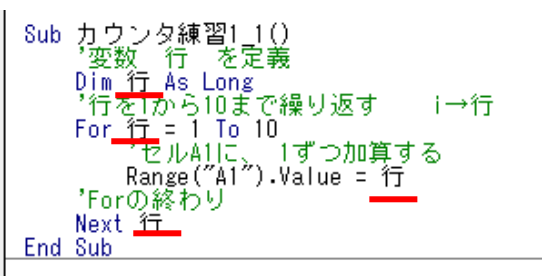
```
Sub 繰り返し2()
    Dim i As Long
    '開始時刻
    Range("B1").Value = Time
    'iを1から30000まで繰り返す
    For i = 1 To 30000
        'セルA1へiを代入する
        Cells(1, "A").Value = i 'iをセルA1へ
    'Forの終わり
    Next i
    '終了時刻
    Range("B2").Value = Time

End Sub
```

4．変数名

Q　：変数ｉ　ｊ　ｋ　とありますが、いまいちピンときません。
A　：ｉ　とは、　iteration（**イテレーション**）反復、繰り返しという意味の英単語の頭文字という説や、INDEX の頭文字という説があるようです。 ｊ ｋはｉの次のアルファベットだから。キーボードも近い位置にあるから。 わかりにくいと思われる方は、日本語でもよいです。

下記は、同じ動きをします。

変数　ｉ	変数　行
	

```
Sub カウンタ練習1()
    '変数　ｉ　を定義
    Dim i As Long
    'iを1から10まで繰り返す
    For i = 1 To 10
        'セルA1に、1ずつ加算する
        Range("A1").Value = i
    'Forの終わり
    Next i
End Sub
```

```
Sub カウンタ練習1_1()
    '変数　行　を定義
    Dim 行 As Long
    '行を1から10まで繰り返す　　i→行
    For 行 = 1 To 10
        'セルA1に、1ずつ加算する
        Range("A1").Value = 行
    'Forの終わり
    Next 行
End Sub
```

（３）　　VBA　実行編

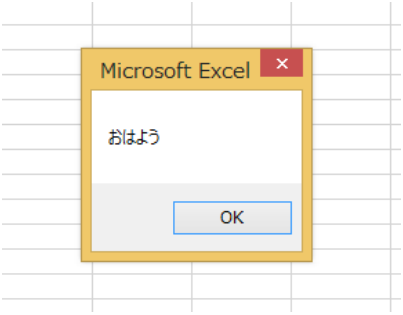
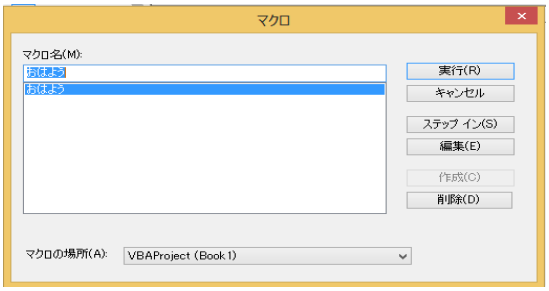
１．マクロを実行した時に選択画面が出るケース

〔F5〕でマクロを実行しようとした時に、そのまま実行されるケースと選択画面が出るケースがあります。

<table>
<tr><td colspan="2">

```
Option Explicit

Sub おはよう()
    MsgBox "おはよう"
End Sub
```

</td></tr>
<tr><td>そのまま実行
 
</td><td>選択画面
 
</td></tr>
<tr><td colspan="2">カーソルの位置が関係しているようです。</td></tr>
<tr><td>コード外（下）・・・マクロ実行

コード内・・・マクロ実行</td><td>コード外（上）・・・・選択画面

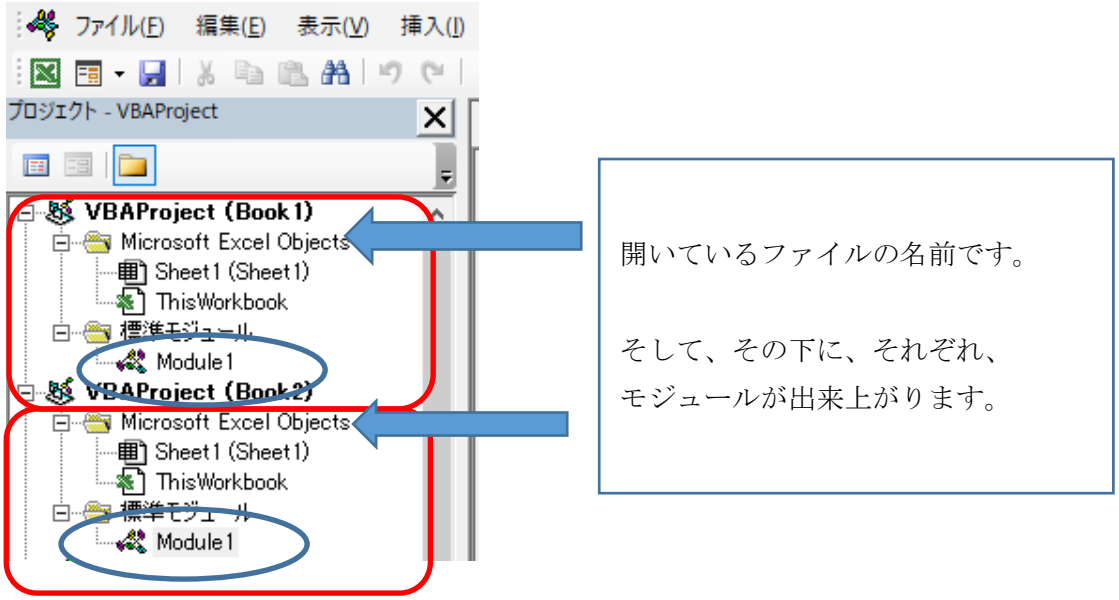
モジュール・・・選択画面</td></tr>
</table>

２．図形にマクロを登録

手順１	手順２	手順３
［挿入］タブ →［図］グループ →［図形］ボタン と進み、どれか一つを選択し	配置後、右クリック、	マクロの登録 登録したいマクロ名を選択し、 OK

3．　複数のファイルを開いたときのモジュール

複数のファイルを開くと、それぞれにモジュールが出来上がります。



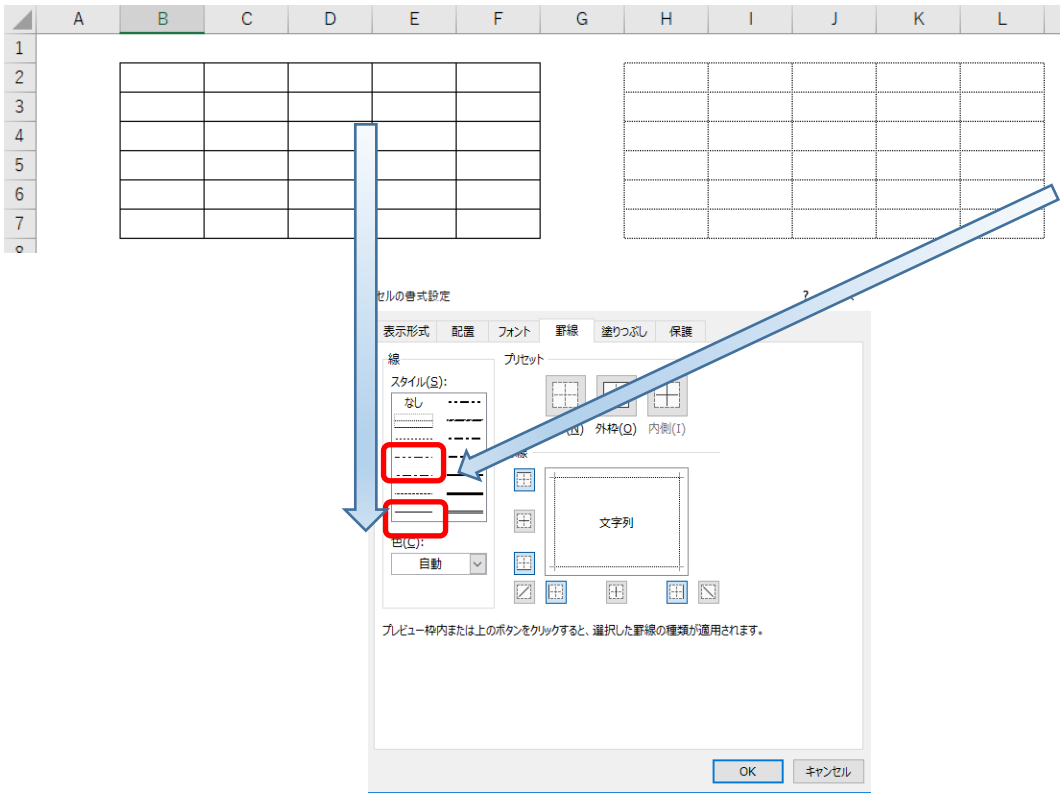
開いているファイルが表示されるので、わかりにくい方は、不要なファイルを閉じておくとよいです。

（4）　　マクロ記録　編

1．『マクロの記録』とは・・・

(例)　マクロの記録　『罫線を引く』

Excel 際に行った操作を VBA のコードとして記録する機能です。



左は、【ホーム】タブのフォント　→　罫線　→　格子で、3 手間

右は、【ホーム】タブのフォント　→　罫線　→　その他の罫線　→　線のスタイルから、細い線を選び

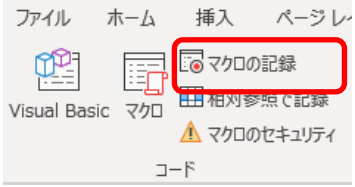
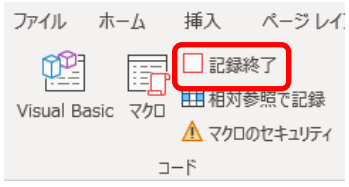
→　外枠と内側　→　OK　で、6 手間

『マクロの記録』だと、一度その手間をかけるだけで、その操作を、

【開発】タブ　→　マクロ　→　細い罫線　→　実行　で、4 手間

頻繁に行う操作を記録しておくと、手間の削減になります。

2．マクロの記録ボタン　と　記録終了ボタン

両者は同じボタンです。記録を開始すると、そのボタンが記録終了に変わります。	
	

（５）　　機能編

１．色の仕組み　RGB

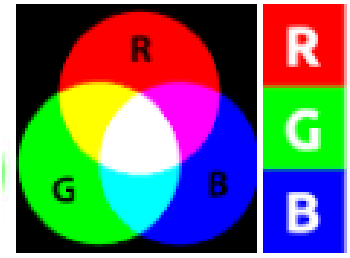
BGB（red, green, blue）　　RGB（ 赤　，　緑　，青）

指定項目	説明
red	色の赤の成分を表す 0 ～ 255 の範囲内の数値を指定します。
green	色の緑の成分を表す 0 ～ 255 の範囲内の数値を指定します。
blue	色の青の成分を表す 0 ～ 255 の範囲内の数値を指定します。

VBA で、セルの塗りつぶしを設定する場合、下記のように書きます。

Range("A1").Interior.Color = RGB(0, 0, 0)　　'黒　を　設定

色	赤の値	緑の値	青の値
黒	0	0	0
青	0	0	255
緑	0	255	0
シアン	0	255	255
赤	255	0	0
マゼンダ	255	0	255
黄色	255	255	0
白	255	255	255



RGB とは、色の表現法の一種で、赤（Red）、緑（Green）、青（Blue）の三つの原色を混ぜて幅広い色を再現する加法混合の一種である。 RGB は三原色の頭文字である。 ブラウン管（CRT）や液晶ディスプレイ（LCD）、デジタルカメラなどで画像再現に使われている。

自分の好みの色を設定する方法

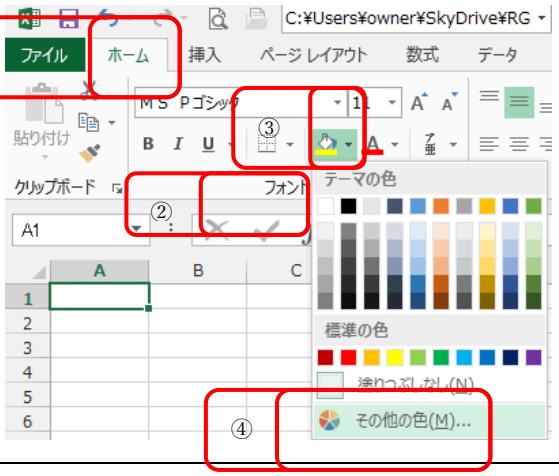
手順１

①［ホーム］タブ

②［フォント］グループ

③塗りつぶしの色

④その他の色



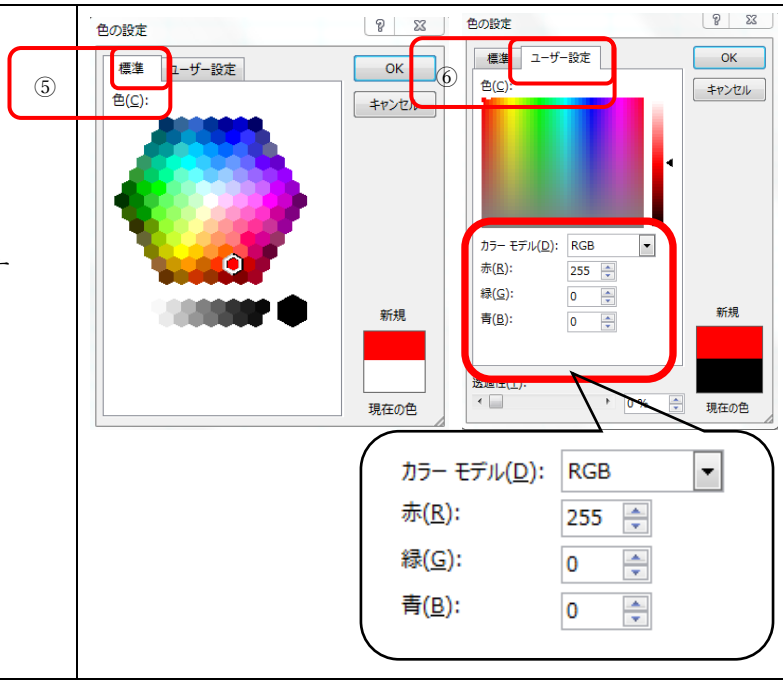
手順２

⑤［標準］タブ

ここで任意の色を選ぶと

⑥［ユーザー設定］タブ

では、番号が表示されます

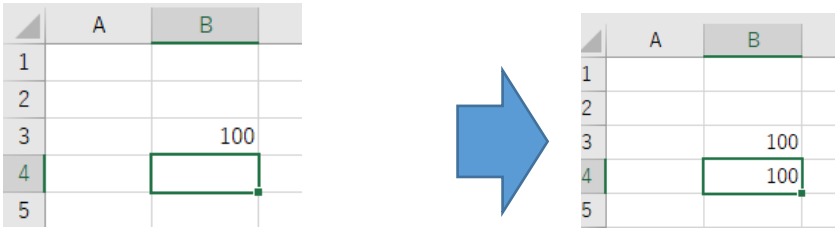


（拡大図）

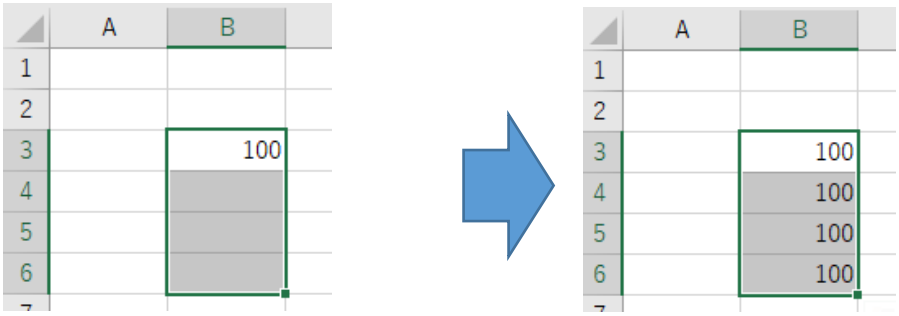
２．ショートカットキー

Ctrl　を押しながら、 D	上のデータ（値・数式）をコピー

（セル１つを選択）



（複数セルを選択）



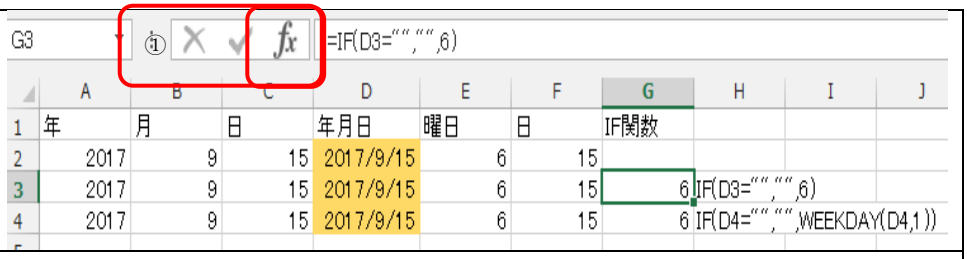
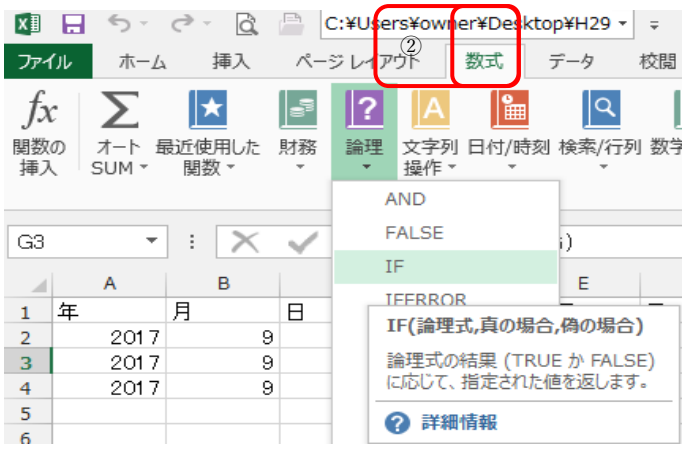
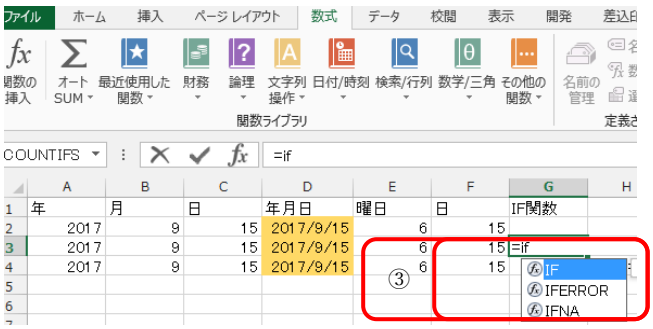
一番上を選択範囲にコピーしてくれます。（とても便利です。）

D が、Down（下へ）だとすると・・・

Ctrl　を押し ながら、 ↓ キー	データの中で、一番下へ移動 （データが何もないと一番下へ） ↑ キー、 → キー、 ← キー　も同様にその方向へ

（６）　　関数編

１．関数の入力　（３種類紹介します。）

①関数の挿入を使う方法	
②[数式]タブから選ぶ方法	
③手入力する方法	 ※作業効率のアップ キーボードに手を置いている場合は、マウス持ち替えるより、少ない手順で入力できます。

（７）　　参考サイト

１．今回の講座で、参考にさせていただいたサイトです

実践ワークシート協会	http://www.pwa.or.jp/
Office　TANAKA	http://officetanaka.net/
VBA エキスパート公式サイト	http://vbae.odyssey-com.co.jp/index.html