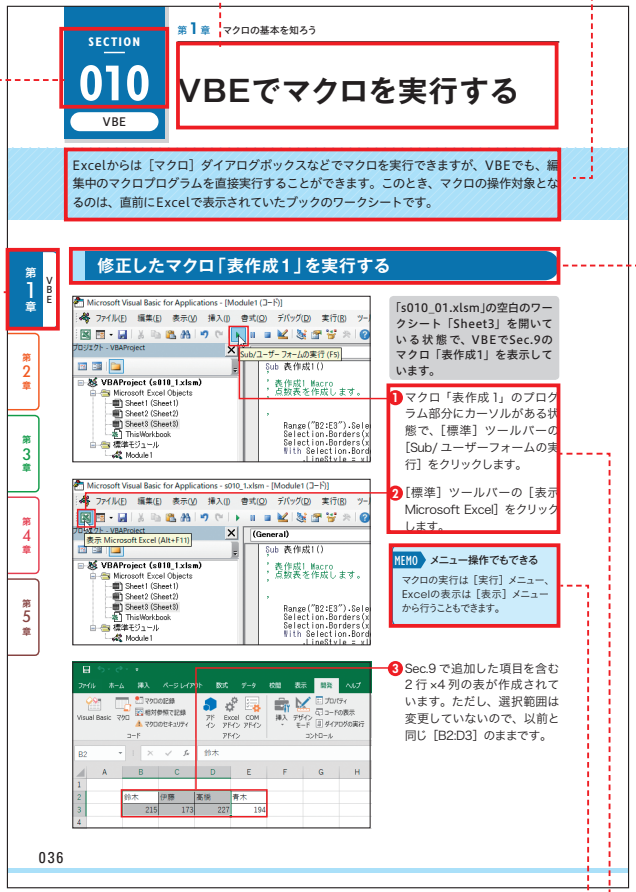
本書の使い方

セクションごとに機能を順番に解説しています。

セクション名は具体的な作業を示しています。

セクションの解説内容のまとめを表しています。

操作内容の見出しです。



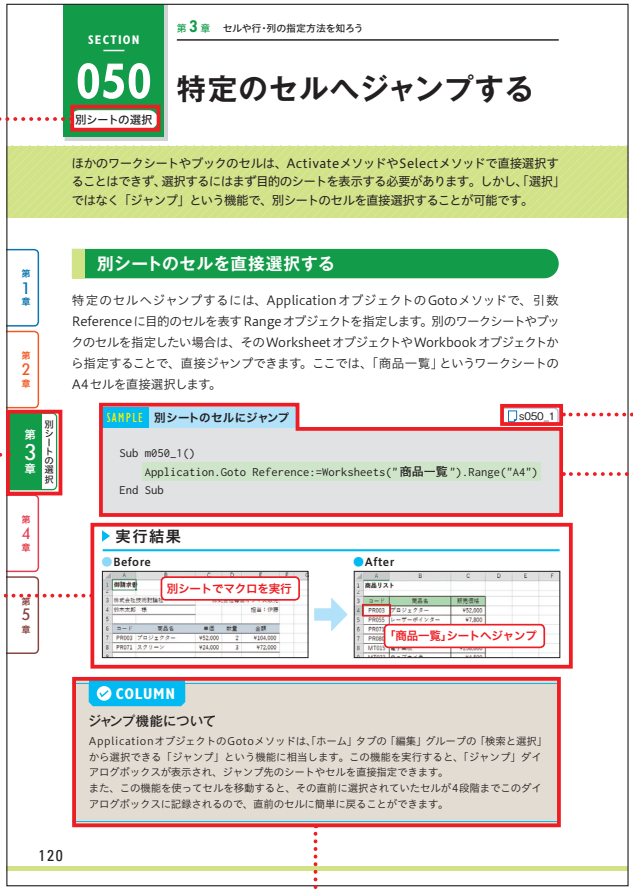
章が探しやすいようにセクションの分類を表示しています。

読者が抱く小さな疑問を予測して解説しています。

番号付きの記述で操作の順番が一目瞭然です。

探しやすいようにセクションの分類を表示しています。

サンプルファイル名を表示しています（P.4参照）。



プログラムの結果を実行前と実行後の画面で解説しています。

重要な補足説明を解説しています。

サンプルプログラムのコードを表示しています。

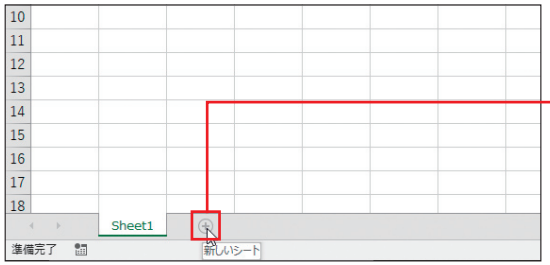
SECTION 003

マクロを実行する

記録マクロ

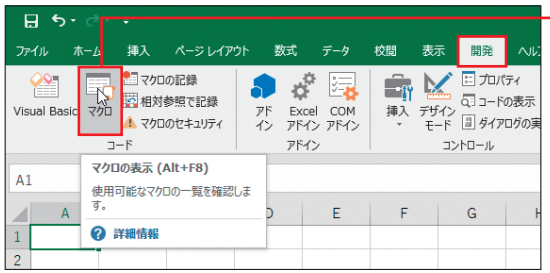
ここでは、記録機能で作成したマクロを実行する手順を解説します。ただし、マクロを記録したワークシートでマクロを実行しても、前と同じ表が重ねて作成されるだけで、見た目は何も変わりません。そこで、新規シートを作成してから、表作成マクロを実行します。

マクロを記録する

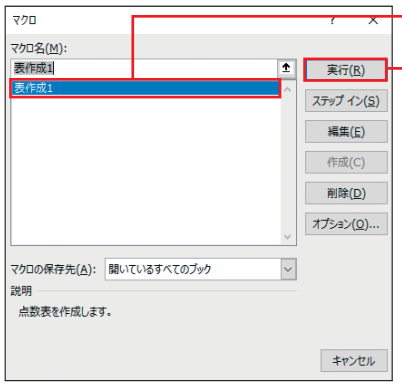


新しいシートを追加してからマクロを実行します。

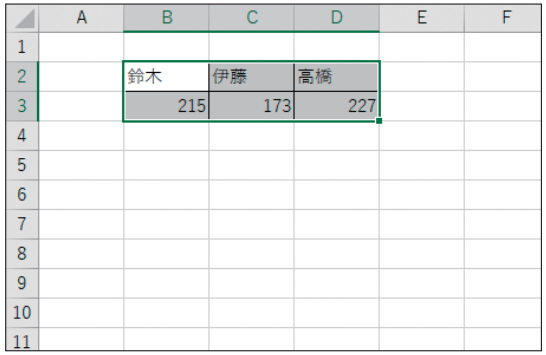
1. シート見出しの右側にある［新しいシート］ボタンをクリックします。
2. 新しいワークシートが作成されます。



3. ［開発］タブの［コード］グループの［マクロ］をクリックします。
4. ［マクロ］ダイアログボックスが表示されます。



5. ［マクロ名］欄で「表作成 1」を選びます。
6. ［実行］をクリックすると、このダイアログボックスが閉じ、マクロが実行されます。



	A	B	C	D	E	F
1						
2		鈴木	伊藤	高橋		
3		215	173	227		
4						
5						
6						
7						
8						
9						
10						
11						

7. 記録時と同様に［B2:D3］のセル範囲が選択され、格子の罫線が設定され、さらにその各セルにデータが入力されます。

MEMO 記録時と同じセル範囲

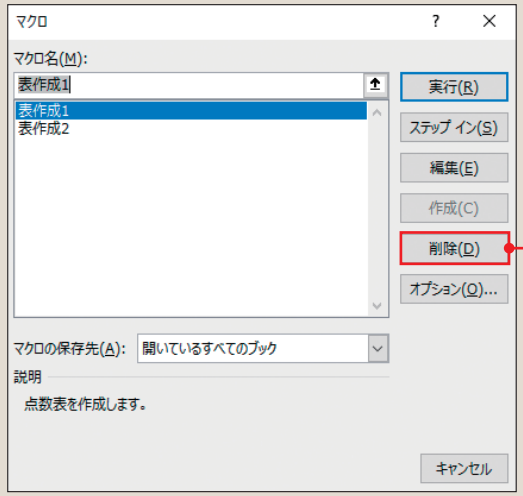
前節で作成したマクロは、アクティブセルの位置などに関係なく、常に［B2:D3］のセル範囲に表を作成します。

COLUMN

マクロを操作する

［マクロ］ダイアログボックスでは、作成したマクロを実行するほかに、削除や編集といった操作も行えます。不要なマクロを削除したい場合は、［マクロ名］ボックスで対象のマクロを選択し、［削除］をクリックします。ただし、ここですべてのマクロを削除しても、マクロの痕跡（標準モジュール）はブックに残っています。［名前を付けて保存］でファイルの種類を「Excelブック」にして保存し直せば、こうしたマクロの痕跡まで完全に削除できます。

一方、［編集］をクリックすると、選択したマクロに対応するVBAのプログラムが表示されます（P.32参照）。［ステップイン］は、マクロプログラムを1行単位で実行し、その動作を確認するために使います。また、［オプション］では、マクロのショートカットキーの設定（P.267参照）や説明を変更することができます。

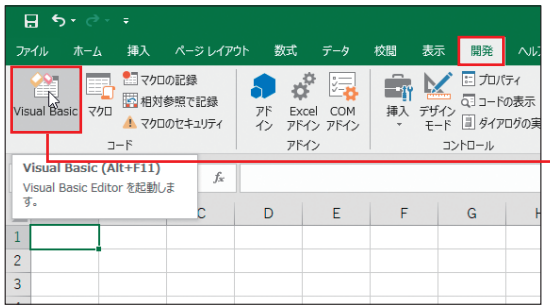


SECTION 009 VBE

マクロを修正する

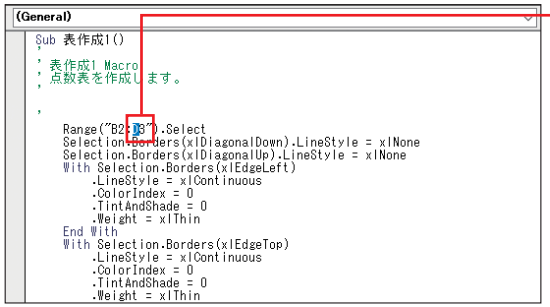
新しいマクロプログラムを1から作成するよりも、記録機能を使って作成したプログラムを修正したほうが早い場合もあります。ここではマクロ「表作成1」を、2行×4列のセル範囲の表を作成するプログラムに修正してみましょう。

マクロ「表作成1」の内容を修正する



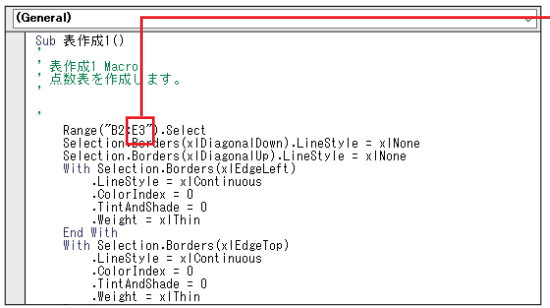
Excelで「s009_1.xlsm」を開いている状態でVBEを開き、このブックに含まれているマクロ「表作成1」を修正します。

❶［開発］タブの［コード］グループの［Visual Basic］をクリックします。



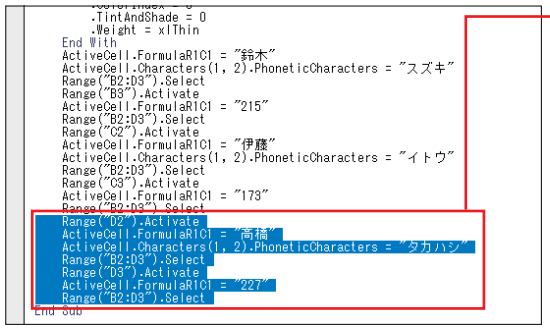
```
(General)
Sub 表作成1()
'
' 表作成1 Macro
' 点数表を作成します。
'
'
    Range("B2:D3").Select
    Selection.Borders(xlDiagonalDown).LineStyle = xlNone
    Selection.Borders(xlDiagonalUp).LineStyle = xlNone
    With Selection.Borders(xlEdgeLeft)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
    End With
    With Selection.Borders(xlEdgeTop)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
```

❷8行目の「Range("B2:D3")」の「D」の部分をドラッグして選択します。



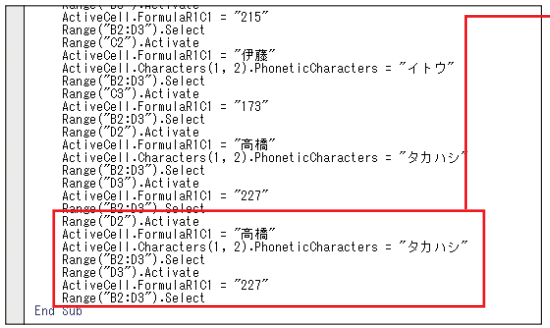
```
(General)
Sub 表作成1()
'
' 表作成1 Macro
' 点数表を作成します。
'
'
    Range("B2:E3").Select
    Selection.Borders(xlDiagonalDown).LineStyle = xlNone
    Selection.Borders(xlDiagonalUp).LineStyle = xlNone
    With Selection.Borders(xlEdgeLeft)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
    End With
    With Selection.Borders(xlEdgeTop)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
```

❸「E」と入力して選択範囲の文字「D」と置き換えます。



```
        .TintAndShade = 0
        .Weight = xlThin
    End With
    ActiveCell.FormulaR1C1 = "鈴木"
    ActiveCell.Characters(1, 2).PhoneticCharacters = "スズキ"
    Range("B2:D3").Select
    Range("B3").Activate
    ActiveCell.FormulaR1C1 = "215"
    Range("B2:D3").Select
    Range("C2").Activate
    ActiveCell.FormulaR1C1 = "伊藤"
    ActiveCell.Characters(1, 2).PhoneticCharacters = "イトウ"
    Range("B2:D3").Select
    Range("C3").Activate
    ActiveCell.FormulaR1C1 = "173"
    Range("B2:D3").Select
    Range("D2").Activate
    ActiveCell.FormulaR1C1 = "高橋"
    ActiveCell.Characters(1, 2).PhoneticCharacters = "タカハシ"
    Range("B2:D3").Select
    Range("D3").Activate
    ActiveCell.FormulaR1C1 = "227"
    Range("B2:D3").Select
End Sub
```

❹コードウィンドウを下のほうへスクロールしていき、プログラムの60～66行をドラッグして選択します。

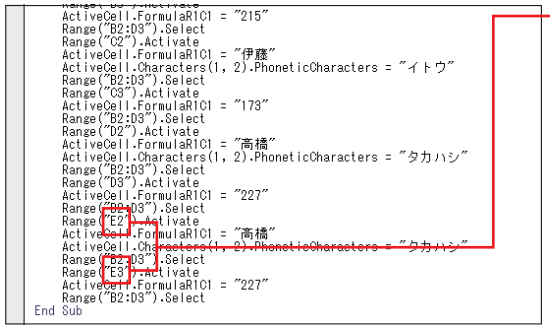


```
    ActiveCell.FormulaR1C1 = "215"
    Range("B2:D3").Select
    Range("C2").Activate
    ActiveCell.FormulaR1C1 = "伊藤"
    ActiveCell.Characters(1, 2).PhoneticCharacters = "イトウ"
    Range("B2:D3").Select
    Range("C3").Activate
    ActiveCell.FormulaR1C1 = "173"
    Range("B2:D3").Select
    Range("D2").Activate
    ActiveCell.FormulaR1C1 = "高橋"
    ActiveCell.Characters(1, 2).PhoneticCharacters = "タカハシ"
    Range("B2:D3").Select
    Range("D3").Activate
    ActiveCell.FormulaR1C1 = "227"
    Range("B2:D3").Select
    Range("D2").Activate
    ActiveCell.FormulaR1C1 = "高橋"
    ActiveCell.Characters(1, 2).PhoneticCharacters = "タカハシ"
    Range("B2:D3").Select
    Range("D3").Activate
    ActiveCell.FormulaR1C1 = "227"
    Range("B2:D3").Select
End Sub
```

❺選択した範囲をコピーして、67行目以下に貼り付けます。

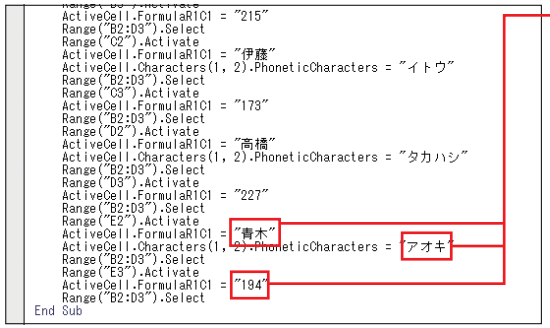
MEMO コピーと貼り付け

コピーと貼り付けの操作はWordなどと同様で、メニュー、ツールバー、ショートカットキー、ドラッグアンドドロップのいずれも使用できます。



```
    ActiveCell.FormulaR1C1 = "215"
    Range("B2:D3").Select
    Range("C2").Activate
    ActiveCell.FormulaR1C1 = "伊藤"
    ActiveCell.Characters(1, 2).PhoneticCharacters = "イトウ"
    Range("B2:D3").Select
    Range("C3").Activate
    ActiveCell.FormulaR1C1 = "173"
    Range("B2:D3").Select
    Range("D2").Activate
    ActiveCell.FormulaR1C1 = "高橋"
    ActiveCell.Characters(1, 2).PhoneticCharacters = "タカハシ"
    Range("B2:D3").Select
    Range("D3").Activate
    ActiveCell.FormulaR1C1 = "227"
    Range("B2:D3").Select
    Range("E2").Activate
    ActiveCell.FormulaR1C1 = "高橋"
    ActiveCell.Characters(1, 2).PhoneticCharacters = "タカハシ"
    Range("B2:D3").Select
    Range("E3").Activate
    ActiveCell.FormulaR1C1 = "227"
    Range("B2:D3").Select
End Sub
```

❻67行目の「D2」を「E2」に、71行目の「D3」を「E3」に修正します。



```
    ActiveCell.FormulaR1C1 = "215"
    Range("B2:D3").Select
    Range("C2").Activate
    ActiveCell.FormulaR1C1 = "伊藤"
    ActiveCell.Characters(1, 2).PhoneticCharacters = "イトウ"
    Range("B2:D3").Select
    Range("C3").Activate
    ActiveCell.FormulaR1C1 = "173"
    Range("B2:D3").Select
    Range("D2").Activate
    ActiveCell.FormulaR1C1 = "高橋"
    ActiveCell.Characters(1, 2).PhoneticCharacters = "タカハシ"
    Range("B2:D3").Select
    Range("D3").Activate
    ActiveCell.FormulaR1C1 = "227"
    Range("B2:D3").Select
    Range("E2").Activate
    ActiveCell.FormulaR1C1 = "青木"
    ActiveCell.Characters(1, 2).PhoneticCharacters = "アオキ"
    Range("B2:D3").Select
    Range("E3").Activate
    ActiveCell.FormulaR1C1 = "194"
    Range("B2:D3").Select
End Sub
```

❼68行目の「高橋」を「青木」に、69行目の「タカハシ」を「アオキ」に、72行目の「227」を「194」に修正します。

MEMO 修正後のマクロ

修正したマクロの実行結果はSec.10で紹介します。

SECTION 012

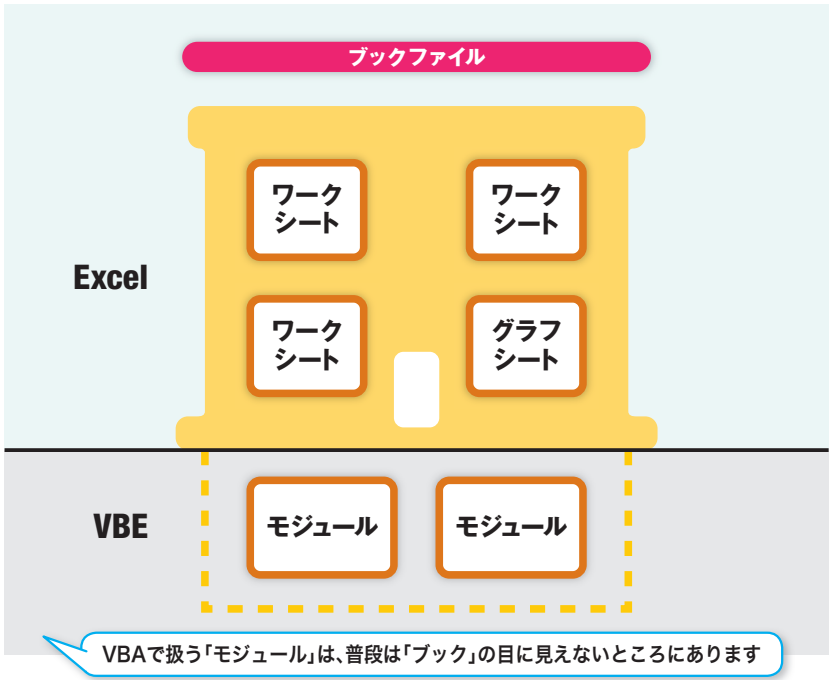
モジュールとは？

モジュール

VBAのプログラムは、ブックの中に用意した「モジュール」に保存します。モジュールにはいくつかの種類があり、保存するプログラムの種類や目的に応じて使い分けます。一般的な「マクロ」のプログラムは、「標準モジュール」に保存します。

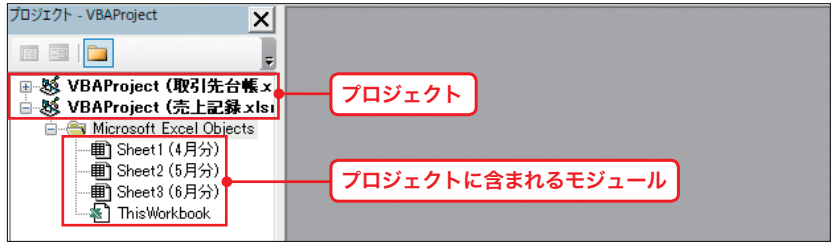
ブックとモジュール

Excelの文書ファイルである「ブック」を1つの大きな建物とすると、その中にワークシートやグラフシートといった“部屋”があり、それぞれにデータが収められています。VBAで作成したマクロなどのプログラムも、これらと同様にブックの中に保存されますが、プログラムを収めるために専用の“部屋”が用意されます。ワークシートなどは建物の地上階にある目に見える部屋ですが、プログラムを収めるための場所は目に見えない地下室のようなものです。この地下に当たるのがVBEであり、地下室に当たる場所を「モジュール」と呼びます。



プロジェクトとモジュール

VBEを表示すると、「プロジェクトエクスプローラー」の中に、Excelで現在開いている各ブックがそれぞれ「VBAProject」という名前の「プロジェクト」として表示されています。プロジェクトとは、いわばブックを、プログラムの保存場所として見たときの呼び方です。Excelで開いて作業している状態をブックの表の顔とするなら、VBEにおけるプロジェクトは、同じブックの裏の顔です。



Excelのブックに対応するプロジェクトの中には、「Sheet1」や「ThisWorkbook」などのアイコンが含まれています。これらのアイコンが「モジュール」と呼ばれます。VBAのプログラムは、それぞれの用途に応じたモジュールの中に保存します。

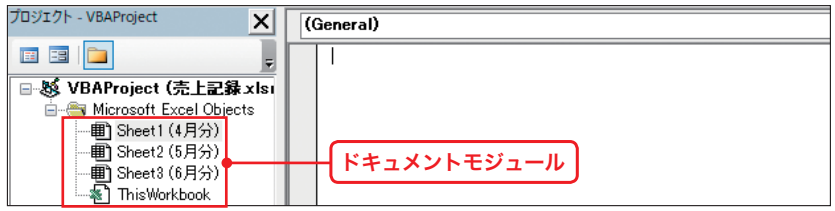
モジュールの種類

プロジェクトに含めることができるモジュールには、次のような種類があります。

1 ドキュメントモジュール

「Sheet1」や「ThisWorkbook」などのモジュールで、「Microsoft Excel Objects」というフォルダーに分類されています。「Sheet1」などはワークシート、「ThisWorkbook」はこのプロジェクトのブックそのものに対応します。これらのモジュールは、主に、そのオブジェクトのイベントマクロ（P.272参照）のプログラムを記述するのに使用します。

Excelブックのプロジェクトには、最初から1つのThisWorkbookモジュールと、ブック内のすべてのシートに対応するシートのモジュールが含まれており、VBEの操作でこれらを追加・削除することはできません。



SECTION 015

オブジェクト

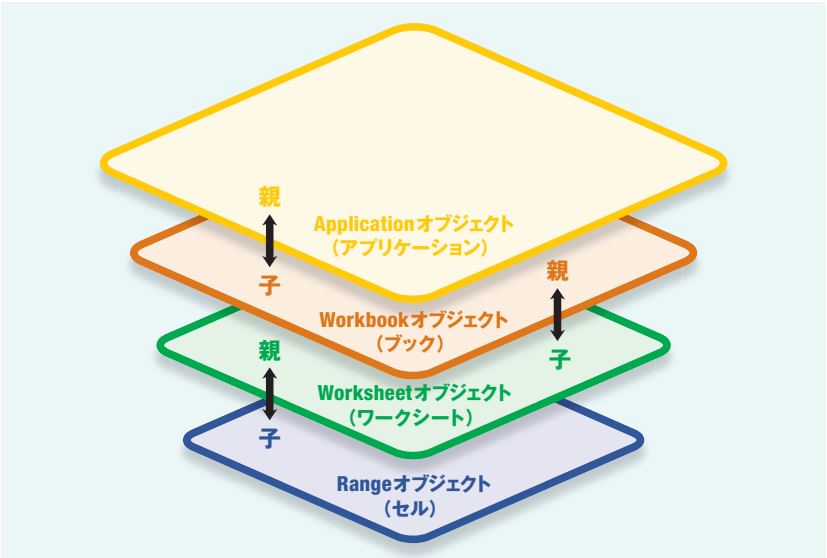
オブジェクトの階層とは？

Excelの「ブック」には、1つ以上の「ワークシート」が含まれています。さらに、「ワークシート」の中には数多くの「セル」が含まれています。このような階層的な構造は、そのままExcel VBAのオブジェクトにも反映されています。

親オブジェクトと子オブジェクト

Excel VBAで、特定のブックは「Workbookオブジェクト」として表されます。ブックに含まれる特定のワークシートは、そのWorkbookオブジェクトに属する「Worksheetオブジェクト」として表されます。また、ワークシートに含まれるセル（範囲）は、Worksheetオブジェクトに属する「Rangeオブジェクト」として表されます。さらに、Workbookオブジェクトは、Excelアプリケーションを表す「Applicationオブジェクト」に属しています。

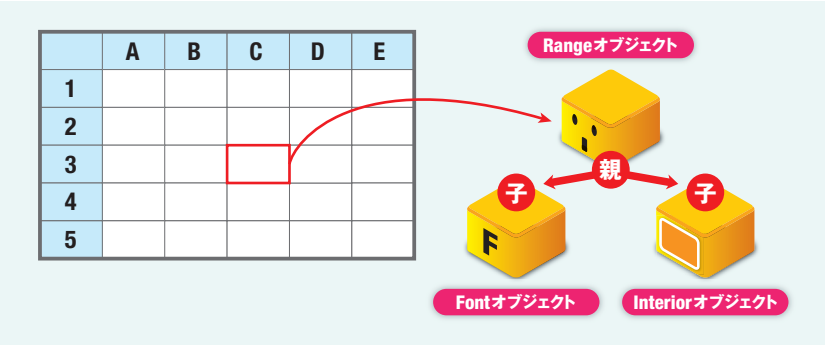
このように、Excelの操作対象を表す各オブジェクトは、その実体と同じように階層的な構造になっています。あるオブジェクトを基準にしたとき、その1つ上の階層のオブジェクトを「親オブジェクト」と呼び、その1つ下の階層のオブジェクトを「子オブジェクト」と呼びます。



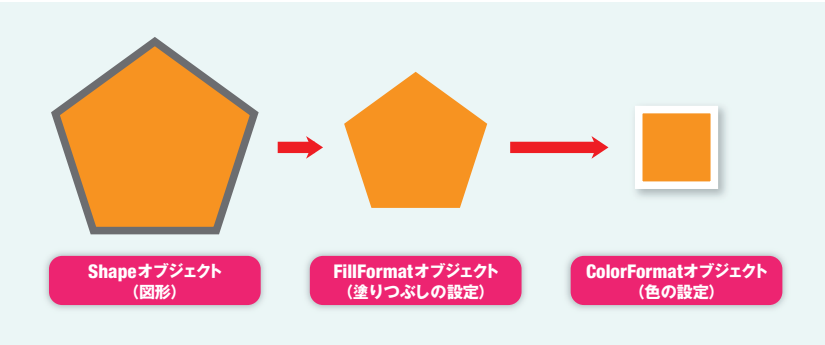
機能を表すオブジェクトの親子関係

機能や設定を表す“目に見えない”オブジェクトは、多くの場合、設定が反映される対象のオブジェクトの子オブジェクトです。

たとえば、セルには、フォントや塗りつぶしといったさまざまな書式を設定できます。フォントに関する設定を変更したい場合は「Fontオブジェクト」、塗りつぶしに関する設定を変更したい場合は「Interiorオブジェクト」に対して操作を行います。FontオブジェクトもInteriorオブジェクトも、対象のセル（範囲）を表すRangeオブジェクトの子オブジェクトになります。



また、ワークシート上に配置される図形は、Worksheetオブジェクトの子オブジェクトの「Shapeオブジェクト」として表されます。Shapeオブジェクトには、塗りつぶしの設定を表す「FillFormatオブジェクト」などの子オブジェクトがあり、FillFormatオブジェクトにはさらに色の設定を表す「ColorFormatオブジェクト」などの子オブジェクトがあります。このように、機能を表すオブジェクトも、1つの機能の全体を表す親オブジェクトと、その中に含まれる細かい設定を表す子オブジェクトといった階層構造になっている場合があります。



SECTION 029 配列

配列を利用する

前項では配列の概念について説明しましたが、ここでは実際にコードの中で配列を利用する方法について解説します。配列も最初に宣言して使用しますが、配列に収める要素の数を固定するか、可変にするかによって、宣言の仕方が変わってきます。

1次元配列を使用する

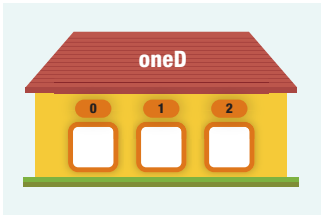
コードの中で配列を使用するには、普通の変数と同様に、事前にDimステートメントを使って宣言します。配列の名前は変数と同様に付けることができますが、名前の後に「()」を付けることで、配列変数であることを示します。データ型を指定すると、配列のすべての要素が同じデータ型になります。

配列には、大きく分けて「固定長配列」(静的配列)と「動的配列」の2種類があります。配列の要素の数が変動しない「固定長配列」を宣言するときは、変数名の後の「()」の中にそのインデックスの最大値を指定します。通常、配列のインデックスは0から始まるため、実際に指定するのは、設定したい要素の順番よりも1少ない値です。

次の例は、整数型で要素数が3つの1次元配列「oneD」を宣言するコードです。

SAMPLE 1次元配列の宣言

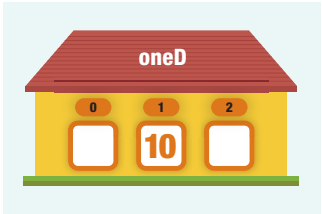
```
Dim oneD(2) As Integer
```



宣言した配列は、「()」にインデックスを指定して使います。3要素の配列の2番目の要素に10という数値を代入する操作は、次の例のようになります。配列からデータを取り出す場合も、同様にインデックスで位置を指定します。

SAMPLE 1次元配列への値の代入

```
oneD(1) = 10
```

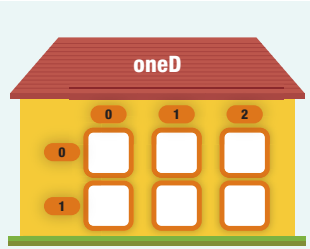


2次元配列を使用する

コードの中で2次元配列を使用する場合は、2つのインデックスの最大値を「,」(半角カンマ)で区切って指定します。最初のインデックスが行、2番目のインデックスが列の指定になります。先に「2階建て以上」と書きましたが、1行だけの2次元配列もあります。次の例は、文字列型で要素数が2行×3列の2次元配列「twoD」を宣言するコードです。

SAMPLE 2次元配列の宣言

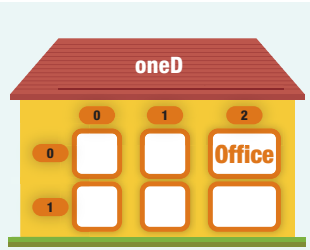
```
Dim twoD(1, 2) As String
```



以後のコードでこの配列を使用する場合も、2つのインデックスで要素の位置を指定し、代入や値の取り出しを行います。次の例は、この配列の1行目で3列目の位置に「Office」という文字列を代入するコードです。

SAMPLE 2次元配列への値の代入

```
twoD(0, 2) = "Office"
```



COLUMN

インデックスの最小値と最大値

通常、配列のインデックスの最小値は0ですが、これをユーザーが設定することも可能です。モジュールの宣言セクション(すべてのプロシージャよりも上の行)に「Option Base 1」と入力しておくと、そのモジュール内の配列のインデックスの最小値が1になります。

また、配列を宣言する際に、インデックスの最小値と最大値を「To」でつないで指定することもできます。たとえば、次のように宣言すると、インデックスが2から5までの4要素の配列になります。

SAMPLE インデックスを指定した配列の宣言

```
Dim hairetsu(2 To 5)
```

SECTION 037

セル範囲の選択

始点と終点を指定してセル範囲を選択する

指定した2つのセルを左上端および右下端とする長方形のセル範囲を、Rangeオブジェクトとして取得できます。また、指定した2つのセル範囲を両方含む長方形のセル範囲を取得することも可能です。

範囲の始点と終点を指定して選択する

Rangeプロパティには、2つの引数を指定する使い方もあります。セル番地を表す2つの文字列を指定すると、それらを左上端および右下端とする長方形のセル範囲を、Rangeオブジェクトとして取得できます。ここでは、B4～E6のセル範囲を選択します。

SAMPLE 2つの番地で範囲を指定　s037_1

```
Sub m037_1()
    Range("B4", "E6").Select
End Sub
```

実行結果

●Before

	店舗別販売数記録				
	店名	4月	5月	6月	合計
	中野店	532	567	496	1595
	高円寺店	608	575	613	1796
	阿佐ヶ谷店	501	468	523	1492
	合計	1641	1610	1632	4883

●After

	店舗別販売数記録				
	店名	4月	5月	6月	合計
	中野店	532	567	496	1595
	高円寺店	608	575	613	1796
	阿佐ヶ谷店	501	468	523	1492
	合計	1641	1610	1632	4883

B4～E6の範囲を選択

COLUMN

引数にRangeオブジェクトを指定する

Rangeプロパティの引数には、セル番地を表す文字列のほかに、Rangeオブジェクトを表すプロパティを指定することもできます。次の例は、B4セルとアクティブセルを左上端および右下端とするセル範囲を選択するコードです。

SAMPLE アクティブセルまでの範囲を選択　s037_2

```
Sub m037_2()
    Range("B4", ActiveCell).Select
End Sub
```

2つの範囲を含む長方形の範囲を選択する

Rangeプロパティの2つの引数に、それぞれ単独のセルではなくセル範囲を指定した場合は、その2つのセル範囲のうち、上下左右のそれぞれ最も端にあるセルを含む長方形の範囲が取得されます。ここでは、2つの引数にB5～D6とC3～E8の各セル範囲を指定して、それらの範囲を含む長方形のセル範囲を選択します。

SAMPLE 2つの範囲を含む範囲を選択　s037_3

```
Sub m037_3()
    Range("B5:D6", "C3:E8").Select
End Sub
```

実行結果

●Before

	店舗別販売数記録				
	店名	4月	5月	6月	合計
	中野店	532	567	496	1595
	高円寺店	608	575	613	1796
	阿佐ヶ谷店	501	468	523	1492
	合計	1641	1610	1632	4883

●After

	店舗別販売数記録				
	店名	4月	5月	6月	合計
	中野店	532	567	496	1595
	高円寺店	608	575	613	1796
	阿佐ヶ谷店	501	468	523	1492
	合計	1641	1610	1632	4883

2つの範囲を含む範囲を選択

セル範囲と選択範囲を含む範囲を選択する

Rangeプロパティの引数に、選択範囲を表すSelectionプロパティを指定することも可能です。ここでは、2つの引数にB2～D4のセル範囲と選択範囲を指定して、それらの範囲を含む長方形のセル範囲を選択します。

SAMPLE 指定範囲と選択範囲を含む範囲を選択　s037_4

```
Sub m037_4()
    Range("B2:D4", Selection).Select
End Sub
```

実行結果

●Before

	店舗別販売数記録				
	店名	4月	5月	6月	合計
	中野店	532	567	496	1595
	高円寺店	608	575	613	1796
	阿佐ヶ谷店	501	468	523	1492
	合計	1641	1610	1632	4883

●After

	店舗別販売数記録				
	店名	4月	5月	6月	合計
	中野店	532	567	496	1595
	高円寺店	608	575	613	1796
	阿佐ヶ谷店	501	468	523	1492
	合計	1641	1610	1632	4883

選択範囲を含む範囲を選択

SECTION 061

連続データの入力

セル範囲に規則的なデータを入力する

セル範囲を指定して、その先頭のセルのデータを範囲全体にコピーしたり、連続的なデータを入力したりする方法を紹介します。また、先頭の2つのセルを基準として、その差の分だけ自動的に増減する連続データを入力することも可能です。

基準のセルを範囲全体にコピーする

セル範囲の上端のセルの値と書式を、同じ列の下にあるすべてのセルにコピー(フィル)します。その範囲を表すRangeオブジェクトのFillDownメソッドを使用するのが簡単です。ここでは、C3セルの内容をC4:C6のセル範囲にフィルします。

SAMPLE C3セルの内容を下のセル範囲にフィル　s061_1

```
Sub m061_1()
    Range("C3:C6").FillDown
End Sub
```

実行結果

●Before

	A	B	C	D	E	F
1						
2		科目	担当			
3		国語	伊藤武史			
4		数学				
5		英語				
6		物理				
7						

●After

	A	B	C
1			
2		科目	担当
3		国語	伊藤武史
4		数学	伊藤武史
5		英語	伊藤武史
6		物理	伊藤武史
7			

下のセル範囲にフィル

COLUMN

その他の方向にフィルする

ここでは1列のセル範囲の上端セルを下方向にフィルしましたが、下端セルを上方向にフィルするには、FillUpメソッドを使います。また、セル範囲の左端セルを同じ行の右方向にフィルするにはFillRightメソッドを、右端セルを左方向にフィルするにはFillLeftメソッドを使います。右のコードはFillRightメソッドの使用例です。

SAMPLE 右のセル範囲にフィル　s061_2

```
Sub m061_2()
    Range("C5:F7").FillRight
End Sub
```

連続データを入力する

先頭セルの数値が1ずつ増加していく連続データをフィルするには、先頭セルを表すRangeオブジェクトのAutoFillメソッドで、引数Destinationに入力対象のセル範囲を表すRangeオブジェクトを、引数Typeに定数xlFillSeriesを指定します。

SAMPLE 1ずつ増加する連続データを入力　s061_3

```
Sub m061_3()
    Range("C2").AutoFill Destination:=Range("C2:F2"), _
        Type:=xlFillSeries
End Sub
```

実行結果

●Before

	A	B	C	D	E	F	G	H
1								
2			月	4				
3			販売数	156	182	193	175	
4								

●After

	A	B	C	D	E	F
1						
2			月	4	5	6	7
3			販売数	156	182	193	175

連続データを入力

2つのセルを基準に連続データを入力する

先頭の2つのセルに数値を入力し、これらのセルを基準としてオートフィルの操作を行うと、以下、その2つの数値の差の分だけ増減していく連続データを入力できます。ここでは、最初の2つのセルに「2」と「5」と入力し、以下3ずつ増えていく連続データを入力します。AutoFillメソッドの引数Typeは定数xlFillDefaultを指定するか、省略します。

SAMPLE 3ずつ増加する連続データを入力　s061_4

```
Sub m061_4()
    Range("B3").Value = 2
    Range("B4").Value = 5
    Range("B3:B4").AutoFill Destination:=Range("B3:B8")
End Sub
```

実行結果

●Before

	A	B	C	D	E	F	G
1							
2		日	特売商品				
3			コロッケ				
4			ロースカツ				
5			メンチカツ				
6			チキンカツ				
7			一口カツ				
8			唐揚げ				

●After

	A	B	C	D	E	F	G
1							
2		日	特売商品				
3		2	コロッケ				
4		5	ロースカツ				
5		8	メンチカツ				
6		11	チキンカツ				
7		14	一口カツ				
8		17	唐揚げ				

連続データを入力

SECTION 109

ワークシートを検索する

検索

ワークシートのセルに入力されている特定のデータを検索するコードを紹介します。Excelの通常の検索機能では見つかったセルを選択しますが、VBAの場合は単にRangeオブジェクトとして取得できるので、選択せずにそのまま別の操作を行うことが可能です。

特定の文字列を検索する

「本店」という文字列を含むセルを検索し、その4つ右にあるセルの値を表示します。

セルの検索は、対象のセル範囲を表すRangeオブジェクトのFindメソッドで実行します。ここでは、Cellsプロパティで作業中のワークシートのすべてのセルを表すRangeオブジェクトを取得し、検索対象としています。

検索のオプションは、すべてFindメソッドの引数として指定します。検索文字列を指定するWhatは必須ですが、それ以外の引数はすべて省略可能です。引数を省略した場合は、既定の設定またはExcelを起動してから実行した検索設定が適用されます。次の例では、引数はWhatとLookAtだけを指定しています。

検索を実行し、該当するセルが見つかった場合は、戻り値としてそのセルを表すRangeオブジェクトを取得できます。しかし、発見できなかった場合はオブジェクトがないことを意味する「Nothing」というキーワードの状態になり、この戻り値をそのまま何らかのセル操作に使用するとエラーになります。そこで、戻り値を一度オブジェクト変数fRngにセットし、Ifステートメントで「Is」演算子を使用してNothingでないことを確認してから、その4つ右側のセルの値をメッセージ画面に表示させています。

SAMPLE 「本店」の行の販売数合計を表示　s109_1

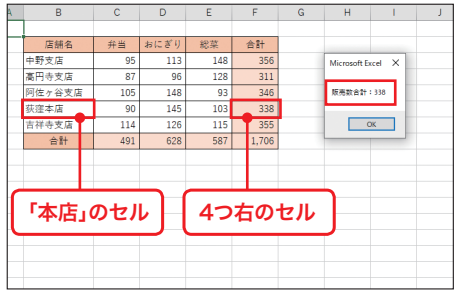
```
Sub m109_1()
    Dim fRng As Range
    Set fRng = Cells.Find(What:="本店", LookAt:=xlPart)
    If Not fRng Is Nothing Then
        MsgBox "販売数合計:" & fRng.Offset(ColumnOffset:=4).Value
    End If
End Sub
```

実行結果

Before

店舗名	弁当	おにぎり	総菜	合計
中野支店	95	113	148	356
高円寺支店	87	96	128	311
阿佐ヶ谷支店	105	148	93	346
荻窪本店	90	145	103	338
吉祥寺支店	114	126	115	355
合計	491	628	587	1,706

After



COLUMN

Findメソッドの引数

Findメソッドに指定できるWhat以外の引数には、次のような種類があります。

引数	指定内容	指定値
After	検索の起点となるセル	Rangeオブジェクト
LookIn	検索対象を表す定数	数式：xlFormulas、値：xlValues、コメント：xlComments
LookAt	セルと完全に一致するものだけを検索するかどうかを表す定数	完全一致：xlWhole、一部一致：xlPart
SearchOrder	検索方向を表す定数	行方向：xlByRows、列方向：xlByColumns
SearchDirection	起点のセルからどちらに向けて検索を行うかを表す定数	後方：xlNext、前方：xlPrevious
MatchCase	大文字と小文字を区別するかどうかを表す論理値	しない：False、する：True
MatchByte	全角と半角を区別するかどうかを表す論理値	しない：False、する：True
SearchFormat	書式を検索するかどうかを表す論理値	しない：False、する：True

COLUMN

ワイルドカードを使用した検索

Excelの検索機能では、任意の文字列を表す「*」と、任意の1文字を表す「?」という2種類のワイルドカードが使用できます。VBAで検索を行う場合も、これらのワイルドカードは使用可能です。たとえば、「営業第一部」「営業第二部」「営業本部」をすべて検索したい場合は、引数Whatに「営業*部」のように指定します。

SECTION 137 コントロール

チェックボックスでオン／オフを設定する

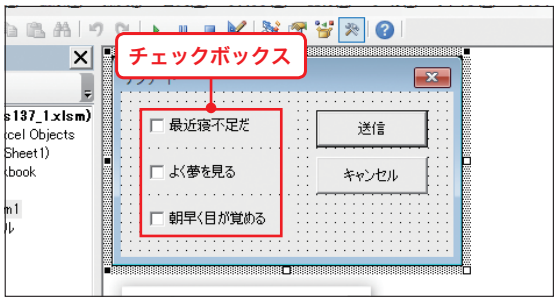
フォーム上にチェックボックスを配置し、そのオン／オフの状態に応じて異なる処理を実行する例を紹介します。このサンプルでは、チェックボックスがクリックされた時点では処理は行わず、[OK] ボタンがクリックされたときにその値だけを参照します。

アンケート回答用のユーザーフォームを設計する

3つの質問に答えるアンケート回答用のフォームを作成します。質問はいずれもチェックボックスで作成し、Yes／Noをそのオン／オフで表します。

3つのチェックボックスのほかには、回答を集計シートに反映させるための[送信] ボタンと、回答せずにフォームを閉じるための[キャンセル] ボタンを用意します。また、ユーザーフォームのCaptionプロパティを「アンケート」とします。

ユーザーフォームの設計例



各ボタンのイベントマクロ

「送信」と表示したCommandButton1のイベントマクロでは、3つのチェックボックスの値を1つずつ調べていきます。チェックボックスのValueプロパティの値は、チェックの付いた状態(オン) をTrue、付いていない状態(オフ) をFalseとする論理値なので、そのままIfステートメントの条件式に指定できます。この値がTrueだった場合は、集計用シートでそれぞれの回答数を集計するセルの値を、現在の値よりも1増やします。3つのチェックボックスについてそれぞれ処理を行ったら、このフォームを閉じます。

一方、「キャンセル」と表示したコマンドボタンのイベントマクロには、このフォームを閉じるだけのコードを記述します。

SAMPLE 「送信」ボタンのイベントマクロ　s137_1

```
Private Sub CommandButton1_Click()
    If CheckBox1.Value Then Range("C4").Value = Range("C4").Value + 1
    If CheckBox2.Value Then Range("C5").Value = Range("C5").Value + 1
    If CheckBox3.Value Then Range("C6").Value = Range("C6").Value + 1
    Unload Me
End Sub
```

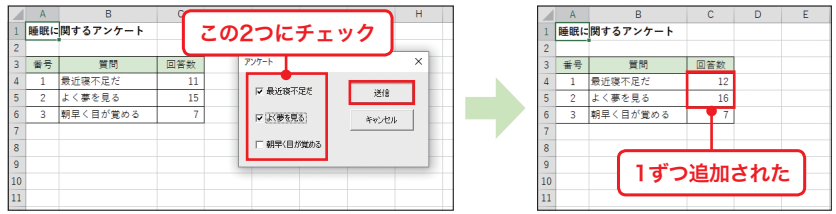
SAMPLE 「キャンセル」ボタンのイベントマクロ　s137_1

```
Private Sub CommandButton2_Click()
    Unload Me
End Sub
```

アンケート回答フォームの使用例

ユーザーフォームを表示させ、各チェックボックスの質問について、Yesならチェックを付け、Noならチェックを外します。3つの質問に答えて[送信] ボタンをクリックすると、Yesと答えた質問の集計セルに1が加算されます。

実行結果



COLUMN

チェックボックスの状態をコードで設定する

ここではチェックボックスはユーザーが操作し、そのオン／オフの状態をコードで判定しています。反対に、コードを使ってチェックボックスのオン／オフを切り替えたい場合は、そのValueプロパティにTrueまたはFalseを設定すればOKです。