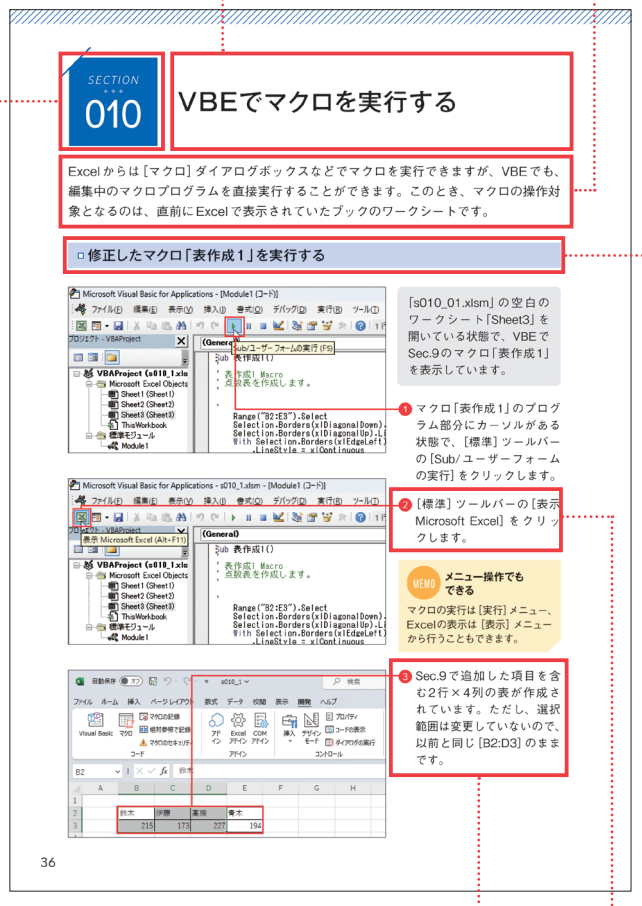
■本書の使い方

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。

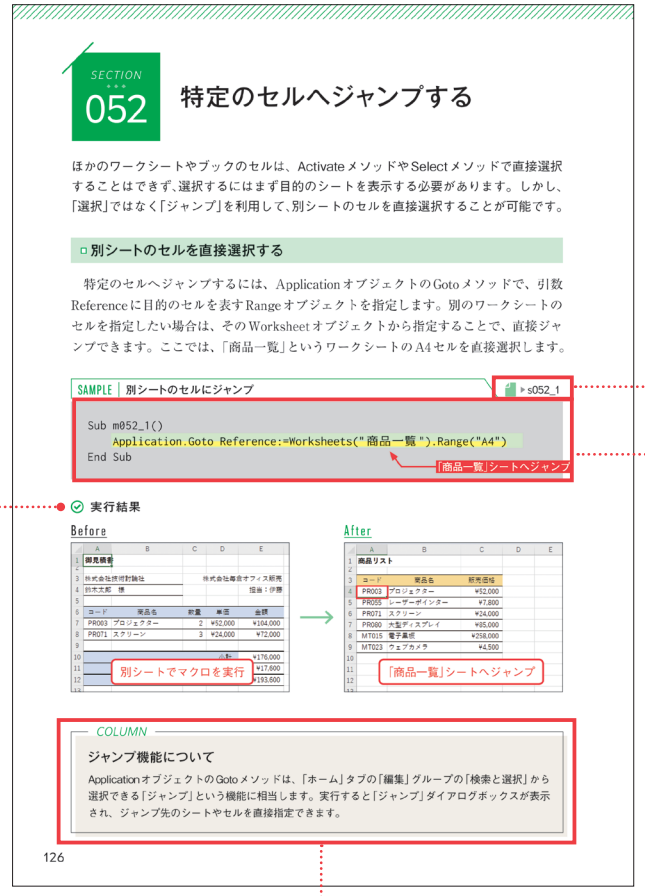
操作内容の見出しです。



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。

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。

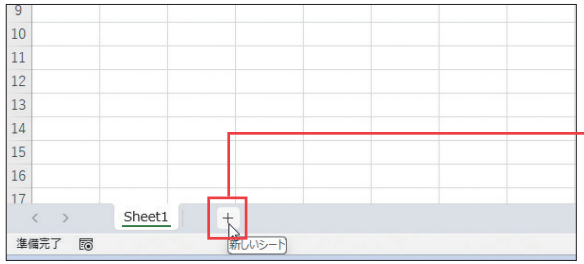
サンプルプログラムのコードを表示しています。

SECTION 003

マクロを実行する

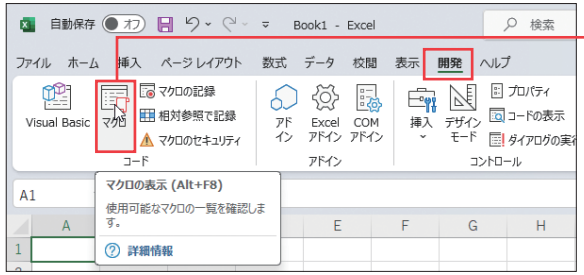
ここでは、記録機能で作成したマクロを実行する手順を解説します。ただし、マクロを記録したワークシートでマクロを実行しても、前と同じ表が重ねて作成されるだけで、見た目は何も変わりません。そこで、新規シートを作成してから、表作成マクロを実行します。

マクロを実行する

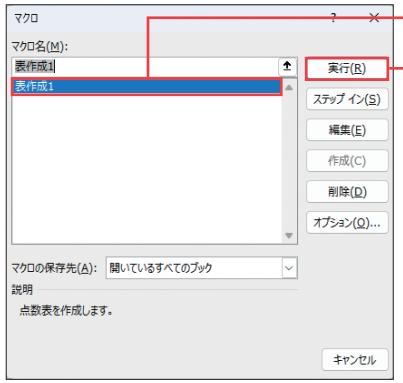


新しいシートを追加してからマクロを実行します。

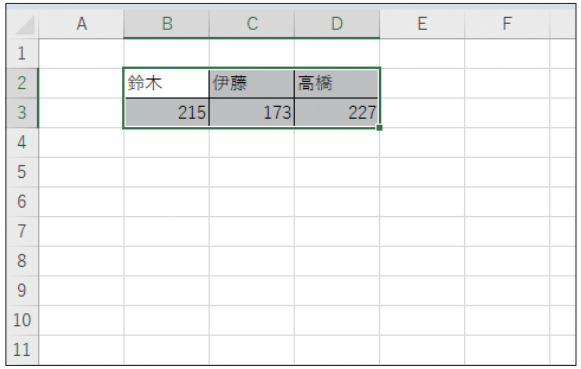
1. シート見出しの右側にある［新しいシート］ボタンをクリックします。
2. 新しいワークシートが作成されます。



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。



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

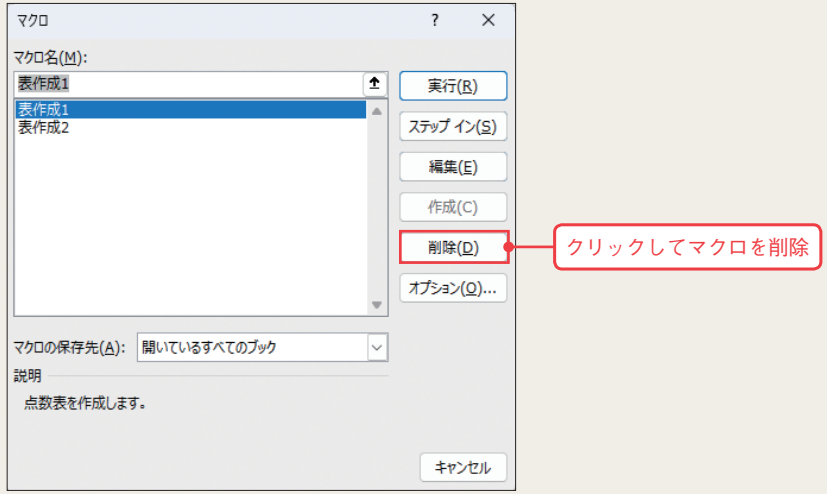
前節で作成したマクロは、アクティブセルの位置などに関係なく、常に［B2:D3］のセル範囲に表を作成します。

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.311参照）や説明を変更することができます。

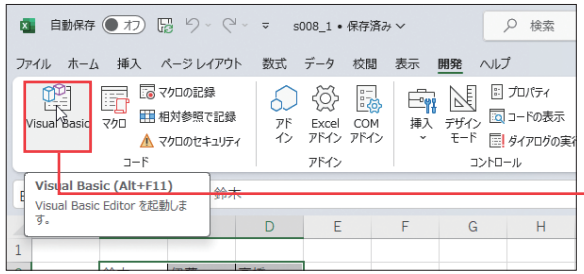


SECTION 008

マクロの中身を見る

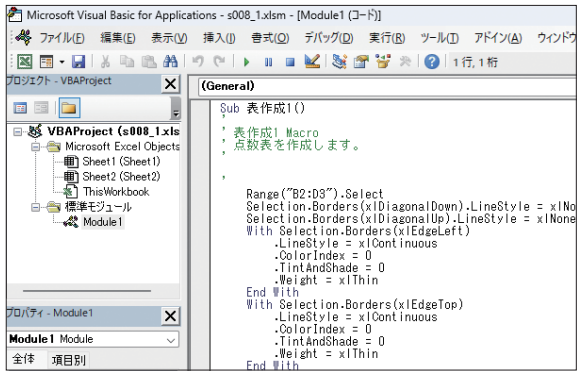
記録機能で作成した「マクロ」が実際にどのようなVBAのプログラムになっているかを確認してみましょう。ここでは、Sec.2で絶対参照で記録したマクロ「表作成1」を、VBEのコードウィンドウに表示します。

マクロ「表作成1」の内容を確認する



Excelで「s008_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を確認します。

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



2 VBEの画面が表示され、このブックに含まれているマクロのプログラムがコードウィンドウに表示されます。

MEMO 表示されないときは

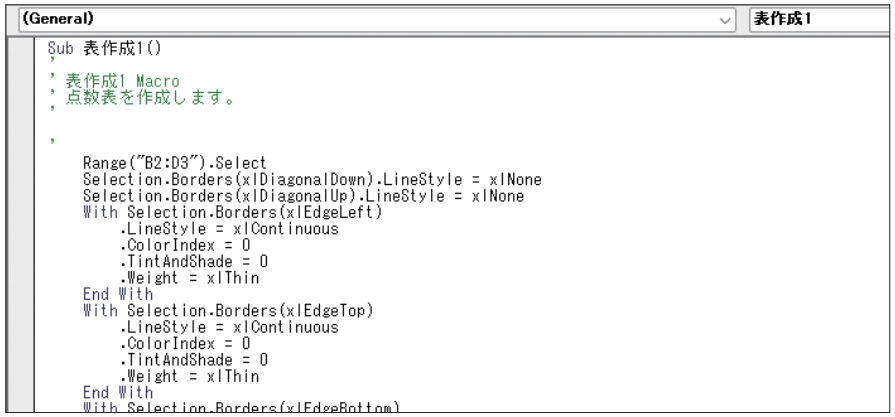
プログラムが表示されない場合は、プロジェクトエクスプローラーの「Module1」のアイコンをダブルクリックします（下のコラム参照）。

COLUMN プロジェクトエクスプローラーのアイコン

プロジェクトエクスプローラーには、さまざまな種類のアイコンが、フォルダーで分類されています。フォルダーの中の各アイコンは、マクロプログラムの“記述場所”である「モジュール」(P.40参照)を表しています。フォルダーのアイコンの左側が[+]の状態になっているときは、クリックすると中のモジュールが表示されます。「Module1」が見つからない場合は、「標準モジュール」フォルダーをクリックして中を確認してください。

作成されたマクロプログラムの中身

ここでは、Sec.2で記録機能を使って作成したマクロ「表作成1」が、具体的にどのようなプログラムになっているかを見てみましょう。



冒頭の「Sub」と「()」で挟まれた部分がマクロ名で、これがそのまま[マクロ]ダイアログボックスに表示されます。また、「'」で始まる緑字の行は、マクロの動作に直接関係しない「コメント」を表します(P.85参照)。

その下の「Range("B2:D3").Select」以下の行が、実際の作業を表すマクロプログラムです。

COLUMN プログラムは英語のように読もう

```
Range("B2:D3").Select
Selection.Borders(xlDiagonalDown).LineStyle = xlNone
Selection.Borders(xlDiagonalUp).LineStyle = xlNone
```

ピリオドで区切られた前半の「Range("B2:D3")」の部分は、操作の“目的語”に当たる「オブジェクト」(P.46参照)を表しています。ここでは、「B2:D3のセル範囲」を意味しています。ピリオドの後の「Select」は操作の“動詞”に当たる「メソッド」(P.58参照)です。これは文字通り、「選択しろ」という命令を表しています。

その次の行はやや複雑ですが、「Selection」は選択範囲、「Borders」は罫線を表します。「LineStyle」は線の種類を表す「プロパティ」(P.50参照)であり、「=」を使って設定値を指定しています。設定値には「None」が入っており、「なし」にすると読み取れます。

このように、VBAのプログラムは、英語の意味からの連想で、その操作の内容がわかることも少なくありません。マクロの記録機能で自動的にプログラムを生成し、その動作の意味を英語のように読み解いていく、というのも、VBAのプログラミングを学習するための有効な方法でしょう。

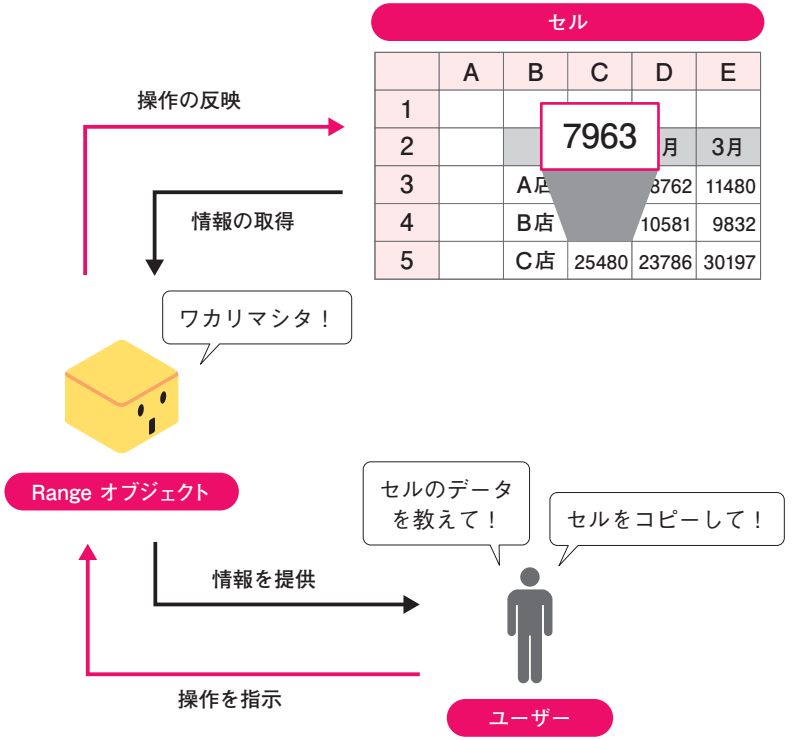
SECTION 014

オブジェクトとは？

ここでは、Excel VBAのプログラミングで重要な「オブジェクト」とは何かを理解しましょう。VBAのプログラムでは、まず「オブジェクト」を指定し、それに対して命令を出すことで、Excelの一連の作業を自動的に実行させます。

VBAの「オブジェクト」

「オブジェクト」とは、操作を行う対象の“モノ”を、プログラムで扱えるような形で指定したもののことです。たとえば、VBAのプログラムで特定のセルを操作したい場合は、まずそのセルを表す「Rangeオブジェクト」を指定します。



VBA の「オブジェクト」は、ユーザーと Excel との橋渡し役として働きます

オブジェクトとは「操作対象」のこと

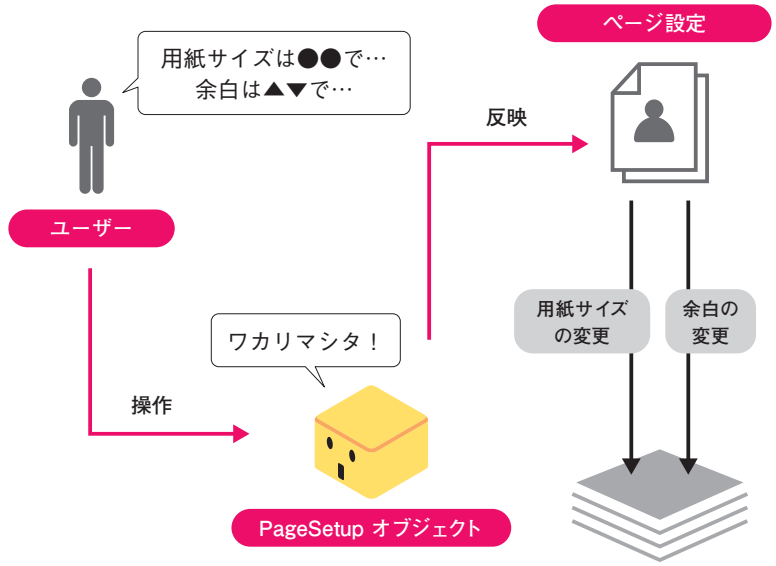
セルに入力されているデータや、セルの縦横のサイズといった情報は、セルを表すRangeオブジェクトから取り出すことができます。反対に、Rangeオブジェクトを通してそうした情報を変更することもでき、これによって“実体”であるセルの内容も変化します（左ページの図参照）。

また、オブジェクトに対する命令という形で、操作を実行することも可能です。たとえば、セルをコピーしたり、データを消去したりといった操作は、そのRangeオブジェクトに対する命令として実行できます。セル以外にも、ワークシートやブック、さらにExcelのアプリケーションなどに、それぞれ対応するオブジェクトが用意されています。

“目に見えない”オブジェクトとは？

セルやワークシートのようなオブジェクトはわかりやすいのですが、実は、こうした“目に見える”オブジェクトばかりがオブジェクトではありません。

「印刷のページ設定」や「フォントの設定」などのようなExcelの「機能」も、VBAでは1つのオブジェクトとして表されます。VBAでそうした機能を利用したいときは、そのオブジェクトに対する命令として実行するわけです。たとえば、用紙サイズの変更といったページ設定に関する操作なら、「PageSetupオブジェクト」を通して実行します。



「ページ設定」などの目に見えない機能にもオブジェクトが用意されています

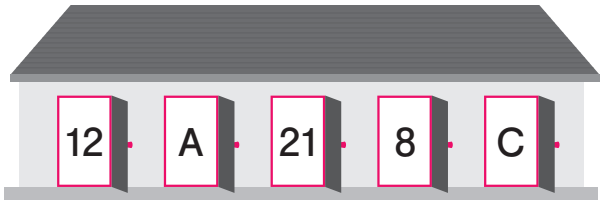
SECTION 028

配列とは？

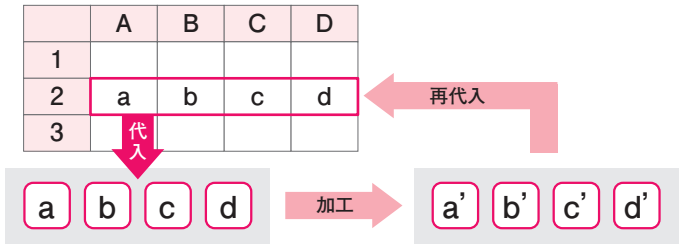
「配列」とは、複数のデータをまとめて扱えるようにしたデータ構造のことです。VBAで単に「配列」といった場合、配列データを収めることができる「配列変数」を意味します。ここではまず、VBAにおける「配列」の概念について説明しましょう。

変数と配列

変数はよく箱のイメージで説明されますが、通常の変数はそこに1つの値だけを代入することができます。これに対し、配列は複数の部屋を持つ建物のイメージです。1つの建物の内部が複数の部屋に仕切られていて、その1つ1つに値を代入することができます。この部屋のことを「要素」といいます。

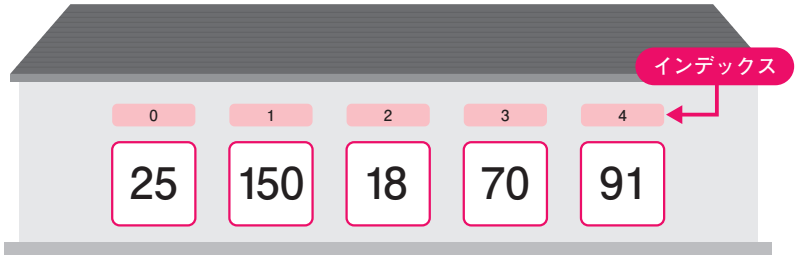


配列には、同じように処理したいデータのグループを収めて使います。処理対象のデータが大量にある場合、それを1つずつ変数に入れていくと大量の変数を用意しなければならなくなります。配列と繰り返し処理（第7章参照）を組み合わせれば、処理するデータがいくつあっても、用意する変数は1つだけで済みます。特に2次元までの配列は、Excelのワークシートのセル範囲とデータの構造に互換性があり、相互に簡単にデータをやり取りすることができます。

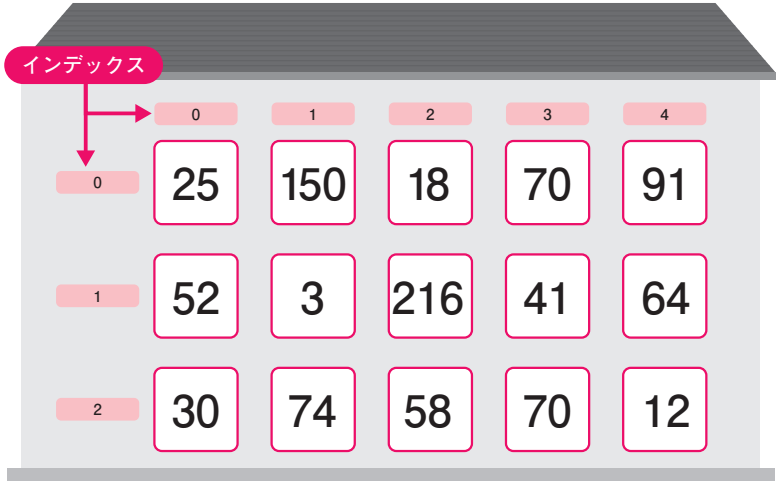


配列のインデックス

配列の中でも最も基本的な「1次元配列」の場合、データを収めるための部屋が、横方向1列に並んでいるイメージです。配列の中のデータを特定するために、各部屋には左から順番に部屋番号が付けられています。この番号のことを「インデックス」といいます。VBAの配列の場合、通常はインデックスの最小値が0になります。



データを収めるための部屋を縦×横の2方向に並べた配列のことを、「2次元配列」といいます。1次元配列を1階建ての長屋とするなら、2次元配列は2階建て以上のアパートです。この場合は、部屋番号（インデックス）も、左→右と上→下の2方向に付けていくことになります。Excelのワークシートのセルに最も近いのが、この2次元配列です。



2次元配列は、Excelのワークシートと同じ構造でデータを収納できます

SECTION 043

セルの行全体・列全体を選択する

対象のセルを含む行全体または列全体を、Rangeオブジェクトとして取得することができます。Rangeオブジェクトを対象とする操作の中には、行全体または列全体に対してしか実行できないものがあるため、個別のセルの選択状態を行または列全体に広げます。

□ 特定のセルを含む行全体を選択する

特定のセルを表すRangeオブジェクトのEntireRowプロパティで、そのセルを含む行全体を表すRangeオブジェクトを取得することができます。ここでは、C6セルを含む行全体を選択します。

SAMPLE | C6セルを含む行全体を選択 ▸s043_1

```
Sub m043_1()
    Range("C6").EntireRow.Select ← 行全体を取得
End Sub
```

実行結果

Before

	A	B	C	D	E	F	G
1							
2		店舗別販売数記録					
3							
4		店名	4月	5月	6月	合計	
5		中野店	532	567	496	1595	
6		高円寺店	608	575	613	1796	
7		阿佐ヶ谷店	501	468	523	1492	
8		合計	1641	1610	1632	4883	
9							
10					※荻窪店は対象外		
11							

After

	A	B	C	D	E	F	G
1							
2		店舗別販売数記録					
3							
4		店名	4月	5月	6月	合計	
5		中野店	532	567	496	1595	
6		高円寺店	608	575	613	1796	
7		阿佐ヶ谷店	501	468	523	1492	
8		合計	1641	1610	1632	4883	
9							
10							
11							

C6セルを含む行全体を選択

COLUMN

複数の行全体を選択する

1つのセルではなく、複数の行に渡るセル範囲を表すRangeオブジェクトのEntireRowプロパティでは、複数の行全体を表すRangeオブジェクトを取得できます。

SAMPLE | 複数の行全体を選択 ▸s043_2

```
Sub m043_2()
    Range("B4:C6").EntireRow.Select
End Sub
```

4〜6行の行全体を取得

□ 特定のセルを含む列全体を選択する

特定のセルを含む列全体を表すRangeオブジェクトを取得するには、EntireColumnプロパティを使用します。ここでは、ActiveCellプロパティでアクティブセルを表すRangeオブジェクトを取得し、そのセルを含む列全体を選択します。

SAMPLE | アクティブセルを含む列全体を選択 ▸s043_3

```
Sub m043_3()
    ActiveCell.EntireColumn.Select ← アクティブセルの列全体を取得
End Sub
```

実行結果

アクティブセルを含む列全体を選択

Before

	A	B	C	D	E	F	G
1							
2		店舗別販売数記録					
3							
4		店名	4月	5月	6月	合計	
5		中野店	532	567	496	1595	
6		高円寺店	608	575	613	1796	
7		阿佐ヶ谷店	501	468	523	1492	
8		合計	1641	1610	1632	4883	
9							
10					※荻窪店は対象外		

After

	A	B	C	D	E	F	G
1							
2		店舗別販売数記録					
3							
4		店名	4月	5月	6月	合計	
5		中野店	532	567	496	1595	
6		高円寺店	608	575	613	1796	
7		阿佐ヶ谷店	501	468	523	1492	
8		合計	1641	1610	1632	4883	
9							
10					※荻窪店は対象外		

□ 選択範囲を含む列全体を選択する

複数の列に渡るセル範囲を表すRangeオブジェクトのEntireColumnプロパティで、その複数の列全体を表すRangeオブジェクトを取得できます。Selectionプロパティで選択範囲を表すRangeオブジェクトを取得し、その列全体を選択します。

SAMPLE | 選択範囲を含む列全体を選択 ▸s043_4

```
Sub m043_4()
    Selection.EntireColumn.Select ← 選択範囲の列全体を取得
End Sub
```

実行結果

選択範囲を含む列全体を選択

Before

	A	B	C	D	E	F	G
1							
2		店舗別販売数記録					
3							
4		店名	4月	5月	6月	合計	
5		中野店	532	567	496	1595	
6		高円寺店	608	575	613	1796	
7		阿佐ヶ谷店	501	468	523	1492	
8		合計	1641	1610	1632	4883	

After

	A	B	C	D	E	F	G
1							
2		店舗別販売数記録					
3							
4		店名	4月	5月	6月	合計	
5		中野店	532	567	496	1595	
6		高円寺店	608	575	613	1796	
7		阿佐ヶ谷店	501	468	523	1492	
8		合計	1641	1610	1632	4883	

SECTION 061

書式だけ・値だけを貼り付ける

セル範囲の値ではなく、書式の設定だけを別のセル範囲にコピー・貼り付けすることができます。反対に、貼り付け先の書式はそのままにして、値だけを貼り付けることも可能です。これらはいずれも、通常とは異なる特殊な貼り付け方法を使用します。

セル範囲の書式を別のセル範囲にコピーする

特定のセル範囲の書式を別のセル範囲にコピーしたい場合も、元のセル範囲を表すRangeオブジェクトのCopyメソッドを実行します。次に貼り付け先のセル範囲を表すRangeオブジェクトのPasteSpecialメソッドで、引数Pasteに定数xlPasteFormatsを指定します。

SAMPLE | B2:C5のセル範囲の書式のみを複製 ▸s061_1

```
Sub m061_1()
    Range("B2:C5").Copy                                  ← コピー
    Range("E3:F4").PasteSpecial Paste:=xlPasteFormats    ← 書式を貼り付け
    Application.CutCopyMode = False
End Sub
```

実行結果

Before

	商品名	価格
3	天丼	¥1,200
4	かつ丼	¥1,000
5	親子丼	¥900

After（書式だけを複製）

	商品名	価格
3	天丼	¥1,200
4	かつ丼	¥1,000
5	親子丼	¥900

COLUMN

PasteSpecial メソッドで貼り付ける

PasteSpecialメソッドはPasteメソッドの代わりにも利用できますが、その対象は貼り付け先のセルを表すRangeオブジェクトです。必ずCopyメソッドと組み合わせて使い、Cutメソッドとの組み合わせでは使えません。

SAMPLE | PasteSpecialメソッドで貼り付ける ▸s061_2

```
Sub m061_2()
    Range("B2:C5").Copy                ← コピー
    Range("E3").PasteSpecial           ← 貼り付け
    Application.CutCopyMode = False
End Sub
```


セル範囲に数値を加算して貼り付ける

数値が入力されているセルをコピーし、別のセル範囲に入力されている数値と演算して貼り付けることができます。ここでは、引数Pasteに定数xlPasteValuesを指定して、値だけを貼り付けます。また、引数Operationに、加算を意味する定数xlPasteSpecialOperationAddを指定します。

SAMPLE | 値を加算して貼り付ける ▸s061_3

```
Sub m061_3()
    Range("E3").Copy                                     ← コピー
    Range("C3:C5").PasteSpecial Paste:=xlPasteValues, _
        Operation:=xlPasteSpecialOperationAdd            ← 加算して貼り付け
    Application.CutCopyMode = False
End Sub
```

実行結果

Before

	商品名	価格		値上げ額
3	天丼	¥1,200		100
4	かつ丼	¥1,000		
5	親子丼	¥900		

After（値上げ額を加算）

	商品名	価格		値上げ額
3	天丼	¥1,300		100
4	かつ丼	¥1,100		
5	親子丼	¥1,000		

COLUMN

PasteSpecialメソッドの引数

PasteSpecialメソッドには4つの引数を指定できますが、ここではPasteとOperationについて説明します。引数Pasteは、貼り付ける内容を指定するものです。指定できる定数の一部を表1に示します。また、引数Operationは、コピーした数値を貼り付け先のセルの値と演算する指定で、表2のような定数を指定できます。

【表1】

定数	貼り付け内容
xlPasteAll	すべて
xlPasteFormulas	数式
xlPasteVaues	値
xlPasteFormats	書式設定
xlPasteFormulasAndNumberFormats	数式と数値の書式
xlPasteValuesAndNumberFormats	値と数値の書式

【表2】

定数	演算の内容
xlPasteSpecialOperationNone	なし
xlPasteSpecialOperationAdd	加算
xlPasteSpecialOperationSubtract	減算
xlPasteSpecialOperationMultiply	乗算
xlPasteSpecialOperationDivide	除算

SECTION 074

2種類のシートの指定方法

シートの集合を表すコレクションから単体のシートのオブジェクトを取り出すには、コレクションにインデックスを指定して取得します。インデックスに指定できるのは、シート見出しの左からの並び順を表す数値、またはシート名の文字列です。

シートの種類を調べる

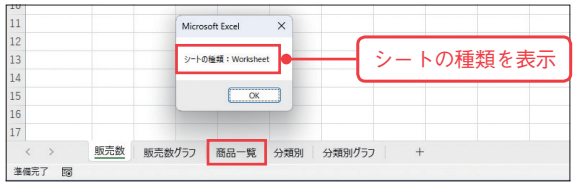
作業中のブックに含まれているすべてのシートの中で、シート見出しの左から3番目の位置に表示されているシートをWorksheetオブジェクトまたはChartオブジェクトとして取得するには、Sheetsコレクションにインデックスとして「3」を指定します。次の例は、データ型を表す文字列を返すTypeName関数で、取得したオブジェクトの種類を調べ、メッセージ画面に表示するコードです。

SAMPLE | 3番目のシートの種類を表示　s074_1

```
Sub m074_1()
    MsgBox "シートの種類:" & TypeName(Sheets(3))
End Sub
```

シートの種類

実行結果



COLUMN

SheetsコレクションからWorksheetオブジェクトを取得する

シートの集合を表すコレクションにインデックスとして指定する数値は、シート見出しの表示位置を左から数えた順番です。ここではSheetsコレクションに数値を指定し、その位置にある単体のオブジェクト（Worksheetオブジェクトまたは Chartオブジェクト）を取得しています。Sheetsコレクションではなく Worksheetsコレクションまたは Chartsコレクションにインデックスを指定して、オブジェクトを取得することも可能です。その場合、ワークシートまたはグラフシートだけを対象として、指定した順番に当たるシートが取得されます。

他シートのセルの値を調べる

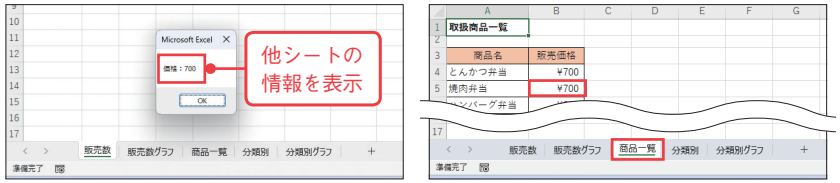
シートの集合を表すコレクションには、インデックスとしてシートの名前を指定し、特定のシートを表すオブジェクトを取得することもできます。次の例は、「商品一覧」というワークシートのB5セルの値をメッセージ画面に表示するコードです。

SAMPLE | 他シートの情報を表示　s074_2

```
Sub m074_2()
    MsgBox "価格:" & Worksheets("商品一覧").Range("B5").Value
End Sub
```

他シートのセルの値

実行結果



特定のグラフシートの名前を調べる

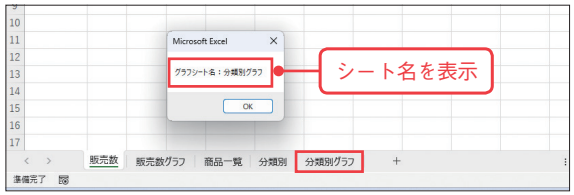
次の例では、Chartsコレクションのインデックスに「2」を指定して、左から2番目のグラフシートを表すChartオブジェクトを取得し、Nameプロパティでそのシート名を取り出して、メッセージ画面に表示しています。

SAMPLE | グラフシートの名前を表示　s074_3

```
Sub m074_3()
    MsgBox "グラフシート名:" & Charts(2).Name
End Sub
```

グラフシート名

実行結果



SECTION 095

メッセージ画面を使って処理を分ける

MsgBox関数は、これまで画面にメッセージを表示するためだけに使用してきましたが、その戻り値を取得して利用することも可能です。ここでは、メッセージ画面に複数のボタンを表示し、クリックされたボタンに応じて処理を分ける方法を紹介します。

▫「OK」と「キャンセル」で処理を分ける

ここまでのMsgBox関数の使用例では、引数として、表示するメッセージを表す第1引数Promptと、メッセージ画面のタイトルの表示を指定する第3引数Titleだけを指定していました。この関数で指定可能な引数は、これ以外にも、表示するボタンを表す第2引数Buttonsなどがあります。

メッセージ画面に「OK」と「キャンセル」というボタンを表示するには、引数Buttonsに定数vbOkCancelを指定します。また、戻り値を取得するには、引数を「( )」内に指定して、式の形で記述します。戻り値は数値で、「OK」がクリックされた場合は定数vbOKで、「キャンセルがクリックされた場合は定数vbCancelで判定できます。

次の例は、「選択範囲をすべてクリアしますか？」というメッセージを表示し、「OK」がクリックされた場合は選択範囲を書式ごとクリアします。「キャンセル」がクリックされた場合は何もしません。

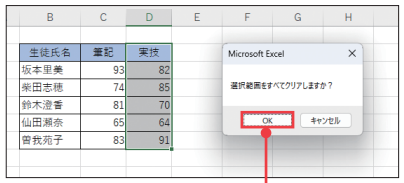
SAMPLE 「OK」がクリックされたらクリア　s095_1

```
Sub m095_1()
    If MsgBox(Prompt:="選択範囲をすべてクリアしますか？", _
        Buttons:=vbOKCancel) = vbOK Then Selection.Clear
End Sub
```

メッセージで確認してクリア

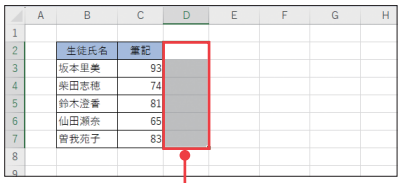
実行結果

Before



「OK」をクリック

After



選択範囲がクリアされる

▫3つのボタンに応じて処理を分ける

メッセージ画面に「はい」「いいえ」の2つのボタンを表示したいときはMsgBox関数の引数Buttonsに定数vbYesNoを、「はい」「いいえ」「キャンセル」の3つのボタンを表示したいときは定数vbYesNoCancelを指定します。戻り値は数値で、「はい」は定数vbYes、「いいえ」は定数vbNo、「キャンセル」は定数vbCancelで判定できます。

次の例では、「確認」というタイトルのメッセージ画面で「店内で召し上がりますか？」と表示し、その戻り値をいったん変数Ansに代入します。「はい」がクリックされた場合はB5セルの数値を1.1倍し、「いいえ」がクリックされた場合はB5セルの数値を1.08倍して、それぞれC5セルに入力します。「キャンセル」がクリックされた場合は何もしません。

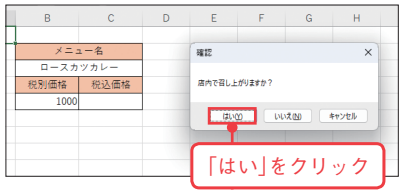
SAMPLE 2つの選択肢とキャンセルで処理を分ける　s095_2

```
Sub m095_2()
    Dim Ans As Integer
    Ans = MsgBox(Prompt:="店内で召し上がりますか？", _
        Buttons:=vbYesNoCancel, Title:="確認")
    If Ans = vbYes Then
        Range("C5").Value = Range("B5").Value * 1.1
    ElseIf Ans = vbNo Then
        Range("C5").Value = Range("B5").Value *1.08
    End If
End Sub
```

押されたボタンに応じて処理を変える

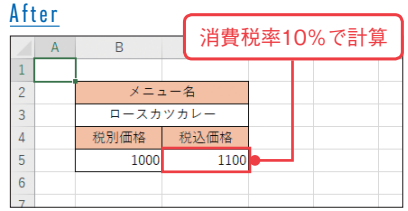
実行結果1

Before



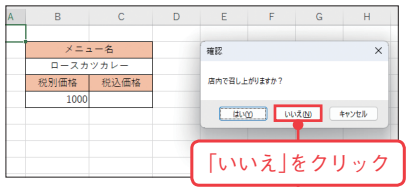
「はい」をクリック

After



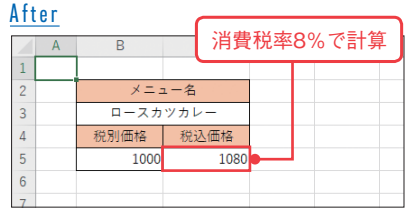
実行結果2

Before



「いいえ」をクリック

After



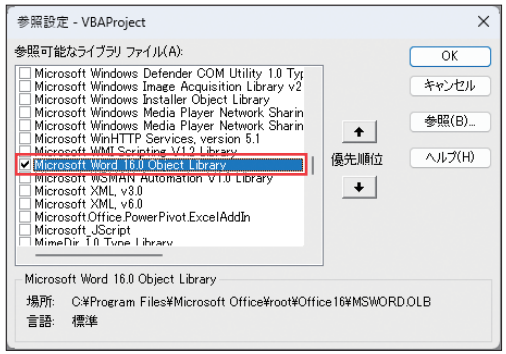
SECTION 126

Excel VBAでWordを操作する

Excel VBAで外部プログラムを操作する例として、まずWordの機能をExcelから利用する方法を紹介します。Wordの文書を新規作成したり、作成済みの文書ファイルから情報を取り出したり、その内容を修正したりすることが可能です。

Wordへの参照を設定する

ここでは、事前にWordのオブジェクトライブラリへの参照を設定します。P.277で解説した手順で［参照設定］ダイアログボックスを表示し、Office 2019以降またはMicrosoft 365版Wordの場合は「Microsoft Word 16.0 Object Library」にチェックを付けて、［OK］をクリックします。



なお、数字の部分はWordのバージョンによって異なります。事前に参照設定をせずにWordを使用可能にするには、CreateObject関数（P.276参照）の引数に「Word.Application.16」という文字列を指定し、その戻り値をオブジェクト変数にセットします。Wordのバージョンを限定したくない場合は、末尾の「.16」を省略することもできます。

Wordで新規文書を作成する

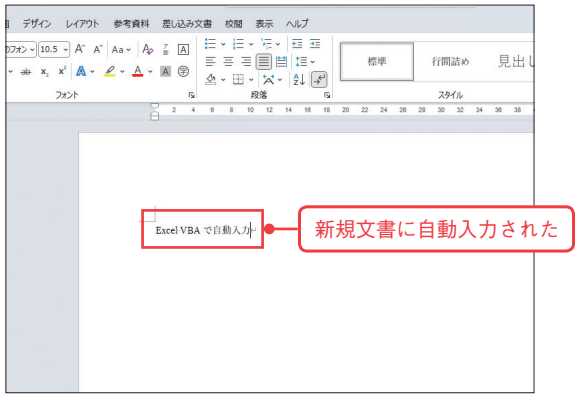
まず、Excel VBAでWordの新規文書を作成し、文章を自動入力する方法を紹介しましょう。Wordの文書は、VBAではDocumentオブジェクトとして表されます。また、開かれているすべてのWord文書はDocumentsコレクションです。

次の例は、まずWordのアプリケーションを表すオブジェクトを作成し、そのDocumentsプロパティでDocumentsコレクションを取得し、そのAddメソッドで新規文書を作成します。この時点ではWordのウインドウは非表示の状態なので、Visibleプロパティに Trueを設定することで見えるようにします。さらに、Selectionプロパティで、Wordのカーソル位置または選択範囲を表すSelectionオブジェクトを取得し、そのTypeTextメソッドで、引数に指定した文字列を新規文書に入力しています。

SAMPLE | Wordの文書を作成して文字列を入力 s126_1

```
Sub m126_1()
    Dim wd As New Word.Application ← Wordアプリケーションを生成
    wd.Documents.Add ← 新規文書を作成
    wd.Visible = True ← Wordを表示
    wd.Selection.TypeText Text:="Excel VBAで自動入力"
End Sub                              ← 文書に文字列を入力
```

実行結果



Documentオブジェクトを直接作成する

WordのDocumentオブジェクトをセットするための変数をNewキーワードを付けて宣言するだけでも、新規文書を作成できます。

次の例は、上のコードと同じ結果になります。VisibleプロパティはWordアプリケーションのプロパティなので、DocumentオブジェクトのApplicationプロパティで設定します。また、Selectionオブジェクトもアプリケーションに属するため、ここではDocumentオブジェクトのRangeメソッドで文書の範囲全体を表すRangeオブジェクトを取得し、そのTextプロパティに代入する形で文字列を入力しています。

SAMPLE | Wordの文書を作成して文字列を入力 s126_2

```
Sub m126_2()
    Dim wDoc As New Word.Document ← 新規文書のオブジェクトを作成
    wDoc.Application.Visible = True ← Wordを表示
    wDoc.Range.Text = "Excel VBAで自動入力" ← 文書に文字列を入力
End Sub
```