EXCEL VBA塾

動画と連動！

初心者OK！

補講編

YouTuber「エクセル兄さん」
たてばやし 淳［著］

マイナビ

もくじ

補講A　よく使うセルのプロパティ　001

レッスン1 ★☆☆　セルのフォントの書式に関するプロパティ　001
レッスン2 ★★☆　セルの罫線や塗りつぶしに関するプロパティ　013
レッスン3 ★★☆　セルの表示形式に関するプロパティ　021

補講B　行や列の操作　032

レッスン1 ★☆☆　行や列を指定するには?　032
レッスン2 ★★☆　行や列のプロパティやメソッドを利用する　036
レッスン3 ★★☆　行や列を挿入・削除する　045

補講C　もっと知っておきたいセル範囲の指定方法　049

レッスン1 ★☆☆　Rangeの応用的な使い方　049
レッスン2 ★★☆　セルを指定する様々な方法　055

補講D　上級編　071

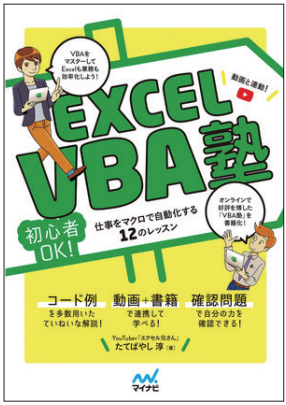
レッスン1 ★★☆　条件分岐:Likeを使った文字列のパターン判定　071
レッスン2 ★★★　条件分岐:論理演算子(And、Or、Not)を使った判定　079
レッスン3 ★★★　VBA関数:MsgBox関数の応用、InputBox関数、Format関数、InStr関数　086
レッスン4 ★★★　フィルターによるデータ抽出:複数の条件による抽出、範囲で抽出　101

補講E　デバッグ エラーが起きたらどうする?　111

補講の確認問題の模範解答　124

本書について

- この『Excel VBA塾　補講編』は、下記の本の補講的な内容をまとめたものです。



Excel VBA塾
たてばやし 淳　[著]
株式会社マイナビ出版
2021年11月刊行
ISBN4-974-8399-7572-2

- 本書中で「**書籍第●章**」といった表記が出てくる場合、「書籍」とは上記『**Excel VBA塾**』のことを指します。

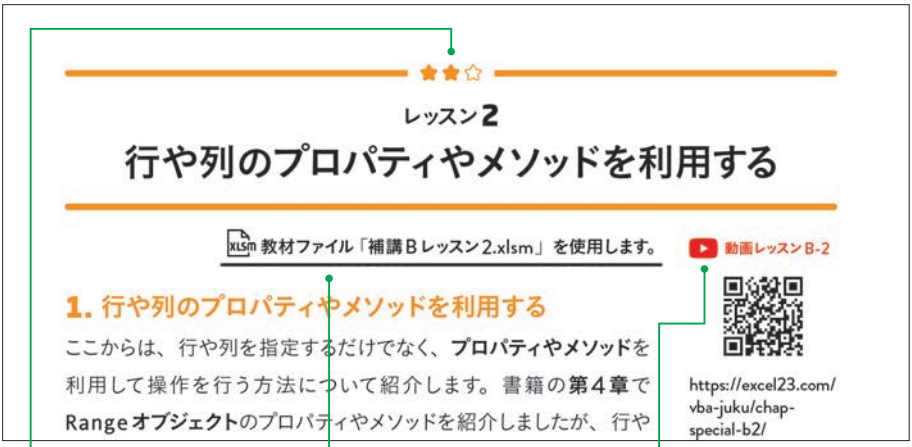
本書のサポートサイト

https://book.mynavi.jp/supportsite/detail/9784839975722-hokou.html

本書のサポートサイトです。教材ファイルはこちらからダウンロードできます。

- 本書に記載された内容は、情報の提供のみを目的としております。したがって、本書を用いての運用はすべてお客様自身の責任と判断において行ってください。
- 本書の制作にあたっては正確な記述につとめましたが、著者や出版社のいずれも、本書の内容に関してなんらかの保証をするものではなく、内容に関するいかなる運用結果についてもいっさいの責任を負いません。あらかじめご了承ください。
- 本書中に掲載している画面イメージなどは、特定の設定に基づいた環境にて再現される一例です。ハードウェアやソフトウェアの環境によっては、必ずしも本書通りの画面にならないことがあります。あらかじめご了承ください。
- 本書は2021年10月段階での情報に基づいて執筆されています。本書に登場するソフトウェアのバージョン、URL、製品のスペックなどの情報は、すべてその原稿執筆時点でのものです。執筆以降に変更されている可能性がありますので、ご了承ください。
- 本書中に登場する会社名および商品名は、該当する各社の商標または登録商標です。本書では®および™マークは省略させていただいております。

本書の構成



難易度：
このレッスンのレベルを示します

教材ファイル：
このレッスンで使用するファイルです

動画レッスン：
QRコードまたはURLから、このレッスンに連動した動画にアクセスできます

書式　**書式：**VBAの書式です

例　**例：**サンプルコードです

補足　コラム

補足、コラム：
併せて知っておきたいことなどを囲みで解説しています

	A	B	C	D	E	F
1	番号	物件名	所在地	価格(万円)	沿線・駅	間取り
2	1	南品川スイートホーム	東京都品川区南品川x	4480	JR京浜東北線「大井町」徒歩12分	2LDK+S
3	2	旗の台グランドステージ	東京都品川区旗の台xx	4480	東急大井町線「旗の台」徒歩5分	1LDK+S
4	3	西品川ゴールドハウス	東京都品川区西品川x-xx-xx	4590	東急大井町線「下神明」徒歩5分	3DK
5	4	東大井プレミアハウス	東京都品川区東大井x	4220	JR京浜東北線「大井町」徒歩5分	2LDK+S
6	5	小山ネクストハウス	東京都品川区小山xxx	6700	東急目黒線「洗足」徒歩5分	3LDK
7	6	東大井リトルプレイス	東京都品川区東大井xxx	5380	JR京浜東北線「大井町」徒歩9分	4LDK
8	7	東大井ガーデンコート	東京都品川区東大井xx	5980	京急本線「立会川」徒歩8分	1LDK+2S

確認問題：レッスンの最後に入っています。理解度の確認に使いましょう

補講A：よく使うセルのプロパティ

実用マクロにおいてよく行う処理は、なんといっても「セルの操作」です。そこで本章では、「セルの操作」について補足解説を行います。

以下のような、実用マクロでよく使うセルのプロパティを紹介します。

- **セルのフォントの書式（太字、斜体、下線、フォントの名前、色など）**
- **セルの罫線や塗りつぶし**
- **セルの表示形式**

★☆☆

レッスン1

セルのフォントの書式に関するプロパティ

教材ファイル「補講Aレッスン1.xlsm」を使用します。

動画レッスンA-1

https://excel23.com/vba-juku/chap-special-a1/

1.「セルのフォントの書式」で、どんなことができるか？

セルのフォントの書式に関するプロパティを利用すると、例えば

- フォントの太字設定
- フォントのサイズ
- フォント名
- フォントの色

などを変更することができます。

2. フォントの書式を変更するには？

フォントの書式を変更するためには、「**Fontオブジェクト**」を利用します。Fontオブジェクトは、セルのフォントに関する情報をまとめているオブジェクトです。Fontオブジェクトを利用するためには、以下のようにコードを記述します。

書式

```
Rangeオブジェクト.Font.[プロパティ] = 設定値
```

- 上記の[プロパティ]の部分には、設定したい書式によって様々なプロパティを記述します。

以下は、よく使われるプロパティの一覧です。

表A-1-1

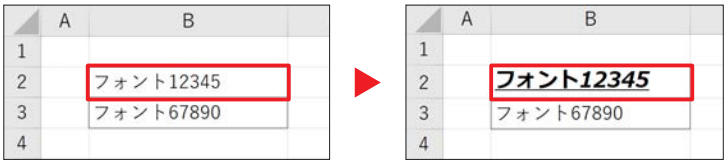
プロパティ名	意味	設定値
Font.Bold	太字	Trueで有効化、Falseで無効化
Font.Italic	斜体	Trueで有効化、Falseで無効化
Font.Underline	下線	Trueで有効化、Falseで無効化
Font.Size	大きさ	数値で指定 （標準値は「Application.StandardFontSize」）
Font.Name	フォント名	文字列でフォント名を指定 （標準値は「Application.StandardFont」）
Font.Color	フォントの色	色定数または数値（RGB関数） 詳しくは**4.**で解説
Font.ColorIndex	（旧）フォントの色	1~56の数値で指定 フォントの色を「自動」に戻すには「xlAutomatic」

3. フォントの書式を変更してみよう（色以外）

2.で紹介したプロパティを利用して、フォントの書式を変更するコード例を紹介します。

[例] セルB2に次の書式を設定する

- 太字、斜体、下線を適用
- フォントの大きさを「12」、フォント名を「メイリオ」に変更



例

```
Sub exampleA_1_3()
    Range("B2").Font.Bold = True          '太字
    Range("B2").Font.Italic = True        '斜体
    Range("B2").Font.Underline = True     '下線
    Range("B2").Font.Size = 12            'フォントの大きさを「12」に変更
    Range("B2").Font.Name = "メイリオ"    'フォント名を「メイリオ」に変更
End Sub
```

- Font.Bold、Font.Italic、Font.Underlineは、「True」を代入するとその書式が有効になります。逆に、「False」を代入すると無効になります。
- Font.Sizeには、フォントの大きさを数値で設定します。
- Font.Nameにはフォント名を記述します。

「Font.Size」や「Font.Name」に代入する設定値は、Excelの「Home」タブで、フォント名やフォントの大きさを入力する値と同様です。

補足

同じオブジェクトは「With構文」で省略できる

3.のコード例では、同じ「Range("B2").Font.…」というコードを何度も繰り返し記述する必要がありました。このような場合は、「With～End With構文」によって省略して記述することができます。以下に、書式を紹介します。

書式

```
With オブジェクト
    「.」以降のコード
End With
```

- 「With オブジェクト」～「End With」で、構文の開始と終了を意味します。

以下に、省略前と省略後のコード例を比較しながら説明します。

[例1] 省略前のコード（赤字は、何度も同じコードを記述している箇所です）

例

```
Sub exampleA_1_3補足a省略前()
    Range("A5").Value = 100
    Range("A5").Copy Range("B5")
    Range("A5").Clear
End Sub
```

[例1'] 省略後のコード

例

```
Sub exampleA_1_3補足a省略後()
    With Range("A5")
        .Value = 100
        .Copy Range("B5")
        .Clear
    End With
End Sub
```

- 省略前のコードでは「Range("A5")」と**何度も記述**する必要がありました。一方、省略後のコードでは、「With Range("A5")」～「End With」と記述することで、その記述を省略することができます。
- With～End Withの構文内では、「.」以降のコードを記述すればよいということになります。
- なお、With～End Withの構文の中のコードは、Tabキーで一段字下げ（インデント）して記述するのが一般的な作法です。

[例2] 省略前のコード（赤字は、何度も同じコードを記述している箇所です）

例

```
Sub exampleA_1_3補足b省略前()
    Range("B2").Font.Bold = True          '太字
    Range("B2").Font.Italic = True        '斜体
    Range("B2").Font.Underline = True     '下線
    Range("B2").Font.Size = 12            'フォントの大きさ
    Range("B2").Font.Name = "メイリオ"      'フォント名
End Sub
```

[例2'] 省略後のコード

例

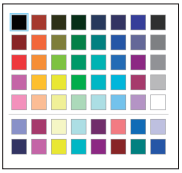
```
Sub exampleA_1_3補足b省略後()
    With Range("B2").Font
        .Bold = True          '太字
        .Italic = True        '斜体
        .Underline = True     '下線
        .Size = 12            'フォントの大きさ
        .Name = "メイリオ"      'フォント名
    End With
End Sub
```

- 省略前のコードでは「Range("B2").Font」と**何度も記述する**必要がありました。一方、省略後のコードでは、「With Range("B2").Font」～「End With」と記述することで、その記述を省略することができます。
- With～End With構文は、「With 親オブジェクト.子オブジェクト」のように親子関係のオブジェクトを「.」でつなげて記述し、まとめて省略することもできます。今回のコード例において「Range("B2").Font」は、Rangeオブジェクトと Fontオブジェクトが親子関係にあたるため、まとめて省略することができます。

4. フォントの色を変更する

フォントの色を変更するプロパティは、次の2種類があります。

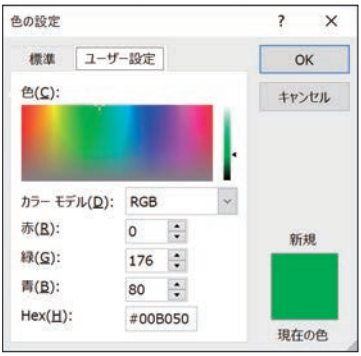
- Excel2003以前ではFont.ColorIndexプロパティ（その後のバージョンでも使用可能です）
- Excel2007以降ではFont.Colorプロパティ



Excel2003以前：Font.ColorIndexプロパティ

Rangeオブジェクト.Font.ColorIndex=設定値

・1～56の数値で設定



Excel2007以降：Font.Colorプロパティ

Rangeオブジェクト.Font.Color=設定値

・カラー定数やRGB関数で設定

本書では、主にFont.Colorプロパティを紹介します。

書式

```
Rangeオブジェクト.Font.Color = カラー定数またはRGB関数
```

- Font.Colorプロパティの設定値は、主に「カラー定数」「RGB関数」という2つの方法で指定できます。
- 「カラー定数」は、色を指定できる8種類の定数です。定数名を記述すれば、それに対応する色を指定できます。

表A-1-2

定数名	色
vbBlack	黒
vbRed	赤
vbGreen	緑
vbYellow	黄

定数名	色
vbBlue	青
vbMagenta	紫
vbCyan	シアン
vbWhite	白

RGB関数は、光の3原色（赤,緑,青）の組み合わせで色を指定できる関数です。RGB関数は、以下の書式で記述します。

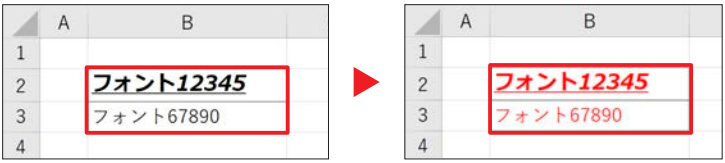
書式

```
RGB(赤,緑,青)
```

- 赤，緑，青の各引数に0～255の整数を記述して、色の濃さを指定します。
- 例えば、「RGB(255,100,0)」と指定した場合、赤は255（最大）、緑は100（少し）、青は0（なし）というバランスの混合色になり、結果としてオレンジっぽい色になります。

以下に具体的なコード例を紹介します。

[例] セルB2のフォントの色をカラー定数で「赤」に変更する
セルB3のフォントの色をRGB関数で「255, 0, 0」に変更する



例

```
Sub exampleA_1_4()
    Range("B2").Font.Color = vbRed
    Range("B3").Font.Color = RGB(255, 0, 0)
End Sub
```

- 上記を実行すると、いずれのセルもフォントの色が**赤色**になります。
- 「vbRed」は、カラー定数で「赤」を意味します。
- RGB(255,0,0)は、赤=255,緑=0,青=0を混ぜた色合いになるため、結果として赤色になります。

補足

なぜRGB関数が必要か?

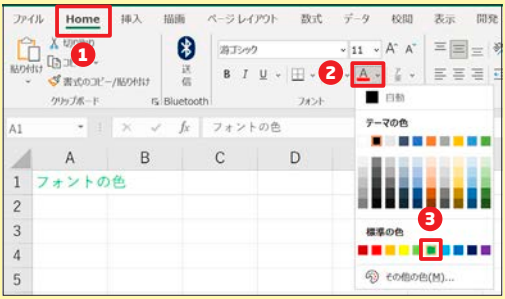
現在のExcelは、約1600万色を表現できます。したがって、Colorプロパティには、0～16777215の数値を代入して色を設定することができます。しかし、このような膨大な数値の範囲から、自分の選びたい色を指定するのは現実的には難しいです。そこで、RGB関数で「赤、緑、青」のバランスにより色を指定する方法がよく使用されています。

補足

指定したい色のRGBの値を調べるには？

自分が指定したい色のRGBの値を調べる方法を紹介します。

1 まずはマクロを使わずに、Excelシート上で手動でフォントの色を設定します。
［Home］タブ（❶）から「フォントの色」ボタン（❷）を押し、任意の色を選ぶ（❸）



2 次に、以下の操作で、その色のRGBの値を調べます。
もう一度［Home］タブの「フォントの色」ボタン（❹）を押し、「その他の色」を選択する（❺）

「ユーザー設定」タブ（❻）の「カラーモデル」が「RGB」になっていることを確認する（❼）。
赤（R）、緑（G）、青（B）という値が表示される（❽）ので、その3つの値をメモしておく

以上の操作で、自分の指定したいフォントの色のRGB値を調べることができます。
その値を使って、RGB関数の引数を記述しましょう。

補足

フォントの設定を標準状態に戻すには？

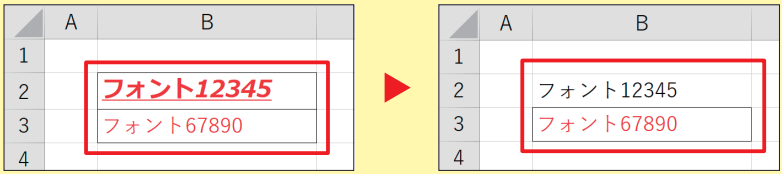
フォントの太字、斜体、下線、フォントの色、フォント名といった値を**すべて元に戻して標準状態にするには**どうしたらいいでしょうか。

最も簡単な方法は、「ClearFormatsメソッド」を使う方法です。ClearFormatsメソッドは、セルの書式だけをクリアして値だけ残します。

[例1] セルB2の書式をクリアする

例

```
Range("B2").ClearFormats
```

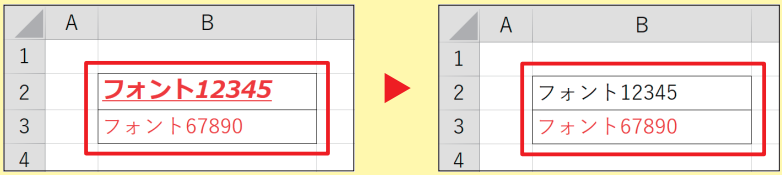


罫線も消えてしまう

しかし、上記の方法の問題点は、**セルの表示形式や罫線や塗りつぶしの色までクリア**されてしまう点です。

そこで、表示形式や罫線や塗りつぶしの色はクリアせずに、**フォントの書式だけを標準状態に戻す**コード例を紹介します。

[例2] セルB2のフォントの書式を標準状態に戻す



罫線を残したまま書式を標準状態に戻す

例

```
Sub exampleA_1_4補足()

    With Range("B2").Font

        '太字,斜体,下線は「False」を代入する
        .Bold = False          '太字
        .Italic = False        '斜体
        .Underline = False     '下線

        'サイズとフォント名は標準値を代入する
        .Size = Application.StandardFontSize
        .Name = Application.StandardFont

        'フォントの色を「自動」に戻す
        .ColorIndex = xlAutomatic

    End With

End Sub
```

- 太字、斜体、下線の書式は、各プロパティに「False」を代入すれば無効化されます。
- フォントサイズには「Application.StandardFontSize」という値を、フォント名には「Application.StandardFont」という値をそれぞれ代入することで、標準のフォントサイズ、標準のフォント名に変更することができます。
- フォントの色は、ColorIndexプロパティに「xlAutomatic」という値を代入することで、カラーパレットの「自動」という設定に戻すことができます。

確認問題

教材ファイル「補講Aレッスン1確認問題.xlsm」を使用します。

	A	B	C	D	E	F
1	番号	物件名	所在地	価格(万円)	沿線・駅	間取り
2	1	南品川スイートホーム	東京都品川区南品川x	4,480	JR京浜東北線「大井町」徒歩12分	2LDK+S
3	2	旗の台グランドステージ	東京都品川区旗の台xx	4,480	東急大井町線「旗の台」徒歩5分	1LDK+S
4	3	西品川ゴールドハウス	東京都品川区西品川x-xx-xx	4,590	東急大井町線「下神明」徒歩5分	3DK
5	4	東大井プレミアハウス	東京都品川区東大井x	4,220	JR京浜東北線「大井町」徒歩5分	2LDK+S
6	5	小山ネクストハウス	東京都品川区小山xxx	6,700	東急目黒線「洗足」徒歩5分	3LDK
7	6	東大井リトルプレイス	東京都品川区東大井xxx	5,380	JR京浜東北線「大井町」徒歩9分	4LDK
8	7	東大井ガーデンコート	東京都品川区東大井xx	5,980	京急本線「立会川」徒歩8分	1LDK+2S
9	8	五反田の新築一戸建て	東京都品川区西五反田x	5,980	JR山手線「五反田」徒歩14分	3LDK
10	9	大井グレートアクセス	東京都品川区大井x	5,020	JR京浜東北線「大森」徒歩11分	2LDK+S
11	10	旗の台セカンドステージ	東京都品川区旗の台xx	5,740	東急大井町線「北千束」徒歩5分	3LDK
12						

1. セル範囲A1:F1のフォントに「太字」を適用してください。
2. セル範囲A1:F1のフォントの色を赤色に変更してください(赤色は、カラー定数で指定するか、RGB関数で赤:255、緑:0、青:0で指定してください)。
3. セル範囲A1:F1のフォント名を「メイリオ」に変更してください。
4. IF構文とFor～Next構文を使って、[間取り]列のセルの値が"3LDK"に一致する場合、そのセルに「太字」を適用してください。
5. If構文とFor～Next構文を使って、[価格(万円)]列のセルの値が5000以上の場合、セルに「太字」を適用してください。

動画レッスンA-1t

https://excel23.com/vba-juku/chap-special-a1test/

※模範解答は、PDFの巻末に掲載しています。

★★☆

レッスン2

セルの罫線や塗りつぶしに関するプロパティ

教材ファイル「補講Aレッスン2.xlsm」を使用します。

動画レッスンA-2

https://excel23.com/vba-juku/chap-special-a2/

1. セルの罫線や塗りつぶしを変更する

セルの罫線や塗りつぶしに関するプロパティを利用すると、

- 表に罫線を設定する
- セル範囲に塗りつぶしの色を設定する

といった処理を行うことができます。

2. セルの塗りつぶしの色を設定するには？

セルの塗りつぶしの色を変更するプロパティは2種類あります。

- Excel2003以前では`Interior.ColorIndex`プロパティ（その後のバージョンでも使用可能です）
- Excel2007以降では`Interior.Color`プロパティ

本書では、主に`Interior.Color`プロパティを紹介します。

どこかで見覚えがある2種類ではないでしょうか？　実は、本章**レッスン1**の**4.**で紹介した、**フォントの色を変更するプロパティと使い方は同じ**なのです（フォントの色は、「`Font.ColorIndex`」と「`Font.Color`」の2種類のプロパティでした）。

書式

```
Rangeオブジェクト.Interior.Color = カラー定数またはRGB関数
```

- `Interior.Color`プロパティの設定値は、主に「カラー定数」「RGB関数」という2つの方法で指定できます。
- カラー定数は、色を指定できる8種類の定数です。定数名を記述すれば、それに対応する色を指定できます（カラー定数は**補講Aレッスン1**の**4.**参照）。

RGB関数は、光の3原色（赤、緑、青）の組み合わせで色を指定できる関数です。RGB関数は、以下の書式で記述します。

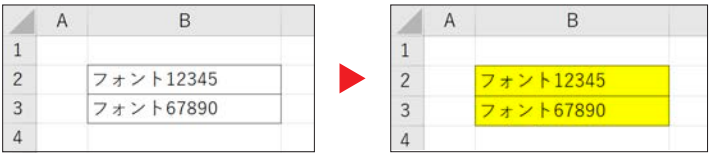
書式

```
RGB(赤,緑,青)
```

- 赤,緑,青の各引数に0～255の整数を記述して、色の濃さを指定します。

以下に具体的なコード例を紹介します。

[例] セルB2の塗りつぶしの色をカラー定数で「黄」に変更する
セルB3の塗りつぶしの色をRGB関数で「255, 255, 0」に変更する



例

```
Sub exampleA_2_2()
    Range("B2").Interior.Color = vbYellow
    Range("B3").Interior.Color = RGB(255, 255, 0)
End Sub
```

- 上記を実行すると、いずれのセルも塗りつぶしの色が黄色になります。
- 「vbYellow」は、カラー定数で「黄」を意味します。
- RGB(255,255,0)は、赤=255,緑=255,青=0を混ぜた色合いになるため、結果として黄色になります。

○○.Colorプロパティや○○.ColorIndexプロパティは、フォントの色やセルの塗りつぶしの色だけでなく、罫線の色、図形の色、グラフの色など、様々なオブジェクトの色に共通のプロパティです。使い方や代入する値も同じなので、共通して把握しておくとよいでしょう。

補足

塗りつぶしの色を「なし」に戻すには？

塗りつぶしの色を「なし」（無色）に戻すには、Interior.ColorIndexプロパティに定数「xlNone」を代入する方法があります。

[例] セルB2の塗りつぶしの色を「なし」に変更する

例

```
Range("B2").Interior.ColorIndex = xlNone
```

3. セルの罫線を設定するには？

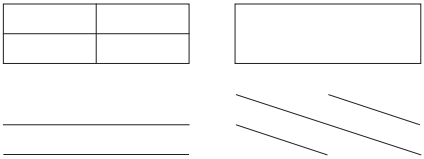
セルの罫線を設定するには、以下のようなコードを記述します。

書式

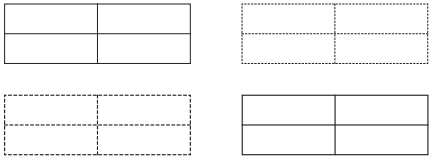
```
Rangeオブジェクト.Borders(位置).LineStyle = 設定値
```

- Borders()の引数で、罫線の**位置**を指定することができます。
- LineStyleに設定値を代入することで、**罫線のスタイル**を指定することができます。

罫線の位置（Borders()の引数で指定）



線のスタイル（LineStyleに代入する値で指定）



Borders()の引数（罫線の位置）

そのセル範囲のどの位置に罫線を指定するか、定数で指定できます。
以下に定数の一覧を紹介します。

表A-2-1

定数	説明
xlDiagonalDown	範囲内の各セルの左上隅から右下方向への罫線
xlDiagonalUp	範囲内の各セルの左下隅から右上への罫線
xlEdgeBottom	範囲内の下側の罫線
xlEdgeLeft	範囲内の左端の罫線
xlEdgeRight	範囲の右端の罫線
xlEdgeTop	範囲内の上側の罫線
xlInsideHorizontal	範囲内の全セルの水平罫線（外枠は除く）
xlInsideVertical	範囲内の全セルの垂直罫線（外枠は除く）

LineStyleに代入する設定値

罫線の種類を定数で指定できます。以下に定数の一覧を紹介します。

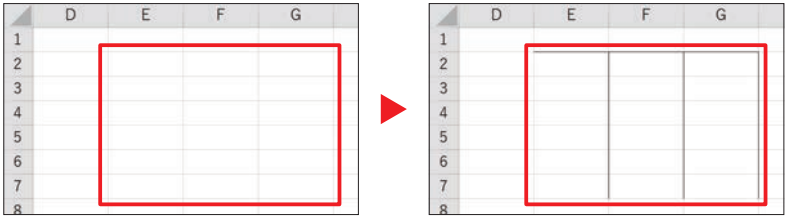
表A-2-2

定数	説明
xlContinuous	実線
xlDash	破線
xlDashDot	一点鎖線
xlDashDotDot	二点鎖線
xlDot	点線
xlDouble	2本線
xlLineStyleNone	線なし（削除）
xlSlantDashDot	斜破線

※罫線を削除する場合は「xlLineStyleNone」を指定すると覚えておきましょう。

以下に具体的なコード例を紹介します。

[例] セル範囲 E2:G7 において、上端、右端、全セルの垂直罫線を適用する（すべて線のスタイルは「実線」とする）



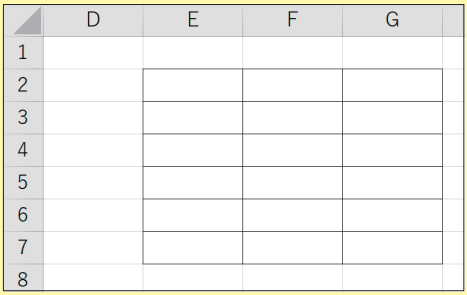
例

```
Sub exampleA_2_3()
    Range("E2:G7").Borders(xlEdgeTop).LineStyle = xlContinuous
    Range("E2:G7").Borders(xlEdgeRight).LineStyle = xlContinuous
    Range("E2:G7").Borders(xlInsideVertical).LineStyle = _
        xlContinuous
End Sub
```

- Borders()の引数にそれぞれ「xlEdgeTop（上端）」「xlEdgeRight（右端）」「xlInsideVertical（全セルの垂直罫線）」を指定しています。
- 例えば上記の引数に「xlEdgeTop（上端）」を指定した場合、セル範囲の全セルの上側に罫線が適用されるわけ**ではない**点に注意しましょう。あくまで**セル範囲の上端**にだけ罫線が適用されます。
- LineStyleプロパティには、いずれも「xlContinuous（実線）」を代入しています。よって、すべて実線の罫線が適用されます。

補足

もっと簡単に罫線（格子）を適用するウラ技コード



ここまで紹介したコードの記述方法ですと、罫線の位置を1つ1つ指定する必要があります。そのため、格子の罫線を引くためには、上下左右、水平・垂直方向にそれぞれ罫線を適用するコードを書かなければなりません。

これではコードも多くなってしまい、作業効率も良くありませんね。そこで、もっと簡単に手早く格子の罫線を適用できるウラ技のようなコードを紹介します。

書式

```
Sub exampleA_2_3補足()
    Range("E2:G7").Borders().LineStyle = xlContinuous
End Sub
```

- 上記のコードだけで、セル範囲全体に格子の罫線を引くことができます。
- Borders()の引数には何も記述せず省略します。すると、セル範囲の個々のセルの上下左右の位置が指定されます（「Borders()」の()を省略して「Borders」とだけ記述しても可です）。
- LineStyleに代入する値を「xlContinuous」（実線）とすることで、個々のセルの上下左右に実線が適用されます。
- 上記の結果、セル範囲全体に格子の罫線を引くことができるのです。

なお、罫線を削除する方法として、上記のコードの「xlContinuous」を「xlLineStyleNone（線なし）」に変更すれば、セル範囲の罫線がすべて削除されます。こちらも知っておきましょう。

4. 罫線の色を設定するには

罫線の色を設定するには、以下のようなコードを記述します。

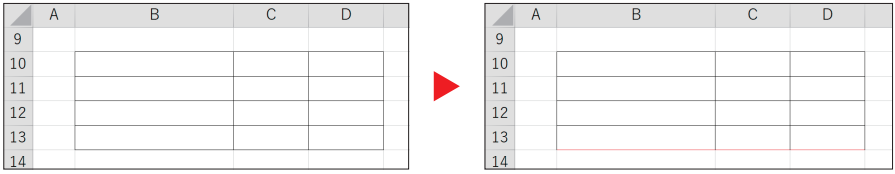
書式

```
Rangeオブジェクト.Borders(位置).Color = カラー定数またはRGB関数
```

- Borders() の引数で、罫線の**位置**を指定することができます。
- Color に設定値を代入することで、**罫線の色**を指定することができます。
- 色の設定値は、フォントの色やセルの塗りつぶしの色と同様で、カラー定数またはRGB関数で指定できます（詳細は、**レッスン1**の**4.**を参照してください）。

以下に具体的なコード例を紹介します。

[例] セル範囲 B10:D13 の下端の罫線の色を、赤色に設定する



例

```
Sub exampleA_2_4()
    Range("B10:D13").Borders(xlEdgeBottom).Color = vbRed
End Sub
```

- 「Borders(xlEdgeBottom)」と記述することで、セル範囲の下端の位置が指定されます。
- Color プロパティの設定値を「vbRed」とすることで、「赤」になります。

5. 罫線の太さを設定するには

罫線の太さを設定するには、以下のようなコードを記述します。

書式

```
Rangeオブジェクト.Borders(位置).Weight = 設定値
```

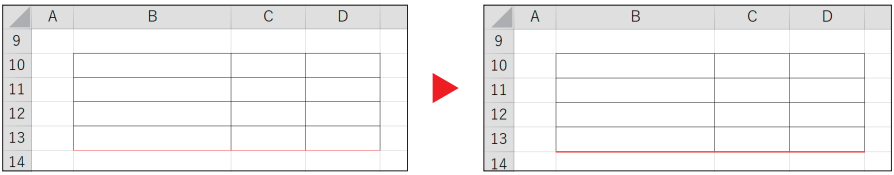
- Borders()の引数で、罫線の**位置**を指定することができます。
- Weightに設定値を代入することで、**罫線の太さ**を指定することができます。
- 設定値は、定数を記述することで、対応する太さを指定できます。設定値に使用できる定数の一覧を紹介します。

表A-2-3

定数	説明
xlHairline	最も細い
xlMedium	中間の太さ
xlThick	太線
xlThin	細い線（標準）

以下に具体的なコード例を紹介します。

[例] セル範囲B10:D13の下端の罫線の太さを「太線」に変更する



例

```
Sub exampleA_2_5()
    Range("B10:D13").Borders(xlEdgeBottom).Weight = xlThick
End Sub
```

- Borders(xlEdgeBottom)と記述することで、セル範囲の下端の位置が指定されます。
- Weightの設定値を「xlThick」とすることで、「太線」になります。

確認問題

教材ファイル「補講Aレッスン2確認問題.xlsm」を使用します。

	A	B	C	D	E	F	G
1							
2		売上日	店舗	商品型番	販売単価	売上数量	
3		2021/04/07	KC	D88-F-PPL	30,320	14	
4		2021/04/13	KJ	T32-S-RED	18,600	15	
5		2021/04/17	KC	Y46-M-RED	25,200	14	
6		2021/04/25	SJ	J01-T-PPL	25,000	15	
7		2021/05/01	KC	L01-T-RED	52,000	15	
8		2021/05/07	SJ	D88-F-BLK	14,980	9	
9		2021/05/13	YH	Y46-M-GRN	23,000	6	
10		2021/05/19	KJ	Q81-S-BLU	28,500	10	
11		2021/05/27	KJ	J01-T-PPL	25,000	14	
12		2021/05/31	HO	J01-T-PPL	25,000	6	
13		2021/06/06	AK	Z20-M-BLU	9,800	6	
14							

動画レッスンA-2t

https://excel23.com/vba-juku/chap-special-a2test/

1. セル範囲B2:F2に塗りつぶしの色（黄色）を適用してください。
   ※黄色は、カラー定数で指定するか、RGBの赤255、緑:255、青:0で指定してください。
2. 表全体に罫線（格子）を適用してください。
3. セル範囲B2:F2の下側に罫線（二重線）を適用してください。

※模範解答は、PDFの巻末に掲載しています。

★★☆

レッスン3

セルの表示形式に関するプロパティ

教材ファイル「補講Aレッスン3.xlsm」を使用します。

動画レッスンA-3

https://excel23.com/vba-juku/chap-special-a3/

1. セルの表示形式を変更する

セルの表示形式に関するプロパティを利用すると、日付の表示形式を変更したり、数値をカンマ区切りのスタイルに変更することができます。

2. セルの表示形式とは？

セルの表示形式とは、セルの見た目上の表示のことです。

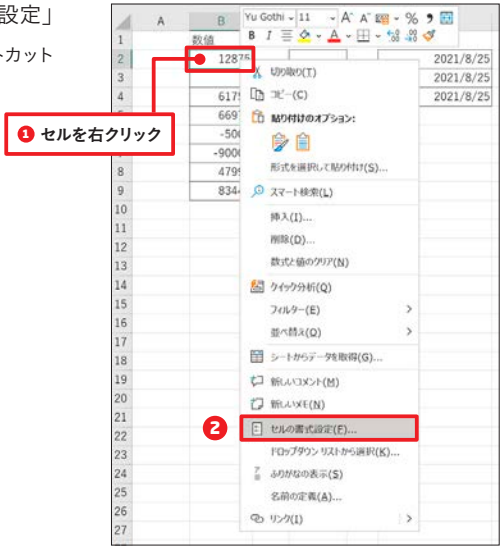
例えば、セルの値は同じ「99999」でも、セルの表示形式によって右のように見た目を変更できます。

表 A-3-1

見た目	説明
99,999	カンマ区切り
¥99999	通貨（円）
99999.0	小数点以下1桁

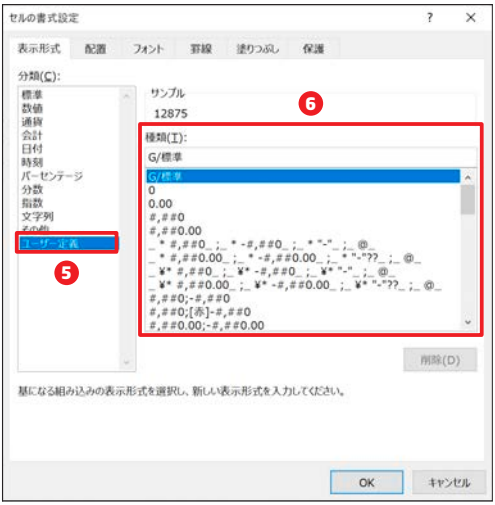
マクロを使用しない場合、セルの表示形式は次の操作で変更できます。

1 セルを右クリック（❶）して「セルの書式設定」を選択（❷）する（またはCtrl+1のショートカットキー）。



2 「表示形式」タブ（❸）で、分類（❹）を選択する（数値、通貨、会計…etcから選択）

3 また、「ユーザー定義」(❺)を選択すると、**書式記号**(❻)という特殊な記号を使って自由に表示形式を指定することができます。
VBAで操作する場合は**3.**から紹介するNumberFormatlLocalプロパティで設定することができます。



セルの表示形式の設定をしっかり行っていなかった場合、データが誤った形でセルに表示されてしまうことがあります。例えば"12-30"という文字列をセルに代入した結果、それが日付データだと解釈されてしまい、セルには「12月30日」などと表示されてしまうことがあります。このような事態を避けるためには、代入する前に、あらかじめセルの表示形式を「文字列」に変更しておくなど、表示形式を適切に設定しておく必要があります。

3. セルの表示形式を変更する

セルの表示形式は、NumberFormatLocalプロパティで設定できます。

書式

```
Rangeオブジェクト.NumberFormatLocal = 書式記号
```

- 右辺には、「書式記号」と呼ばれる文字列を記述します(「書式指定文字」と呼ばれることもあります)。
- 書式記号は、**書籍第12章レッスン4**で紹介した「Format関数」で使える記号とほぼ同様です。

以下に、よく使われる書式記号を紹介します。

表A-3-2　日付関連の書式記号（よく使われるもの）

書式記号	説明
yyyy	西暦の年
m	月（1桁の月の場合は1桁で表記）
mm	月（1桁の月の場合は先頭に0を付加して「01」などと表記）
d	日（1桁の日の場合は1桁で表記）
dd	日（1桁の日の場合は先頭に0を付加して「01」などと表記）
ggg	元号（例：「令和」）
e	和暦の年
ggge	元号＋年（例：「令和3」）

表A-3-3　数値関連の書式記号（よく使われるもの）

書式記号	説明
#	1桁の数値（その桁の数値がない場合は何も入らない）
0	1桁の数値（その桁の数値がない場合も0が入る）
#,###	3桁おきにコンマ区切り（0の場合は何も入らない）
#,##0	3桁おきにコンマ区切り（0の場合は0が入る）

表A-3-4　文字列関連の書式記号

書式記号	説明
@	文字列

補足

「NumberFormatLocal」と「NumberFormat」の違い

「NumberFormatLocalプロパティ」の他にも、「NumberFormatプロパティ」という名前の似たプロパティが存在します。これらの違いは何でしょうか。

ひとことでいうと、NumberFormatLocalの方が**日本語環境に対応している**という違いがあります。

例えば、セルの表示形式を「標準」に変更するとき、NumberFormat**Local**の場合は"G/標準"という文字列を代入します。一方、NumberFormatの場合は"G/標準"という文字列には対応していないため、"General"という文字列を代入する必要があります。

例

```
Range("A1").NumberFormatLocal = "G/標準"   '日本語環境に対応
Range("A1").NumberFormat = "General"       '対応していない
```

また、セルの表示形式を¥のついた数値（例えば¥1000と表示する）に変更するとき、NumberFormatLocalの場合は"¥0"という文字列を代入すれば可能です。一方、NumberFormatの場合は"¥0"には対応しておらず、"""¥""0"という文字列を代入する必要があります。

例

```
Range("A1").NumberFormatLocal = "¥0"      '日本語環境に対応
Range("A1").NumberFormat = """¥""0"       '対応していない
```

- """¥""0"という記述についてですが、外側の両端の「"」は文字列として記述するための「"」です。また、その中に文字列として「"」を記述したい場合は、「""」と2つ続けて記号を書かなければいけないのがVBAのルールであるため、2つ続けて書いています。

以上のように、NumberFormatLocalプロパティの方が日本語環境に対応しているため、国内で利用する上ではNumberFormatLocalプロパティの方がより自然に使用できるといえます。

4. 数値の表示形式を変更する

ここでは、数値の表示形式を変更する具体例を紹介します。

	A	B
1		数値
2		12875
3		0
4		61755
5		66971
6		-5000
7		-90000
8		47991
9		83445
10		

標準

	A	B
1		数値
2		12,875
3		
4		61,755
5		66,971
6		-5,000
7		-90,000
8		47,991
9		83,445
10		

例1の結果

	A	B
1		数値
2		12,875
3		0
4		61,755
5		66,971
6		-5,000
7		-90,000
8		47,991
9		83,445
10		

例2の結果

	A	B
1		数値
2		12875.0
3		0.0
4		61755.0
5		66971.0
6		-5000.0
7		-90000.0
8		47991.0
9		83445.0
10		

例3の結果

[例1] セル範囲 B2:B9 の表示形式を "#,###" に変更する

例

```
Sub exampleA_3_4a()
    Range("B2:B9").NumberFormatLocal = "#,###"
End Sub
```

- "#,###" と指定すると、3桁おきに桁区切りのカンマ（,）が表示されます。
- ただし、値が0のセルは、0が省略されてしまいセルは空白の表示になります。

[例2] セル範囲 B2:B9 の表示形式を "#,##0" に変更する

例

```
Sub exampleA_3_4b()
    Range("B2:B9").NumberFormatLocal = "#,##0"
End Sub
```

- [例1] と同じく、3桁おきに桁区切りのカンマ（,）が表示されます。
- 値が0のセルも、省略されずに「0」と表示されます。

[例3] セル範囲 B2:B9 の表示形式を "0.0" に変更する

例

```
Sub exampleA_3_4c()
    Range("B2:B9").NumberFormatLocal = "0.0"
End Sub
```

- 小数第1位まで表示されます。
- 小数第1位が「0」だった場合も、省略されずに「0」と表示されます。

5. 表示形式を「文字列」に変更する

例えばセルに "00025" のような文字列を代入しようとしても、セルの表示形式が標準のままだと、数値データとみなされてしまいます。すると、上位の桁の「0」が省略され、「25」

標準の場合

	C	D	E
1		文字列	
2		25	
3			

という数値だけがセルに代入される結果になってしまいます。それを避けるためには、前もってセルの表示形式を「文字列」に変更しておく方法があります。セルの表示形式が「文字列」になっていれば、"00025"のような文字列が代入されても上位の桁の「0」は省略されません。セルには「00025」と表示されます。
以下にコード例を紹介します。

表示形式が「文字列」の場合

	C	D	E
1		文字列	
2		00025	
3			

[例] セルD2の表示形式を文字列に変更し、セルD2に「00025」という文字列を代入

例

```
Sub exampleA_3_5()
    Range("D2").NumberFormatLocal = "@"
    Range("D2").Value = "00025"
End Sub
```

- NumberFormatLocalプロパティに"@"を代入すると、表示形式が文字列になります。
- その後、"00025"という文字列をセルD2に代入しています。

6. 日付の表示形式を変更する

セルの日付が同じでも、表示形式によって見た目は大きく変わります。ここでは、日付の表示形式を変更するコード例を紹介します。

	E	F
1		日付
2		2022/1/25
3		2022/1/25
4		2022/1/25

▼

	E	F
1		日付
2		2022/01/25
3		2022年1月25日
4		令和4年1月25日(火)

[例] セルF2の表示形式を"yyyy/mm/dd"に変更する
セルF3の表示形式を"yyyy年m月d日"に変更する
セルF3の表示形式を"ggge年m月d日(aaa)"に変更する

例

```
Sub exampleA_3_6()
    Range("F2").NumberFormatLocal = "yyyy/mm/dd"
    Range("F3").NumberFormatLocal = "yyyy年m月d日"
    Range("F4").NumberFormatLocal = "ggge年m月d日(aaa)"
End Sub
```

- yyyyは年（西暦）、mmは月（2桁表示）、ddは日（2桁表示）を表示します。
- yyyy年m月d日と記述すると、年,月,日という文字列が間に挿入されます。
- ggeは元号＋年（和暦）、aaaは曜日の先頭1文字を表示します。

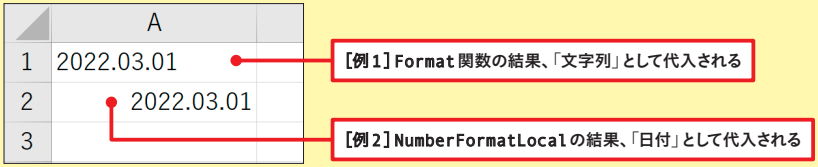
補足

「Format関数」とNumberFormatLocalの違いとは？

よくある質問として「Format関数とNumberFormatLocalの違いは？」という質問があります。
どちらも書式記号を使うという点で似ているのですが、役割や用途は異なります。

- Format関数は、あるデータの書式を変更し、文字列として返すVBA関数です。
- NumberFormatLocalは、セルの表示形式を設定します。

以下に、例を用いて説明します。



[例1] Format関数を用いて、"2022/3/1"という日付を、"2022.03.01"という文字列に変換し、その結果をセルA1に代入するコード例

例

```
Sub exampleA_3_6補足1()
    Range("A1").Value = _
        Format("2022/3/1", "yyyy.mm.dd")
End Sub
```

上記のコードを実行すると、文字列として"2022.03.01"がセルに代入されますが、セル上では日付データとして扱われず、あくまで文字列として扱われます。

その証拠に、セル内で値が左揃えになっています（［例1］の上の図を参照）。このセルの値は、あくまで文字列として扱われるので、日付として計算したり日付として抽出や並べ替えをすることができません。
上記の原因は、Format関数の役割は文字列を返すことであり、セルの表示形式を変えることではないからです（たまたまFormat関数の戻り値によってセルの表示形式も自動的に変更されることもありますが、それはたまたまの現象です）。

［例2］NumberFormatLocalプロパティを用いて、セルA2の表示形式を"yyyy.mm.dd"に変更し、そのセルに"2022/3/1"という日付を代入するコード例

例

```
Sub exampleA_3_6補足2()
    Range("A2").NumberFormatLocal = "yyyy.mm.dd"
    Range("A2").Value = "2022/3/1"
End Sub
```

上記のコードの結果、セルの表示形式は"yyyy.mm.dd"に変更され、見た目上にも「2022.03.01」と表示されます（［例1］の上の図を参照）。また、セルに代入されたデータは日付として扱われるため、セル内で右揃えになっています。

以上のように、Format関数とNumberFormatLocalプロパティは似ているので混同しがちですが、もともと役割と用途は異なるので、気をつけて選択しましょう。

補足

実は奥が深い、セルの表示形式

セルの表示形式は、実はとても奥が深く、かなり細かい設定まで行うことができます。しかし、結論をいうと、実務上はそこまで複雑なセルの表示形式を使いこなさなくてもよい、と筆者は考えます。

例えば、ユーザー定義の書式設定では、以下のような設定をすることも可能です。

書式

```
¥#,##0;[赤]"△"¥#,##0;0;@ "様"
```

正の数の書式;負の数の書式;ゼロの書式;文字列の書式

上記の意味をまとめると、以下のようにあらわせます。

表A-3-5

正の数の場合	¥つきで桁区切りに表示	¥98,765
負の数の場合	赤色で△に続けて同上	△¥98,765
ゼロの場合	0と表示	0
文字列の場合	文字列に続けて"様"を表示	鈴木 様

上記のように、ユーザー定義の書式設定では、「;」で区切って指定することで、次のようにデータの種類別に書式を指定できます。

書式

正の数の書式;負の数の書式;ゼロの書式;文字列の書式

その他にも、

- 等号や不等号などの比較演算子を使用し、条件によって表示形式を分ける
- セル幅いっぱいに「-----」などを表示させる

などの独自のカスタマイズを行う方法もあります。

しかし、先述しましたが、実務上はそこまで複雑な表示形式を使いこなさなくてもよいと筆者は考えます。
なぜなら、この機能自体、あまり一般のExcelユーザーに認知されていないからです。表示形式についてよく知らずにシートを触った人が「このセル、いったいどうなってるの?」と混乱する可能性があります。
また、VBAと併用する場合にはもっと注意が必要です。

- セルの表示形式で条件分岐するのか？
- VBAのコードで条件分岐するのか？

どちらで処理をするのかルール化しておかないと、どちらで何を条件分岐しているのか分からなくなり、混乱を生む原因になりかねません。
結論として、セルの表示形式については、深入りしすぎず、基本的なことさえ押さえておくに留めることをおすすめします。

確認問題

教材ファイル「補講Aレッスン3確認問題.xlsm」を使用します。

	A	B	C	D	E	F	G
1	番号	売上日	店舗	商品型番	販売単価	売上数量	販売金額
2	1	2021/4/7	KC	D88-F-PPL	30320	14	424480
3	2	2021/4/13	KJ	T32-S-RED	18600	15	279000
4	3	2021/4/17	KC	Y46-M-RED	25200	14	352800
5	4	2021/4/25	SJ	J01-T-PPL	25000	15	375000
6	5	2021/5/1	KC	L01-T-RED	52000	15	780000
7	6	2021/5/7	SJ	D88-F-BLK	14980	9	134820
8	7	2021/5/13	YH	Y46-M-GRN	23000	6	138000
9	8	2021/5/19	KJ	Q81-S-BLU	28500	10	285000
10	9	2021/5/27	KJ	J01-T-PPL	25000	14	350000
11	10	2021/5/31	HO	J01-T-PPL	25000	6	150000
12	11	2021/6/6	AK	Z20-M-BLU	9800	6	58800

動画レッスンA-3t

https://excel23.com/vba-juku/chap-special-a3test/

1. E列～G列の表示形式を、カンマ区切りの数値形式に変更してください。
2. B列の表示形式を、"yyyy/mm/dd"形式に変更してください。
3. A列の表示形式を、3桁の数値（桁が空白の場合でも0を表示すること）に変更してください。

※模範解答は、PDFの巻末に掲載しています。

補講B：行や列の操作

実用マクロにおいては、行や列を操作することが頻繁にあります。
そこで本章では、行や列を操作する基本について以下の補足解説を行います。

- **行や列を指定する**
- **行や列のプロパティやメソッド（値のクリア、コピー、表示形式、列幅や行の高さ）**
- **行や列の挿入・削除**

★☆☆

レッスン1

行や列を指定するには？

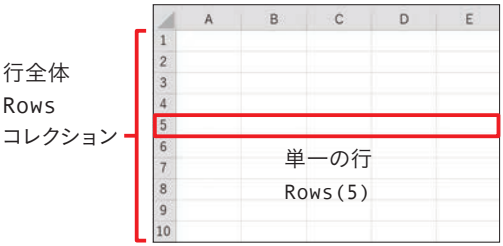
教材ファイル「補講Bレッスン1.xlsm」を使用します。

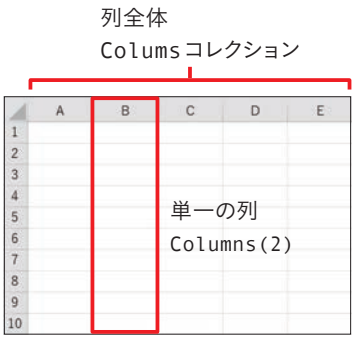
動画レッスンB-1

https://excel23.com/vba-juku/chap-special-b1/

1. 行や列は「コレクション」に属している

行や列を指定する上で、「コレクション」という概念を理解しておきましょう。
「コレクション」とは、同じ種類のオブジェクトの集合のことです。
例えば、行や列もコレクションになっています。





- **行全体の集合＝Rowsコレクション**
- **列全体の集合＝Columsコレクション**

（どちらも複数形の「s」が付いていることに注意しましょう）

そして、単一の行や単一の列は、それぞれ**コレクションに属する1つのオブジェクト**です。
例えば、単一の行や列を指定するには、以下のようなコードを記述します。

例

```
Rows(5)        'シートの5行目を指定する
Columns(2)     'シートの2列目を指定する
```

続いて**2.**で、行や列の指定方法についてもう少し詳しく解説していきます。

2. 行を指定するには「Rows」

行を指定する「Rows」、列を指定する「Columns」については**書籍第11章レッスン1**で簡単に説明しました。もう少し詳しく解説していきましょう。
以下はRowsの書式です。

書式

```
Rows(行番号)
```

- 行番号は、シートの上端から「1,2,3…」と数えた数値で行を指定できます。
- 行番号は、"3:5"のように記述すると3～5行目を範囲指定できます。このとき、""で囲って文字列として記述する必要があります。

例

```
Rows(3)        'シートの3行目を指定
Rows("3:5")    'シートの3～5行目を範囲指定
```

以下に具体的なコード例を紹介します。

[例] シートの3～5行目を選択する（Selectメソッドを使用）

例

```
Sub exampleB_1_2()
    Rows("3:5").Select
End Sub
```

- コードを実行すると、シートの3～5行目が選択されます。
- 「.Select」はSelectメソッドといい、指定したセル範囲を選択します。

3. 列を指定するには「Columns」

列を指定する「Columns」の書式は以下になります。

書式

```
Columns(列番号)
```

- 列番号は、シートの左端から「1,2,3…」と数えた数値で列を指定できます。
- 列番号は、"A"や"B"のように列名を指定することもできます（""で囲う必要あり）。
- 列番号は、"B:D"のように記述するとB～D列を複数指定できます（""で囲う必要あり）。
- ただし、"2:4"のように番号で範囲指定することは仕様上、できません。

例

```
  Columns(2)        '2列目(すなわちB列)
  Columns("B")      'B列を指定
  Columns("B:D")    'B:D列を範囲指定
× Columns("2:4")    '※これは不可能で、エラーになります
```

- "2:4"のように記述して複数列を指定することはできません。Rowsではそのようなことが可能なのですが、Columnsでは不可能なので注意しましょう。

以下に具体的なコード例を紹介します。

[例] シートのC～E列を選択する（Selectメソッド）

例

```
Sub exampleB_1_3()
    Columns("C:E").Select
End Sub
```

- コードを実行すると、シートのC～E列が選択されます。

よくある質問に「列を指定する場合には以下のどちらがよいですか？」というものがあります。

- 「Columns("A")」のように列名を記述する書式
- 「Columns(1)」のように番号を記述する書式

結論からいうと「場合による」という回答になります。
例えば、シート上で範囲が決まっている列を指定する場合には、「Columns("A")」のように列名を記述した方が、コードを読みやすいというメリットがあります。
一方、変数を使って「Columns(num)」（変数numには整数が入る）のようにコードを記述する場合は、番号を使用する書式の方が適しています。
場合によってどちらも記述できるようにしておくとよいでしょう。

確認問題

教材ファイル「補講Bレッスン1確認問題.xlsm」を使用します。

	A	B	C	D	E	F
1	番号	物件名	所在地	価格(万円)	沿線・駅	間取り
2	1	南品川スイートホーム	東京都品川区南品川x	4,480	JR京浜東北線「大井町」徒歩12分	2LDK+S
3	2	旗の台グランドステージ	東京都品川区旗の台xx	4,480	東急大井町線「旗の台」徒歩5分	1LDK+S
4	3	西品川ゴールドハウス	東京都品川区西品川x-xx-xx	4,590	東急大井町線「下神明」徒歩5分	3DK
5	4	東大井プレミアハウス	東京都品川区東大井x	4,220	JR京浜東北線「大井町」徒歩5分	2LDK+S
6	5	小山ネクストハウス	東京都品川区小山xxx	6,700	東急目黒線「洗足」徒歩5分	3LDK
7	6	東大井リトルプレイス	東京都品川区東大井xxx	5,380	JR京浜東北線「大井町」徒歩9分	4LDK
8	7	東大井ガーデンコート	東京都品川区東大井xx	5,980	京急本線「立会川」徒歩8分	1LDK+2S
9	8	五反田の新築一戸建て	東京都品川区西五反田x	5,980	JR山手線「五反田」徒歩14分	3LDK
10	9	大井グレートアクセス	東京都品川区大井x	5,020	JR京浜東北線「大森」徒歩11分	2LDK+S
11	10	旗の台セカンドステージ	東京都品川区旗の台xx	5,740	東急大井町線「北千束」徒歩5分	3LDK
12						

1. シートの8行目を選択（Select）してください。
2. シートのD列を選択（Select）してください。

3. シートの3～6行目を選択（Select）してください。
4. シートのB:E列を選択（Select）してください。
5. Long型の変数「num」を宣言し、numに「5」を代入してください。
次に、num行目を選択（Select）してください。
6. Long型の変数「maxRow」を宣言し、表の最終行をmaxRowに代入してください。
次に、maxRow行目を選択（Select）してください。

動画レッスンB-1t

https://excel23.com/vba-juku/chap-special-b1test/

※模範解答は、PDFの巻末に掲載しています。

★★☆

レッスン2

行や列のプロパティやメソッドを利用する

教材ファイル「補講Bレッスン2.xlsm」を使用します。

動画レッスンB-2

https://excel23.com/vba-juku/chap-special-b2/

1. 行や列のプロパティやメソッドを利用する

ここからは、行や列を指定するだけでなく、**プロパティやメソッド**を利用して操作を行う方法について紹介します。書籍の**第4章**で**Rangeオブジェクト**のプロパティやメソッドを紹介しましたが、行や列においても**同様のプロパティやメソッド**を使用できるものが多々あります。また、行や列に特有のプロパティやメソッドもあります。

本レッスンでは、行や列のプロパティやメソッドのうち、実務的によく使うものを紹介します。

- ClearContentsメソッド　：値をクリアする
- Copyメソッド　：コピーする
- NumberFormatLocalプロパティ　：表示形式を取得または設定する
- AutoFitメソッド　：行の高さや列幅を自動調整する
- RowHeightプロパティ　：行の高さを取得または設定する
- ColumnWidthプロパティ　：列幅を取得または設定する

2. 行や列の値をクリアする（ClearContentsメソッド）

	A	B	C	D	E	F
1	番号	物件名	所在地	価格(万円)		間取り
2	1	南品川スイートホーム	東京都品川区南品川x	4480		2LDK+S
3	2	旗の台グランドステージ	東京都品川区旗の台xx	4480		1LDK+S
4	3	西品川ゴールドハウス	東京都品川区西品川x-xx-xx	4590		3DK
5						
6	5	小山ネクストハウス	東京都品川区小山xxx	6700		3LDK
7	6	東大井リトルプレイス	東京都品川区東大井xxx	5380		4LDK
8	7	東大井ガーデンコート	東京都品川区東大井xx	5980		1LDK+2S
9	8	五反田の新築一戸建て	東京都品川区西五反田x	5980		3LDK
10	9	大井グレートアクセス	東京都品川区大井x	5020		2LDK+S
11	10	旗の台セカンドステージ	東京都品川区旗の台xx	5740		3LDK
12						

行の値をクリア

列の値をクリア

ClearContentsメソッドを使用すると、行や列の値をクリアできます。

書式

```
Rows(引数).ClearContents        '行の値をクリアする
Columns(引数).ClearContents     '列の値をクリアする
```

- 上記の「引数」には、RowsまたはColumnsの引数を記述します。
- ClearContentsメソッドは、**書籍第４章**で紹介したセルのClearContentsメソッドと同じものです。

[例1] 5行目の値をクリアする

例

```
Sub exampleB_2_2a()
    Rows(5).ClearContents
End Sub
```

- 「Rows(5)」でシートの5行目を指定し、「.ClearContents」で値をクリアします。

[例2] E列の値をクリアする

例

```
Sub exampleB_2_2b()
    Columns("E").ClearContents
End Sub
```

●「Columns("E")」でE列を指定し、「.ClearContents」で値をクリアします。

「ClearContentsメソッド」では値のみがクリアされますが、その他にも、書式と値をすべてクリアする「Clearメソッド」や、書式のみをクリアする「ClearFormats」メソッドも利用できます（**書籍第4章**で紹介したRangeオブジェクトのメソッドと同様です）。

3. 行や列をコピーして別の場所に転記する（Copyメソッド）

	A	B	C	D	E	F
1	番号	物件名	所在地	価格(万円)	所在地	間取り
2	1	南品川スイートホーム	東京都品川区南品川x	4480	東京都品川区南品川x	2LDK+S
3	2	旗の台グランドステージ	東京都品川区旗の台xx	4480	東京都品川区旗の台xx	1LDK+S
4	3	西品川ゴールドハウス	東京都品川区西品川x-xx-xx	4590	東京都品川区西品川x-xx-xx	3DK
5	2	旗の台グランドステージ	東京都品川区旗の台xx	4480	東京都品川区旗の台xx	1LDK+S
6	5	小山ネクストハウス	東京都品川区小山xxx	6700	東京都品川区小山xxx	3LDK
7	6	東大井リトルプレイス	東京都品川区東大井xxx	5380	東京都品川区東大井xxx	4LDK
8	7	東大井ガーデンコート	東京都品川区東大井xx	5980	東京都品川区東大井xx	1LDK+2S
9	8	五反田の新築一戸建て	東京都品川区西五反田x	5980	東京都品川区西五反田x	3LDK
10	9	大井グレートアクセス	東京都品川区大井x	5020	東京都品川区大井x	2LDK+S
11	10	旗の台セカンドステージ	東京都品川区旗の台xx	5740	東京都品川区旗の台xx	3LDK
12						

Copyメソッドを使用すると、行や列をコピーして別の場所に転記できます。

書式

```
Rows(引数).Copy Rows(引数)          '行をコピーする
```

コピー元　貼り付け先

```
Columns(引数).Copy Columns(引数)    '列をコピーする
```

コピー元　貼り付け先

●コピー元となる行や列を指定し、「.Copy」を記述します。

●引数には、貼り付け先の行や列を記述します。

[例1] 3行目をコピーして5行目に貼り付ける

例

```
Sub exampleB_2_3a()
    Rows(3).Copy Rows(5)
End Sub
```

●「Rows(3)」がコピー元で、貼り付け先が「Rows(5)」となります。

[例2] 3列目（C列）をコピーして5列目（E列）に貼り付ける

例

```
Sub exampleB_2_3b()
    Columns(3).Copy Columns(5)
End Sub
```

- 「Columns(3)」がコピー元で、貼り付け先が「Columns(5)」となります。
- 上記の代わりに「Columns("C")」や「Columns("E")」と記述することも可能です。

[例3] 3～5行目をコピーして10行目に貼り付ける

例

```
Sub exampleB_2_3c()
    Rows("3:5").Copy Rows(10)
End Sub
```

- 「Rows("3:5")」がコピー元で、貼り付け先が「Rows(10)」となります。
- このように、複数行をコピーする場合でも、貼り付け先は単一の行を指定すればよいという仕組みになっています。

Excelを手動で操作するときも、複数行をコピーして貼り付ける場合は、貼り付け先は単一の行でもよいという仕組みになっています。それとVBAのCopyメソッドも、それと同じ仕組みと覚えておくとよいでしょう。

4. 行や列の表示形式を設定する（NumberFormatLocalプロパティ）

NumberFormatLocalプロパティを使用すると、行や列のセルの表示形式を変更できます。

セルの表示形式（NumberFormatLocalプロパティ）については、詳しくは**補講A：よく使うセルのプロパティ**の**レッスン3**で解説しています。
本レッスンでは詳しい説明は割愛し、使い方だけを説明しています。

書式

```
Rows(引数).NumberFormatLocal = 設定値        '行の表示形式を変更する
Columns(引数).NumberFormatLocal = 設定値     '列の表示形式を変更する
```

- 設定値には、書式記号を記述します。
- 書式記号についての詳細説明は、**補講A**の**レッスン3**を参照ください。

[例1]1行目のセルの表示形式を"@"(文字列)に変更する

例

```
Sub exampleB_2_4a()
    Rows(1).NumberFormatLocal = "@"
End Sub
```

- "@"を代入することで、表示形式を文字列に変更しています。

[例2]7列目(G列)の表示形式を"#,##0"に変更する

例

```
Sub exampleB_2_4b()
    Columns(7).NumberFormatLocal = "#,##0"
End Sub
```

- "#,##0"を代入することで、3桁ごとにカンマ区切りのある表示形式に変更しています。

5. 行の高さや列の幅を自動調整する(AutoFitメソッド)

AutoFitメソッドを使用すると、行の高さや列の幅を自動調整できます。

書式

```
Rows(引数).AutoFit          '行の高さを自動調整する
Columns(引数).AutoFit       '列幅を自動調整する
```

[例1]1行目の行の高さを自動調整する

例

```
Sub exampleB_2_5a()
    Rows(1).AutoFit
End Sub
```

- 「Rows(1)」でシートの1行目を指定し、「AutoFit」で行の高さを自動調整します。

[例2]5列目(E列)の列幅を自動調整する

例

```
Sub exampleB_2_5b()
    Columns(5).AutoFit
End Sub
```

- 「Columns(5)」で5列目を指定し、「AutoFit」で列幅を自動調整します。
- 上記は「Columns("E")」と記述しても同じ列を指定できます。

[例3]1〜3列目(A:C列)の列幅を自動調整する

例

```
Sub exampleB_2_5c()
    Columns("A:C").AutoFit
End Sub
```

- 上記のように、複数列を範囲指定してAutoFitメソッドを実行することも可能です。

補足

飛び飛びの列を自動調整するには?

上記では「A～C列」のように複数の列が一続きになっている範囲を自動調整しました。ですが、例えば「A列、C:D列、F列」のように飛び飛びの列範囲を自動調整したい場合は、どのように記述すればよいでしょうか。
なお、Columnsを使って「Columns("A,C:D,F")」のように記述することはできません。

1つ目の方法は、

```
Columns("A").AutoFit
Columns("C:D").AutoFit
Columns("F").AutoFit
```

のように記述する方法です。飛び飛びの列ならば、列の塊ごとにコードを分ければよいということで、単純ですが確実な方法です。

2つ目の方法は、

```
Range("A1,C1:D1,F1").EntireColumn.AutoFit
```

のように記述する方法です。
まず、「Range("A1,C1:D1,F1")」のように記述することで、飛び飛びのセル範囲を一度に指定できます。また、「.EntireColumn」で、そのセルを含んでいる列全体を指定することができます。
したがって、A列、C:D列、F列を同時に指定することができます。なお、行で同様のことを行う場合は、「.EntireColumn」の代わりに「.EntireRow」を使用します。

6. 列幅や行の高さを数値で設定する(RowHeight、ColumnWidthプロパティ)

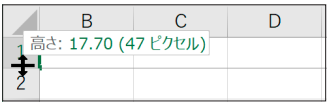
5.のように自動調整するのではなく、列幅や行の高さを特定の数値に変更する方法もあります。行の高さはRowHeightプロパティ、列幅はColumnWidthプロパティで変更できます。

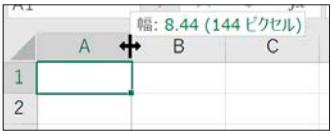
書式

```
Rows(引数).RowHeight = 設定値                    '行の高さを変更する
Columns(引数).ColumnsWidth = 設定値              '列幅を変更する
```

- 設定値には、整数または小数を記述します。
- 単一の行や列だけでなく、複数の行、複数の列を範囲指定してRowHeightやColumnWidthを変更することも可能です。

Excelにおいては、行の高さはポイントという単位で、列幅は文字数単位で設定されます。以下の図は、セルの行の境界線をドラッグした状態と、列の境界線をドラッグした状態に表示される設定値です。図では、行の高さが「17.70（ポイント）」、列幅が「8.44（文字）」として設定されていることがわかります。





VBAで行の高さや列幅を指定する前に、まずは図のように手動で行の高さや列幅を操作してみて、数値の目安を確かめてみるとよいでしょう。

[例]1行目の行の高さを「25」に、5列目の列の幅を「18」に変更する

例

```
Sub exampleB_2_6()
    Rows(1).RowHeight = 25                  '行の高さ
    Columns(5).ColumnWidth = 18             '列幅
End Sub
```

- 「Rows(1)」でシートの1行目を指定し、「.RowHeight = 25」で行の高さを25に変更しています。
- 「Columns(5)」で5列目を指定し、「.ColumnWidth = 18」で列幅を18に変更しています。

確認問題

教材ファイル「補講Bレッスン2確認問題.xlsm」を使用します。

	A	B	C	D	E	F
1	番号	物件名	所在地	価格(万円)	沿線・駅	間取り
2	1	南品川スイートホーム	東京都品川区南品川x	4480	JR京浜東北線「大井町」徒歩12分	2LDK+S
3	2	旗の台グランドステージ	東京都品川区旗の台xx	4480	東急大井町線「旗の台」徒歩5分	1LDK+S
4	3	西品川ゴールドハウス	東京都品川区西品川x-xx-xx	4590	東急大井町線「下神明」徒歩5分	3DK
5	4	東大井プレミアハウス	東京都品川区東大井x	4220	JR京浜東北線「大井町」徒歩5分	2LDK+S
6	5	小山ネクストハウス	東京都品川区小山xxx	6700	東急目黒線「洗足」徒歩5分	3LDK
7	6	東大井リトルプレイス	東京都品川区東大井xxx	5380	JR京浜東北線「大井町」徒歩9分	4LDK
8	7	東大井ガーデンコート	東京都品川区東大井xx	5980	京急本線「立会川」徒歩8分	1LDK+2S
9	8	五反田の新築一戸建て	東京都品川区西五反田x	5980	JR山手線「五反田」徒歩14分	3LDK
10	9	大井グレートアクセス	東京都品川区大井x	5020	JR京浜東北線「大森」徒歩11分	2LDK+S
11	10	旗の台セカンドステージ	東京都品川区旗の台xx	5740	東急大井町線「北千束」徒歩5分	3LDK
12						

1. 11行目の値をクリアしてください。
2. B～D列をコピーして、H列に貼り付けてください。
3. D列の表示形式を、カンマ区切りの数値形式に変更してください。
4. B:F列の列幅を自動調整してください。
5. 1行目の行の高さを「24」に変更してください。

動画レッスンB-2t

https://excel23.com/vba-juku/chap-special-b2test/

※模範解答は、PDFの巻末に掲載しています。

★★☆

レッスン3

行や列を挿入・削除する

教材ファイル「補講Bレッスン3.xlsm」を使用します。

動画レッスンB-3

https://excel23.com/vba-juku/chap-special-b3/

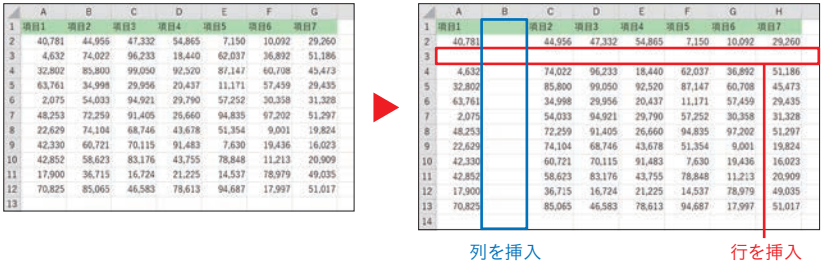
1. 行や列を挿入する（Insertメソッド）

Insertメソッドを使用すると、指定の場所に行や列を挿入できます。

書式

```
Rows(引数).Insert          '行を挿入する
Columns(引数).Insert       '列を挿入する
```

- 行や列が挿入されると、元にあった行や列は、**挿入された分だけ後ろへ（番号の大きい方へ）スライド移動**します。
- なお、複数行や複数列を範囲指定してInsertメソッドを使用した場合、**指定した行数または列数の分だけ**、空白の行や列が挿入されます。



[例1] 単一の行や列を挿入する（3行目に行を挿入、2列目に列を挿入）

例

```
Sub exampleB_3_1a()
    Rows(3).Insert          '行を挿入
    Columns(2).Insert       '列を挿入
End Sub
```

- 「Rows(3)」でシートの3行目を指定し、「.Insert」で行を挿入します。
- 「Columns(2)」で2列目を指定し、「.Insert」で列を挿入します。

[例2] 複数の行や列を挿入する（5～8行目に行を挿入、D～F列に列を挿入）

例

```
Sub exampleB_3_1b()
    Rows("5:8").Insert          '複数行を挿入
    Columns("D:F").Insert       '複数列を挿入
End Sub
```

- 「Rows("5:8")」でシートの5～8行目を範囲指定しています。
- 「Columns("D:F")」でシートのD～F列を範囲指定しています。
- 挿入された行や列の数だけ、もとにあった行や列は後ろへ（番号の大きい方へ）スライド移動します。

2. 行や列を削除する（Deleteメソッド）

Deleteメソッドを使用すると、行や列を削除できます。

書式

```
Rows(引数).Delete           '行を削除
Columns(引数).Delete        '列を削除
```

- 行や列を削除すると、なくなった空白を埋めるように、**以降の行や列が前方に（番号の小さい方へ）スライド移動**します。

- なお、複数行や複数列を範囲指定してDeleteメソッドを使用した場合、**なくなった行数または列数の分だけ、以降の行や列は、前方に（番号の小さい方へ）スライド移動**します。

[例1] 単一の行や列を削除する（3行目と2列目を削除する）

例

```
Sub exampleB_3_2a()
    Rows(3).Delete          '行を削除
    Columns(2).Delete       '列を削除
End Sub
```

- 「Rows(3)」でシートの3行目を指定し、「.Delete」で行を削除します。
- 「Columns(2)」で2列目を指定し、「.Delete」で列を削除します。

[例2] 複数の行や列を削除する（5～8行目とD～F列を削除する）

例

```
Sub exampleB_3_2b()
    Rows("5:8").Delete          '複数行を削除
    Columns("D:F").Delete       '複数列を削除
End Sub
```

- 「Rows("5:8")」でシートの5～8行目を範囲指定しています。
- 「Columns("D:F")」でシートのD～F列を範囲指定しています。
- 上記のコードを実行すると、削除された行数または列数の分だけ、以降の行や列は、前方に（番号の小さい方へ）スライド移動します。

確認問題

教材ファイル「補講Bレッスン3確認問題.xlsm」を使用します。

	A	B	C	D	E	F	G	H	I
1	項目1	項目2	項目3	項目4	項目5	項目6	項目7	項目8	項目9
2	34,176	12,227	25,941	91,376	11,942	1,487	85,549	50,417	15,546
3	52,199	28,136	7,828	43,392	90,774	81,875	92,541	58,930	1,460
4	17,544	57,303	56,497	46,044	96,552	50,620	53,609	21,323	57,801
5	55,191	99,681	22,052	4,785	49,590	60,175	92,521	97,953	69,809
6	18,009	75,855	85,800	80,128	24,223	92,104	77,084	78,073	64,690
7	10,240	18,551	74,478	21,566	32,743	54,942	76,422	38,130	89,344
8	27,151	51,274	14,192	80,763	90,609	27,331	56,419	20,496	40,346
9	6,532	59,581	85,198	39,621	92,602	44,572	35,966	45,473	20,092
10	7,487	27,599	47,537	22,869	39,764	61,367	79,492	51,586	30,345
11	75,080	2,159	89,725	52,827	83,708	4,922	6,474	78,316	24,713
12	59,303	45,664	7,572	96,160	48,417	73,068	28,984	82,038	77,633
13	11,997	31,676	77,968	31,916	76,885	48,425	6,386	27,053	87,952
14	47,620	39,945	11,901	79,611	13,497	52,402	15,841	96,899	6,218
15	17,164	82,107	58,365	74,274	79,269	12,945	83,056	57,988	18,486
16	85,552	29,131	51,448	67,282	90,721	5,828	59,095	32,900	23,733
17	28,220	60,868	57,114	36,295	37,046	20,475	70,699	99,096	79,029
18	98,978	6,841	54,998	73,216	5,301	15,760	37,254	21,438	89,222
19									

1. C列に列を挿入してください。
2. 5行目に行を挿入してください。
3. E～G列を削除してください。
4. 7～10行目を削除してください。

動画レッスンB-3t

https://excel23.com/vba-juku/chap-special-b3test/

※模範解答は、PDFの巻末に掲載しています。

補講C

もっと知っておきたいセル範囲の指定方法

実用マクロにおいては、セル範囲を指定して操作することが頻繁にあります。また様々なケースに対応できるように、セル範囲を指定する方法にも色々なパターンがあります。そこで本章では、様々なセル範囲の指定方法について解説を行います。

- **Rangeを使った応用的なセル範囲の指定方法**
- **表全体を一括で指定するCurrentRegionプロパティ**
- **端へ飛んだセルを指定するEndプロパティ**
- **離れた場所を指定するOffsetプロパティ**
- **セル範囲の大きさを変更するResizeプロパティ**

★☆☆

レッスン1
Rangeの応用的な使い方

教材ファイル「補講Cレッスン1.xlsm」を使用します。

動画レッスンC-1

https://excel23.com/vba-juku/chap-special-c1/

1. Range（始点セル，終点セル）で範囲を指定する

以下の書式で、Rangeを使って始点セル〜終点セルまでのセル範囲を指定できます。

書式

```
Range(始点セル,終点セル)
```

- 始点セルや終点セルは、RangeやCellsを使って記述できます。

具体的なコード例を紹介します。

[例1] セル範囲A1:C5を選択する（Selectメソッドを使用）

例

```
Sub exampleC_1_1a()
    Range(Range("A1"), Range("C5")).Select
End Sub
```

始点セル　終点セル

- 始点セルとしてRange("A1")、終点セルとしてRange("C5")を指定しています。
- 上記の結果、セル範囲A1:C5が指定されます。
- 「.Select」により、指定の範囲が選択されます。

[例2] セル範囲A1:C5を選択する（始点セルと終点セルには「Cells」を使用）

例

```
Sub exampleC_1_1b()
    Range(Cells(1, "A"), Cells(5, "C")).Select
End Sub
```

始点セル　終点セル

- 始点セルとしてCells(1, "A")、終点セルとしてCells(5, "C")を指定しています。
- 上記の結果、セル範囲A1:C5が指定されます。

上記の[例1],[例2]はどちらも同じセル範囲を選択するコードですが、引数にRangeを使用するか、Cellsを使用するかという点で違いがあります。

2. Range("行番号:行番号")で行を指定する

以下の書式で、Rangeを使って行を指定することができます。

書式

```
Range("行番号:行番号")
```

- 引数には、例えば"1:3"と記述すると1〜3行目を範囲指定できます。
- また、例えば"3:3"などと同じ数値を記述すると、3行目だけを単一で指定できます。

[例1]1〜5行目を選択する

例

```
Sub exampleC_1_2a()
    Range("1:5").Select
End Sub
```

- Rangeの引数に「"1:5"」と記述することで、1〜5行目を範囲指定しています。

[例2]3行目(単一の行)を選択する

例

```
Sub exampleC_1_2b()
    Range("3:3").Select
End Sub
```

- Rangeの引数に「"3:3"」と記述することで、3行目を単一で指定しています。

3. Range("列名:列名")で列を指定する

行と同様に、Rangeを使って列を指定することができます。

書式

```
Range("列名:列名")
```

- 引数には、例えば"A:C"と記述するとA～C列を指定できます。
- また、例えば"C:C"などと同じ列名を記述すると、C列だけを単一で指定できます。

[例1] A～E列を選択する

例

```
Sub exampleC_1_3a()
    Range("A:E").Select
End Sub
```

- Rangeの引数に「"A:E"」と記述することで、A～E列を範囲指定しています。

[例2] C列を選択する

例

```
Sub exampleC_1_3b()
    Range("C:C").Select
End Sub
```

- Rangeの引数に「"C:C"」と記述することで、C列を単一で指定しています。

補足

列番号によって列を範囲指定することはできる？ その方法とは

3.で紹介した書式では、列範囲を指定するために「Range("A:C")」のように**列名を使用する必要がありました。**しかし、例えば「1列目〜3列目まで」のように、**列番号で範囲指定したいケース**もあります。そのような場合は、以下の書式で指定できます。

書式

```
Range(Columns(列番号),Columns(列番号))
```

始点の列　終点の列

- Rangeの第1引数のColumns()の中に、**始点の列の列番号**を記述します。
- Rangeの第2引数のColumns()の中に、**終点の列の列番号**を記述します。

[例]1〜3列目の複数列を選択する（すなわちA〜C列）

例

```
Sub exampleC_1_3補足()
    Range(Columns(1),Columns(3)).Select
End Sub
```

始点の列　終点の列

- 第1引数の「Columns(1)」で始点の列、第2引数のColumns(3)で終点の列を指定しています。
- 上記の結果、1列目〜3列目（すなわちA〜C列）が範囲指定されます。

確認問題

教材ファイル「補講Cレッスン1確認問題.xlsm」を使用します。

	A	B	C	D
1	商品コード	商品名	商品分類	在庫数
2	a00001	Wordで役立つビジネス文書術	Word教材	75
3	a00002	Excelデータ分析初級編	Excel教材	56
4	a00003	パソコンをウィルスから守る術	パソコン教材	90
5	a00004	超速タイピング術	パソコン教材	80
6	a00005	PowerPointプレゼン術	PowerPoint教材	75
7	a00006	PowerPointアニメーションマスター	PowerPoint教材	38
8	a00007	はじめからWindows入門	パソコン教材	100
9	a00008	初心者脱出のExcel術	Excel教材	88

1. Range(始点セル,終点セル)を使って、セル範囲A1:D9を選択(Select)してください。
2. Rangeを使って、シートの3～5行目を選択(Select)してください。
3. Rangeを使って、シートのC～D列を選択(Select)してください。

動画レッスンC-1t

https://excel23.com/vba-juku/chap-special-c1test/

※模範解答は、PDFの巻末に掲載しています。

★★☆

レッスン2

セルを指定する様々な方法

教材ファイル「補講Cレッスン2.xlsm」を使用します。

動画レッスンC-2

https://excel23.com/vba-juku/chap-special-c2/

1. 表全体を一括で指定する(CurrentRegionプロパティ)

セル

	A	B	C	D	E	F
1	購入ID	日付	氏名	商品コード	商品名	商品分類
2	1	2021/1/1	相川 なつこ	a00001	Wordで役立つビジネス	Word教材
3	2	2021/1/9	長坂 美代子	a00002	Excelデータ分析初級	Excel教材
4	3	2021/1/17	長坂 美代子	a00003	パソコンをウィルスカ	パソコン教材
5	4	2021/1/25	篠原 哲雄	a00001	Wordで役立つビジネス	Word教材
6	5	2021/2/2	布 施寛	a00004	超速タイピング術	パソコン教材
7	6	2021/2/10	布 施寛	a00005	PowerPointプレゼン	PowerPoint教材
8	7	2021/2/18	水戸 陽太	a00006	PowerPointアニメー	PowerPoint教材
9	8	2021/2/26	水戸 陽太	a00007	はじめからWindowsパ	パソコン教材
10	9	2021/3/6	山田 大地	a00008	初心者脱出のExcel術	Excel教材
11	10	2021/3/14	山田 大地	a00003	パソコンをウィルスカ	パソコン教材
12						

CurrentRegion

CurrentRegionプロパティを使用すると、**あるセルを含んでいる表全体**を自動で指定することができます。

書式

```
Rangeオブジェクト.CurrentRegion
```

- Rangeオブジェクトは、表の中に含まれる1つ以上のセルをRangeやCellsで指定します。

[例] セルA1を含んでいる表全体を選択する

例

```
Sub exampleC_2_1()
    Range("A1").CurrentRegion.Select
End Sub
```

- マクロを実行すると、セルA1を含んでいる表全体が選択された状態になります。

CurrentRegionプロパティで表全体を指定することを具体的にイメージするために、シート上で次の操作をしてみましょう。

1. 表の中に含まれる1つのセルを選択する
2. ショートカットキーのCtrl+Shift+ : を押す
3. 表全体が選択される

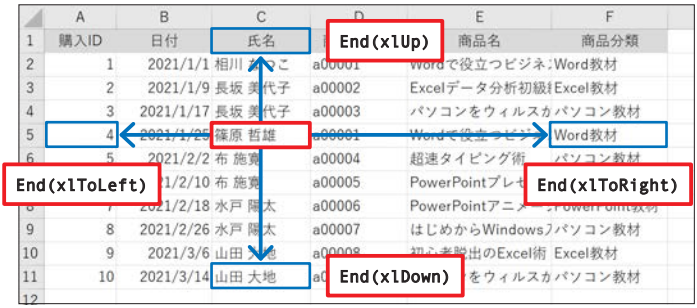
ショートカットキーの「Ctrl+Shift+ : 」は、セルが隣同士つながっているエリア全体を選択します。
それが、CurrentRegionで指定できるセル範囲のイメージです。

実務では、**表全体**を操作することがよくあります。
例えば、

- **表全体**をコピーして別の場所に転記する
- **表全体**を削除する
- **表全体**に罫線（格子）を適用する

このような場面では、CurrentRegionを使えば表全体を自動で指定できるため便利です。

2. 端へ飛んだセルを指定する（Endプロパティ）



Endプロパティを使用すると、あるセルから**データの端まで飛んだセル**を指定できます。
飛ぶ方向は、**上下左右の４つの方向**を引数によって指定できます。

書式

Rangeオブジェクト.End(方向)

[正式な引数名]

```
Rangeオブジェクト.End(Direction)
```

- Rangeオブジェクトは、RangeやCellsで指定できます。
- End(方向)の引数には、定数を使って上下左右の4つの方向を指定できます。

表C-2-1

定数	説明
xlDown	下方向
xlToLeft	左方向
xlToRight	右方向
xlUp	上方向

以下に具体的なコードを紹介します。

	A	B	C	D	E	F
1	購入ID	日付	氏名	商品コード	商品名	商品分類
2	1	2021/1/1	相川 なつこ	a00001	Wordで役立つビジネ	Word教材
3	2	2021/1/9	長坂 美代子	a00002	Excelデータ分析初級	Excel教材
4	3	2021/1/17	長坂 美代子	a00003	パソコンをウィルスカ	パソコン教材
5	4	2021/1/25	篠原 哲雄	a00001	Wordで役立つビジネ	Word教材
6	5	2021/2/2	布 施寛	a00004	超速タイピング術	パソコン教材
7	6	2021/2/10	布 施寛	a00005	PowerPointプレゼン	PowerPoint教材
8	7	2021/2/18	水戸 陽太	a00006	PowerPointアニメー	PowerPoint教材
9	8	2021/2/26	水戸 陽太	a00007	はじめからWindows	パソコン教材
10	9	2021/3/6	山田 大地	a00008	初心者脱出のExcel術	Excel教材
11	10	2021/3/14	山田 大地	a00003	パソコンをウィルスカ	パソコン教材
12						

[例1] セルD6から右端へ飛んだセルを選択する

例

```
Sub exampleC_2_2a()
    Range("D6").End(xlToRight).Select
                    右方向
End Sub
```

- Endプロパティの引数に「xlToRight」という定数を指定することで、右方向へ飛んだセルを指定します。
- 上記のコードの結果、D6から右方向に飛んだセルのF6が選択されます。

[例2] セルD6から下端へ飛んだセルを選択する

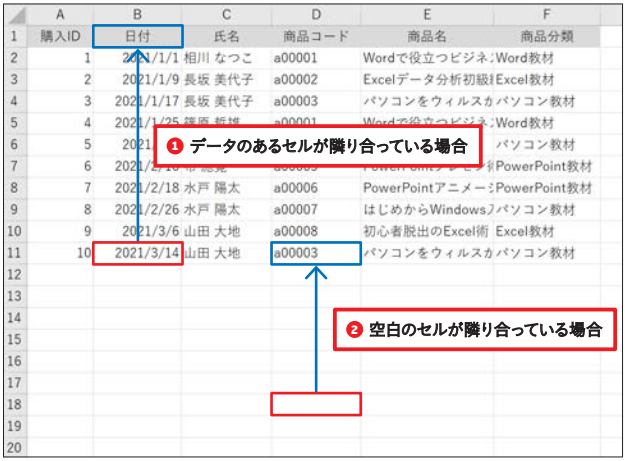
例

```
Sub exampleC_2_2b()
    Range("D6").End(xlDown).Select
End Sub
```

下方向

- Endプロパティの引数に「xlDown」という定数を指定することで、下方向へ飛んだセルを指定します。
- 上記のコードの結果、D6から下に飛んだセルのD11が選択されます。

Endプロパティは、❶データのあるセルが隣り合っている場合には、その端まで飛んだセルを指定します。❷データのないセル(空白のセル)が隣り合っている場合は、空白のセルをすべて通過していき、データのあるセルに出会ったらそのセルを指定します。**書籍第7章**で紹介した「最終行の取得」のコードでは、この性質を利用して最終行のセルを指定しています。



	A	B	C	D	E	F
1	購入ID	日付	氏名	商品コード	商品名	商品分類
2	1	2021/1/1	相川 なつこ	a00001	Wordで役立つビジネ	Word教材
3	2	2021/1/9	長坂 美代子	a00002	Excelデータ分析初級	Excel教材
4	3	2021/1/17	長坂 美代子	a00003	パソコンをウィルスか	パソコン教材
5	4	2021/1/25	篠原 哲雄	a00001	Wordで役立つビジネ	Word教材
6	5	2021				パソコン教材
7	6	2021/2/16	[illegible]	[illegible]	PowerPointプレゼン	PowerPoint教材
8	7	2021/2/18	水戸 陽太	a00006	PowerPointアニメー	PowerPoint教材
9	8	2021/2/26	水戸 陽太	a00007	はじめからWindows	パソコン教材
10	9	2021/3/6	山田 大地	a00008	初心者脱出のExcel術	Excel教材
11	10	2021/3/14	山田 大地	a00003	パソコンをウィルスか	パソコン教材
12						
13						
14						
15						
16						
17						
18						
19						
20						

補足

表の最終［列］を取得するコード

書籍第7章で紹介した「最終［行］の取得」と似ていますが、Endプロパティを応用することで、表の最終［**列**］を取得することができます。

End(xlToRight)

	A	B		E	F	G
1	購入ID	日付	氏名	商品コード　商品名	商品分類	
2	1	2021/1/1	相川 なつこ	a00001 Wordで役立つビジネ	Word教材	
3	2	2021/1/9	長坂 美代子	a00002 Excelデータ分析初級	Excel教材	
4	3	2021/1/17	長坂 美代子	a00003 パソコンをウィルスか	パソコン教材	
5	4	2021/1/25	篠原 哲雄	a00001 Wordで役立つビジネ	Word教材	
6	5	2021/2/2	布 施寛	a00004 超速タイピング術	パソコン教材	
7	6	2021/2/10	布 施寛	a00005 PowerPointプレゼン	PowerPoint教材	
8	7	2021/2/18	水戸 陽太	a00006 PowerPointアニメー	PowerPoint教材	
9	8	2021/2/26	水戸 陽太	a00007 はじめからWindows	パソコン教材	
10	9	2021/3/6	山田 大地	a00008 初心者脱出のExcel術	Excel教材	
11	10	2021/3/14	山田 大地	a00003 パソコンをウィルスか	パソコン教材	
12						

例

```
Dim maxCol As Long
maxCol = Range("A1").End(xlToRight).Column
```

セルA1　右方向へ飛ぶ　列番号

- まず、変数「maxCol」を宣言しています（名前の「Col」は、「列」という英単語の「Column」に由来します）。
- 次に、「Range("A1").End(xlToRight)」により、表のタイトル行にあるセルA1から、Endプロパティで右方向へ飛んだセルを指定します。これによって、表の一番右（最終列）のセルが指定されます。
- さらに、「.Column」で、指定したセルが属している列の列番号を取得します。今回のシートにおいては、F列（6番目の列）が最終列なので、「6」が取得されます。
- 以上により、変数「maxCol」に、最終列である「6」が代入されます。

なお、**書籍第7章**の「最終［**行**］の取得」の説明では、猿のイラストで説明したように、シートの末端から逆戻りして最終行を取得するコードを紹介しました。

最終［列］の取得においても、シートの末端から逆戻りして最終列を指定する方法もあります。その場合は、以下のように記述します。

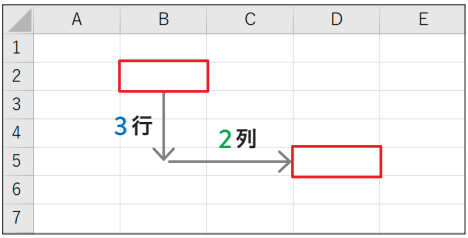
例

```
Dim maxCol As Long
maxCol = Cells(1,Columns.Count).End(xlToLeft).Column
```

シートの最終端の列　左方向へ飛ぶ　列番号

- Cellsの第2引数の「Columns.Count」は、シートの最大列の列数を返します。
- よって、「Cells(1,Columns.Count)」というコードで、シートの最終端の列のセルを指定しています。
- また、「End(xlToLeft)」というコードで、左方向へ飛んで辿り着いたセルを指定します。
- 最後に、「.Column」により、そのセルの列番号を取得します。

3. 離れた場所を指定する（Offsetプロパティ）




```
Range("B2").Offset(3, 2)
```

行数　列数

Offsetプロパティを使用すると、セルから指定の行数・列数だけ離れたセルを指定できます。

書式

```
Rangeオブジェクト.Offset(行数,列数)
```

[正式な引数名]

```
Rangeオブジェクト.Offset(RowOffset, ColumnOffset)
```

- 引数にはそれぞれ整数を指定できます（プラスの値、マイナスの値を指定できます）。
- 行数は、プラスの値で下方向を、マイナスの値で上方向を指定します。
- 列数は、プラスの値で右方向を、マイナスの値で左方向を指定します。
- Rangeオブジェクトは、単一のセルだけでなく、セル範囲も指定できます。セル範囲でOffsetプロパティを使用すると、セル範囲ごとスライド移動したセル範囲を指定できます。

以下に具体的なコードを紹介します。

	A	B	C	D	E	F
1	購入ID	日付	氏名	商品コード	商品名	商品分類
2	1	2021/1/1	相川 なつこ	a00001	Wordで役立つビジネ	Word教材
3	2	2021/1/9	長坂 美代子	a00002	Excelデータ分析初級	Excel教材
4	3	2021/1/17	長坂 美代子	a00003	パソコンをウィルスカ	パソコン教材
5	4	2021/1/25	篠原 哲雄	a00001	Wordで役立つビジネ	Word教材
6	5	2021/2/2	布 施寛	a00004	超速タイピング術	パソコン教材
7	6	2021/2/10	布 施寛	a00005	PowerPointプレゼン	PowerPoint教材
8	7	2021/2/18	水戸 陽太	a00006	PowerPointアニメー	PowerPoint教材
9	8	2021/2/26	水戸 陽太	a00007	はじめからWindows	パソコン教材
10	9	2021/3/6	山田 大地	a00008	初心者脱出のExcel術	Excel教材
11	10	2021/3/14	山田 大地	a00003	パソコンをウィルスカ	パソコン教材
12						

[例1] セルD5から3行2列離れたセルを選択する

例

```
Sub exampleC_2_3a()
    Range("D5").Offset(3, 2).Select
                       行  列
End Sub
```

- 「Range("D5")」でセルD5を指定し、「.Offset(3, 2)」で下に3行、右に2列離れたセルを指定します。
- 結果として、セルF8が選択されます。

[例2] セルD5から－3行－2列離れたセルを選択する

例

```
Sub exampleC_2_3b()
    Range("D5").Offset(-3, -2).Select
                       行   列
End Sub
```

- Offsetプロパティの引数にはマイナスの値を記述できます。
- 「Range("D5")」でセルD5を指定し、「.Offset(-3, -2)」で上に3行、左に2列離れたセルを指定します。
- 結果として、セルB2が選択されます。

[例3] セル範囲A1:F11から1行0列離れたセルを選択する

例

```
Sub exampleC_2_3c()
    Range("A1:F11").Offset(1,0).Select
                           行 列
End Sub
```

- セル範囲に対してもOffsetプロパティを使用できます。
- 「Range("A1:F11")」でセル範囲A1:F11を範囲指定し、「.Offset(1, 0)」で下に1行、右に0列離れたセル範囲を指定します。
- 結果、A2:F12が選択されます。

[例3]のようにOffsetプロパティを使用すると、**表のタイトル行だけを除いて、データ範囲だけを選択**することができます。
ただし、このコードのデメリットは、Offsetでセル範囲を下にずらした分、選択範囲の末尾に1行だけ空行が含まれます。実用面では、マクロの流れ上、末尾に1行の空行が含まれていても問題がないかどうか確認した上で使用してください。

補足

Offsetの引数が「0」のとき省略できる

Offsetプロパティの引数のいずれかが「0」の場合、その引数を省略できます。

- 第1引数を省略する場合は「,」とだけ入力する。
  例：Rangeオブジェクト.Offset(,列数)
- 第2引数を省略する場合は、何も入力せず省略する。
  例：Rangeオブジェクト.Offset(行数)

[例1] セルC3から1行0列離れたセルを選択する

例

```
Sub exampleC_2_3補足1()
    Range("C3").Offset(1).Select
                       行　（列は省略）
End Sub
```

- 省略せずに記述するならOffset(1,0)と記述しますが、第2引数の「0」を省略して、「Offset(1)」と記述しています。

[例2] セルC3から0行2列離れたセルを選択する

例

```
Sub exampleC_2_3補足2()
    Range("C3").Offset(, 2).Select
                （行は省略）列
End Sub
```

- 省略せずに記述するなら「Offset(0,2)と記述しますが、第1引数の「0」を省略して、「Offset(,2)」と記述しています。

4. セル範囲の大きさを変更する（Resizeプロパティ）

Resizeプロパティを利用すると、セル範囲の大きさを変更できます。

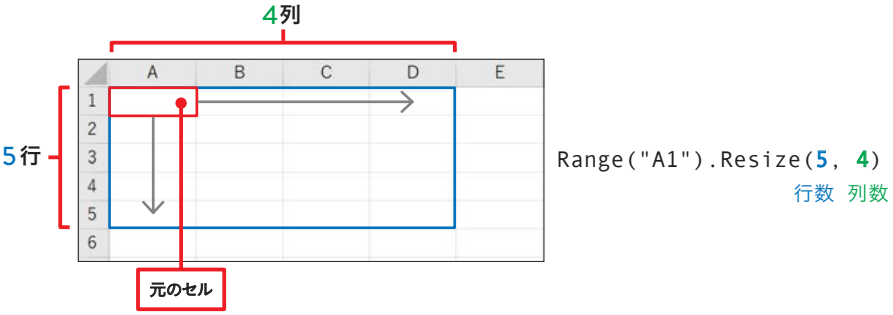
書式

```
Rangeオブジェクト.Resize(行数,列数)
```

［正式な引数名］

```
Rangeオブジェクト.Resize(RowSize, ColumnSize)
```

- 引数にそれぞれ整数を入力すると、その行数×列数の大きさのセル範囲に変更できます。
- 引数には、**マイナスの数や「0」は指定できません。**



［例］セルA1をもとに、5行4列分のセル範囲を選択する

例

```
Sub exampleC_2_4()
    Range("A1").Resize(5, 4).Select
                       行数 列数
End Sub
```

- セルA1をもとに、5行×4列のセル範囲が選択されます。
- 上記の結果、A1:D5が選択されます。

Resizeプロパティのイメージは、「ネットを広げる感じ」と考えるのがおすすめです。
上記の例では、元のセル範囲が「Range("A1")」でした。それを元に、Resizeでネットを広げるようにセル範囲を拡大し、5行×4列のセル範囲を指定しています。

Resizeプロパティは、Offsetプロパティと書式が似ていますが、似て非なるものだと考えましょう。

- Offsetプロパティは、元のセル範囲から○行○列だけ**移動したセル範囲**を指定する
- Resizeプロパティは、元のセル範囲を○行○列分の**大きさのセル範囲に変更**する

といった違いがあります。また、引数に使用できる数値の範囲も異なります。

- Offsetプロパティの引数には、プラスの数、マイナスの数、0を使用できます。
- Resizeプロパティの引数には、プラスの数のみ使用できます。マイナスの数や0は使用できません。

補足

Resizeで、行数または列数のどちらかだけを変更する

Resizeプロパティを利用して、行数または列数のどちらかだけ変更することもできます。

①元のセル範囲から、行数は変更せず列数だけを変更する場合

第1引数を省略して記述します。

書式

```
Rangeオブジェクト.Resize(,列数)
```

②列数は変更せずに行数だけを変更する場合

第2引数を省略して記述します。

書式

```
Rangeオブジェクト.Resize(行数)
```

[例1] セル範囲A1:D4（4行×4列の範囲）から、（行数は変更せず）列数だけを5列に変更して選択する

例

```
Sub exampleC_2_4補足1()
    Range("A1:D4").Resize(, 5).Select
End Sub
```

（行数は省略）　列数

- もとのセル範囲はA1:D4の4行×**4列**のセル範囲です。それを元にResizeプロパティによって4行×5列のセル範囲に変更しています。

[例2] セル範囲A1:D4（4行4列の範囲）から、（列数は変更せず）行数だけを5行に変更して選択する

例

```
Sub exampleC_2_4補足2()
    Range("A1:D4").Resize(5).Select
End Sub
```

行数　（列数は省略）

- もとのセル範囲はA1:D4の**4行**×4列のセル範囲です。それを元にResizeプロパティによって5行×4列のセル範囲に変更しています。

5. Offset、Resize、CurrentRegionを使った実用コード例

Offset、Resize、CurrentRegionなどを使った実用的なコード例を紹介します。

[例1] 表の中で、セルA3を含んでいる行だけを、薄い青（RGBで230,230,245）に塗りつぶす

	A	B	C	D	E	F
1	購入ID	日付	氏名	商品コード	商品名	商品分類
2	1	2021/1/1	相川 なつこ	a00001	Wordで役立つビジネ	Word教材
3	2	2021/1/9	長坂 美代子	a00002	Excelデータ分析初級	Excel教材
4	3	2021/1/17	長坂 美代子	a00003	パソコンをウィルスカ	パソコン教材
5	4	2021/1/25	篠原 哲雄	a00001	Wordで役立つビジネ	Word教材
6	5	2021/2/2	布 施寛	a00004	超速タイピング術	パソコン教材
7	6	2021/2/10	布 施寛	a00005	PowerPointプレゼン	PowerPoint教材
8	7	2021/2/18	水戸 陽太	a00006	PowerPointアニメー	PowerPoint教材
9	8	2021/2/26	水戸 陽太	a00007	はじめからWindows	パソコン教材
10	9	2021/3/6	山田 大地	a00008	初心者脱出のExcel術	Excel教材
11	10	2021/3/14	山田 大地	a00003	パソコンをウィルスカ	パソコン教材
12						

例

```
Sub exampleC_2_5a()
    Cells(3, "A").Resize(, 6).Interior.Color = _改行
        RGB(230, 230, 245)
End Sub
```

- 「Cells(3, "A")」によりセルA3を指定し、「.Resize(, 6)」で1列×6行のセル範囲に変更しています。
- また「.Interior.Color = RGB(230, 230, 245)」により、セル範囲の塗りつぶしの色を変更しています。
- 上記の結果、セルA3から1行×6列のセル範囲が塗りつぶされます。

表の列数が6であるため、Resizeプロパティの第2引数に「6」を指定しました。
しかし、実務では、最終列が「6」以外に変化することもあります。最終列が変化することが想定される場合は、「最終［列］の取得」のコード（このPDFの59ページ参照）を使って最終列を取得し、変数に代入して、「6」と置き換えるとよいでしょう。

[例2] 上記の例1と同様の処理を1行おきに実行し、表を交互に塗りつぶす

	A	B	C	D	E	F
1	購入ID	日付	氏名	商品コード	商品名	商品分類
2	1	2021/1/1	相川 なつこ	a00001	Wordで役立つビジネ	Word教材
3	2	2021/1/9	長坂 美代子	a00002	Excelデータ分析初級	Excel教材
4	3	2021/1/17	長坂 美代子	a00003	パソコンをウィルスカ	パソコン教材
5	4	2021/1/25	篠原 哲雄	a00001	Wordで役立つビジネ	Word教材
6	5	2021/2/2	布 施寛	a00004	超速タイピング術	パソコン教材
7	6	2021/2/10	布 施寛	a00005	PowerPointプレゼン	PowerPoint教材
8	7	2021/2/18	水戸 陽太	a00006	PowerPointアニメー	PowerPoint教材
9	8	2021/2/26	水戸 陽太	a00007	はじめからWindows	パソコン教材
10	9	2021/3/6	山田 大地	a00008	初心者脱出のExcel術	Excel教材
11	10	2021/3/14	山田 大地	a00003	パソコンをウィルスカ	パソコン教材
12						

例

```
Sub exampleC_2_5b()

    '最終行を取得する
    Dim maxRow As Long
    maxRow = Cells(Rows.Count,1).End(xlUp).Row

    '行を塗りつぶす
    Dim i As Long
    For i = 3 To maxRow Step 2
        Cells(i, "A").Resize(, 6).Interior.Color = _改行
            RGB(230, 230, 245)
    Next

End Sub
```

- 「`For i = 3 To maxRow Step 2`」と記述することで、変数iは、開始値「3」から終了値「maxRow」まで、2ずつ増加していきます。よって、変数iは「3、5、7、9…」と変化していきます。
- 上記の結果、**表を1行おきに交互に塗りつぶす**結果になります。

[例3] セルA1の属している表全体のうち、タイトル行だけを除いたデータ範囲を選択する

例

```
Sub exampleC_2_5c()
    Range("A1").CurrentRegion.Offset(1).Select
End Sub
```

- CurrentRegionプロパティで表全体を指定し、そのセル範囲を「Offset(1)」で1行分、下へスライド移動させています。
- よって、**表のタイトル行を除いたデータ範囲**を選択することができます。
- ただし、このコードのデメリットは、Offsetでセル範囲を下にずらした分、選択範囲の末尾に1行だけ空行が含まれます。実用面では、マクロの流れ上、末尾に1行の空行が含まれていても問題がないかどうか確認した上で使用してください。

[例4] 上記の例3と同様に、表のタイトル行だけを除いたデータ範囲をコピーして、セルH1に貼り付ける

例

```
Sub exampleC_2_5d()
    Range("A1").CurrentRegion.Offset(1).Copy Range("H1")
End Sub
```

- 上記は例3と同様に指定したセル範囲を、Copyメソッドでコピーし、セルH1に貼り付けています。

確認問題

教材ファイル「補講Cレッスン2確認問題.xlsm」を使用します。

	A	B	C	D	E
1	商品コード	商品名	商品分類	在庫数	
2	a00001	Wordで役立つビジネス文書術	Word教材	75	
3	a00002	Excelデータ分析初級編	Excel教材	56	
4	a00003	パソコンをウィルスから守る術	パソコン教材	90	
5	a00004	超速タイピング術	パソコン教材	80	
6	a00005	PowerPointプレゼン術	PowerPoint教材	75	
7	a00006	PowerPointアニメーションマスター	PowerPoint教材	38	
8	a00007	はじめからWindows入門	パソコン教材	100	
9	a00008	初心者脱出のExcel術	Excel教材	88	
10					

1. セルA1を含んでいる表全体を選択してください。
2. セルA1から右方向へデータの端まで飛んだセルを選択してください。
3. セルA1から下方向へ3つ、右方向へ3つ離れたセルを選択してください。

動画レッスンC-2t

https://excel23.com/vba-juku/chap-special-c2test/

※模範解答は、PDFの巻末に掲載しています。

補講D：上級編

本章は「上級編」として、より応用的な内容を解説していきます。紹介するのは以下となります。

- 第８章「条件分岐」応用：Like演算子の詳しい使い方
- 第８章「条件分岐」応用：論理演算子を使った判定方法
- 第９章「VBA関数」応用：MsgBox関数の応用、InputBox関数、Format関数、InStr関数
- 第10章「フィルターによるデータ抽出」応用：複数の条件による抽出、範囲で抽出

★★☆

レッスン１

条件分岐：Likeを使った文字列のパターン判定

教材ファイル「補講 D レッスン1.xlsm」を使用します。

動画レッスンD-1

https://excel23.com/vba-juku/chap-special-d1/

1. Like演算子を使って文字列のパターン判定をする

書籍第８章レッスン１で登場した「Like」について、より詳しい使い方を解説します。まず、Like演算子を使った条件式の書式は以下となります。

書式

左辺 Like 文字列パターン

文字列パターンには「ワイルドカード」という記号（「*」,「?」など）を使用して、様々なパターンを指定できます。

以下にワイルドカードの一覧表を掲載しますが、一覧だけではイメージが湧かないと思いますので、**2.** では具体的なコード例を紹介します。

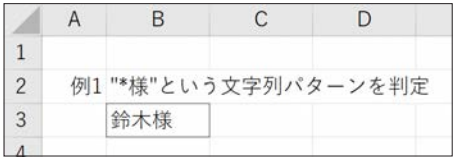
表 D-1-1

記号	意味	使用例	使用例の説明
*	任意の文字列 (文字数を問わない)	対象 Like "*様"	"様"で終わる文字列なら一致
?	任意の1文字の文字列	対象 Like "??様"	2文字の文字列+"様"なら一致
#	任意の1桁の数字	対象 Like "#番地"	1桁の数字+"番地"なら一致
[ ]	[ ]内のうち1文字	対象 Like "[ABC][123]"	例えば"A1"、"A2"、"A3"、"B1"、…などは一致
[!]	[ ]内の文字以外の1文字	対象 Like "[!ABC] "	例えば"D"、"E"、"F"などは一致
[-]	[ ]内の範囲の1文字	対象 Like "[A-Z][0-9]"	A～Z(A,B,C, …X,Y,Z)のうち1文字+0～9のうち1文字なら一致

2. Like演算子を使った具体的なコード例

それでは、Like演算子とワイルドカードを組み合わせた様々な使用方法を紹介します。

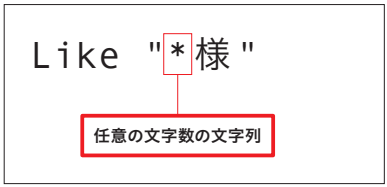
[例1] セルB3の値が「*様」という文字列パターンに一致するか判定する。一致するなら、メッセージボックスで"一致しました"と出力する。



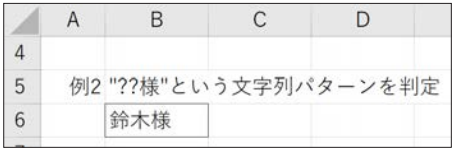
例

```
Sub exampleD_1_2a()
    If Range("B3").Value Like "*様" Then
        MsgBox "一致しました"
    End If
End Sub
```

- ワイルドカードの「*」は、任意の文字数の文字列を意味します。
- 文字列パターンに「Like "*様"」と記述することで、"様"の左に任意の文字数の文字列がある、つまり**"様"で終わっている文字列パターン**であることを指しています。



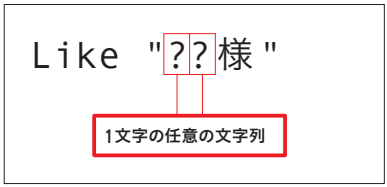
[例2] セルB6の値が「??様」という文字列パターンに一致するか判定する。
一致するなら、メッセージボックスで"一致しました"と出力する。



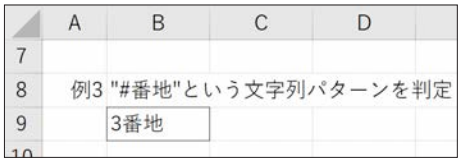
例

```
Sub exampleD_1_2b()
    If Range("B6").Value Like "??様" Then
        MsgBox "一致しました"
    End If
End Sub
```

- ワイルドカードの「?」は、1文字の任意の**文字列**を意味します。
- 文字列パターンに「Like "??様"」と記述することで、"様"の左に2文字の任意の文字列がある、つまり**"〇〇様"という文字列パターン**であることを指しています。



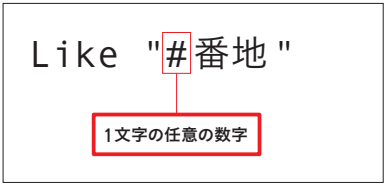
[例3] セルB9の値が「#番地」という文字列パターンに一致するか判定する。
一致するなら、メッセージボックスで"一致しました"と出力する。



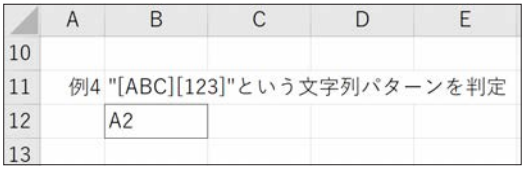
例

```
Sub exampleD_1_2c()
    If Range("B9").Value Like "#番地" Then
        MsgBox "一致しました"
    End If
End Sub
```

- ワイルドカードの「#」は、1文字の任意の**数字**を意味します。
- 文字列パターンに「Like "#番地"」と記述することで、"番地"の左に1文字の任意の**数字があるパターン**であることを指しています。



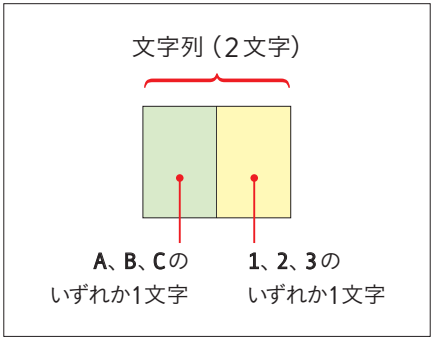
[例4] セルB12の値が「A、B、C」のいずれか1文字と、「1、2、3」のいずれか1文字が続いている文字列パターンに一致するか判定する。
一致するなら、メッセージボックスで"一致しました"と出力する。



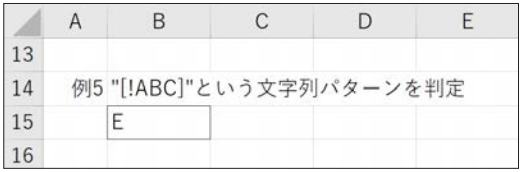
例

```
Sub exampleD_1_2d()
    If Range("B12").Value Like "[ABC][123]" Then
        MsgBox "一致しました"
    End If
End Sub
```

- ワイルドカードの［ ］は、その内の1文字が入ることを意味します。
- 文字列パターンに「Like "[ABC][123]"」と記述することで、「A、B、C」のいずれか1文字と、「1、2、3」のいずれか1文字が続いている**パターン**であることを指しています。



［例5］セルB15の文字列が、「A、B、C」のいずれでもない1文字であるか判定する。
一致するなら、メッセージボックスで"一致しました"と出力する。

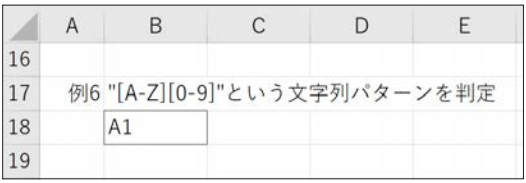


例

```
Sub exampleD_1_2e()
    If Range("B15").Value Like "[!ABC]" Then
        MsgBox "一致しました"
    End If
End Sub
```

- ワイルドカードの［！］は、その内のどの文字にも該当しない1文字を意味します。
- 文字列パターンに「`Like "[!ABC]"`」と記述することで、「**A、B、C」のいずれでもない1文字**であることを指しています。

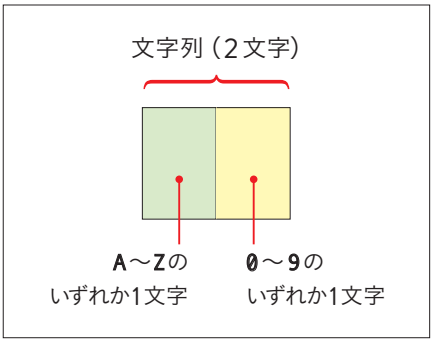
［例6］セルB18の文字列が、「A、B、C…Z」のいずれか1文字と、「0、1、2…9」のいずれか1文字が続いた文字列パターンであるか判定する。
一致するなら、メッセージボックスで"一致しました"と出力する。



例

```
Sub exampleD_1_2f()
    If Range("B18").Value Like "[A-Z][0-9]" Then
        MsgBox "一致しました"
    End If
End Sub
```

- ワイルドカードの［ - ］は、始点から終点の間にあるいずれか1文字を意味します。
- 文字列パターンに「`Like "[A-Z][0-9]"`」と記述することで、「A～Z」のいずれか1文字と、「0～9」のいずれか1文字が続いた文字列パターンであることを指しています。



確認問題

教材ファイル「補講Dレッスン1確認問題.xlsm」を使用します。

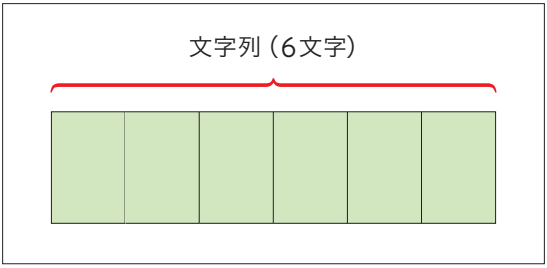
	A	B	C
1	商品コード	商品名	商品コードチェック
2	a00001	Excelデータ分析初級編	
3	a000002	Wordで役立つビジネス文書術	
4	a00003	パソコンをウィルスから守る術	
5	e00004	超速タイピング術	
6	a00o05	PowerPointプレゼン術	
7	a00006	PowerPointアニメーションマスター	
8	x00007	はじめからWindows入門	
9	a000080	初心者脱出のExcel術	
10	a00009	Excelこれから入門	
11	c00010	Wi-Fiでつながるインターネット超入門	
12	a00011	Excelピボットテーブル完全攻略	
13	a00012	Wordじっくり入門	
14			

動画レッスンD-1t

https://excel23.com/vba-juku/chap-special-d1test/

1. ［商品コード］列のセルの値が6文字の文字列が続いた文字列パターンと一致するか判定してください。

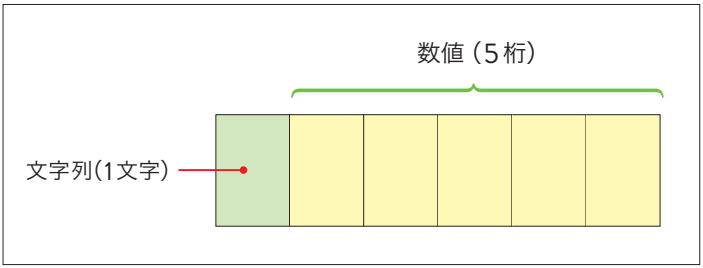
一致するなら［商品コードチェック］列のセルに"OK"という文字列を代入してください。
一致しないなら［商品コードチェック］列のセルに"NG"という文字列を代入してください。



2. ［商品コード］列のセルの値が1文字の文字列＋5桁の数値が続いた文字列パターンと一致するか判定してください。

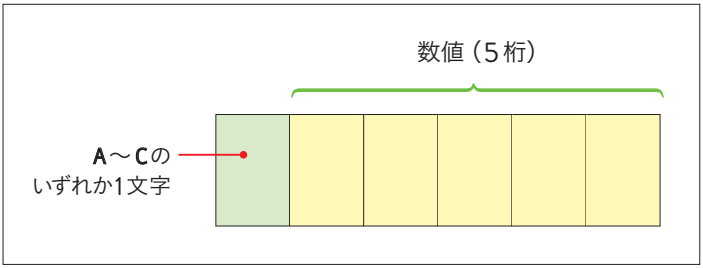
一致するなら［商品コードチェック］列のセルに"OK"という文字列を代入してください。

一致しないなら[商品コードチェック]列のセルに"NG"という文字列を代入してください。



3. [商品コード]列のセルの値が「a～c」のいずれか1文字+5桁の数値が続いた文字列パターンと一致するか判定してください。

一致するなら[商品コードチェック]列のセルに"OK"という文字列を代入してください。
一致しないなら[商品コードチェック]列のセルに"NG"という文字列を代入してください。



※模範解答は、PDFの巻末に掲載しています。

レッスン2

条件分岐：論理演算子（And、Or、Not）を使った判定

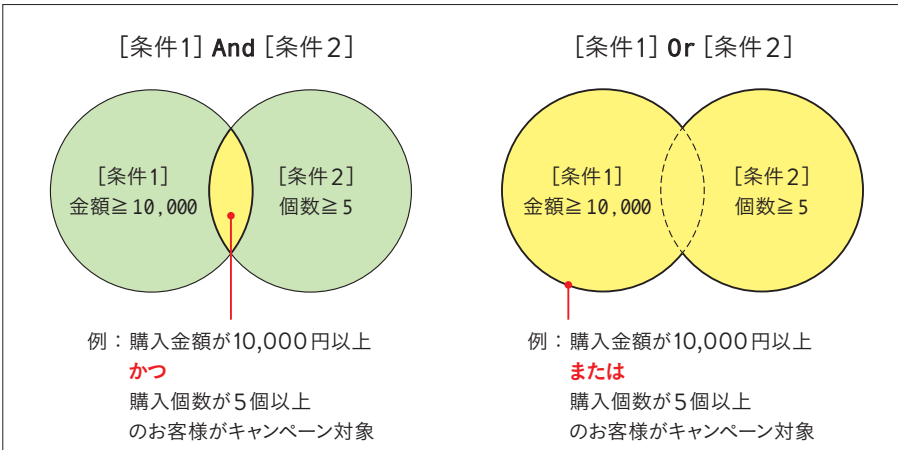
教材ファイル「補講Dレッスン2.xlsm」を使用します。

動画レッスンD-2

https://excel23.com/vba-juku/chap-special-d2/

1. And演算子（かつ）、Or演算子（または）で複数条件を指定する

2つ以上の条件を組み合わせて判定する場合、「And演算子」「Or演算子」を使う方法があります。



2つの条件を同時に満たす（And演算子）

例えば、「［条件式1］を満たす かつ［条件式2］を満たす」という複数条件で判定したい場合、2つの条件式の間にAnd演算子を記述します。

書式

```
If ［条件式1］ And ［条件式2］ Then
    処理内容
End If
```

- 上記のコードでは、条件式1と条件式2の両方に一致するならば、Trueの場合の処理が実行されます。

2つの条件のどちらかを満たす（Or演算子）

例えば、「［条件式1］または［条件式2］のどちらかを満たす」という複数条件で判定したい場合、2つの条件式の間に**Or**演算子を記述します。

書式

```
If ［条件式1］ Or ［条件式2］ Then
    処理内容
End If
```

- 上記のコードでは、条件式1と条件式2のいずれか片方に一致するならば、Trueの場合の処理が実行されます。

2. And、Orを使った具体的なコード例

具体的なコード例を用いて説明します。次の表では、以下の2つの条件によって、**その顧客が「特典」の対象であるか**どうかを判定します。

- ［金額］が10,000円以上である
- ［個数］が5個以上である

2つの条件をAnd演算子**（かつ）**またはOr演算子**（または）**で判定するコードについて、それぞれ見てみましょう。

	A	B	C	D	E	F
1	日付	氏名	金額	個数	特典(Andで判定)	特典(Orで判定)
2	2022/1/1	相川なつこ	11,880	4		
3	2022/1/2	長坂美代子	5,940	7		
4	2022/1/4	篠原哲雄	7,920	4		
5	2022/1/5	布施寛	13,860	6		
6	2022/1/7	水戸陽太	9,900	3		
7	2022/1/9	山田大地	11,880	3		
8	2022/1/13	小林アキラ	15,840	5		
9	2022/1/15	吉川真衣	11,880	7		
10	2022/1/17	白川美優	11,880	5		
11	2022/1/18	若山佐一	9,900	5		
12						

［例1］［金額］が10,000以上　かつ　［個数］が5以上なら、E列に「有」という文字列を代入する。そうでなければ「無」を代入する。

例

```
Sub exampleD_2_2a()

    Dim i As Long
    For i = 2 To 11
        If Cells(i, "C").Value >= 10000 And _ 改行
           Cells(i, "D").Value >= 5 Then
            Cells(i, "E").Value = "有"
        Else
            Cells(i, "E").Value = "無"
        End If
    Next i

End Sub
```

- 条件式1は「Cells(i, "C").Value >= 10000」
- 条件式2は「Cells(i, "D").Value >= 5」
- 上記の2つの間にAnd演算子を記述することで、2つの条件を同時に満たすかどうかを判定しています。

補足

1行のコードが長くなってきたら改行しよう

And演算子やOr演算子を使って複数の条件式を記述すると、1行のコードが長くなってきます。1行のコードが長いと可読性（読みやすさ）を損ないやすくなるので、コードを途中で改行しましょう。コードを途中で改行する方法は、**書籍第4章**のコラムで紹介していますので、参照してください。

一般的には、And演算子やOr演算子の後ろに「 _」を記述して改行し、次の行に2つ目の条件式を記述します。

[例2] [金額]が10,000以上　または　[個数]が5以上なら、F列に「有」という文字列を代入する。そうでなければ「無」を代入する。

例

```
Sub exampleD_2_2b()

    Dim i As Long
    For i = 2 To 11

        If Cells(i, "C").Value >= 10000 Or _改行
           Cells(i, "D").Value >= 5 Then
            Cells(i, "F").Value = "有"
        Else
            Cells(i, "F").Value = "無"
        End If
    Next i

End Sub
```

- 2つの条件式の間にOr演算子を記述することで、どちらかを満たしているかどうか判定しています。

補足

注意！ And演算子の使い方で間違えやすいケース

And演算子を使う際によくある間違った書き方を紹介します。
例えば、「セルA1の値が300より大きく、かつ500より小さい」という判定をするコードの例です。

[間違った書き方]

```
If Range("A1").Value >300 And <500 Then
```

[正しい書き方]

```
If Range("A1").Value >300 And Range("A1").Value <500 _
Then
```

「間違った書き方」では、「Range("A1").Value」を1つ目の条件式にしか記述していません。ついつい、このように書きたくなってしまうのですが、これは間違いです。「正しい書き方」のように、Range("A1").Valueを2つの条件式のどちらにも記述する必要があります。

3.「〜でない」のNot演算子

「Not演算子」を使用すると、「〜でない」という条件を判定できます。

書式

```
If Not 条件式 Then
    True(真)の場合の処理
End If
```

- 「Not」の後に条件式を記述することで、その条件を**満たさない場合はTrue、満たす場合はFalse**ということになり、通常とは逆の結果になります。
- 上記に加えて、「Else」や「ElseIf」を書き加えて処理を分岐することもできます。

以下に具体的なコード例を紹介します。

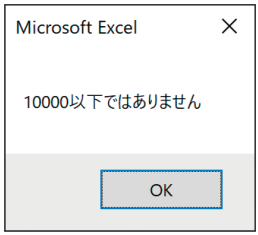
	A	B	C	D
1	日付	氏名	金額	個数
2	2022/1/1	相川なつこ	11,880	4
3	2022/1/2	長坂美代子	5,940	7
4	2022/1/4	篠原哲雄	7,920	4
5	2022/1/5	布施寛	13,860	6
6	2022/1/7	水戸陽太	9,900	3
7	2022/1/9	山田大地	11,880	3
8	2022/1/13	小林アキラ	15,840	5
9	2022/1/15	吉川真衣	11,880	7
10	2022/1/17	白川美優	11,880	5
11	2022/1/18	若山佐一	9,900	5

[例] セルC2の値が10,000以下でないことを判定し、Trueならばメッセージボックスで"10000以下ではありません"と出力する

例

```
Sub exampleD_2_3()

    If Not Range("C2").Value <= 10000 Then
        MsgBox "10000以下ではありません"
    End If

End Sub
```

- 条件式を「Not Range("C2").Value <= 10000」とすることで、セルC2の値が10000以下でないことを判定します。
- セルCの値は11880であり、10000以下ではありません。そのため条件式の結果はTrueとなり、メッセージボックスが出力されます。



確認問題

教材ファイル「補講Dレッスン2確認問題.xlsm」を使用します。

	A	B	C	D	E	F
1	商品型番	販売金額	販売数量	大人気	人気	不人気
2	D88-F-PPL	424,480	33			
3	T32-S-RED	279,000	56			
4	Y46-M-RED	352,800	65			
5	J01-T-PPL	375,000	33			
6	L01-T-RED	780,000	63			
7	D88-F-BLK	134,820	62			
8	Y46-M-GRN	138,000	55			
9	Q81-S-BLU	285,000	58			
10	J01-T-PPL	350,000	62			
11	J01-T-PPL	150,000	56			
12	Z20-M-BLU	58,800	64			

動画レッスンD-2t

https://excel23.com/vba-juku/chap-special-d2test/

1. 以下の2つの条件の両方を同時に満たしている場合、[大人気]列のセルに"Yes"という文字列を代入してください。そうでなければ"No"という文字列を代入してください。

   (条件1)[販売金額]列のセルの値が300000より大きい

   (条件2)[販売数量]列のセルの値が60より大きい

2. 1.と同じ2つの条件のどちらか片方でも満たしている場合、[人気]列のセルに"Yes"という文字列を代入してください。そうでなければ"No"という文字列を代入してください。

3. 1.と同じ2つの条件のいずれも満たさない場合、[不人気]列のセルに"Yes"という文字列を代入してください。そうでなければ"No"という文字列を代入してください。

   ヒント：Not [条件式1] And Not [条件式2] という条件式になります。

※模範解答は、PDFの巻末に掲載しています。

レッスン3

VBA関数：MsgBox関数の応用、InputBox関数、Format関数、InStr関数

教材ファイル「補講Dレッスン3.xlsm」を使用します。

動画レッスンD-3

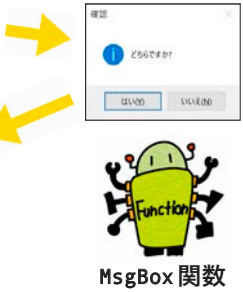
https://excel23.com/vba-juku/chap-special-d3/

1. MsgBox関数の応用。ボタンで条件分岐する

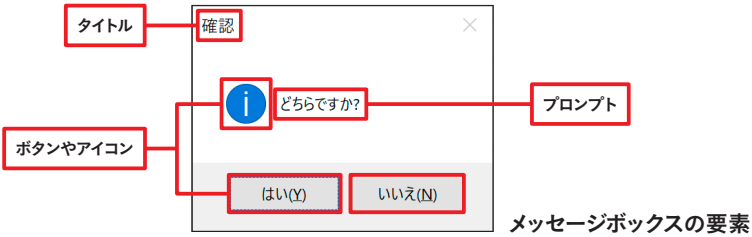
「MsgBox関数」について書籍ではメッセージを出力するだけの目的で使用してきました。しかし、応用すればそれ以上のことができます。例えばメッセージと同時に「はい」「いいえ」というボタンを表示させ、押されたボタンによって条件分岐して、

- 「はい」を押された場合の処理
- 「いいえ」を押された場合の処理

に分けるといったことも可能になります。



2. MsgBox関数の使い方



メッセージボックスの要素

MsgBox関数は以下のような書式で記述します。

書式

MsgBox(プロンプト,[ボタンやアイコン],[タイトル])

▶ [ ]は省略可能　※他にも引数がありますが使用頻度が低いので割愛します。

[正式な引数名]

```
MsgBox(Prompt,[Button,][Title])
```

- **第1引数**は、メッセージボックスに表示させる文字列を指定します（もしメッセージを出力するだけの目的なら第1引数だけを記述し、第2引数以降は省略できます）。
- **第2引数**は、ボタンやアイコンを数値や定数で指定できます。もしボタンとアイコンの両方を指定する場合は、数値や定数を「+」で加算して記述します。
- **第3引数**は、タイトルバーに表示する文字列を指定します。

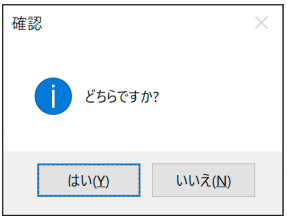
第2引数には、以下の定数を使用できます（黄色はボタンに関する定数、緑はアイコンに関する定数、青はボタンの既定を指定する定数です）。

表D-3-1

定数名	値	説明
vbOKOnly	0	[OK] ボタンのみを表示
vbOKCancel	1	[OK] ボタンと [キャンセル] ボタンを表示
vbAbortRetryIgnore	2	[中止]、[再試行]、[無視] の各ボタンを表示
vbYesNoCancel	3	[はい]、[いいえ]、[キャンセル] の各ボタンを表示
vbYesNo	4	[はい] ボタンと [いいえ] ボタンを表示
vbRetryCancel	5	[再試行] ボタンと [キャンセル] ボタンを表示
vbCritical	16	[重大なメッセージ] アイコンを表示
vbQuestion	32	[警告クエリ] アイコンを表示
vbExclamation	48	[警告メッセージ] アイコンを表示
vbInformation	64	[情報メッセージ] アイコンを表示
vbDefaultButton1	0	1番目のボタンが既定
vbDefaultButton2	256	2番目のボタンが既定
vbDefaultButton3	512	3番目のボタンが既定
vbDefaultButton4	768	4番目のボタンが既定

では、具体的なコード例を紹介します。

［例］以下のようなメッセージボックスを出力する



- プロンプト："どちらですか?"
- ボタン：「はい/いいえ」ボタン
- アイコン：「情報メッセージ」アイコン
- タイトル："確認"

例

```
MsgBox("どちらですか?", vbYesNo + vbInformation, "確認")
```

プロンプト　ボタンやアイコン　タイトル

- 第2引数は「vbYesNo + vbInformation」と記述しています。
- ボタンやアイコンを指定する定数を「+」で加算して記述すれば、ボタンとアイコンの両方を指定することができます。これは、いずれの定数も中身は数値が入っているためです。

また、MsgBox関数は、**どのボタンが押されたのか**を戻り値として整数で返します。
例えば、「はい」ボタンが押された場合は「6」が戻り値となり、「いいえ」が押された場合は「7」が戻り値となります。
しかし、「6」や「7」といった数値のままでは、条件式がわかりにくくなります。そこで、数値の代わりに定数を使用することができます。例えば、「6」の代わりに「vbYes」という定数名、「7」の代わりに「vbNo」という定数名で代用できます。
以下は、戻り値の判定のために使える定数です。

表D-3-2

定数	値	説明
vbOK	1	OK
vbCancel	2	キャンセル
vbAbort	3	中止
vbRetry	4	再試行

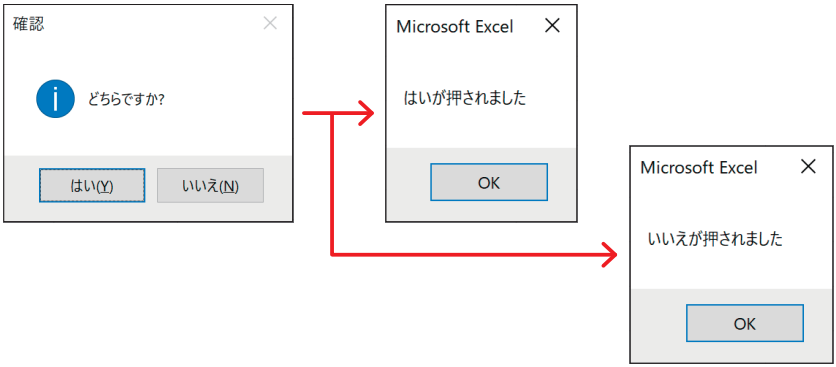
定数	値	説明
vbIgnore	5	無視
vbYes	6	はい
vbNo	7	いいえ

上記を利用して、**どのボタンが押されたのか、MsgBox関数の戻り値によって条件分岐する方法**を**3.**で紹介します。

3. MsgBox（応用）の実用ケース

どのボタンが押されたかによって条件分岐するケースを紹介します。

［例］以下のようなメッセージボックスを出力する



- 「はい」を押した場合：メッセージボックスで " はいが押されました " と出力
- 「いいえ」を押した場合：メッセージボックスで " いいえが押されました " と出力

例

```
Sub exampleD_3_3()

    Dim 回答 As Long
    回答 = MsgBox("どちらですか?", vbYesNo + vbInformation, "確認")

    If 回答 = vbYes Then
        MsgBox "はいが押されました"
    ElseIf 回答 = vbNo Then
        MsgBox "いいえが押されました"
    End If

End Sub
```

- 変数「回答」を宣言しています。変数の型を「Long」にしている理由は、MsgBox関数の戻り値は整数で返ってくるため、整数を代入できる「Long」が適切なためです。
- 次に、「回答 = MsgBox("どちらですか？", vbYesNo + vbInformation, "確認")」というコードでは、MsgBox関数の戻り値を、変数「回答」に代入します。
- If～ElseIf～End Ifの構文では、変数「回答」の値が、定数「vbYes」または「vbNo」に一致するかどうかを判定して条件分岐しています。
- 上記の結果がTrueの場合は「はい」ボタンが押されたと判定し、Falseなら「いいえ」ボタンが押されたことを判定しています。

補足

MsgBoxの引数は()がいる？ いらない？

よくある質問で、「MsgBoxの引数に()を書く場合と書かない場合の違いは何ですか？」という質問があります。結論からいうと、「もしメッセージを出力するだけの用途ならば()を書く必要はありません。ボタンによって条件分岐したいなら()を書く必要があります。」という回答になります。

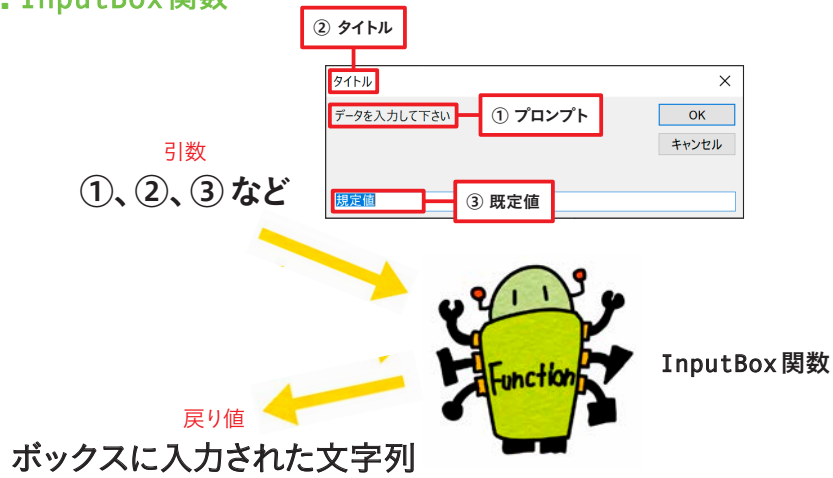
メッセージを出力するだけの用途ならば、「MsgBox "こんにちは"」のように()なしで記述します。これは、メッセージを出力するだけで処理が終わるため、戻り値がないからです。

一方、本章で紹介したように、ボタンによって条件分岐したいなら、()を書く必要があり、「MsgBox("どちらですか？", vbYesNo)」のように記述します。

書籍第9章レッスン1の**4.**で説明したように、戻り値があるVBA関数は、引数を()で囲うということがルールだからです。

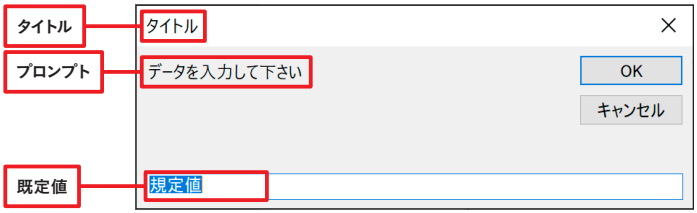
なお、メッセージを出力するだけの目的でも、MsgBox("こんにちは")のように()で囲って書いても結果として正常にマクロが動作します。しかし、厳密に正しい記述方法としては、この場合は()なしで書きます。

4. InputBox関数



InputBox関数は、ユーザーが入力できるダイアログボックスを出力できる関数です。MsgBox関数と似ていますが、InputBox関数は、ユーザーが文字列を入力でき、その入力された値をマクロの処理に利用できるという違いがあります。

5. InputBox関数の使い方



InputBox関数は、ダイアログボックスを出力し、ユーザーが文字列を入力して「OK」をクリックすると、その文字列を返します。なお「キャンセル」をクリックした場合は""（空白の文字列）を返します。

書式

InputBox(プロンプト,[タイトル],[既定値])

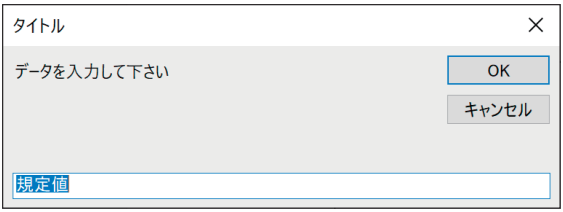
- [ ]は省略可能　※他にも引数がありますが使用頻度が低いので割愛します。

[正式な引数名]

```
InputBox(Prompt,[Title],[Default])
```

- **第1引数**では、ボックスに表示するメッセージを文字列で指定します（例えば"データを入力してください"のように記述）。
- **第2引数**では、ボックスのタイトルバーに表示する文字列を指定できます（省略すると空白になります）。
- **第3引数**では、ボックスの入力欄にデフォルトで表示させる文字列を指定できます（省略すると空白になります）。

[例] 下記のようなダイアログボックスを表示するコード例



例

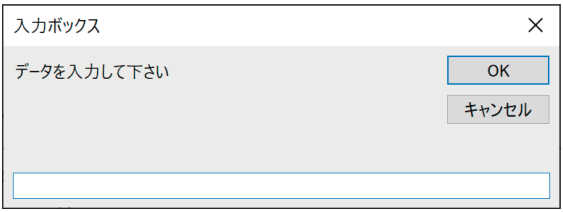
```
InputBox("データを入力して下さい", "タイトル", "規定値")
```

プロンプト　タイトル　既定値

6. InputBoxの実用ケース

次に、InputBox関数でユーザーに文字列を入力させ、その結果をセルに代入するケースを紹介します。

[例] 以下のようなInputBoxを出力し、入力された値をセルA1に代入する



- プロンプト："データを入力して下さい"
- タイトル　："入力ボックス"
- 既定値　　：省略

例

```
Sub exampleD_3_6()
    Dim 回答 As String
    回答 = InputBox("データを入力して下さい", "入力ボックス")
    Range("A1").Value = 回答
End Sub
```

- 変数「回答」(String型)を宣言し、その変数にInputBox関数の結果を格納しています。
- 最後に、セルA1の値に変数「回答」の値を代入しています。

7. Format関数

Format関数は、値の書式を変換して、文字列として返します。
例えば、「2021/5/1」という日付データがあるとします。これをFormat関数で「yyyy年m月d日」という書式(yyyyは西暦、mは月、dは日を意味します)に変換し、文字列として返すことができます。
Format関数については、**書籍第12章レッスン4**の補足で解説していますので、書式のみ再掲しておきます。

書式

```
Format(対象,[書式])
```

▸ [ ]は省略可能　※他にも引数がありますが使用頻度が低いので割愛します。

8. Format関数の使い方

具体的な記述方法を紹介します。

［例1］"2022/1/1"という日付を、"2022年01月01日"という書式の文字列に変換して返すコード例

例

```
Format("2022/1/1", "yyyy年mm月dd日")
```

対象　書式

- 書式記号のyyyyは西暦の年、mmは月、ddは日を意味します。
- 「mm」と記述した場合、月が1桁の場合、十の位に「0」を入れて「01」、「02」、「03」のような形式となります（「dd」も同様です）。

［例2］123という数値を、"000123"という書式の文字列に変換して返すコード例

例

```
Format(123, "000000")
```

対象　書式

- 書式記号「0」は1桁の数値を意味します。「000000」と記述した場合、6桁の数値の書式を意味しますが、各桁に数値がない場合は「0」が入ります。その結果、このコードでは「000123」という文字列が返ります。

［例3］98000という数値を、"98,000"（3桁おきにコンマ区切り）の書式の文字列に変換して返すコード例

例

```
Format(98000, "#,##0")
```

対象　書式

- 書式記号を#,##0とすることで、3桁おきにコンマ区切りの書式を指定できます。
- なお、書式記号を#,###としても3桁おきにコンマ区切りの書式に変換することはできます。 ただし、#,###とした場合は、もし変換前の値が「0」だった場合は、0は省略

されて空白の文字列が戻り値として返されます。

一方、#,##0とした場合は、0は省略されず、戻り値「0」が返されます。

9. Format関数の実用ケース

次に、Format関数の実用ケースを紹介します。右の表において、日付をもとに、「請求書番号」を作成して代入するマクロの例です。

[例] セルC8の日付を「yyyymmdd」形式に変換し、その後ろに「-」と［番号］列の値を付加する。その結果を［請求書番号］列に代入する（例:日付が「2021/5/3」で、番号が「1」の場合、請求書番号は「20210503-1」となる）

例

```
Sub exampleD_3_9()

    '変数に日付を代入
    Dim 日付 As Date
    日付 = Range("C8").Value

    '繰り返し
    Dim i As Long
    For i = 11 To 18
        Cells(i, "C").Value = _改行
          Format(日付, "yyyymmdd") & "-" & Cells(i, "B").Value
    Next i

End Sub
```

- 変数「日付」(Date型) を宣言し、セルC8の値を変数に代入しています。
- Format関数で、変数「日付」の日付データを”yyyymmdd”形式に変換しています。
- また、上記に"-"とCells(i, "B")の値をそれぞれ「&」で文字列として結合しています。

10. InStr関数

InStr関数は、文字列の中で、**特定の文字列が何文字目にあるか**を数値で返します。例えば、**" 鈴木 太郎 "**という文字列があるとします。姓と名の間には" "(半角スペース)が入っています。InStr関数を利用すると、「" "(半角スペース)は文字列の何文字目にあるか?」を検索し、「3」という数値を返します。

11. InStr関数の使い方

InStr関数は、文字列の中で特定の文字列が何文字目にあるかを整数で返します(もし存在しない場合は0を返します)。

書式

```
InStr(対象, 検索文字列)
```

▶ 他にも引数がありますが使用頻度が低いので割愛します。

[正式な引数名]

```
InStr(String1,String2)
```

［例］"鈴木 太郎"という文字列の中で、半角スペースが何文字目にあるかを返すコード例

例

```
InStr("鈴木 太郎", " ")
```

対象　検索文字列

12. InStr関数の実用ケース

次に、InStr関数の実用ケースを紹介します。

以下の表で、氏名の一覧から、「氏」と「名」に分けて転記するマクロを紹介します。

	A	B	C	D
25				
26		氏名	氏	名
27		相川 なつこ		
28		長坂 美代子		
29		篠原 哲雄		
30		布 施寛		
31		水戸 陽太		
32		山田 大地		
33		小林 アキラ		
34		吉川 真衣		
35		白川 美優		
36		若山 佐一		
37				

▶

	A	B	C	D
25				
26		氏名	氏	名
27		相川 なつこ	相川	なつこ
28		長坂 美代子	長坂	美代子
29		篠原 哲雄	篠原	哲雄
30		布 施寛	布	施寛
31		水戸 陽太	水戸	陽太
32		山田 大地	山田	大地
33		小林 アキラ	小林	アキラ
34		吉川 真衣	吉川	真衣
35		白川 美優	白川	美優
36		若山 佐一	若山	佐一
37				

氏名の間には半角スペースが挿入されています。InStr関数を使用して、「半角スペースは何文字目にあるか」を調べ、その前と後ろの文字列を**Left**,**Mid**関数で取得すれば、「氏」と「名」を転記することができます。

［例］B列の［氏名］において、半角スペースより前の文字列と後ろの文字列に分割し、それぞれC列とD列に転記する。
例えば、氏名が"相川 なつこ"というデータは、"相川"と"なつこ"に分割してそれぞれC列とD列に転記する。

例

```
Sub exampleD_3_12()

    '変数を宣言
```

```
    Dim 氏名 As String

     '繰り返し
     Dim i As Long
     For i = 27 To 36

         '変数に氏名を代入
         氏名 = Cells(i, "B").Value

         '氏と名を転記
         Cells(i, "C").Value = Left(氏名, InStr(氏名, " ") - 1)
         Cells(i, "D").Value = Mid(氏名, InStr(氏名, " ") + 1)

     Next i

 End Sub
```

- 変数「氏名」(String型) を宣言し、氏名を代入しています。
- 繰り返しのFor～Next構文では、「For i = 27 To 36」というコードにより、変数iは、開始値「27」から終了値「36」まで1ずつ増えながら繰り返します。
- 「InStr(氏名, " ")」というコードで、変数「氏名」の文字列において、**半角スペースが何文字目にあるか**を調べます。
- 「氏」を転記するコードの「Left(氏名, InStr(氏名, " ") - 1)」では、Left関数により、文字列の先頭から半角スペースの1文字手前までを抜き出しています。
- 「名」を転記するコードの「Mid(氏名, InStr(氏名, " ") + 1)」では、Mid関数により、半角スペースの1文字後ろから文字列の最後までを抜き出しています（※Mid関数では、第3引数を省略すると、文字列の末尾まですべて抜き出すことができます)。
- 上記で「氏」と「名」が抜き出され、それぞれ「Cells(i,"C")」、「Cells(i,"D")」の値に代入されます。

確認問題

教材ファイル「補講Dレッスン3確認問題.xlsm」を使用します。

	A	B	C	D	E	F	G	H	I
1	問題1					問題4			
2		単価	数量	金額			分類-コード	分類	コード
3		8,000	50				Word教材-a00001		
4							Excel教材-a00002		
5							パソコン教材-a00003		
6	問題2						パソコン教材-a00004		
7		氏名	氏名+敬称				PowerPoint教材-a00005		
8		鈴木太郎					PowerPoint教材-a00006		
9		佐藤次郎					パソコン教材-a00007		
10		加藤香					Excel教材-a00008		
11		渡辺祥子							
12									
13									
14	問題3								
15		番号	請求書番号						
16		2022/1/10							
17		2022/1/19							
18		2022/1/28							
19		2022/2/6							
20		2022/2/15							
21									

1. 以下のメッセージボックスを出力してください。

動画レッスンD-3t

https://excel23.com/vba-juku/chap-special-d3test/

- プロンプト：" 送料500円を加算しますか？ "
- ボタン：「はい」「いいえ」

また、メッセージボックスで押したボタンによって条件分岐をしてください。

- 「はい」を押した場合、セルB3の値×セルC3の値 +500 の結果をセルD3に代入
- 「いいえ」を押した場合、セルB3の値×セルC3の値 の結果をセルD3に代入

2. 以下のInputBoxを出力してください。

- プロンプト：" 敬称を入力してください。 "
- [氏名] 列のセルの末尾に、上記InputBoxで入力された文字列を付加して [氏名+敬称] 列のセルに代入してください（例えば、"鈴木太郎"に"様"を付加して、"鈴木太郎様"という文字列を代入します）。

3. [番号] 列のセルの日付を、"yyyy_mm_dd" 形式の文字列に変換して、[請求書番号] 列のセルに代入してください（例えば、日付が2022/1/10なら"2022_01_10"に変換して代入します）。

4. **[分類 - コード] 列のセルの文字列について以下の処理を行ってください。**

"-" より前の文字列を取得して、[分類] 列のセルに代入する
"-" より後ろの文字列を取得して、[コード] 列のセルに代入する（例えば、"Word 教材 - a00001" という文字列ならば "Word 教材" と "a00001" に分割します）

※模範解答は、PDFの巻末に掲載しています。

コラム

ダミーデータを作って小さくテストする方法

例えば元データが1万件ある場合、マクロで全件テストすると時間がかかってしまうことがあります。そんな場合は、まずは100件のダミーデータなどを自作して、小さくテストしてみるのも手です。テクニックをいくつか紹介します。

1. RANDBETWEEN関数（ワークシート関数）を使う

最小値から最大値までの整数の乱数（ランダムな値）を返します。

書式

```
=RANDBETWEEN(最小値,最大値)
```

- Excel2010以降で有効な関数です。
- RANDBETWEEN関数は、再計算のたびに異なる値を返します。再計算させないためには、セル範囲をコピーして値を貼り付け直すとよいでしょう。

2. 疑似個人情報（ダミーデータ）出力サービスを利用する

以下のようなサイトでは、疑似個人情報を出力してくれます。

- **疑似個人情報データ生成サービス**
  https://hogehoge.tk/personal/
- **なんちゃって個人情報**
  http://kazina.com/dummy/

レッスン4

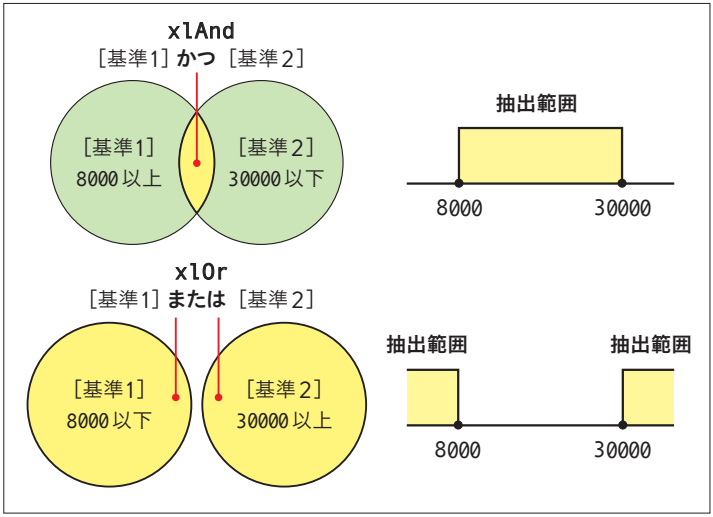
フィルターによるデータ抽出：複数の条件による抽出、範囲で抽出

教材ファイル「補講Dレッスン4.xlsm」を使用します。

動画レッスンD-4

https://excel23.com/vba-juku/chap-special-d4/

1. 複数の基準でデータを抽出する（かつ・または）



AutoFilterメソッドを使用して、1つのフィールド（列）に対して**2つの基準**でデータ抽出をすることができます。

例えば、以下のような条件で抽出できます。

- 金額が8000以上**かつ**30000以下のデータを抽出する（2つの基準を同時に満たす）
- 金額が8000以下**または**30000以上のデータを抽出する（少なくとも基準を1つは満たす）

引数は、「基準1」「基準2」という2つの基準を記述し、「オペレーター」という引数で、「かつ」「または」という条件を指定することができます。

書式

```
Rangeオブジェクト.AutoFilter フィールド, 基準1, オペレーター, 基準2
```

[正式な引数名]

```
Rangeオブジェクト.AutoFilter Field, Criteria1, Operator, Criteria2
```

- 基準1、基準2に、それぞれ条件式を記述します。
- 第3引数（オペレーター）には、以下の2種類の定数を入力して「**かつ**」「**または**」を選択できます。

定数「**xlAnd**」を記述 ➡「かつ」という条件
定数「**xlOr**」を記述 ➡「または」という条件

以下に、具体的なコード例を紹介します。

[例1] [金額] が8000以上かつ30000以下のデータを抽出する

例

```
Sub exampleD_4_1a()
    Range("A1").AutoFilter                    '前回のフィルターを解除
    Range("A1").AutoFilter 9, ">=8000", xlAnd, "<=30000"
                           フィールド 基準1 オペレーター 基準2
End Sub
```

- AutoFilterメソッドの第3引数に「**xlAnd**」と記述することで「**かつ**」という意味になり、基準1と基準2**の両方を同時に満たす**データを抽出することになります。
- 上記の結果、金額が8000以上30000以下の範囲内のデータが抽出されます。

	A	B	C	D	E	F	G	H	I
1	購入ID	日付	氏名	商品コード	商品名	商品分類	単価	数量	金額
2	1	2022/1/1	相川 なつこ	a00001	Wordで役立つビジネス文書術	Word教材	3,980	3	11,940
4	3	2022/1/17	長坂 美代子	a00003	パソコンをウィルスから守る術	パソコン教材	3,980	4	15,920
5	4	2022/1/25	篠原 哲雄	a00001	Wordで役立つビジネス文書術	Word教材	3,980	5	19,900
7	6	2022/2/10	布 施寛	a00005	PowerPointプレゼン術	PowerPoint教材	4,200	2	8,400
8	7	2022/2/18	水戸 陽太	a00006	PowerPointアニメーションマスター	PowerPoint教材	2,480	9	22,320
10	9	2022/3/6	山田 大地	a00008	初心者脱出のExcel術	Excel教材	2,480	7	17,360
14	13	2022/4/7	小林 アキラ	a00009	Excelこれから入門	Excel教材	2,480	4	9,920
17	16	2022/5/1	吉川 真衣	a00004	超速タイピング術	パソコン教材	4,500	4	18,000
18	17	2022/5/9	白川 美優	a00011	Excelピボットテーブル完全攻略	Excel教材	3,600	7	25,200
19	18	2022/5/17	若山 佐一	a00012	Wordじっくり入門	Word教材	2,480	8	19,840
21	20	2022/6/2	小林 アキラ	a00009	Excelこれから入門	Excel教材	2,480	7	17,360
23	22	2022/6/18	吉川 真衣	a00005	PowerPointプレゼン術	PowerPoint教材	4,200	3	12,600

[例2] [商品名] 列が「Word」という文字列を含む、または「パソコン」という文字列を含むデータを抽出する

例

```
Sub exampleD_4_1b()
    Range("A1").AutoFilter                          '前回のフィルターを解除
    Range("A1").AutoFilter 5, "*Word*", xlOr, "*パソコン*"
                           フィールド  基準1   オペレーター   基準2
End Sub
```

- AutoFilterメソッドの第3引数に「**xlOr**」と記述することで「**または**」という意味になり、少なくとも基準を1つは満たすデータを抽出することになります。
- 上記の結果、商品名に「Word」または「パソコン」を含むデータが抽出されます。

	A	B	C	D	E	F	G	H	I
1	購入ID	日付	氏名	商品コード	商品名	商品分類	単価	数量	金額
2	1	2022/1/1	相川 なつこ	a00001	Wordで役立つビジネス文書術	Word教材	3,980	3	11,940
4	3	2022/1/17	長坂 美代子	a00003	パソコンをウィルスから守る術	パソコン教材	3,980	4	15,920
5	4	2022/1/25	篠原 哲雄	a00001	Wordで役立つビジネス文書術	Word教材	3,980	5	19,900
11	10	2022/3/14	山田 大地	a00003	パソコンをウィルスから守る術	パソコン教材	3,980	10	39,800
13	12	2022/3/30	山田 大地	a00001	Wordで役立つビジネス文書術	Word教材	3,980	2	7,960
19	18	2022/5/17	若山 佐一	a00012	Wordじっくり入門	Word教材	2,480	8	19,840
20	19	2022/5/25	山田 大地	a00001	Wordで役立つビジネス文書術	Word教材	3,980	8	31,840

補講D 上級編

補足

「数量」と「金額」など、異なる複数のフィールドで抽出したい場合は?

1. で紹介した方法では、**同じフィールド（列）**において2つの基準でデータ抽出しました。

例えば、

［金額］列が8000以上かつ30000以下のデータを抽出

といった条件です。上記は、**［金額］という1つのフィールド**が抽出対象になっています。

それとは別に、**異なる複数のフィールド（列）**でデータ抽出する方法もあります。例えば、

［数量］が5以下　かつ　［金額］が15000以下のデータを抽出

といった条件です。上記は、［数量］［金額］という2つのフィールドが抽出対象になっています。
このような場合にはどのようなコードを記述すればよいか、例を紹介します。

［例］［数量］が5以下 かつ ［金額］が15000以下のデータを抽出する

例

```
Sub exampleD_4_1補足()
    Range("A1").AutoFilter                  '前回のフィルターを解除
    Range("A1").AutoFilter 8, "<=5"         '数量で抽出
    Range("A1").AutoFilter 9, "<=15000"     '金額で抽出
End Sub
```

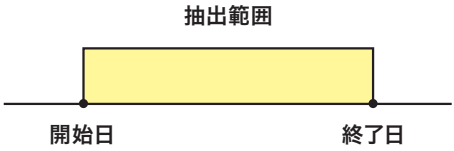
・2行目の「Range("A1").AutoFilter 8, "<=5"」により、［数量］列をもとにデータ抽出しています。

・3行目の「Range("A1").AutoFilter 9, "<=15000"」により、［金額］列をもとにデータ抽出しています。

以上のように、2つ以上のフィールドをもとにデータ抽出する場合は、**フィールドの数だけAutoFilterメソッドを記述すればよい**ということです。

2. 日付の範囲でデータを抽出する

1.で解説した方法を応用して、**日付の範囲**（開始日〜終了日）のデータを抽出する方法を紹介します。



書式

```
'開始日より後、終了日より前を抽出する
Rangeオブジェクト.AutoFilter フィールド, ">開始日", xlAnd, "<終了日"

'開始日以降、終了日以前を抽出する
Rangeオブジェクト.AutoFilter フィールド, ">=開始日", xlAnd, "<=終了日"
```

- 開始日、終了日はそれぞれyyyy/m/dの書式で記述します。
- ">開始日"で開始日**より後**、"<終了日"で終了日**より前**という意味になります。
- ">=開始日"で開始日**以降**、"<=終了日"で終了日**以前**という意味になります。

以下に具体的なコード例を紹介します。

[例] [日付]が2022/3/14以降、2022/5/9以前のデータを抽出する

例

```
Sub exampleD_4_2()
    Range("A1").AutoFilter                    '前回のフィルターを解除
    Range("A1").AutoFilter 2, ">=2022/3/14", xlAnd, "<=2022/5/9"
                           フィールド 開始日以降     かつ     終了日以前
End Sub
```

- 基準1を">=2022/3/14"と記述することで、2022/3/14**以降**という意味になります。
- 基準2を"<=2022/5/9"と記述することで、2022/5/9**以前**という意味になります。
- 上記の結果、[日付]が2022/3/14**以降**、2022/5/9**以前**のデータを抽出できます。

補足

3つ以上の基準で抽出したい場合は?

残念ながら、AuoFilterメソッドの引数には基準1と基準2しかないため、そのままでは3つ以上の基準で抽出することはできません。

対策方法として、以下の2つを紹介します。

①Array関数を使う方法

②作業列を使う方法

①Array関数を使う方法

以下のように記述すると、3つ以上の基準で抽出できます。

[例][単価]の列が3980、4500、1980のいずれかに一致するデータを抽出

書式

```
Sub exampleD_4_2補足a()
    Range("A1").AutoFilter          '前回のフィルターを解除
    Range("A1").AutoFilter _改行
     7, Array("3,980", "4,500", "1,980"), xlFilterValues
End Sub
```

- 第2引数には「Array("3,980", "4,500", "1,980")」と記述しています。これは「Array関数」という関数を用いることで、複数の値を「配列」というひとまとめのデータとして指定しています（配列という概念については、本書では踏み込んだ解説は割愛いたします）。
- 上記のように記述する場合、第3引数（オペレーター）には「**xlFilterValues**」という定数を指定します。
- この記述方法は、複数の数値に一致するデータを指定する方法なので、セルの表示形式通りに引数を記述する必要があります。例えば、元の表で「3,980」「4,500」「1,980」とカンマ区切りの書式で表示されている場合は、コードの引数も"3,980","4,500","1,980"とカンマ区切りで記述します。

上記のコードの記述方法では、複数の数値のいずれか1つ以上に一致するデータを抽出することができます（ワイルドカード「*」「?」も使用できます）。
しかし、**「xより大きい/小さい」「y以上/以下」のような条件式を用いることはできません。**また、**「条件1、条件2、条件3を同時に満たす」といったAnd条件での抽出にも対応していません。**そこで、次の②の方法も紹介します。

②作業列を使う方法

以下の図のように、「作業列」と呼ばれる一時的な列を作り、条件に一致するデータは作業列に"OK"などの値を代入して、目印を付けておきます。その目印をもとにフィルターで抽出するという方法です。

	A	B	C	D	E	F	G	H	I	J
1	購入ID	日付	氏名	商品コード	商品名	商品分類	単価	数量	金額	作業列
2	1	2022/1/1	相川 なつこ	a00001	Wordで役立つビジネス文書術	Word教材	3,980	3	11,940	
3	2	2022/1/9	長坂 美代子	a00002	Excelデータ分析初級編	Excel教材	4,500	10	45,000	OK
4	3	2022/1/17	長坂 美代子	a00003	パソコンをウィルスから守る術	パソコン教材	3,980	4	15,920	
5	4	2022/1/25	篠原 哲雄	a00001	Wordで役立つビジネス文書術	Word教材	3,980	5	19,900	
6	5	2022/2/2	布 施寛	a00004	超速タイピング術	パソコン教材	4,500	10	45,000	OK
7	6	2022/2/10	布 施寛	a00005	PowerPointプレゼン術	PowerPoint教材	4,200	2	8,400	OK
8	7	2022/2/18	水戸 陽太	a00006	PowerPointアニメーションマスター	PowerPoint教材	2,480	9	22,320	
9	8	2022/2/26	水戸 陽太	a00007	はじめからWindows入門	パソコン教材	1,980	3	5,940	
10	9	2022/3/6	山田 大地	a00008	初心者脱出のExcel術	Excel教材	2,480	7	17,360	OK
11	10	2022/3/14	山田 大地	a00003	パソコンをウィルスから守る術	パソコン教材	3,980	10	39,800	OK
12	11	2022/3/22	山田 大地	a00002	Excelデータ分析初級編	Excel教材	4,500	9	40,500	OK
13	12	2022/3/30	山田 大地	a00001	Wordで役立つビジネス文書術	Word教材	3,980	2	7,960	
14	13	2022/4/7	小林 アキラ	a00009	Excelこれから入門	Excel教材	2,480	4	9,920	
15	14	2022/4/15	小林 アキラ	a00010	Wi-Fiでつながるインターネット超入門	パソコン教材	1,980	1	1,980	OK
16	15	2022/4/23	吉川 真衣	a00005	PowerPointプレゼン術	PowerPoint教材	4,200	8	33,600	OK
17	16	2022/5/1	吉川 真衣	a00004	超速タイピング術	パソコン教材	4,500	4	18,000	
18	17	2022/5/9	白川 美優	a00011	Excelピボットテーブル完全攻略	Excel教材	3,600	7	25,200	OK
19	18	2022/5/17	若山 佐一	a00012	Wordじっくり入門	Word教材	2,480	8	19,840	
20	19	2022/5/25	山田 大地	a00001	Wordで役立つビジネス文書術	Word教材	3,980	8	31,840	OK
21	20	2022/6/2	小林 アキラ	a00009	Excelこれから入門	Excel教材	2,480	7	17,360	OK
22	21	2022/6/10	小林 アキラ	a00010	Wi-Fiでつながるインターネット超入門	パソコン教材	1,980	4	7,920	
23	22	2022/6/18	吉川 真衣	a00005	PowerPointプレゼン術	PowerPoint教材	4,200	3	12,600	OK

このように、データに目印をつけるために付加するデータは「フラグ」と呼ばれています。

［例］次の3つの条件に一致するデータを抽出する。
［金額］が30000以上 または［金額］が5000以下 または［金額］が7の倍数

書式

```
Sub exampleD_4_2補足b()
```

```
    Range("A1").AutoFilter              '前回のフィルターを解除

    '作業列を作成する
    Cells(1, "J").Value = "作業列"

    Dim 金額 As Long
    Dim i As Long
    For i = 2 To 23

        '金額を変数に取得する
        金額 = Cells(i, "I").Value

        '3つの条件のいずれかに一致する場合、作業列に"OK"を代入する
        If 金額 >= 30000 Or 金額 <= 5000 Or _ 改行
            金額 Mod 7 = 0 Then
            Cells(i, "J").Value = "OK"
        End If

    Next i

    'フィルター抽出
    Range("A1").AutoFilter 10, "OK"

End Sub
```

- 「Cells(1, "J").Value = "作業列"」というコードで、10列目（J列）に**作業列**を作成しています。
- 「If 金額 >= 30000 Or 金額 <= 5000 Or 金額 Mod 7 = 0 Then」という条件分岐により、3つの条件式のいずれかに該当する場合は、作業列に"OK"という文字列を代入します。
- 最後の「Range("A1").AutoFilter 10, "OK"」というコードで、［作業列］が"OK"に一致するデータを抽出します。

- 上記の結果、［金額］が30000以上 または ［金額］が5000以下 または［金額］が7の倍数のデータを抽出することができます。

	A	B	C	D	E	F	G	H	I	J
1	購入ID	日付	氏名	商品コード	商品名	商品分類	単価	数量	金額	作業列
3	2	2022/1/9	長坂 美代子	a00002	Excelデータ分析初級編	Excel教材	4,500	10	45,000	OK
6	5	2022/2/2	布 施寛	a00004	超速タイピング術	パソコン教材	4,500	10	45,000	OK
7	6	2022/2/10	布 施寛	a00005	PowerPointプレゼン術	PowerPoint教材	4,200	2	8,400	OK
10	9	2022/3/6	山田 大地	a00008	初心者脱出のExcel術	Excel教材	2,480	7	17,360	OK
11	10	2022/3/14	山田 大地	a00003	パソコンをウィルスから守る術	パソコン教材	3,980	10	39,800	OK
12	11	2022/3/22	山田 大地	a00002	Excelデータ分析初級編	Excel教材	4,500	9	40,500	OK
15	14	2022/4/15	小林 アキラ	a00010	Wi-Fiでつながるインターネット超入門	パソコン教材	1,980	1	1,980	OK
16	15	2022/4/23	吉川 真衣	a00005	PowerPointプレゼン術	PowerPoint教材	4,200	8	33,600	OK
18	17	2022/5/9	白川 美優	a00011	Excelピボットテーブル完全攻略	Excel教材	3,600	7	25,200	OK
20	19	2022/5/25	山田 大地	a00001	Wordで役立つビジネス文書術	Word教材	3,980	8	31,840	OK
21	20	2022/6/2	小林 アキラ	a00009	Excelこれから入門	Excel教材	2,480	7	17,360	OK
23	22	2022/6/18	吉川 真衣	a00005	PowerPointプレゼン術	PowerPoint教材	4,200	3	12,600	OK
24										

以上の方法は、表に作業列を設けなければいけないというデメリットはありますが、3つ以上の基準で柔軟にデータ抽出できるというメリットもあります。

確認問題

教材ファイル「補講Dレッスン4確認問題.xlsm」を使用します。

	A	B	C	D
1	販売日	商品型番	販売金額	販売数量
2	2022/1/5	D88-F-PPL	424,480	33
3	2022/1/20	T32-S-RED	279,000	56
4	2022/2/8	Y46-M-RED	352,800	65
5	2022/2/19	J01-T-PPL	375,000	33
6	2022/3/10	L01-T-RED	780,000	63
7	2022/3/20	D88-F-BLK	134,820	62
8	2022/3/30	Y46-M-GRN	138,000	55
9	2022/4/13	Q81-S-BLU	285,000	58
10	2022/4/24	J01-T-PPL	350,000	62
11	2022/5/7	J01-T-PPL	150,000	56
12	2022/5/20	Z20-M-BLU	58,800	64
13				

動画レッスンD-4t

https://excel23.com/vba-juku/chap-special-d4test/

1. 表において、次の条件に一致するデータを抽出してください。

- **フィールド：［販売金額］列**
- **基準：300,000以上、500,000以下**

※上記の処理の前に、前回のフィルターを解除するコードも書いてください。

2. 表において、次の条件に一致するデータを抽出してください。

- フィールド：[商品型番] 列
- 基準：文字列が "D88" を含んでいる、または文字列が "RED" を含んでいる

※上記の処理の前に、前回のフィルターを解除するコードも書いてください。

3. 表において、次の条件に一致するデータを抽出してください。

- フィールド：[販売日] 列
- 基準：2022/3/10 より後、2022/4/25 より前

※上記の処理の前に、前回のフィルターを解除するコードも書いてください。

4. 表において、次の条件をどちらも満たすデータを抽出してください。

- (条件1) [販売金額] 列のセルの値が 300000 より大きい
- (条件2) [販売数量] 列のセルの値が 60 より大きい

※上記の処理の前に、前回のフィルターを解除するコードも書いてください。

※模範解答は、PDFの巻末に掲載しています。

補講E：デバッグ エラーが起きたらどうする？

教材ファイル「補講E.xlsm」を使用します。

動画レッスンE

https://excel23.com/vba-juku/chap-special-e/

VBAのコードを書いていると、必ずといっていいほど「エラー」が起こります。この補講では、エラーが起きたらどのように対処すればよいか、対処方法を解説します。
また、エラーの原因箇所を探し、コードを修正する作業を「デバッグ」といいますが、そのデバッグのための基本的な操作も解説します。

1. エラーの種類を知っておこう

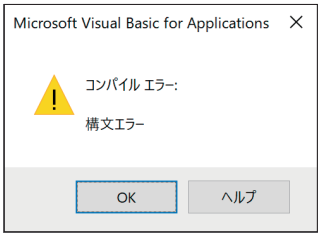
マクロを作成する私達にとって「エラー」というものが「敵」だとすれば、「敵が何であるか？」を明確にして対処する必要があります。まずは、エラーの種類について知っておきましょう。ひとことで「エラー」と言っても、大きく分けて3つの種類があります。

表E-1-1

	構文チェック	マクロの実行	実行結果
①コンパイルエラー	×構文エラーで止まる		
②実行時エラー	○構文エラーなし	×実行時にエラーで止まる	
③論理エラー	○構文エラーなし	○実行時エラーなし	×本来の目的通りの結果にならない

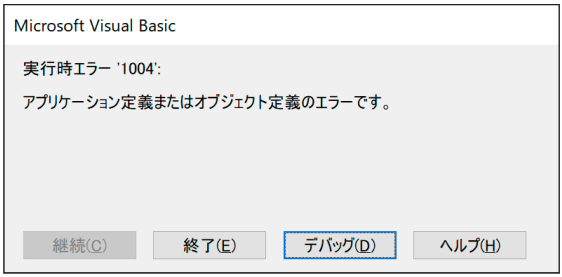
①コンパイルエラー

そもそもマクロを実行する前に、構文が間違っていることが原因で起きるエラーです。
コードの入力中や、マクロの実行ボタンを押した直後に「コンパイルエラー」という警告メッセージが表示されてマクロが停止します。



②実行時エラー

マクロの実行中に起きるエラーです。



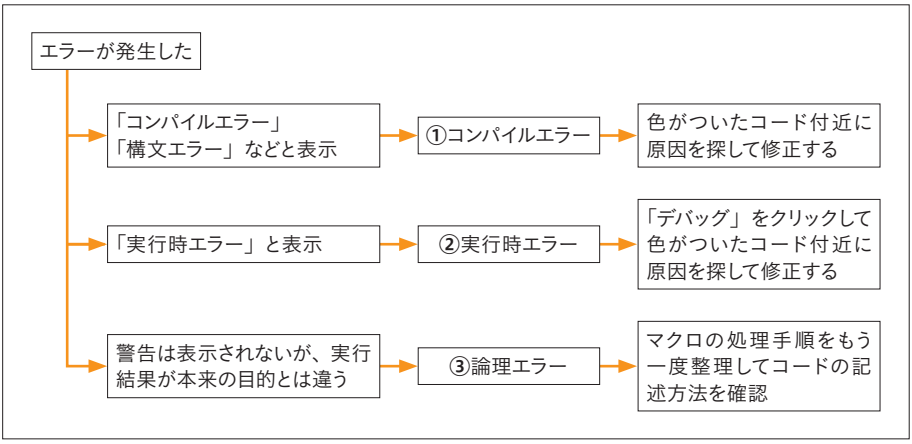
①の「構文エラー」は、マクロを実行する前に構文が間違っていることが原因でエラーになりますが、②の「実行時エラー」は、構文は間違っておらず、マクロを実行できたものの、途中で不具合が起きるためエラーとなります。図のような警告メッセージが表示されます。

③論理エラー

コードは間違っていないのですが、作成者の意図した結果にならないことを指します。
この種類のエラーでは、具体的な警告メッセージは表示されません。
マラソンに例えると、「スタート地点を出発して42.195kmを走り切ることはできたものの、目指していたゴール地点とは違う場所にたどり着いてしまった……」というイメージです。
何も警告が表示されず、一見エラーではないように見えてしまうので、注意が必要です。

2. エラーの種類別の対処法

1.では3つの種類のエラーを説明しました。では、実際にエラーが起きたらどう対処すればいいか、種類別に解説します。



①コンパイルエラーのよくある原因と対処法

コンパイルエラーは、一般的によく起こるエラーです。「構文エラー」とも呼ばれ、VBAの構文が間違っていることで起こります。マクロはVBAのコードを実行する前に、コンピューターが理解できる「機械語」というものに変換する作業を行います。その作業を「コンパイル」といいます。しかし、VBAの構文が間違っていると、コンパイルがうまくいかずにエラーとなります。

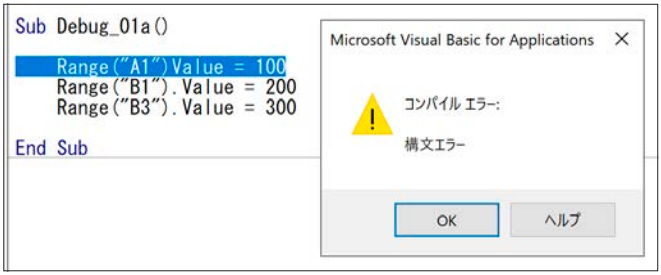
よくある原因として以下のようなものが考えられます。まずは落ち着いて、コードの間違った箇所を探すことから始めましょう。

- **スペルを間違えている**
- **スペースを空けるべき箇所にスペースが入っていない**
- **上の句に対する下の句を忘れている（`If`ステートメントに対する`End If`を忘れているなど）**
- **間違った位置で改行している**
- **宣言していない変数名を使用している**

  など

具体例を見ていきましょう。

1. 以下は、`Value`プロパティの前の「 . 」の入力を忘れてマクロを実行してしまい、コンパイルエラーが表示された例です。



構文エラーは、多くの場合、コードの間違った箇所に色がついてハイライトされたり、フォントが赤くなったりします。

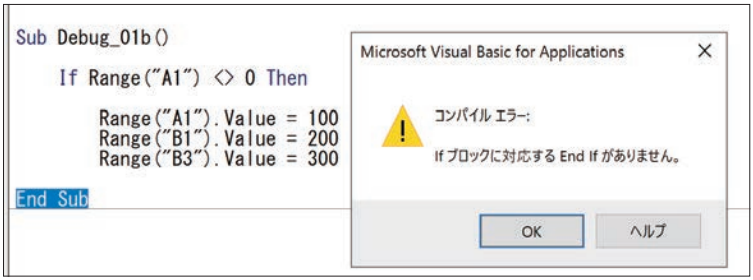
2.「OK」をクリックすると、マクロが一時停止状態のままコードを編集できます。ここでは、「Value」の前に「.」を入力して修正します。

```
Sub Debug_01a()

    Range("A1")Value = 100
    Range("B1").Value = 200
    Range("B3").Value = 300

End Sub
```

「.」を入力

3.コードを修正できたら、「継続」ボタンを押してマクロを再実行するか、または、「リセット」ボタンを押して一度マクロの実行を終わり、もう一度マクロを実行し直します。

時々、ハイライトされた部分とは別の場所に間違いがあり、それが原因でハイライトされている場合もあります。例えば、以下の例を見てみましょう。



上記は、条件分岐のIf～End If構文（**書籍第8章**で解説）において、終わりの「End If」を書き忘れたことが原因でコンパイルエラーが表示されました。ところが、ハイライトされたのは「End Sub」の行になっています。このように、ハイライトされたコードと実際のエラーの原因箇所が異なっている場合があるので注意しましょう。

②実行時エラーのよくある原因と対処法

「実行時エラー」は、コードの構文は間違っていないものの、マクロの実行途中で不具合が起きてしまうケースです。

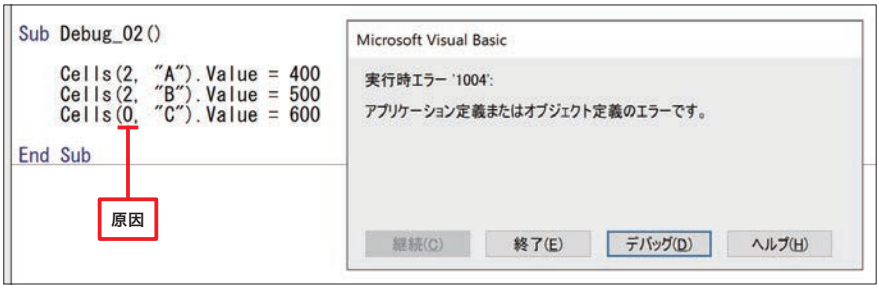
よくある原因としては、次のことが考えられます。

- 存在しないセルやシートやブックなどを指定している
- 存在しないプロパティやメソッドを記述している
- 変数の型に合わないデータを代入しようとしている

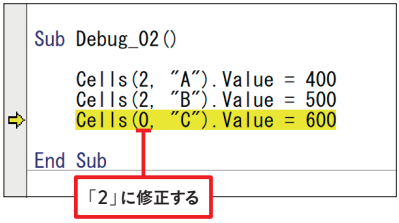
など

具体例を見てみましょう。

以下は、「Cells(0,1)」と記述したために実行時エラーが起きた例です（Cellsの第1引数に「0」と入力していますが、Excelシートに「0行目」は存在ないためエラーとなります）。



1. ［デバッグ］をクリックすると、エラーの原因箇所が黄色く表示されます。そのままマクロが一時停止状態になり、コードを編集することができます（ここでは、Cellsの第1引数の「0」を「2」に修正します）。



2. コードを修正できたら、「継続」ボタンを押してマクロを再実行するか、または、「リセット」ボタンを押して一度マクロの実行を終わり、もう一度マクロを実行し直します。

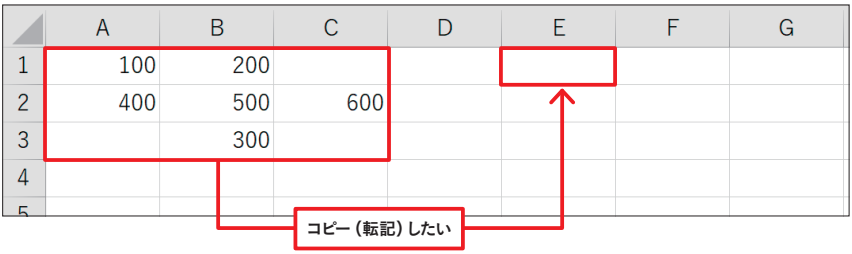
①のコンパイルエラーと同じく、ハイライトされた部分とは別の箇所にエラーの原因がある場合もあります。注意しましょう。

③論理エラーのよくある原因と対処法

論理エラーは、コードは間違っていないのですが、作成者の意図した結果にならないことを指します。ここまで紹介した2つのエラーとは違って、論理エラーは、エラーが起きてもマクロが停止しません。そのため、エラーに気づいて原因を探すのが難しいので注意が必要です。

具体的には、以下の例を見てみましょう。

[例] セル範囲A1:C3をコピーし、E1に貼り付けたい




```
Sub Debug_03()
    Range("A1:C3").Cut Range("E1")
End Sub
```

上記のコードを実行すると、セル範囲A1:C3はセルの値がなくなってしまいました。「コピー」をするつもりでコードを書いたのですが、コピーではなく「切り取る」処理が行われてしまったのです。

原因は、コピーするためにはCopyメソッドを記述すべきところを、誤ってCutメソッド（切り取る）を記述してしまったことです。修正するには、「.Cut」という部分を書き直して「.Copy」に書き換える必要があります。

以上は「論理エラー」の簡単な例でした。構文は間違っていないですし、エラーの警告が表示されることはありませんが、作成者の意図した結果にならないという状況が理解できたでしょうか。
そのため、3種類のエラーの中でも「論理エラー」は気づくのが難しく、最もデバッグが難しいと言えます。論理エラーが起きた場合は、自分が作ろうとしていたマクロの処理手順をもう一度整理し直し、コードの記述方法に間違いがなかったか注意深く確認する必要があります。

補足

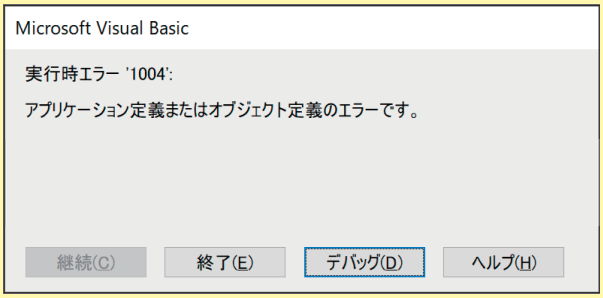
Webで検索して調べる秘訣！
エラー文やエラーコードで検索しよう

VBAの学習において、「自分でエラーの解決方法を調べる」というスキルは非常に大事です。中でも、Googleなどの検索エンジンで検索する方法は有効です。しかし、やみくもに検索をしても、欲しい情報は見つかりません。
エラーの原因について検索する前に、以下の情報を整理しておきましょう。

① エラーの説明文
② エラーコード（番号）
③ エラーに至った経緯や、書いているコード

これらの情報をもとに検索すれば、高い確率で有益なWebサイトにたどり着けるようになります。以下に具体的な例を用いて説明します。

［例］以下のようなエラーメッセージが表示されたとします



実行したコード

```
Sub Debug_04()
    Cells(0, 1).Value = 10
End Sub
```

まずは情報を整理してみましょう。

①エラーの説明文

「アプリケーション定義またはオブジェクト定義のエラーです。」

②エラーコード（番号）

「実行時エラー　'1004'」

③エラーに至った経緯や、書いているコード

「`Cells(0, 1).Value = 10`」というコードを記述して実行したら、上記のエラーメッセージが出た。「デバッグ」ボタンを押すと、そのコードが黄色くハイライトされた。

上記の情報を元に、Googleで検索するキーワードを考えていきます。
例えば、以下のような検索キーワードが考えられます。

- 「VBA Cells 実行時エラー 1004」
- 「VBA Cells アプリケーション定義またはオブジェクト定義のエラー」
- 「VBA Cells 1004」

- ポイントは、「VBA」というキーワードを含めることです。VBAに関するWebサイトを検索しやすくなります。
- また、「`Cells`」というキーワードを含めます。理由は、エラーが出たコードに「`Cells`」が含まれていたので、「`Cells`に関わるエラーではないか？」という仮説を立てたからです。
- 上記のキーワードに続けて、［エラーコード］や、［エラーの説明文］なども複合キーワードとして組み合わせることも重要です。

今回は3つの検索ワードを挙げましたが、1つの検索ワードで必ず求める情報が見つかるとは限りません。1つ目の検索ワードで探しても思うような情報が見つからなかった場合は、2つ目、3つ目も試してみます。

上記のキーワードで検索してみましょう。複数のサイトが表示されます。理想としては、1つではなく複数のサイトを比較してみることが望ましいです。理由は、Webサイトの情報は、そのサイトの管理者個人の視点から説明されているため、複数の人の色んな角度から情報を得た方がバランスのよい情報が得られるからです。

●

いくつかサイトを見たところ、エラーの原因を推測することができました。
Cellsの引数に、ありえない数値を記述してしまったことが原因でエラーになったのではないか？（具体的には、引数に「0」を入力してしまっている）
上記を仮説として、コードを修正します。

修正後のコード

```
'Cellsの引数の「0」を「1」に書き換える
Sub Debug_04()
    Cells(1, 1).Value = 10
End Sub
```

上記のように修正した結果、エラーが起こらなくなりました。

●

いかがだったでしょうか？
以上のような流れで検索すると、エラーの原因を発見できる確率がぐっと高くなります。
VBAの学習においては、「検索して調べる」というスキルも非常に重要なスキルとなってきます。
やみくもに検索するのではなく、情報を整理してから検索し、戦略的に自分の求める情報を探していきましょう。

3.「ステップ実行」で、コードを1行ずつ確かめる！

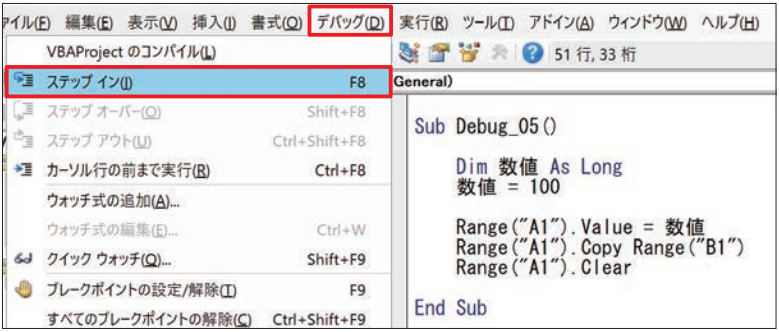
VBAのデバッグをするすべての人におすすめしたいのが「ステップ実行」という手法です。
ステップ実行とは、コードを1行ずつ実行しながら動作確認をする手法です。
この方法を活用すれば、

- 自分の書いたコードの動作確認やエラーの原因調査をする
- 他人の書いたコードの意味を読み取り、理解する

上記の両方に役に立ちます。ぜひ積極的に活用してみましょう。

ステップ実行を行うには、VBEの「ステップイン」という機能を利用します。

1 VBEの「デバッグ」メニューの「ステップイン」を押すか、**F8キー**を押すと、ステップ実行が開始されます。



2 コードが1行ずつ黄色でハイライトされ、F8キーを押すたびにその行が実行されていきます。

3 最後の「End Sub」の行がハイライトされ、そのままF8キーを押すとマクロが終了します。

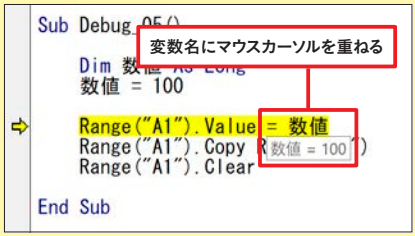
最後まで実行する前に「リセット」ボタンを押してマクロを途中で終了することも可能です。

「F8キー」でステップインを実行する方法は非常に便利です。ぜひ積極的に利用しましょう。

補足

変数の値を確かめる「自動データヒント」

ステップ実行中に、コード内の変数名にマウスカーソルを重ねると、その変数に代入されている値がポップアップで表示されます（この機能を「自動データヒント」といいます）。



コードを実行しながら変数の値を簡単に確認したい場合に便利です。

4.「ブレークポイント」で、指定した行でマクロを一時停止する！

3.で紹介した「ステップ実行」は、コードを1行ずつ実行していく機能でした。ですが、**長いコードになってくると、1からステップインで順番に進めていくのが大変になってきます。**そこで「ブレークポイント」という機能も知っておきましょう。「ブレークポイント」を利用すると、**指定した行までマクロを一気に進行した後、その行で一時停止することができます。**

1 指定したいコードの**左側の余白をクリックする**か、行を選択して**F9キー**を押します。

2 すると、左側の余白に**赤い丸**が表示されます。これが「ブレークポイント」です。

3 コードの実行ボタンを押す（またはF5キーで実行する）と、ブレークポイントまで一気に実行され、ブレークポイントの行で**一時停止状態**になります。

```
Sub Debug_05()

    Dim 数値 As Long
    数値 = 100

    Range("A1").Value = 数値
    Range("A1").Copy Range("B1")
    Range("A1").Clear

End Sub
```

この行の実行前で一時停止する

ブレークポイントを**解除**するには、赤い丸をもう一度クリックするか、その行を選択してF9ボタンを押すことで、赤い丸を消します。

- ブレーク中には、「**自動データヒント**」を利用できます。変数名にマウスカーソルを重ねると、現在変数に代入されている値がポップアップ表示されます。ブレーク時点の変数の値を確かめることができます。

```
Sub Debug_05()

    Dim 数値 As Long
    数値 = 100

    Range("A1").Value = 数値
    Range("A1").Copy R数値 = 100")
    Range("A1").Clear

End Sub
```

変数名にマウスカーソルを重ねる

- ブレークした箇所から **3.** の「**ステップイン（F8キー）**」を始めることができ、以降、F8キーを押すごとに1行ずつコードを動作確認できます。
- 実際にデバッグするときには、ブレークポイントを使って目的とする行まで一気に飛んでから、F8キーのステップインで1行ずつコードを動作確認していくのがおすすめです。

補足

デバッグに役立つ！「Debug.Print」

書籍第3章レッスン2で紹介した「Debug.Print」も、デバッグに役立ちます。このコードは、例えば「Debug.Print "おはよう"」などと記述すると、イミディエイトウィンドウ（Ctrl+Eで表示）という場所に、"おはよう"という文字列がメッセージのように出力されるという説明をしました。

他にも、Debug.Printはデバッグに便利な使い方が多数あります。すべてを紹介すると非常に奥が深いので、この場では一部のみを紹介します。

例

```
Debug.Print 変数名                    '変数の値を出力する
Debug.Print Range("A1").Value         'セルA1の値を出力する
Debug.Print Range("A1").Value * 100   'セルA1の値×100の演
                                       算結果を出力する
```

- 上記のいずれも、「Debug.Print」の引数に記述したコードの結果が、イミディエイトウィンドウに出力されます。

上記のコード例のようにDebug.Printを利用すると、

- 変数の値を確認したい
- セルの値を確認したい
- 演算結果を確認したい

といったときに役に立ちます。

書籍で説明した通り、イミディエイトウィンドウは、VBAの開発者側にしか見ることができないウィンドウです。したがって、ユーザーには見せることなく、開発者側だけが確認できる情報を出力する目的でDebug.Printは役に立ちます。

補講の確認問題の模範解答

補講A レッスン1

```
Sub 問題1()
    Range("A1:F1").Font.Bold = True
End Sub

Sub 問題2()
    Range("A1:F1").Font.Color = RGB(255, 0, 0) 'vbRedでも可能
End Sub

Sub 問題3()
    Range("A1:F1").Font.Name = "メイリオ"
End Sub

Sub 問題4()
    Dim i As Long
    For i = 2 To 11
        If Cells(i, "F").Value = "3LDK" Then
            Cells(i, "F").Font.Bold = True
        End If
    Next i
End Sub

Sub 問題5()
    Dim i As Long
    For i = 2 To 11
        If Cells(i, "D").Value >= 5000 Then
            Cells(i, "D").Font.Bold = True
        End If
    Next i
End Sub
```

補講A レッスン2

```
Sub 問題1()
    Range("B2:F2").Interior.Color = RGB(255, 255, 0)  'vbYellowも可
End Sub

Sub 問題2()
    Range("B2").CurrentRegion.Borders().LineStyle = xlContinuous
End Sub

Sub 問題3()
    Range("B2:F2").Borders(xlEdgeBottom).LineStyle = xlDouble
End Sub
```

補講A レッスン3

```
Sub 問題1()
    Columns("E:G").NumberFormatLocal = "#,##0"
End Sub
```

```
Sub 問題2()
    Columns("B").NumberFormatLocal = "yyyy/mm/dd"
End Sub

Sub 問題3()
    Columns("A").NumberFormatLocal = "000"
End Sub
```

補講B レッスン1

```
Sub 問題1()
    Rows(8).Select
End Sub

Sub 問題2()
    Columns("D").Select
End Sub

Sub 問題3()
    Rows("3:6").Select
End Sub

Sub 問題4()
    Columns("B:E").Select
End Sub

Sub 問題5()
    Dim num As Long
    num = 5
    Rows(num).Select
End Sub

Sub 問題6()
    '最終行を取得
    Dim maxRow As Long
    maxRow = Cells(Rows.Count, 1).End(xlUp).Row

    '行を選択
    Rows(maxRow).Select
End Sub
```

補講B レッスン2

```
Sub 問題1()
    Rows(11).ClearContents
End Sub

Sub 問題2()
    Columns("B:D").Copy Columns("H")
End Sub

Sub 問題3()
    Columns("D").NumberFormatLocal = "#,###"
End Sub

Sub 問題4()
    Columns("B:F").AutoFit
End Sub
```

```
Sub 問題5()
    Rows(1).RowHeight = 24
End Sub
```

補講B レッスン3

```
Sub 問題1()
    Columns("C").Insert
End Sub

Sub 問題2()
    Rows(5).Insert
End Sub

Sub 問題3()
    Columns("E:G").Delete
End Sub

Sub 問題4()
    Rows("7:10").Delete
End Sub
```

補講C レッスン1

```
Sub 問題1()
    Range(Range("A1"), Range("D9")).Select
End Sub

Sub 問題2()
    Range("3:5").Select
End Sub

Sub 問題3()
    Range("C:D").Select
End Sub
```

補講C レッスン2

```
Sub 問題1()
    Range("A1").CurrentRegion.Select
End Sub

Sub 問題2()
    Range("A1").End(xlToRight).Select
End Sub

Sub 問題3()
    Range("A1").Offset(3, 3).Select
End Sub
```

補講D レッスン1

```
Sub 問題1()
    Dim i As Long
    For i = 2 To 13
```

```
        If Cells(i, "A").Value Like "??????" Then
            Cells(i, "C").Value = "OK"
        Else
            Cells(i, "C").Value = "NG"
        End If
    Next i
End Sub

Sub 問題2()
    Dim i As Long
    For i = 2 To 13
        If Cells(i, "A").Value Like "?#####" Then
            Cells(i, "C").Value = "OK"
        Else
            Cells(i, "C").Value = "NG"
        End If
    Next i
End Sub

Sub 問題3()
    Dim i As Long
    For i = 2 To 13
        If Cells(i, "A").Value Like "[a-c]#####" Then
            Cells(i, "C").Value = "OK"
        Else
            Cells(i, "C").Value = "NG"
        End If
    Next i
End Sub
```

補講D レッスン2

```
Sub 問題1()
    Dim i As Long
    For i = 2 To 12
        If Cells(i, "B").Value > 300000 And _
           Cells(i, "C").Value > 60 Then
            Cells(i, "D").Value = "Yes"
        Else
            Cells(i, "D").Value = "No"
        End If
    Next i
End Sub

Sub 問題2()
    Dim i As Long
    For i = 2 To 12
        If Cells(i, "B").Value > 300000 Or _
           Cells(i, "C").Value > 60 Then
            Cells(i, "E").Value = "Yes"
        Else
            Cells(i, "E").Value = "No"
        End If
    Next i
End Sub
```

```
Sub 問題3()
    Dim i As Long
    For i = 2 To 12
        If Not Cells(i, "B").Value > 300000 And _
           Not Cells(i, "C").Value > 60 Then
            Cells(i, "F").Value = "Yes"
        Else
            Cells(i, "F").Value = "No"
        End If
    Next i
End Sub
```

補講D レッスン3

```
Sub 問題1()
    Dim 回答 As Long
    回答 = MsgBox("送料500円を加算しますか?", vbYesNo)

    If 回答 = vbYes Then
        Range("D3").Value = Range("B3").Value * Range("C3").Value + 500
    Else
        Range("D3").Value = Range("B3").Value * Range("C3").Value
    End If
End Sub

Sub 問題2()
    Dim 敬称 As String
    敬称 = InputBox("敬称を入力して下さい。")

    Dim i As Long
    For i = 8 To 11
        Cells(i, "C").Value = Cells(i, "B").Value & 敬称
    Next i
End Sub

Sub 問題3()
    Dim 請求書番号 As String

    Dim i As Long
    For i = 16 To 20
        請求書番号 = Format(Cells(i, "B").Value, "yyyy_mm_dd")
        Cells(i, "C").Value = 請求書番号
    Next i
End Sub

Sub 問題4()
    Dim 何文字目 As Long

    Dim i As Long
    For i = 3 To 10
        何文字目 = InStr(Cells(i, "G").Value, "-")
        Cells(i, "H").Value = Left(Cells(i, "G").Value, 何文字目 - 1)
        Cells(i, "I").Value = Mid(Cells(i, "G").Value, 何文字目 + 1)
    Next i
End Sub
```

補講D レッスン4

```
Sub 問題1()
    Range("A1").AutoFilter
    Range("A1").AutoFilter 3, ">=300000", xlAnd, "<=500000"
End Sub

Sub 問題2()
    Range("A1").AutoFilter
    Range("A1").AutoFilter 2, "*T32*", xlOr, "*RED*"
End Sub

Sub 問題3()
    Range("A1").AutoFilter
    Range("A1").AutoFilter 1, ">2022/3/10", xlAnd, "<2022/4/25"
End Sub

Sub 問題4()
    Range("A1").AutoFilter
    Range("A1").AutoFilter 3, ">300000"
    Range("A1").AutoFilter 4, ">60"
End Sub
```

STAFF
ブックデザイン：岩本 美奈子
本文イラスト：まつむら まきお
DTP・本文図版：AP_Planning
編集：角竹 輝紀

Excel VBA塾 補講編

2021年11月30日　電子版v1.0

著者　たてばやし 淳
発行者　滝口 直樹
発行所　株式会社マイナビ出版
〒101−0003
東京都千代田区一ツ橋2−6−3 一ツ橋ビル 2F
E-Mail：pc-books@mynavi.jp
URL：https://book.mynavi.jp

© 2021 たてばやし 淳

- 本書の一部または全部の再配布、販売などを禁じます。
- 本書掲載内容の無断転載を禁じます。
- 本書は著作権法上の保護を受けています。本書の無断複写・複製は、著作権法上の例外を除き、禁じられています。