EXCEL VBA 入門教材

急がば回れ！文法から覚えるやさしい VBA 入門

要約

プログラミングは文法こそが肝の部分です。しかし Excel VBA 入門書はたいてい文法を後回しにします。楽しいセルの操作の後に小難しい文法が出てきたら、そこで挫折してしまうのも無理はありません。この入門教材では、全く逆のアプローチで Excel VBA を丁寧に解説します。

本村 貴之

まえがき

この教材は Excel の操作を自動化するためのプログラミング言語である VBA について解説したものです。すでに VBA の入門書はたくさん出回っていますが、私が知る限りはどれも似たような構成になっています。はじめにマクロの自動記録、それからセルの操作などが続いて最後の方に申し訳程度に文法などが紹介されています。

VBA 以外の他のプログラミング言語は、まず文法から学習します。文法はプログラミング言語の要ですので、ここをしっかり理解しないことには、役に立つプログラムは作れません。しかしなぜか VBA に限っては文法を中心に解説された入門書が見つからず、前述のような状況です。

Excel VBA しか経験が無い方で、Excel VBA をバリバリ使いこなしている方には今までお会いしたことがありません。やはり他のプログラミング言語で文法をしっかり学んでいる人のほうが VBA もスマートに使いこなしているようです。これは VBA の入門書の構成にも原因があるのではないかと思います。

まずマクロの自動記録機能ですが、これさえあれば難しい VBA を覚えなくてもマクロを作れると錯覚してしまいます。しかしそれはあり得ません。記録させたコードは再生させても同じことを行うだけで、そのままではプログラムとして役に立ちません。大半の部分を自動記録で済ませて VBA で少しだけ編集することもできなくはないですが、自動記録は大量のコードを出力するので、入門者には手に負えないでしょう。

次にセルの操作ですが、これも文法と組み合わせてこそ効果を発揮します。選択されたセルの背景色とテキスト色を一度に変えたいなどの単純操作なら文法を覚えなくてもできますが、例えば条件にあったセルだけ色を変えたいなどの少し高度なことをしようとすると、文法が必要になります。

途中まで楽しく読み進めてきて、最後に難しい文法にぶちあたる。これって結構がっかりしませんか。結局文法が無いと何もできないんですが、文法は上級者向けの機能だと勘違いしてあきらめてしまうケースもあるようです。

そこで本教材では、まず文法を中心にしっかり解説し、最後の方でセルの操作とマクロの自動記録について解説します。コード例もたくさん掲載しますので、手を動かして実際に実行してみてください。経験上、最初にプログラミング言語を覚えるときは極力コピー&ペーストは使わず手で入力する方が、頭に入ります。

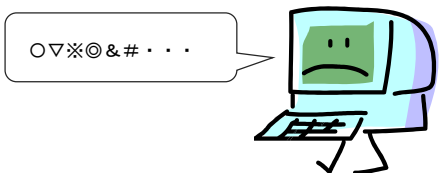
それでは、はじめていきましょう。

目次

はじめに

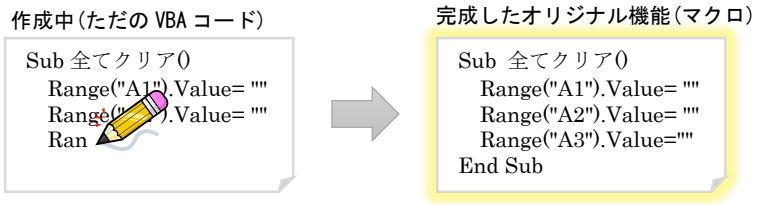
プログラミング言語とは

コンピューターは、人間が命令した通りに働いてくれます。ただし、コンピューターは日本語を理解できません。コンピューターに命令を出すには専門のコトバを覚える必要があります。それが、プログラミング言語です。そのプログラミング言語で書かれた命令書のことを、ソースコード(または単にコード)と呼びます。ソースと略す場合もあります。



VBA とマクロについて

VBA は Visual Basic for Application の略で、Microsoft Office の操作を自動化する際に使用するプログラミング言語です。Excel においては、この VBA で記述された一連の手続きをマクロと呼びます。コードという言葉は単にプログラミング言語で記述された命令や命令の集まりを指すのに対し、マクロという言葉は完成したオリジナル機能としての呼び方です。



実際は作成中でもマクロという呼び方もしますし、完成してもコードであることは変わりませんが、視点の違いで呼び方が変わるということを覚えておいてください。

Excel において「マクロの自動記録」機能は VBA のコードを自動的に生成してくれるものであって、この機能を使って記録されたものだけがマクロではありませんので注意してください。

VBA の特徴

世には様々なプログラミング言語がありますが、VBA はオブジェクト指向というカテゴリーに入ります。**オブジェクト**は日本語でモノのことです。テレビ・冷蔵庫・洗濯機などもオブジェクトですし、車・人・動物・書類など現実世界に存在するものはすべてオブジェクトと言えます。

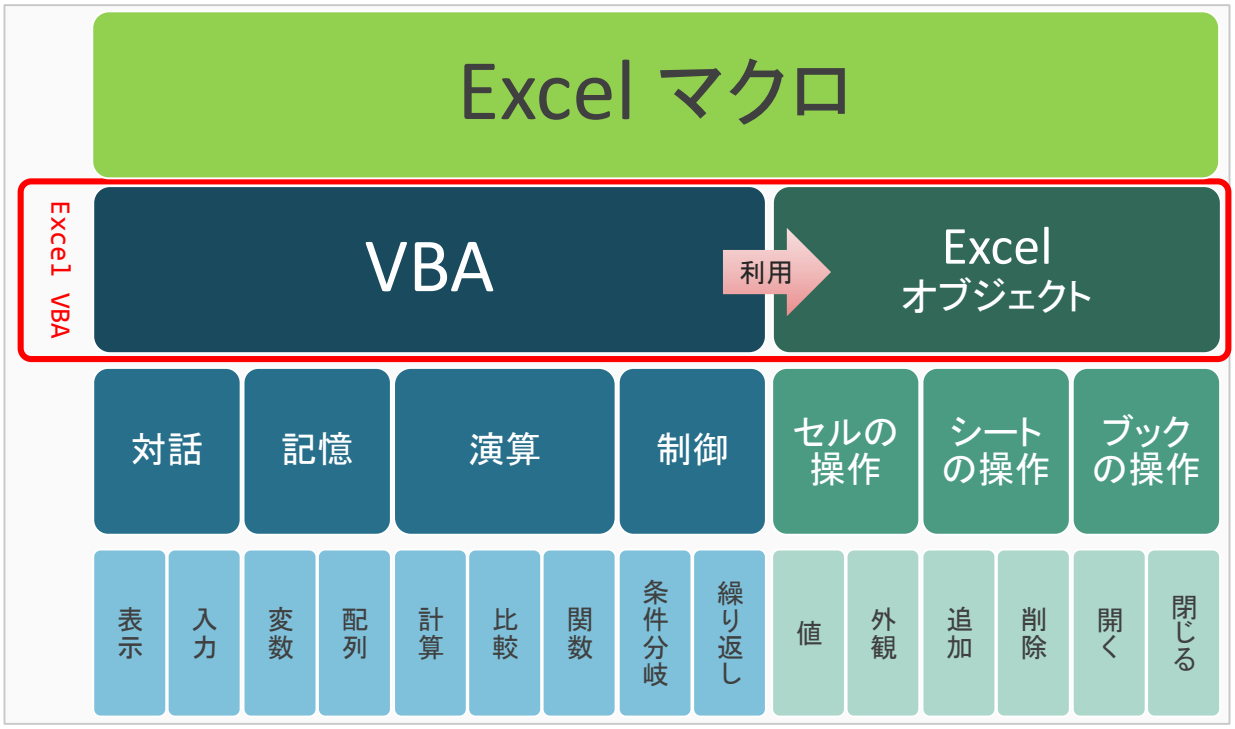
プログラミングにおいてオブジェクトは、機能と性質を持ち合わせたモノを指します。例えば現実世界の車は、走る・曲がる・止まるなどの機能と、ガソリン量・排気量・車高などの性質があります。Excel も、セルの色を変えたりシートを削除したりといった機能と、選択されているセル・セルの値・シートの枚数などの性質を持ちますので、コンピューター上のオブジェクトであると言えます。オブジェクトの機能のことを**メソッド**、性質のことを**プロパティ**と呼びますので

覚えておいてください。オブジェクトのメソッドやプロパティを利用したプログラミングスタイルをオブジェクト指向といいます。

Excel マクロの構成

Excel マクロは VBA で記述されますが、純粋に言語としての VBA と、Excel オブジェクトの操作に大別することができます。例えばセルの色を変えたりシートを削除したりといった命令は純粋な VBA 言語の命令ではなく、Excel が備える機能を VBA から利用しているわけです。

通常は Excel オブジェクトの操作もひっくるめて Excel VBA と呼ばれます。



言語としての VBA は、大きく 4 つの機能に分けられます。演算や制御など小難しい言葉が並んでいますが、実際にコードを書いてみればそんなに難しいものではないので今は気にしないでください。

この教材で学習するのはその純粋な言語としての VBA ですが、Excel オブジェクトの操作についても後半で少し扱います。Excel オブジェクトの操作は実際の Excel 操作の種類だけ存在するた

め、全てを覚えるのは不可能です。代表的な操作をいくつか紹介しますので、その他の操作については必要になった時に書籍やインターネット等で調べてください。

純粋に言語としての VBA は Office 製品で共通となっています。Word マクロも Power Point マクロも扱うオブジェクトは異なりますが VBA の文法を覚えれば応用が利きます。

コラム　マクロの語源

マクロとはもともと「大きい」という意味のギリシャ語です。これが英語に取り入れられて、「おおきな○○」といった意味の英単語を作る際の接頭語(頭に付ける語)として定着しました。

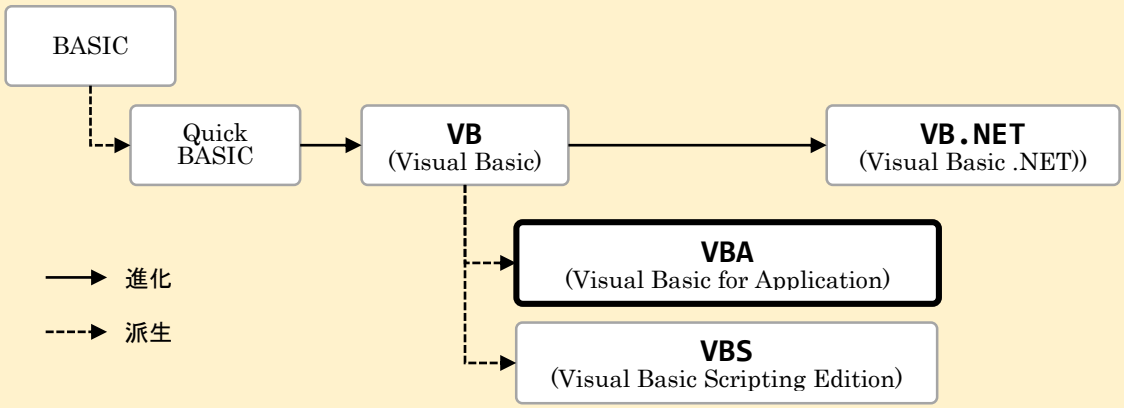
コンピューター用語としてのマクロは、正式にはマクロ命令といいます。小さな命令を複数組み合わせることで、ひとつの大きな命令として扱えるようにしたものがマクロ命令です。今では正式名で呼ばれることはほとんど無いので、「マクロ命令」と言っても通じない方のほうが多いと思います。

コラム　VBA の歴史

VBA の先祖は、1964 年に作られた BASIC という初心者向けの言語です。

もともと BASIC は黒い画面に文字だけの言語でしたが、マイクロソフトがこれを派生させて QuickBASIC を作り、さらにボタン等をマウスで配置できるビジュアルな Basic となりました。これが本家の VB です。

そこから、Excel 等のアプリケーション用に派生したものが VBA で、Windows やブラウザの操作用に派生したものが VBS です。本家の方は進化して、VB.NET(ブイビードットネット)という言語になっています。



VB から先は全部兄弟なので、コードの見た目がよく似ていますが、細かい部分が異なりますので、例えば VB のコードをそのまま VBA に貼り付けても動きません。インターネットなどで VBA のサンプルを検索する際には、間違えずに VBA と書かれたものを選んでください。

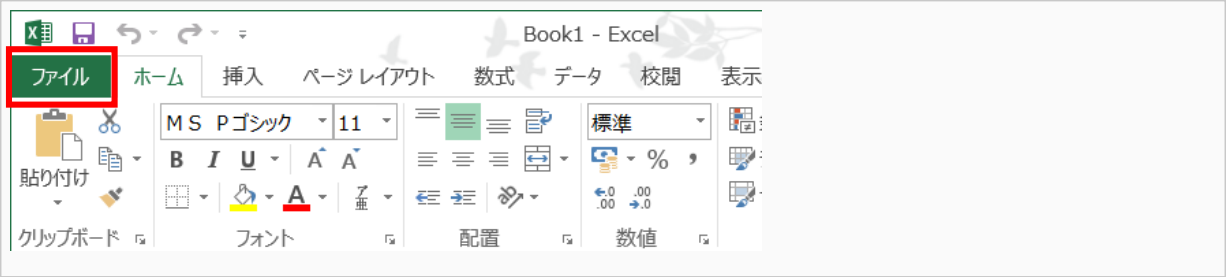
文法などがよく似ていますので、VBA を覚えると、他の VB シリーズもスムーズに理解できるはずです。

プログラミングの準備

では、早速 Excel でプログラミングを始める準備をしましょう。

今開いているワークブックは全て閉じて、一度 Excel を終了してください。そして、新しく Excel を起動します。

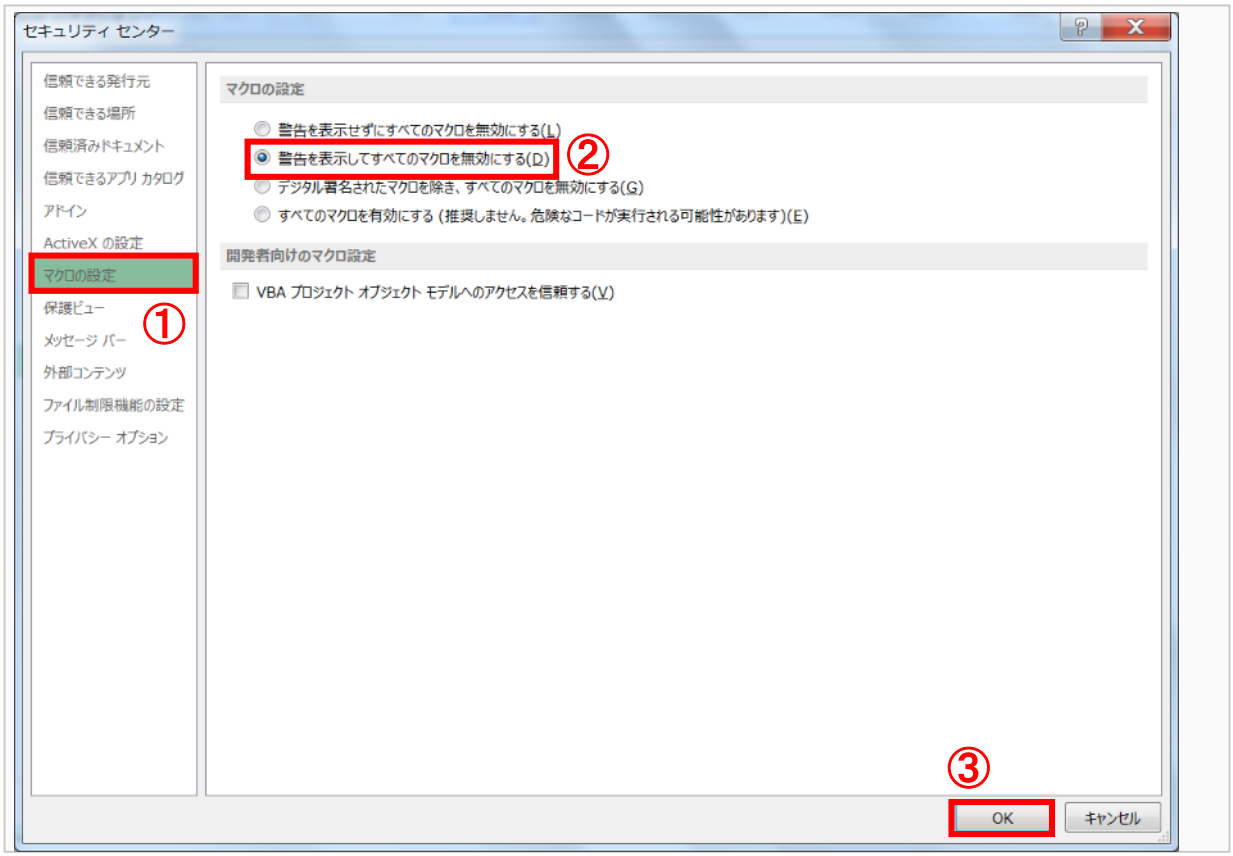
新規 Excel ブックを開き、ファイルをクリックします。



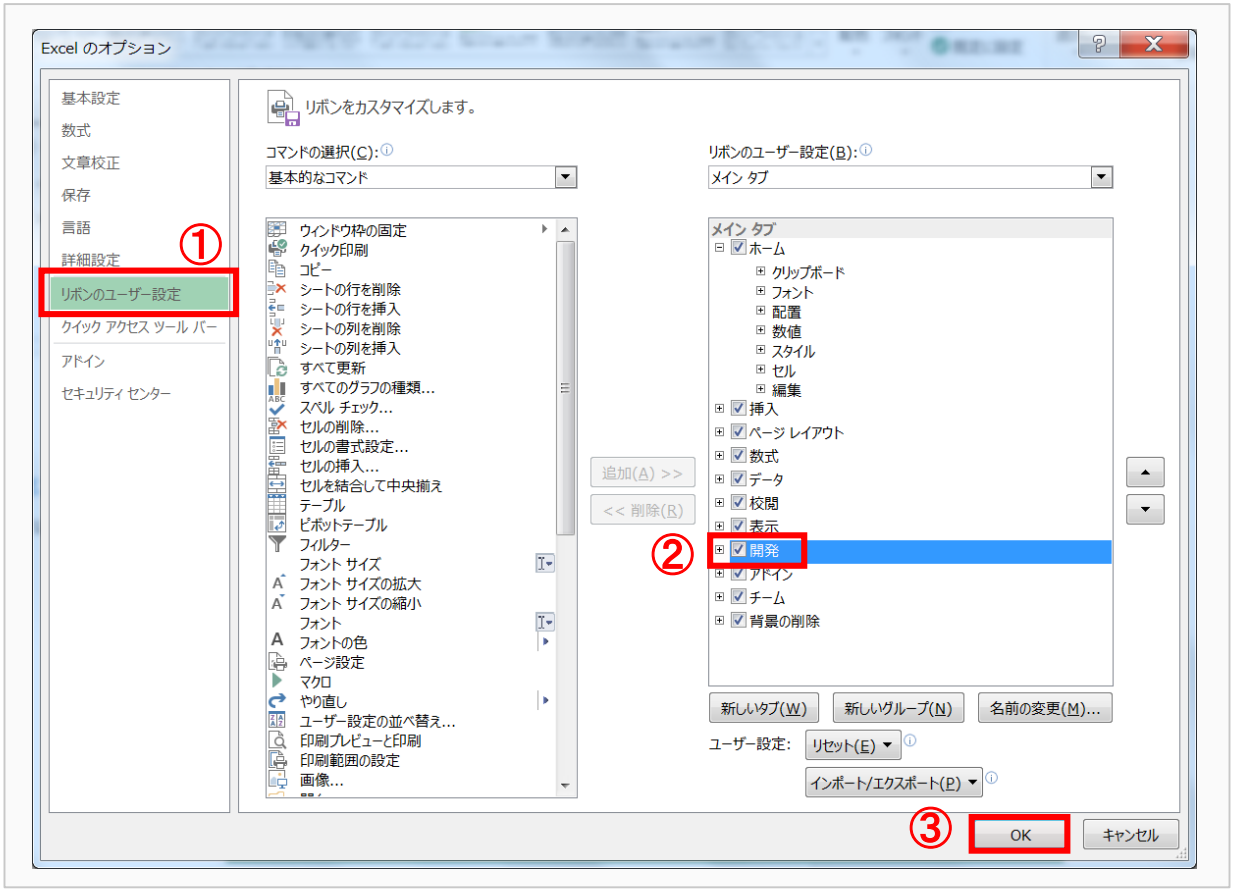
次に、左下にあるオプションを開きます。

Excel のオプションが開いたら、左のメニューからセキュリティセンターを開き、右下に表示されるセキュリティセンターの設定ボタンをクリックします。

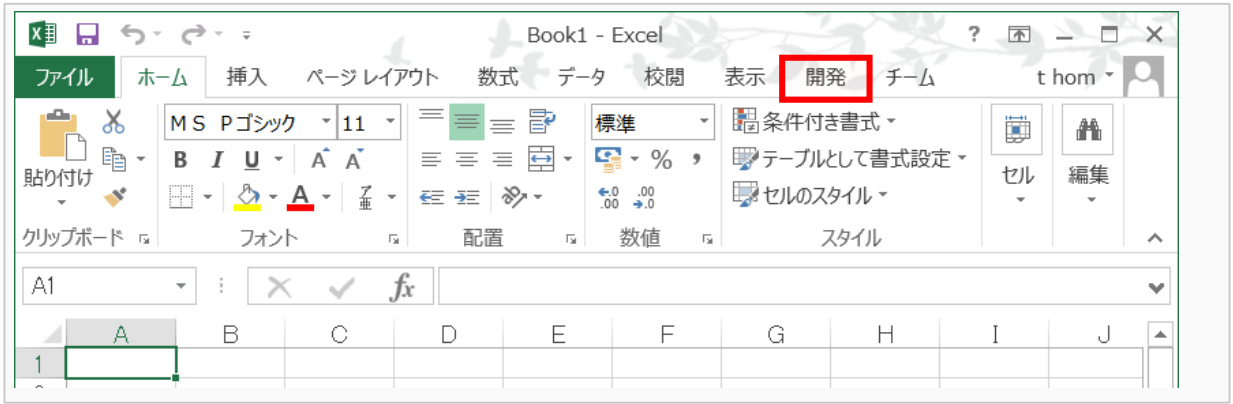
セキュリティセンターの左側のメニューからマクロの設定を開きます。右に表示されたマクロの設定より、**警告を表示して全てのマクロを無効にする**を選択してから、OK をクリックしてください。(これは、マクロを有効にする前に確認を求める警告を出すという意味です。)



Excel のオプションに戻ったら左のメニューよりリボンのユーザー設定を開き、右側のメインタブにある開発にチェックを入れて OK をクリックします。

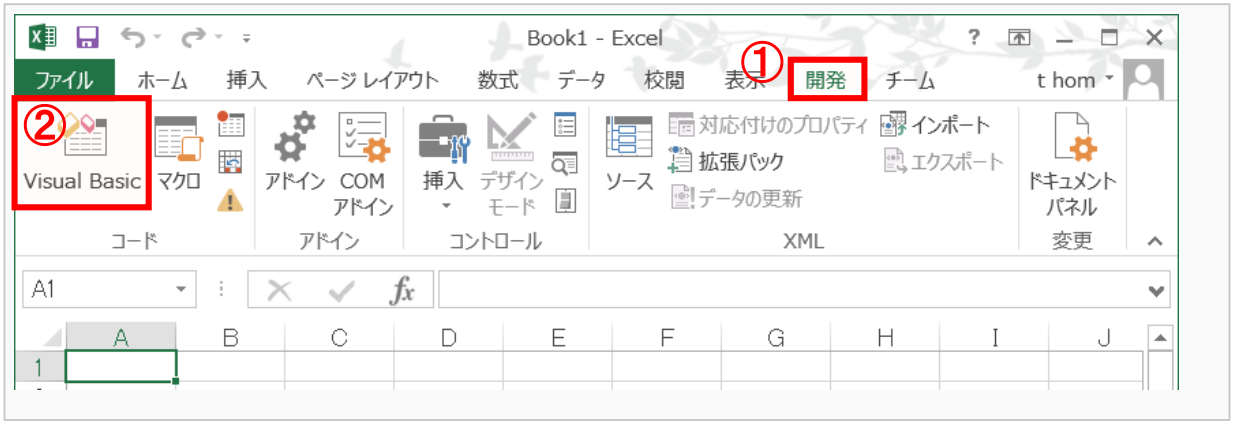


リボンに開発タブが追加されました。これで準備は完了です。



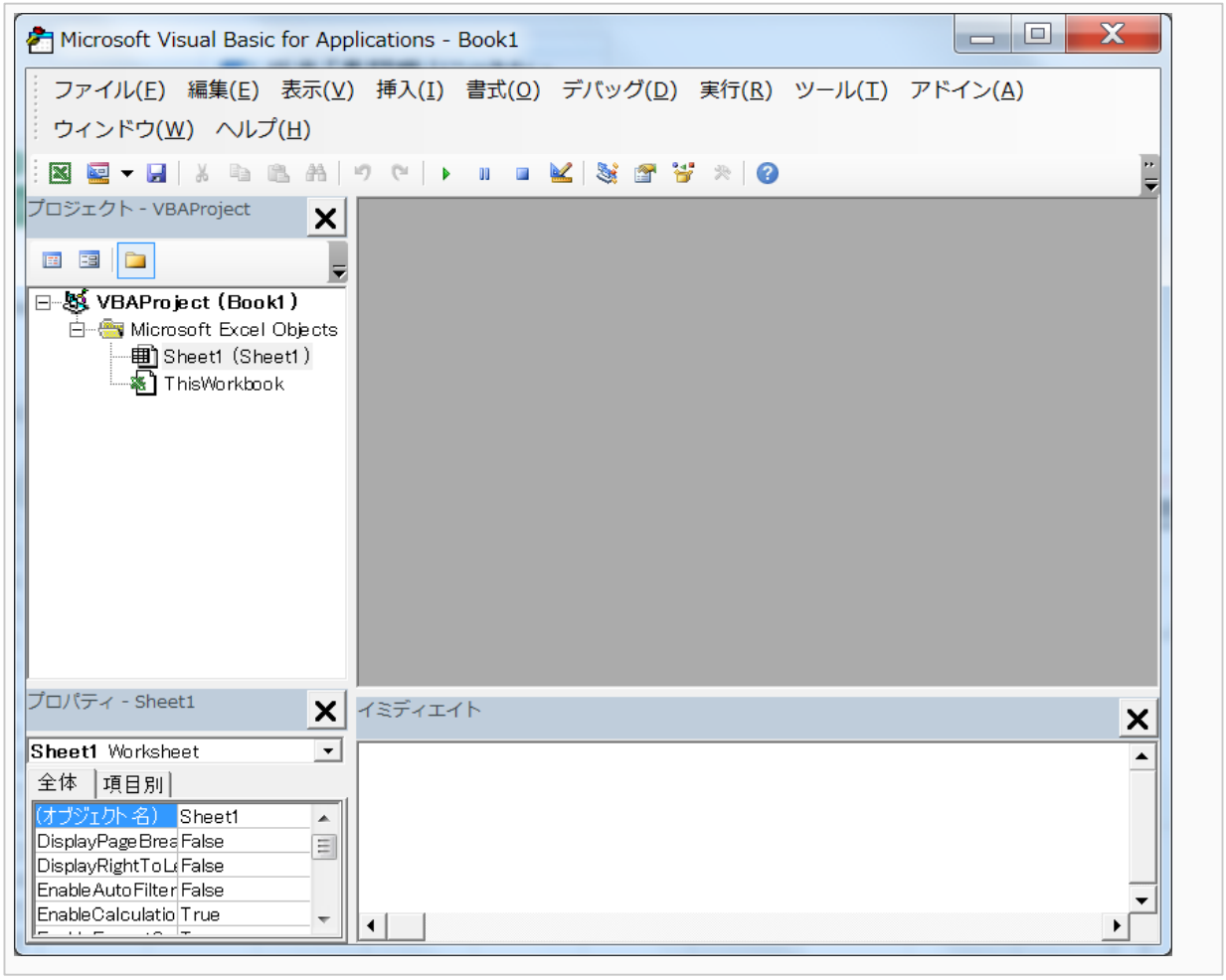
最初のプログラミング

先ほど追加した開発タブを開き、Visual Basic をクリックしてください。

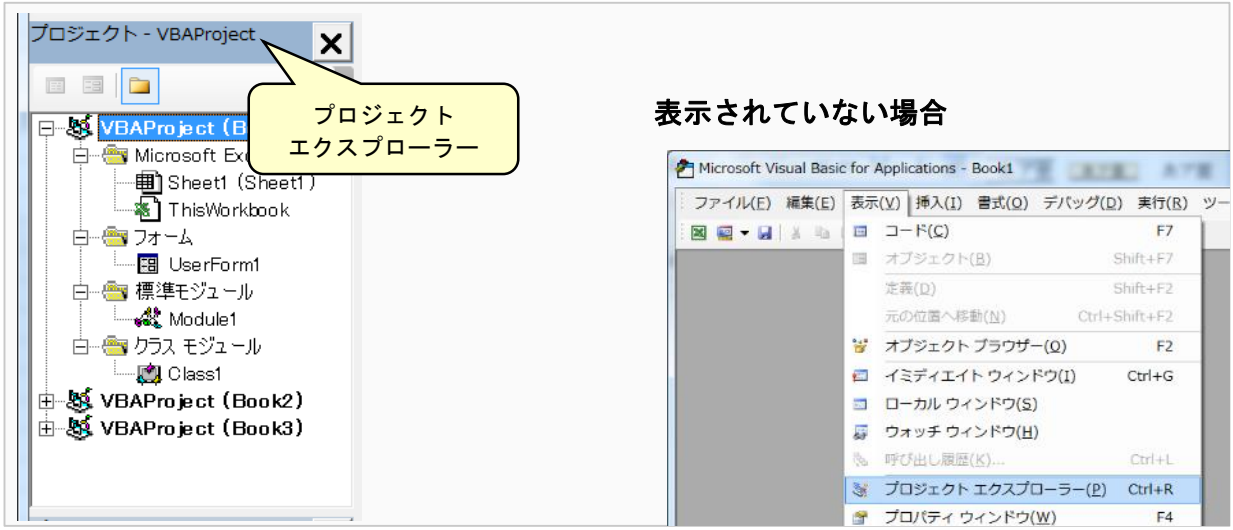


すると、次のような画面が起動します。

この画面は VBE (Visual Basic Editor)と呼ばれる VBA の開発画面です。



初期設定で左上に表示される次のウインドウを**プロジェクトエクスプローラー**と呼びます。表示されていない場合は、表示メニューから表示させてください。他のウインドウは必要になった時に紹介しますので、今は気にしないでください。



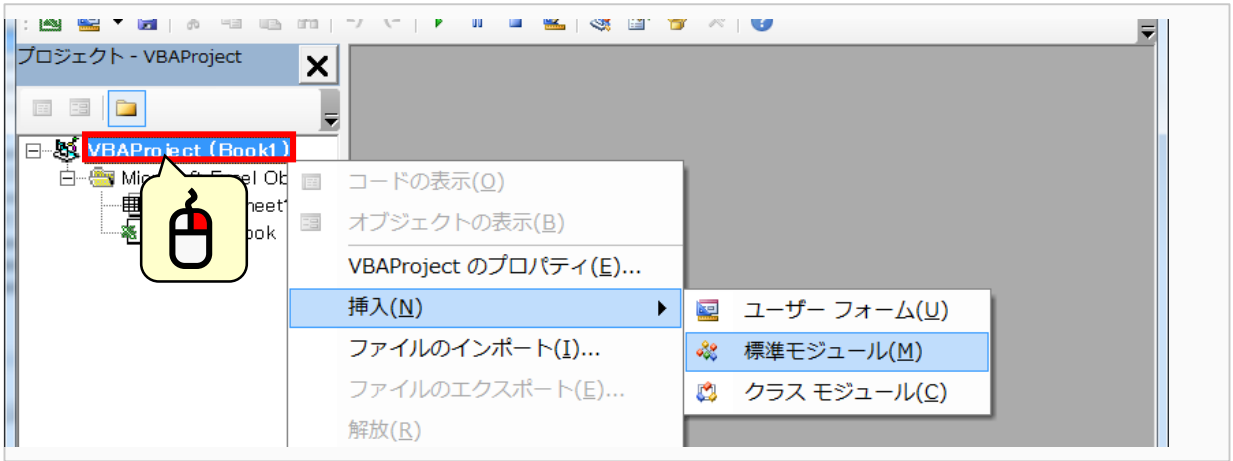
前ページのスクリーンショットでは VBAProject が一つだけ表示されていましたが、ワークブックを複数開いている場合はこのように開いているブックの数だけ VBAProject が表示されます。

フォーム・標準モジュール・クラスモジュールなどのフォルダは最初は表示されていませんが、それぞれモジュールを追加すると表示されるようになります。

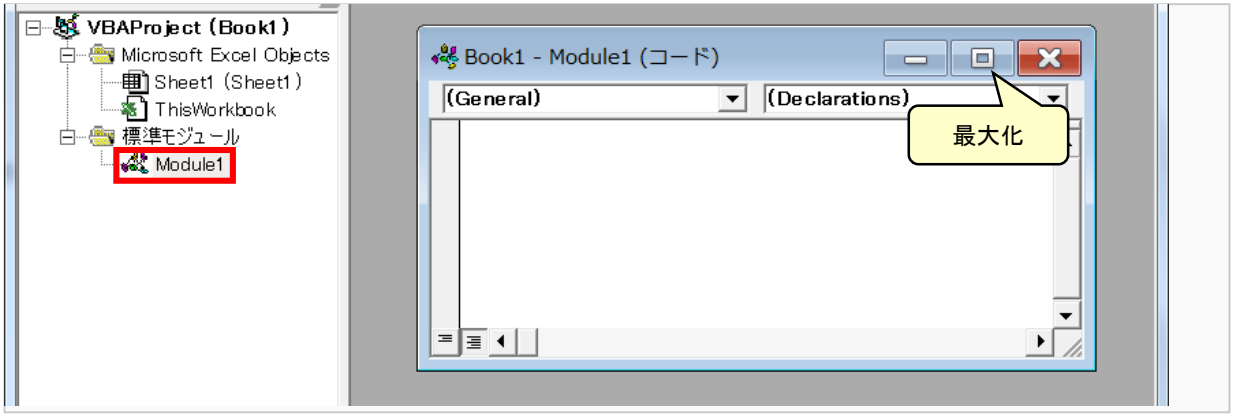
マクロは Microsoft Excel Objects、フォーム、標準モジュール、クラスモジュールのどこでも記述することが出来ますが、書く位置によって動作が異なります。**マクロは標準モジュールに記載するのが一般的**ですので、他の項目は今は気にしないでください。

では、早速**標準モジュール**を追加してみましょう。

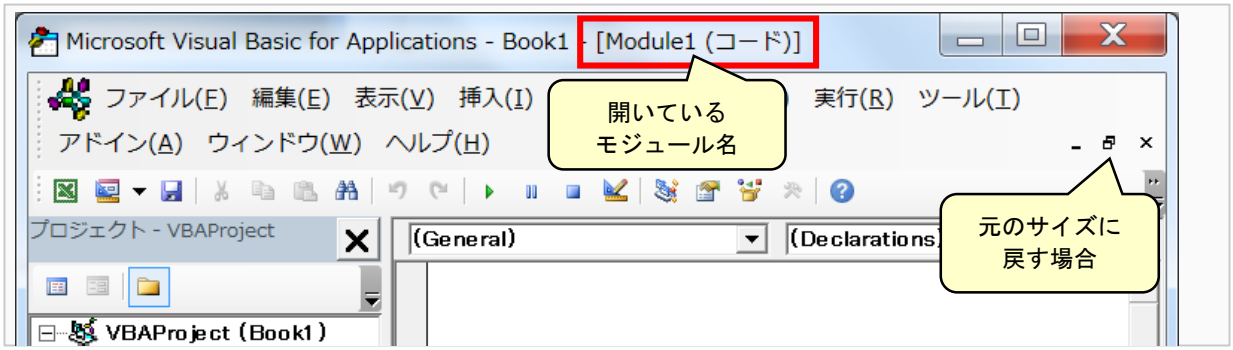
プロジェクトエクスプローラーから VBAProject (Book1)を**右クリック**し、メニューから挿入→標準モジュールをクリックします。



Module1 が追加され、コード画面が表示されます。次のようにコード画面がウインドウで表示された場合は、最大化しておいてください。(はじめから最大化されている場合もあります。)



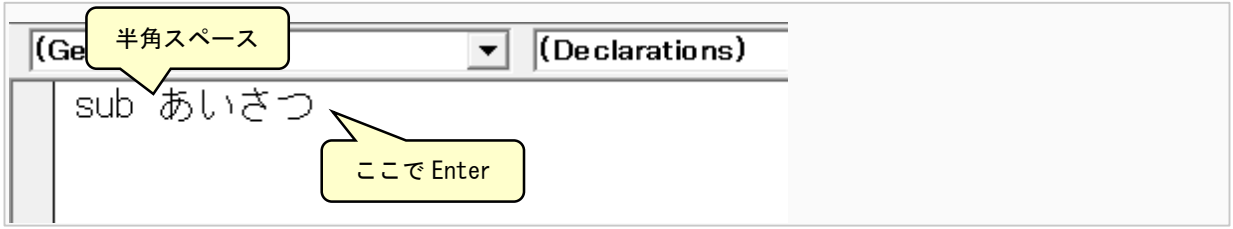
最大化された場合は、今開いているコード画面の名前は VBA のタイトルバーに表示されます。元のサイズに戻したい場合は右上の小さなボタンで戻すことができます。



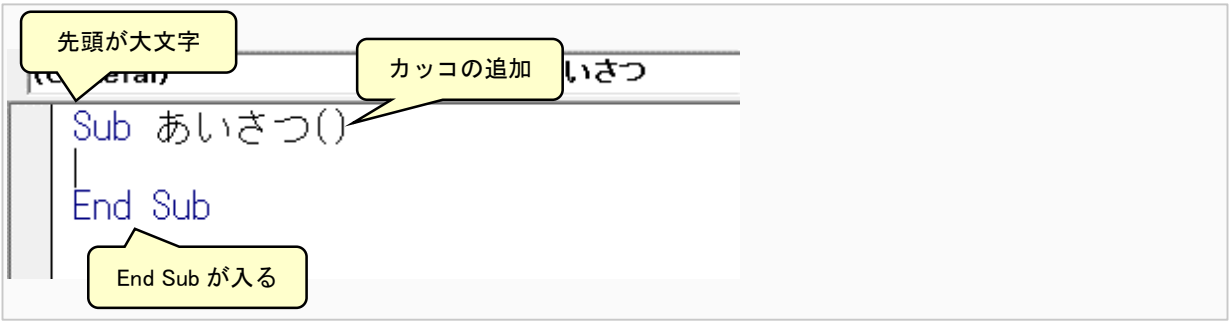
では早速 VBA コードを書いてみましょう。

Module1 のコード画面に次のコードを入力して Enter キーで改行してください。

```
sub あいさつ
```



すると、次のように自動で体裁が整えられます。

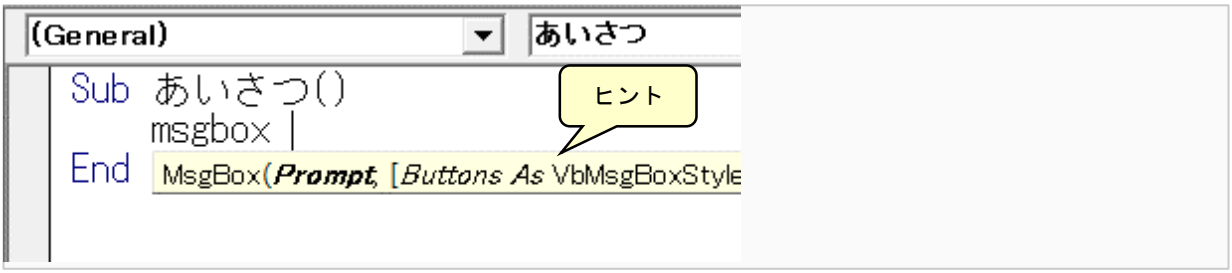


この Sub から End Sub までの間が、ひとつのマクロになります。Sub の後に書いた「あいさつ」がマクロ名です。追加されたカッコにも重要な意味がありますが、今はただの決まり事として覚えてください。

さて、ここから中身を書いていきますが、まず Tab キーでひとつタブを入力してください。

これはコードを見やすくするためで、プログラミングでは**インデント**と呼びます。

次に半角で msgbox と入力して半角スペースを空けます。すると、次のようにヒントが表示されますが、これは中～上級者向けのヒントなので、無視して続けます。

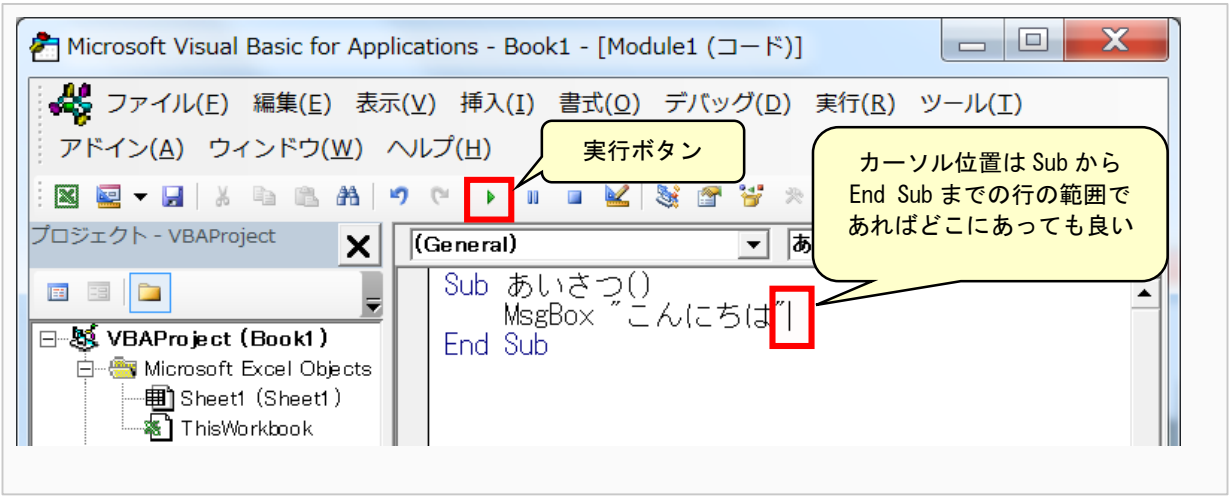


半角のダブルクォーテーション「"」を入力した後、全角で「こんにちは」と入力してください。

そのまま Enter キーで改行すると、自動で体裁が整えられます。

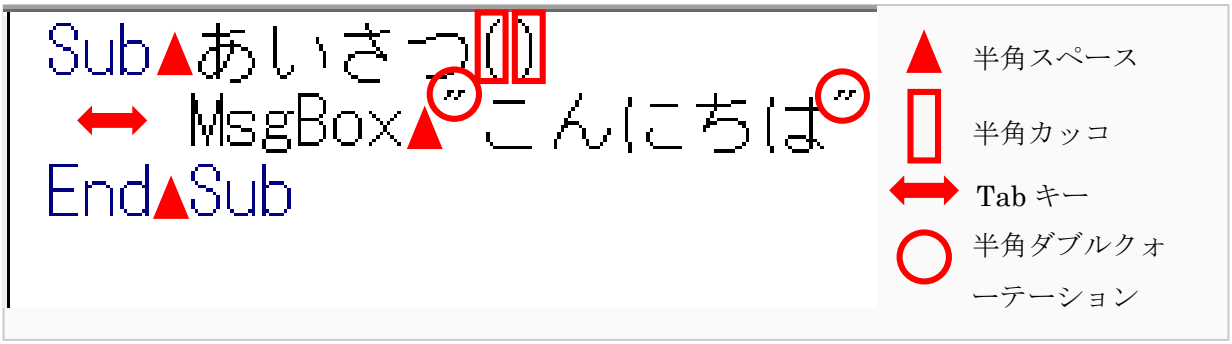
Enter キーを入力した際にできた空行は削除してください。これでコードは完成です。

コード内の任意の場所にカーソルを合わせて、実行ボタンを押します。



実行すると次のように**メッセージボックス**が表示されます。OK ボタンで閉じてください。

うまく実行できなかった場合はスペースや全角・半角などの入力を間違えているのかもしれません。次の図を見ながらもう一度コードを見直してみましょう。



VBE には入力支援機能があるため、Sub や MsgBox などは小文字で入力すると間違いに気づきやすくなります。たとえば、msgbox を間違えて mgsbox と書いてしまったら、それは M と B が大文字にならないので間違いに気づくことができます。

ダブルクォーテーションで囲まれた部分は**文字列**と呼びます。文字が並んでいるから、文字列です。聞き慣れない変な呼び方かもしれませんがよく登場するのでそのうち慣れます。

コラム　Sub の由来

Sub というのはメインとサブのサブです。プログラミングの世界では昔、メイン処理から呼ばれて動くサブプログラムをサブルーチンと呼んでいました。決まり切った単純作業のことを英語でルーチンワークと言いますが、そのルーチンです。

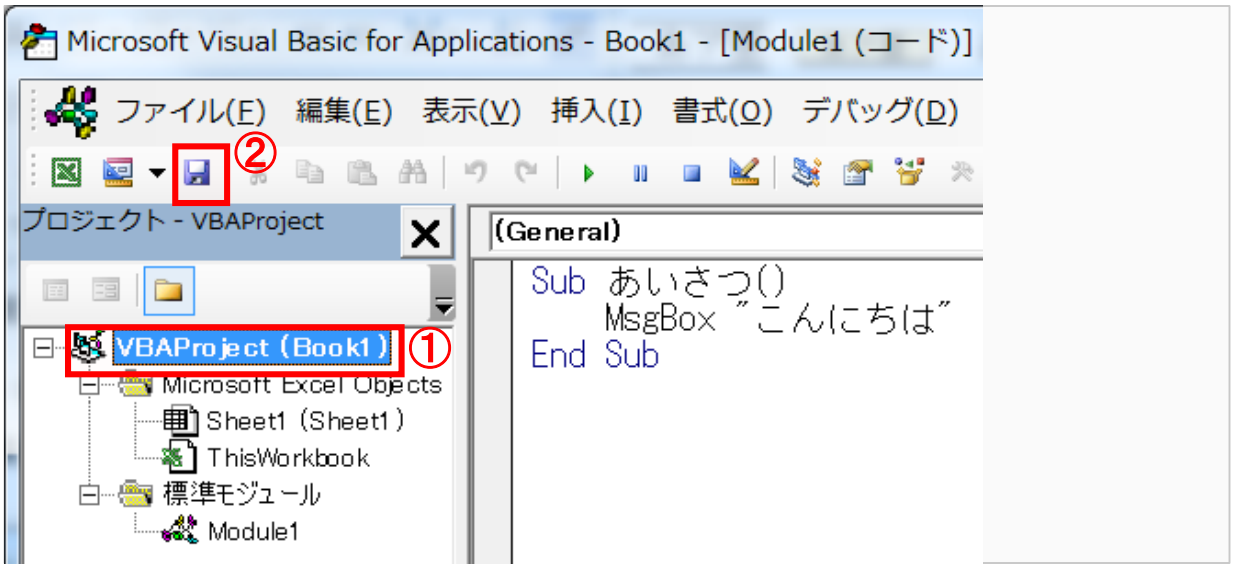
そのうちサブルーチンという言葉が単にひとかたまりの処理を指すようになり、メインから呼ばれるかどうかは関係なくなったようです。

VBA のコードにも決まったメイン処理は無く、どのマクロを実行するかはユーザーが選択するので、サブという用語には若干違和感がありますが、慣例のようなものなので深く考える必要はありません。

プログラムの保存

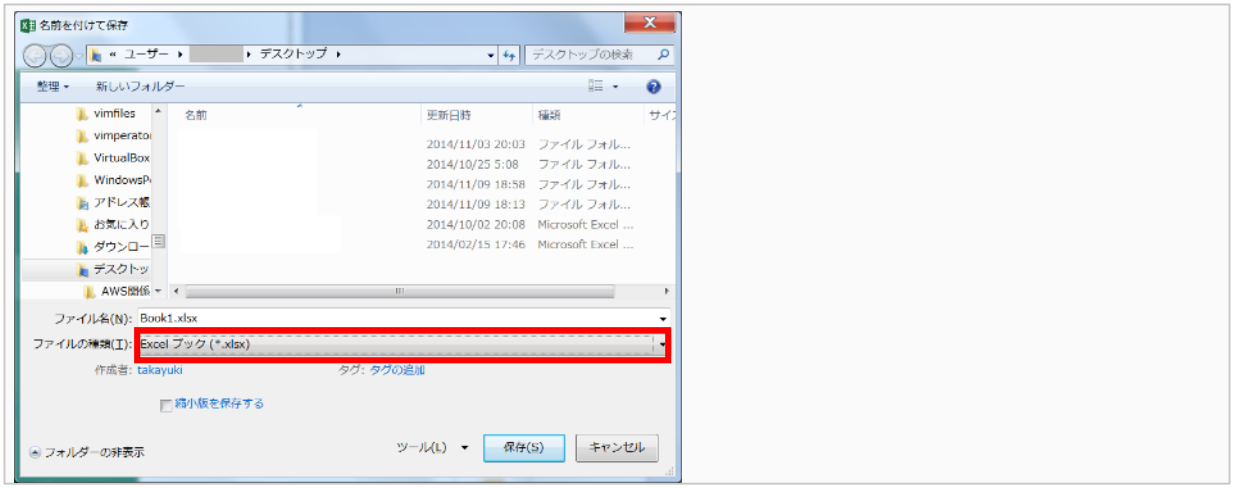
VBA コードはワークブックに保存されます。コードを保存するには、ブックをマクロ有効ブックとして保存する必要があります。そうしないとせっかく書いたプログラムが消えてしまいますので以下の手順に従って慎重に保存してください。

プロジェクトエクスプローラーから、VBAProject (Book1)をクリックして選択してから、VB エディタの保存ボタンをクリックしてください。

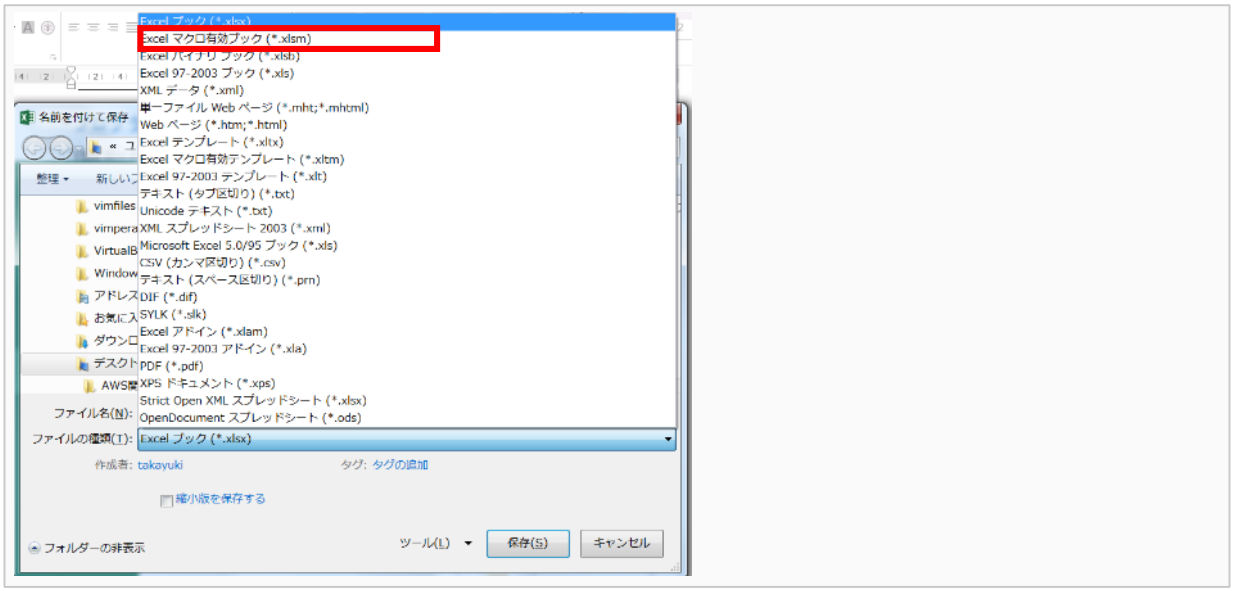


この保存ボタンは VBA コードだけでなく、ブックに対して行った変更が全て保存されますので注意してください。(要は、エクセル上で保存した場合と同じです。逆にエクセルからブックを保存してもコードは保存されます。)

名前を付けて保存が表示されたら、まずファイルの種類をクリックします。



出てきたリストから、Excel マクロ有効ブック(*.xlsm)を選択してください。

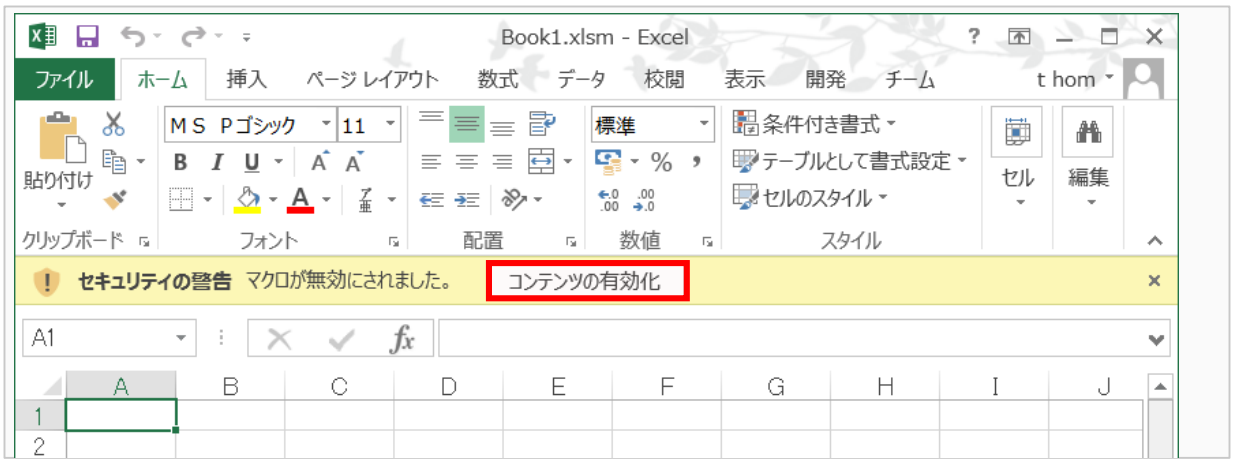


後は普通に保存すれば OK です。保存できたら一度 VB エディタと Excel を×で終了させてください。

Excel でブックを閉じる際にファイルが保存されていない場合は確認メッセージが表示されますが、VB エディタを終了させる際は何も表示されません。これは VB エディタを閉じてもコードはブックに残っているためです。VB エディタでブックを保存し忘れた場合は、必ず Excel 上でブックを保存してください。

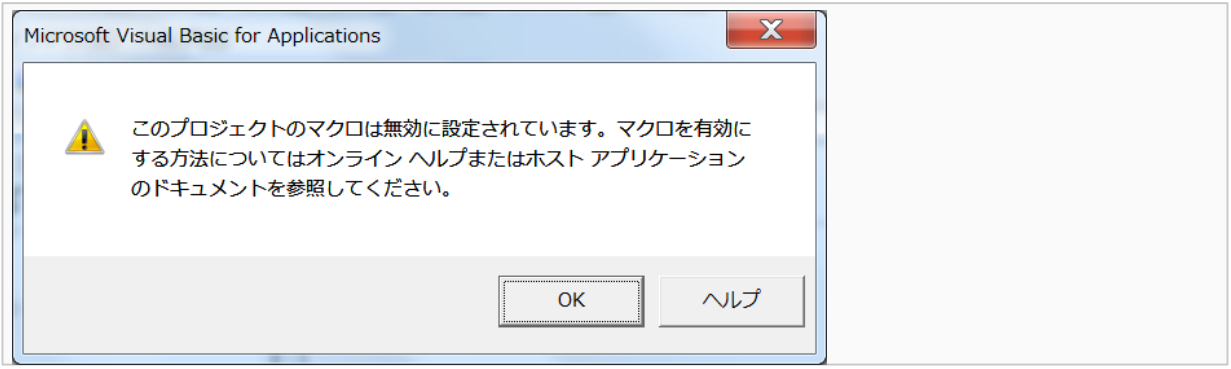
さて、では今保存したファイルを開き直してください。

セキュリティ設定が正しければ、次のような警告が出るはずですので、コンテンツの有効化をクリックしてください。



一度有効化すると次に開き直した時は警告が出なくなりますが、ファイル名を変えたり置き場所を変えるとまた警告が出ます。

コンテンツの有効化を行わずにマクロを実行しようとすると、次のメッセージが表示されます。



一度警告が消えてしまうとコンテンツの有効化が出来ないので、ブックを閉じて開き直してください。

コラム　マクロウイルス

マクロの中にはブックを開いただけで自動実行されるものがあります。正しく使えば便利な機能を実現できますが、インターネットから拾ったブックや知らない相手からメールで送られてきたブックには、悪意を持ったものが紛れ込んでいる可能性があります。そのような悪意を持ったマクロはマクロウイルスと呼ばれ、一昔前に流行っていたようです。

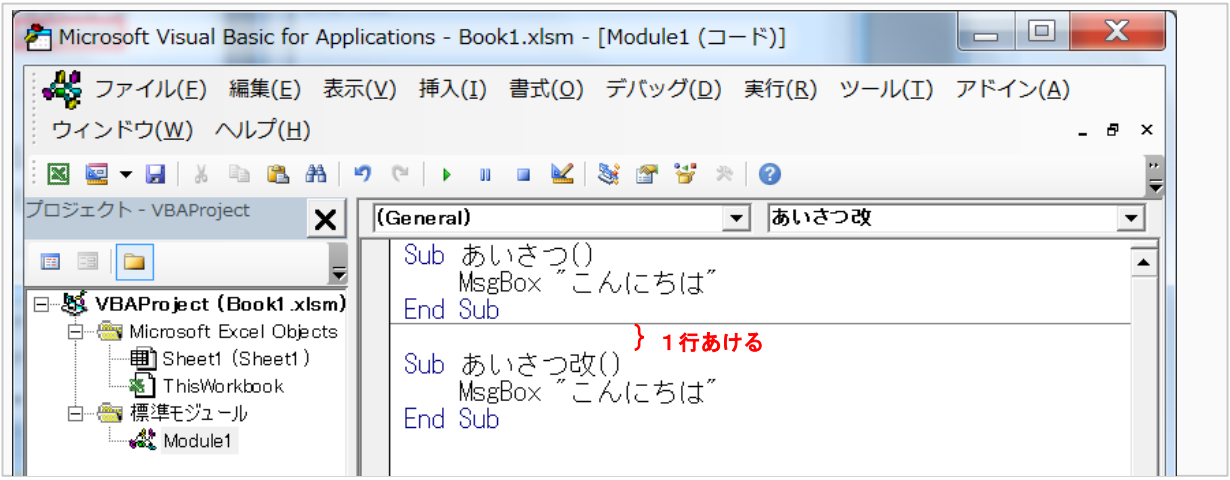
最近マクロウイルスはほとんど見かけなくなりましたが、入手経路が不明なブックを開くときは、すぐにコンテンツの有効化を行わず、不審なコード(例えば許可なくパソコン上のファイルを消すようなコード)が無いかを調べてから、改めてファイルを開き直すと良いでしょう。

条件によって処理を変える

先ほど作成したあいさつマクロですが、いつ実行しても「こんにちは」しか返しません。これを改良して朝なら「おはようございます」、夜なら「こんばんは」と返すようにしてみましょう。

開発タブから Visual Basic を開いてください。コード画面が出ていない場合は、プロジェクトエクスプローラーから Module1 をダブルクリックすることで表示できます。

さて、それではマクロを追加しましょう。ひとまず、先ほど作ったあいさつマクロをコピーして下の行に貼り付け、名前を**あいさつ改**としてください。同じ名前がひとつのモジュールに複数あると実行したときにエラーになります。マクロとマクロの間は水平線で区切られますが、１行空けた方が見やすくて良いです。

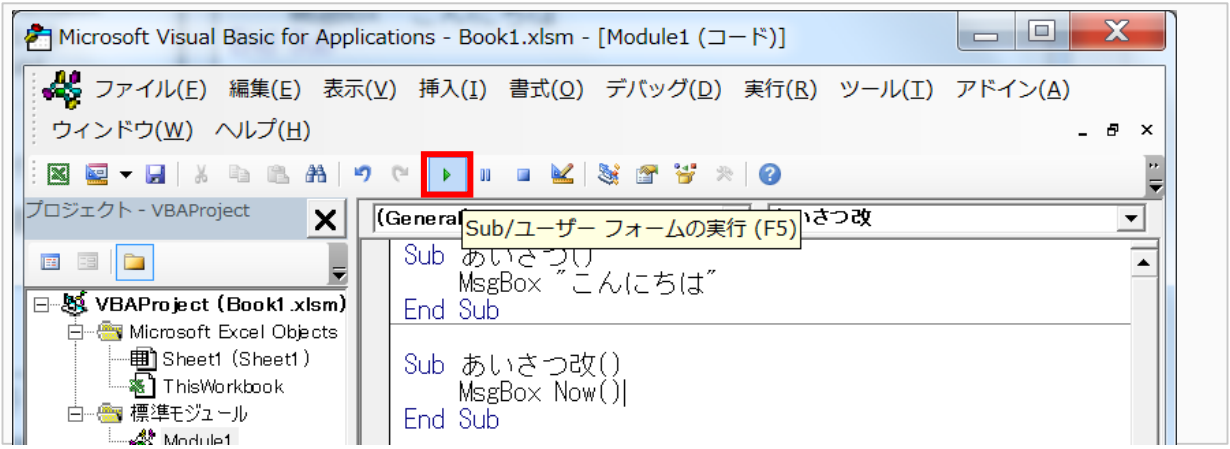


時刻によって返す挨拶を変化させたいわけですから、まずは現在時刻を取得します。現在時刻を得るには Now()と書くだけです。これをメッセージボックスに表示させて見ましょう。あいさつ改を次のように編集してください。

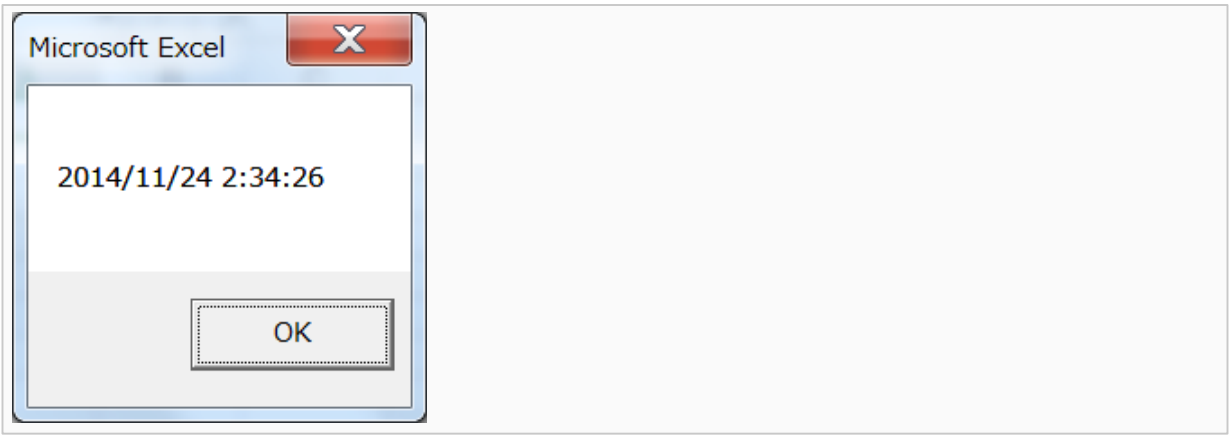
```
Sub あいさつ改()
    MsgBox Now()
End Sub
```

今度の Now()はダブルクォーテーションで囲まないので、注意してください。

では、あいさつ改にカーソルを合わせて、実行ボタンを押してください。



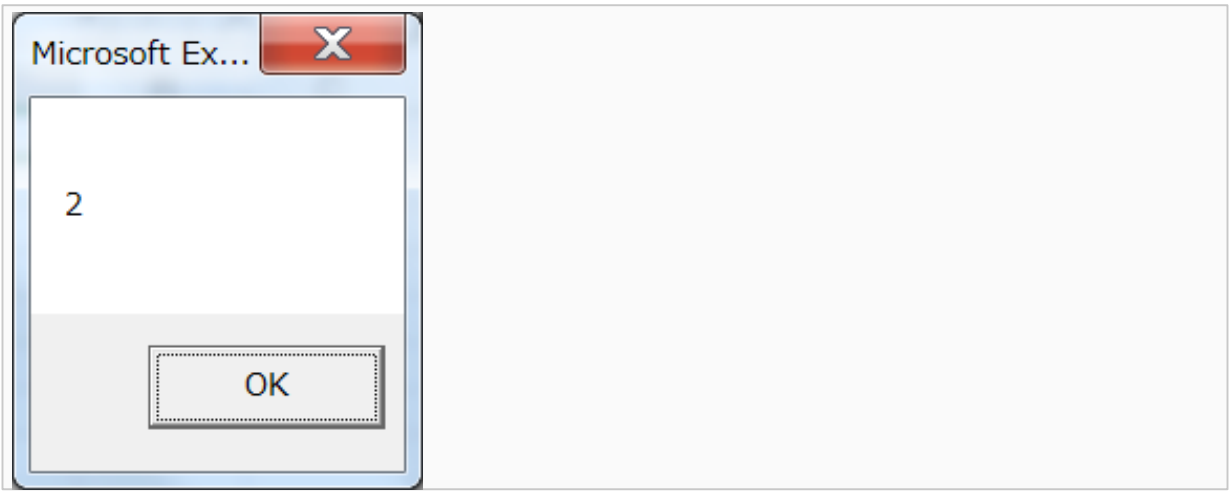
すると、次のように現在日時が表示されます。



ただし挨拶の判断に使うには時間だけ分かれば良いので、これをさらに次のように加工してみます。

```
Sub あいさつ改()
    MsgBox Hour(Now())
End Sub
```

すると、2 時 34 分 26 秒から、2 時の 2 だけを取り出すことができました。



Hour()は、日時を受け取って「時・分」のうちの「時」を返すための文です。ですから Hour のカッコのなかに現在日時を表す Now()を入れると、時刻を 0～23 の数字で返してくれます。このように、何かを受け取って、それを加工して返してくれるものを**関数**と呼びます。ちなみに、何も受け取らずに値を返す関数もあり、先ほどから使用している Now()もその一種です。

さて、時刻の取得方法が分かったところで、次はあいさつの判定です。まずは夕方 5 時以降の挨拶を「こんばんは」に変えてみます。**あいさつ改**のコードを次のように書き換えてください。

```
Sub あいさつ改()
    If Hour(Now()) >= 17 Then
        MsgBox "こんばんは"
    Else
        MsgBox "こんにちは"
    End If
End Sub
```

今度の MsgBox は、If の中身であることを示すためインデント(タブ)をひとつ増やしています。

これで昼間に実行すれば「こんにちは」、夜に実行すれば「こんばんは」と表示されるようになります。ただし、実験するのに時間を待っているとなかなか次にすすめないので、説明だけ読んで考えてみましょう。

まず次の文です。

```
    If Hour(Now()) >= 17 Then
```

If～Then というのは、英語で「もし～だったら」という意味です。真ん中の記号>=は、左右の値を比べています。例えば A>=B なら、A は B 以上であるという意味です。

Else 文は、「～でなければ」という意味です。

そして、最後の End If は条件判定の終わりを示しています。

つまり、日本語で書き表すと次のようになります。

```
もし、Hour(Now()) が 17 以上ならば
    「こんばんは」を表示する
そうでなければ
    「こんにちは」を表示する
判定終わり
```

例えば今が昼の 3 時だとしたら、Hour(Now())の数字は 15 になります。17 未満なので「こんにちは」が表示されます。

しかしこのままだと、深夜 0 時をまわったら「こんにちは」になってしまいます。朝の挨拶を追加する前に、「こんばんは」の条件を追加してみましょう。

先ほどのコードをさらに次のように書き換えてください。

```
Sub あいさつ改()
    If Hour(Now()) >= 17 Or Hour(Now()) <= 3 Then
        MsgBox "こんばんは"
    Else
        MsgBox "こんにちは"
    End If
End Sub
```

If 文に Or という単語が増えています。

```
    If Hour(Now()) >= 17 Or Hour(Now()) <= 3 Then
```

これは、「または」という意味で、左右にあるどちらかの条件が成立したら OK とします。また日本語で書いて見ましょう。

```
もし、Hour(Now())が 17 以上または、Hour(Now())が 3 以下だったら、
```

となります。これで 17 時 00 分から翌日の 3 時 59 分までは「こんばんは」と表示されるようになりました。次に朝の挨拶を加えてみましょう。コードを次のように変更してください。

```
Sub あいさつ改()
    If Hour(Now()) >= 17 Or Hour(Now()) <= 3 Then
        MsgBox "こんばんは"
    ElseIf Hour(Now()) >= 4 And Hour(Now()) <= 10 Then
        MsgBox "おはようございます"
    Else
        MsgBox "こんにちは"
    End If
End Sub
```

ElseIf～Then という文が増えました。また、And という単語も登場しています。

ElseIf～Then は「そうでなくて、もし～だったら」という意味です。And は「かつ」という意味です。

全体を日本語に置き換えると、次のようになります。

```
もし Hour(Now())が 17 以上または Hour(Now())が 3 以下ならば、
    「こんばんは」と表示する
そうではなく、もし Hour(Now())が 4 以上かつ、Hour(Now())が 10 以下ならば
    「おはようございます」と表示する
それ以外の場合は
    「こんにちは」と表示する
判定終わり
```

条件判定の記号は<=、>=以外にも次のようなものがあります。

記号	意味
A < B	A は B より小さい
A > B	A は B より大きい
A = B	A と B は同じである
A <> B	A と B は異なる

条件が不一致の際に何も処理をしない場合、下記コード例のように Else 文は省略できます。

```
If A = B Then
    MsgBox "一致しました"
End If
```

また、条件判断の方法は If 文の他に Select 文というものがあります。If 文だけ覚えれば条件判定は事足りますが、Select 文を使いこなせば次のようにスマートに記述することも出来ます。

```
Sub スマートあいさつ()
    Select Case Hour(Now())
        Case Is >= 17, Is <= 3
            MsgBox "こんばんは"
        Case 4 To 10
            MsgBox "おはようございます"
        Case Else
            MsgBox "こんにちは"
    End Select
End Sub
```

Select 文については紹介にとどめますので、詳しい使い方は書籍やインターネットで調べてみてください。

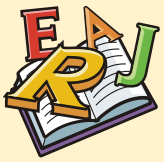
コラム　プログラミングと英語

If、Then、Else などプログラミングではいくつか英単語が出てきますが、プログラミングに英語力はあまり関係ありませんので苦手な方は安心してください。

ただ単語本来の意味が分かっている方が多少プログラムも理解しやすくなりますし、分からない単語はインターネットなどで調べてみるのも良いかもしれません。よく使う単語だけ数えてもせいぜい 50 個程度のものです。

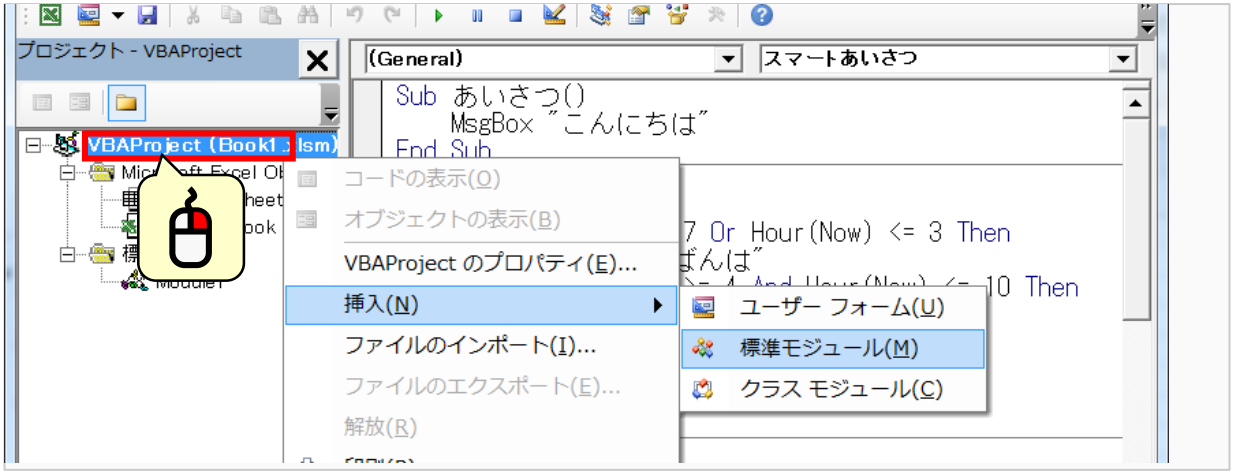
プログラミングにも文法がありますが、英語のようにバリュエーションがあるわけではなく、パターンを一つ覚えてしまえば後はその組み合わせですので、文法そのものは英語よりはるかに簡単にマスターできてしまいます。

あとは覚えたパターンでロジックをどう組み立てるか。そこがプログラミングの難しさでもあり、面白さでもあります。

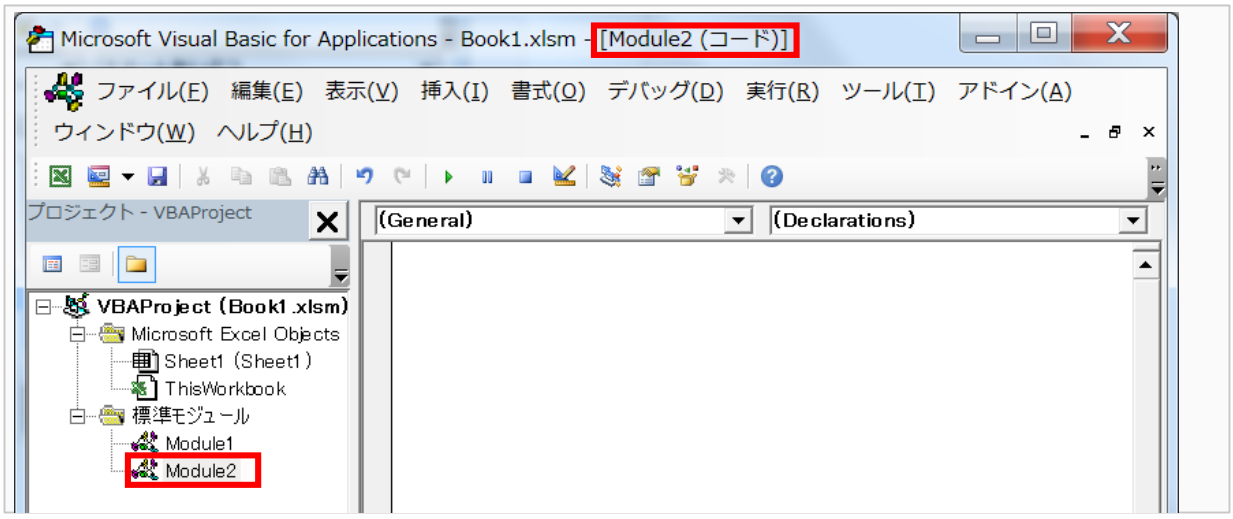


モジュールについて

さて、次のマクロ作成に取りかかる前に、今回はモジュールを一つ追加します。プロジェクトエクスプローラーから VBAProject を右クリックし、挿入→標準モジュールと選択してください。



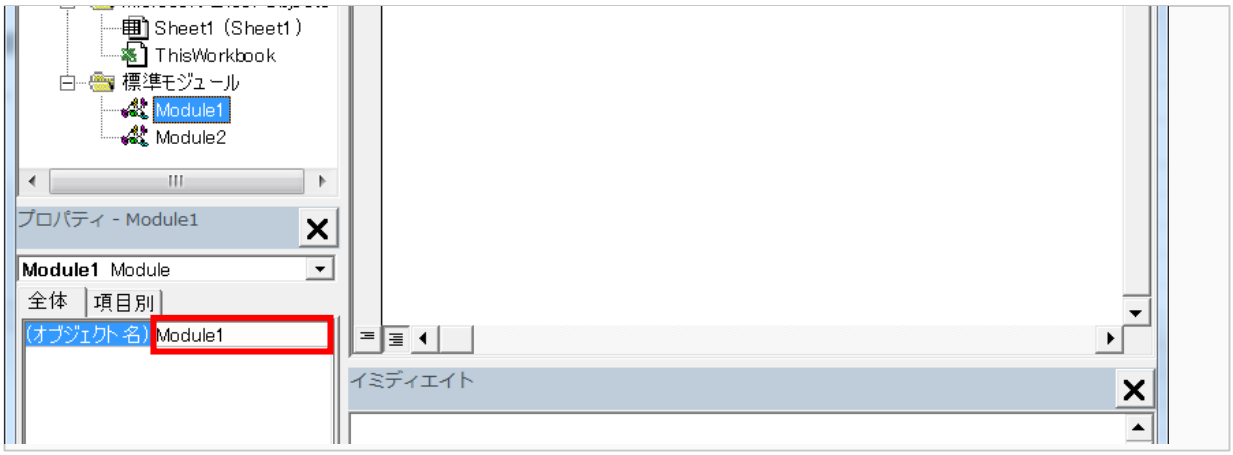
すると、Module2 が追加され、白紙のコード画面が出てきます。タイトルバーで、Module2(コード)が開かれていることを確認してください。



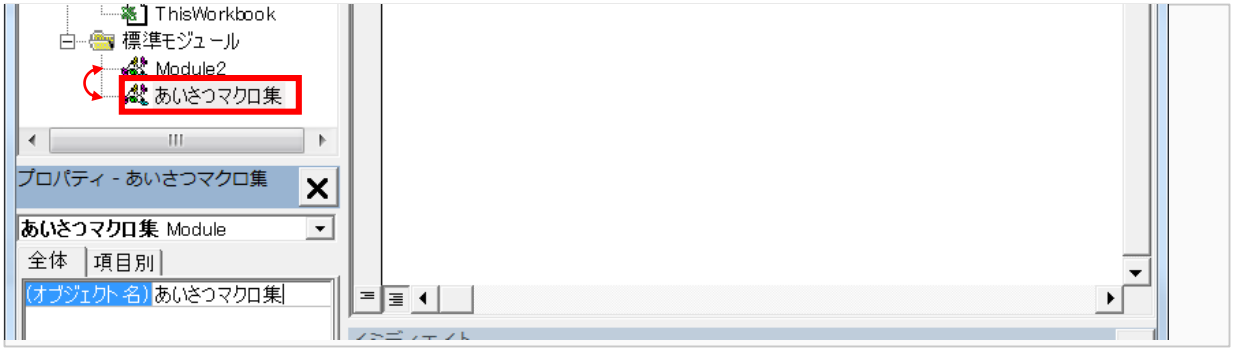
モジュールとは、プログラムにおけるひとかたまりの部品のことです。VBA では、モジュール単位で他のブックにコピーや移動ができます。先ほどの例では**あいさつマクロ**と**あいさつ改マクロ**をひとつのモジュールに含めましたが、何でもかんでも詰め込むと管理がややこしくなりますので、関連性の高いマクロを中心にまとめてください。

また、モジュールには分かりやすい名前を付けることができます。それでは、Module1 の名前をあいさつマクロ集に変更してみましょう。

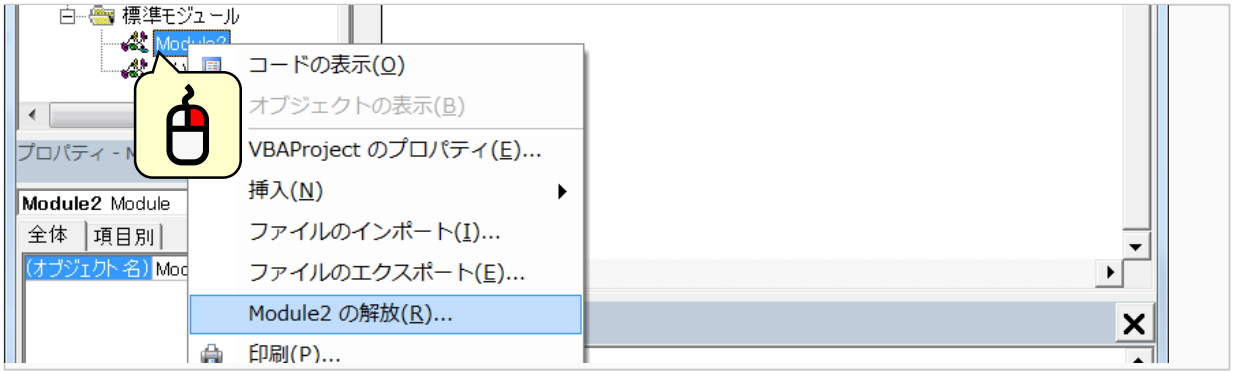
プロジェクトエクスプローラーで Module1 を選択すると、その下のプロパティウインドウに Module1 のオブジェクト名が表示されます。これを編集してください。



オブジェクト名を編集して Enter を押すと、このようにモジュール名が変更されます。モジュール名の昇順に並び変わるため、Module2 と位置が逆転します。



Module の削除方法も覚えてください。プロジェクトエクスプローラーから右クリックして解放を選択すると削除できます。また、ファイルのエクスポートでモジュールを外部ファイルに書き出すことができ、インポートでそれを読み込ませることができます。

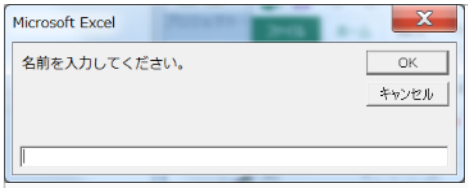


テキスト入力を受け付ける

さて、では新しく追加したモジュールに次のコードを入力し、実行してみてください。

```
Sub あなたの名前は()
    x = InputBox("名前を入力してください。")
    MsgBox "こんにちは、" & x & "さん"
End Sub
```

実行すると次のような入力ボックスが表示されますので、あなたの名前を入れ OK ボタンを押してください。



すると、入力した名前に対してあいさつが返されます。

それではまたコードを確認していきます。まず次の部分です。

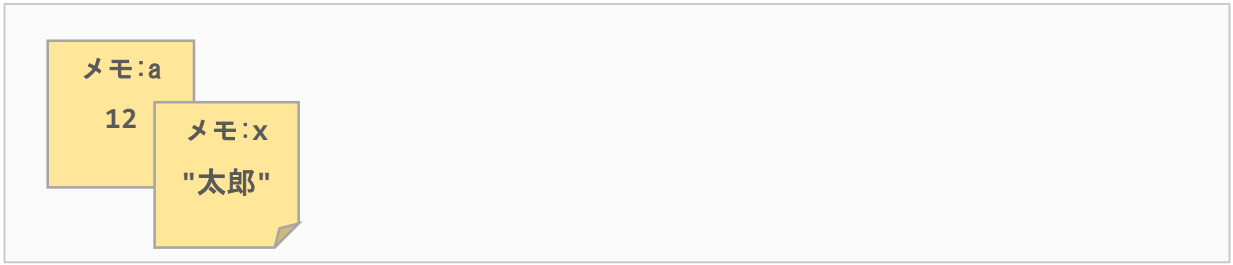
```
    x = InputBox("名前を入力してください。")
```

まず InputBox は関数です。メッセージを渡すと、テキストボックス付きのウインドウを表示し、OK ボタンが押されたら入力された値を返します。

太郎と入力すると InputBox は"太郎"を返しますので、先ほどの文は次のように解釈されます。

```
    x = "太郎"
```

ここで、x は値を保存するメモの名前だと思ってください。プログラミングではこれを**変数**と呼びます。変数には名前を付けることができ、数字や文字列などの値を書き込んだり参照したりできます。



上の 1 行は、x というメモに"太郎"という値を書き込むイメージです。この操作をプログラミングでは「変数:x に値:"太郎"を**代入**する」と言います。

If 文でもイコールが登場しますが、あちらは比較のイコールなので別物だと覚えておきましょう。

変数の名前は x でも y でも YourName でも自分で決めることができます。ただしいくつかの用語は VBA によって使用されており、変数名に使用できません。例えば、Sub や If、Then、MsgBox などプログラムの文法や関数名として使われている用語を変数名にしようとするとエラーになります。

変数名には日本語も使えます。試しにプログラムを次のように書き換えてみてください。

```
Sub あなたの名前は()
    名前 = InputBox("名前を入力してください。")
    MsgBox "こんにちは、" & 名前 & "さん"
End Sub
```

これでも先ほどと同じように動作します。

さて、次はメッセージボックスです。こちらも今までと違う記号が出てきます。

```
    MsgBox "こんにちは、" & x & "さん"
```

"こんにちは" と x と "さん"が&記号で繋がれています。

&記号は文字列同士を結合する際に使用します。例えば次の 2 つのマクロは全く同じメッセージが表示されます。

```
Sub あいさつ1()
    MsgBox "こんにちは"
End Sub

Sub あいさつ2()
    MsgBox "こん" & "にちは"
End Sub
```

メッセージの途中に変数の値を入れたいときは、前述のように&で繋いでやる必要があります。

例えば、"こんにちは x さん"と書いた場合はダブルクォーテーションで囲まれた範囲がただの文字列と判定され、そのまま「こんにちは、x さん」と表示されてしまいます。

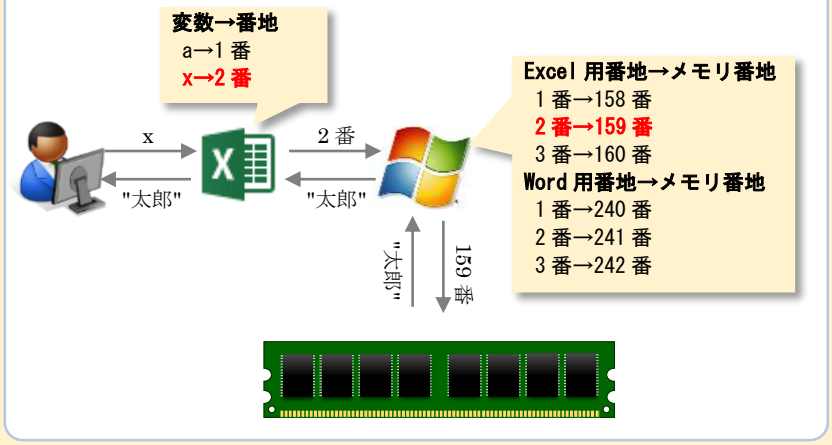
変数は文字列の外に出して&で繋ぐということを覚えておきましょう。

コラム　変数の実態

変数の実態はメモリ上のデータです。

メモリは番地で管理されていますが、さらに OS によってアプリケーションごとに固有の番地に変換されています。そうしないと、もし Excel と Word が同じ番地を指定したらお互いに上書きしあって保存した値がデータが消えてしまうためです。

X を参照する際の実際の動作イメージ



プログラミングを覚えるために必ずしもこういった知識が必要というわけではありませんが、内部の動作も知っておくと理解が深まります。

コラム　プログラミングと数学

プログラミングには数学用語がよく出てきます。変数・代入・関数などの用語は数学由来ですが、プログラミング用語として使う分には数学に精通している必要はありません。

小難しい言葉がでてきますが、やっていることは至極シンプルなので実際に使っているうちにだんだん分かるようになってきます。

以下はプログラミングで使用される代表的な数学用語です。

変数・・・中身が変わるメモ
定数・・・中身が変わらないメモ
　値　・・・文字列や数値など
代入・・・メモに値を書き込むこと
　　すでに値が書き込まれた変数に新しい値を代入すると、古い値は破棄される
関数・・・何かを渡すと、加工した値を返してくれるもの。
　　何も受け取らない関数もある。
演算・・・計算のこと
演算子・・計算につかう記号(+ - * /)
比較演算・2 つの値を比べること

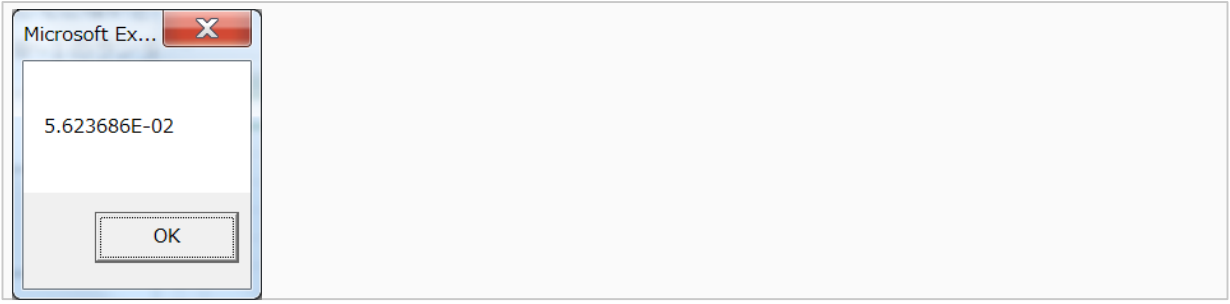
文字入力と If 文の応用

では次に今までに学習した内容の簡単な応用として、じゃんけんゲームを作ります。これはプログラミング初級の題材としてよく出てきます。

さて、じゃんけんですから、まず相手がランダムな手を出してこないとゲームになりません。VBA にはランダムな数を作る関数 Rnd()がありますが、これは 0～1 のランダムな小数を発生させます。試しに次のコードを入力して何度か実行してみてください。

```
Sub じゃんけん()
    MsgBox Rnd()
End Sub
```

たまに次のような数が出て 1 より大きいじゃないかと思われるかもしれませんが、これは小数の特殊な表記方で、これも 1 未満の数です。よく見ると後ろに E-02 といった見慣れない符号がついているのが分かります。これは覚える必要はありません。



さて、このままでは使い勝手が悪いので改良していきましょう。まず 10 倍してみます。

```
Sub じゃんけん()
    MsgBox Rnd() * 10
End Sub
```

何度か実行してみると、整数部に 0～9 の数字がランダムで取れることが分かると思います。

ちなみに「*」は掛け算の記号です。半角文字に「×」が無いためコンピュータではこの記号が使われます。また、割り算の「÷」も同様に「/」で代用されます。

次に、以下のようにしてみましょう。

```
Sub じゃんけん()
    MsgBox Int(Rnd() * 10)
End Sub
```

Int 関数は、与えられた少数から、少数部を捨てて整数部だけ返します。これで 0～9 の整数が取れるようになりました。ただし、じゃんけんですから数字は 3 つで十分です。次のように書き換えてください。

```
Sub じゃんけん()
    MsgBox Int(Rnd() * 10) Mod 3
End Sub
```

すると、何度実行しても 0、1、2 の 3 種類しか出なくなりました。

この Mod というのは、割り算の余りを求める計算の記号です。どんな整数でも、3 で割った余りは 0 か 1 か 2 の 3 種類になるので、これを応用したわけです。ちなみに 4 で割れば 0,1,2,3 の 4 種類、5 で割れば 5 種類の余りが出ますので、任意のランダムな数が欲しいときに使えます。

さて、ではこれを一旦ランダムという名前の変数に入れて、If 文を使って敵の手を作りましょう。

```
Sub じゃんけん()
    '敵の手を決定
    ランダム = Int(Rnd() * 10) Mod 3
    If ランダム = 0 Then
        敵 = "グー"
    ElseIf ランダム = 1 Then
        敵 = "パー"
    Else
        敵 = "チョキ"
    End If
    MsgBox 敵
End Sub
```

これで、ランダムにグー・チョキ・パーを出すようになりました。2 行目に書いた次の文は、コメントといって、プログラムの内容を説明するためのものです。

```
    '敵の手を決定
```

半角のシングルクォーテーション「'」以降はコメントとして扱われ、自由に説明などを書くことができます。次のように命令の後ろにコメントをつけることも出来ます。

```
    MsgBox "こんにちは"  'あいさつ
```

また、次のように、コードの頭に「'」をつけるとコメントとみなされてその行は実行されなくなります。これは開発中に一旦メッセージを無効にしたいが消さずに置いておきたいなどのケースで使われるテクニックです。プログラミング用語では「コメントアウトする」といいます。

```
    'MsgBox "こんにちは"
```

では、プレイヤーの手を決定する部分と勝敗判定を作ります。じゃんけんマクロを次のように変更してください。

```
Sub じゃんけん()
    ’敵の手を決定
    ランダム = Int(Rnd() * 10) Mod 3
    If ランダム = 0 Then
        敵 = "グー"
    ElseIf ランダム = 1 Then
        敵 = "パー"
    Else
        敵 = "チョキ"
    End If

    'プレイヤーの手を決定
    自 = InputBox("グー,チョキ,パーのどれかを入力してください。")

    '勝敗判定
    If 自 = 敵 Then
        MsgBox "あいこです。"
    ElseIf 自 = "グー" And 敵 = "チョキ" Then
        MsgBox "あなたの勝ちです。"
    ElseIf 自 = "チョキ" And 敵 = "パー" Then
        MsgBox "あなたの勝ちです。"
    ElseIf 自 = "パー" And 敵 = "グー" Then
        MsgBox "あなたの勝ちです。"
    Else
        MsgBox "あなたの負けです。"
    End If
End Sub
```

ここまで読み進められた方なら、このコードが何をやっているか分るようになっているかと思います。日が開いて忘れてしまったり、今ひとつ理解できなかった場合はページを戻ってもう一度読み返してみてください。

また、プログラミング初心者のうちは、コードはなるべくコピー&ペーストせず、手入力するようにしてください。手で入力することでプログラムの流れが記憶に残りやすくなり、コードの全体像が頭に入れば今まで分らなかった部分の理解も進みます。

コラム　日本語の変数

プログラマーの間では、マクロ名や変数は必ず英語でなければならないとする主張が多数派です。お客様に納品する商品に使用するコードは、一般的な英語の変数名をつけるべきです。いつ下請けに保守をお願いするかも分かりませんし、外国人の労働者が担当するケースだってあるかもしれません。

ただし、私は本格的な開発でなければ日本語変数を使っても良いと思います。

自分がやっている業務をちょっと効率化するだけのコードなら素人っぽくても正しく動けば問題ありません。まして入門者はただでさえ慣れない文法と格闘しているときに、変数までわざわざ英語で書く必要はありません。日本語変数に抵抗がなければ積極的に活用してみてください。

変数名はできるだけ他人が読んでも分かるものをつけるのが望ましいですが、変数名をどうするか悩んでいると思考がそこで途切れてしまいますので、私は一度 x とか y など適当な変数名をつけておいて、後から意味の分かる内容に書き換えています。

同じ処理を繰り返す

さて、先ほど作ったじゃんけんマクロですが、このままではあいこが出ても終わってしまいます。あいこの場合は終了させずにもう一度勝負させるように変更してみましょう。

そのためには繰り返し処理を学ぶ必要があります。まずは単純なプログラムで説明しますので、じゃんけんマクロの下に次のコードを記述してみてください。

おっとその前にファイルを保存しておいてください。繰り返し処理を覚えたてのころは、コードを誤って無限ループを作ってしまうことがあります。場合によっては Excel を強制終了させるしか無くなるので、そうなるとこれまでの編集が保存できなくなります。

保存できたら以下を入力して実行してください。

```
Sub 繰り返し()
    Do
        なにか = InputBox("なにか入力してください。")
        '0 を入力で終了
        MsgBox なにか & "が入力されました。"
    Loop While なにか <> 0
End Sub
```

文字の入力を求められ、0(ゼロ)を入力するまで繰り返します。

これは Do～Loop 文といい、条件が一致する間、Do から Loop までを繰り返します。While は英語で「～のあいだ」という意味です。記号<>は条件判定の説明で登場したものとおなじで、不一致を表します。

したがって上の文は、変数:なにか が、0 ではないあいだ繰り返すという意味になります。

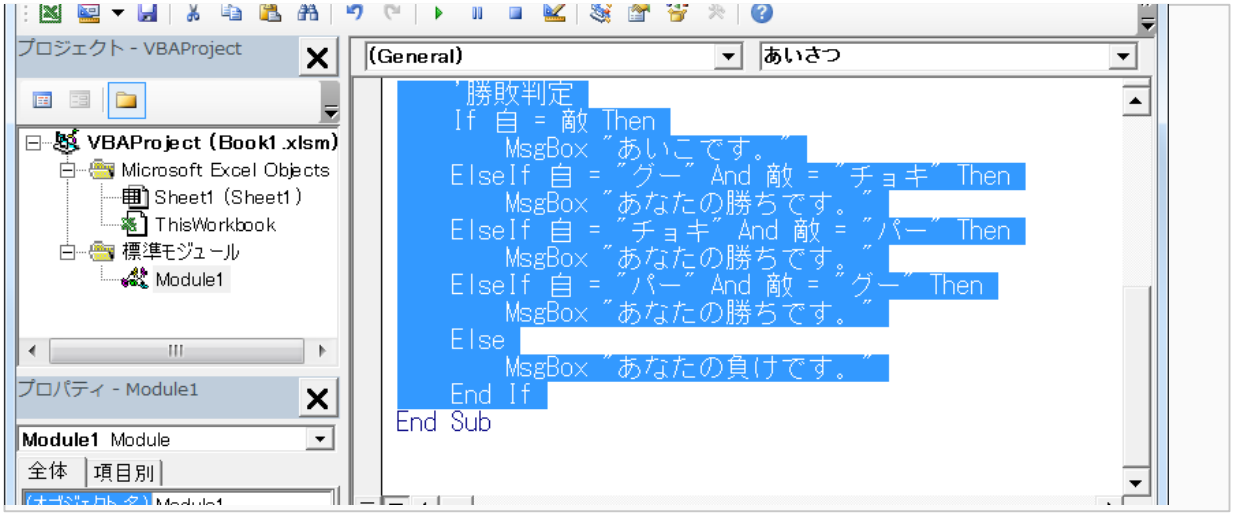
Do～Loop 文の While は、次のように Do 側に書くことも出来ます。

```
Sub 繰り返し()
    なにか = ""
    Do While なにか <> 0
        なにか = InputBox("なにか入力してください。")
        '0 を入力で終了
        MsgBox なにか & "が入力されました。"
    Loop
End Sub
```

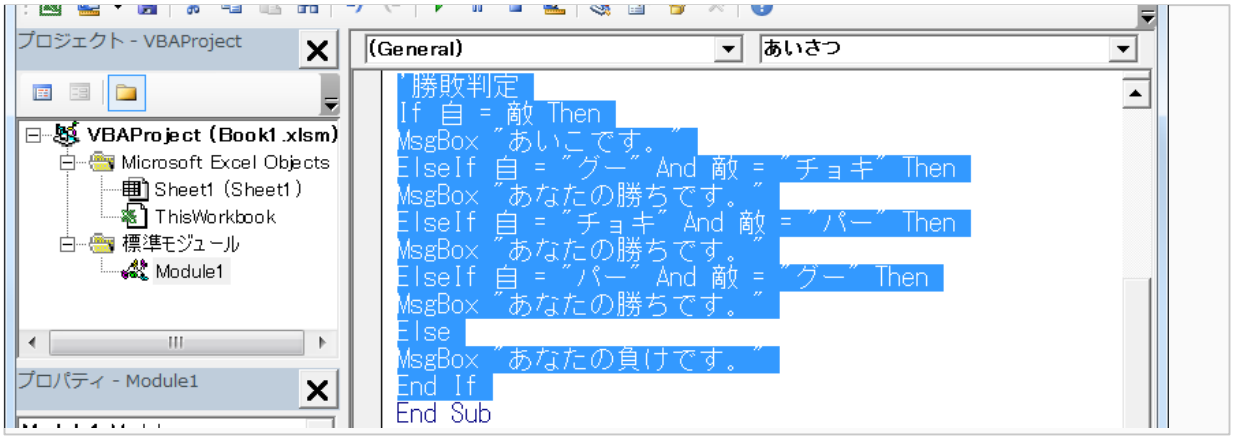
Loop 側に While 文を書いた場合はループの最後に判定されるため必ず一度は実行されますが、Do 側に書くとループの前に判定されますので、条件によっては一度も実行されない場合があります。

さて、この後じゃんけんプログラムを Do～Loop で囲みますが、その際にインデントを 1 行ずつ操作していると面倒なので一気に複数行のインデントを操作する方法をお伝えします。

次のように任意行のコードを選択してから、Tab キーを入力すると、選択された行のインデントがひとつ深くなります。



また、Shift を押しながら Tab で、インデントがひとつ浅くなります。ただ、Shift+Tab でインデントを戻すときは、戻しすぎに注意してください。戻しすぎると下図ように左端で揃ってしまい、個別のインデントを設定しなおすはめになります。



それでは、じゃんけんプログラムを次のように書き換えてください。

```
Sub じゃんけん()
    Do
        あいこ = 0    'あいこフラグを初期化
        '敵の手を決定
        ランダム = Int(Rnd() * 10) Mod 3
        If ランダム = 0 Then
            敵 = "グー"
        ElseIf ランダム = 1 Then
            敵 = "パー"
        Else
            敵 = "チョキ"
        End If

        'プレイヤーの手を決定
        自 = InputBox("グー,チョキ,パーのどれかを入力してください。
")

        '勝敗判定
        If 自 = 敵 Then
            MsgBox "あいこです。"
            あいこ = 1    'あいこフラグを 1 にする。
        ElseIf 自 = "グー" And 敵 = "チョキ" Then
            MsgBox "あなたの勝ちです。"
        ElseIf 自 = "チョキ" And 敵 = "パー" Then
            MsgBox "あなたの勝ちです。"
        ElseIf 自 = "パー" And 敵 = "グー" Then
            MsgBox "あなたの勝ちです。"
        Else
            MsgBox "あなたの負けです。"
        End If
    Loop While あいこ = 1    'あいこフラグが 1 の間繰り返す。
End Sub
```

Do～Loop で囲った他に、変数:あいこ を新しく登場させています。これはあいこだった場合に While 条件にそれを伝えるための変数で、このような使い方をフラグを立てると言います。

まず Do の直後にあいこを 0 にしています。これで、一旦は繰り返しを無効にしておきます。

```
        あいこ = 0     'あいこフラグを初期化
```

次のように、あいこになった場合だけフラグを立てて繰り返しを有効にするわけです。

```
        If 自 = 敵 Then
            MsgBox "あいこです。"
            あいこ = 1     'あいこフラグを 1 にする。
```

最後の Loop While でフラグが立っていれば、Do に戻って繰り返します。

```
    Loop While あいこ = 1     ’あいこフラグが 1 の間繰り返す。
```

ここでフラグの初期化を Do の外に出すと大変です。（実際にはしないでください。）

```
    あいこ = 0     'あいこフラグを初期化
    Do
        ’敵の手を決定
```

初期化だから、一番最初だけすれば良いと勘違いして Do の外に出してしまうと、一度あいこになったら次のループからフラグが立ちっぱなしになり、無限ループに陥ってしまいます。

プログラムを作っていると間違って無限ループを作ってしまうことが良くあるので、無限ループプログラムを実行してしまった際の対処についてお伝えしましょう。

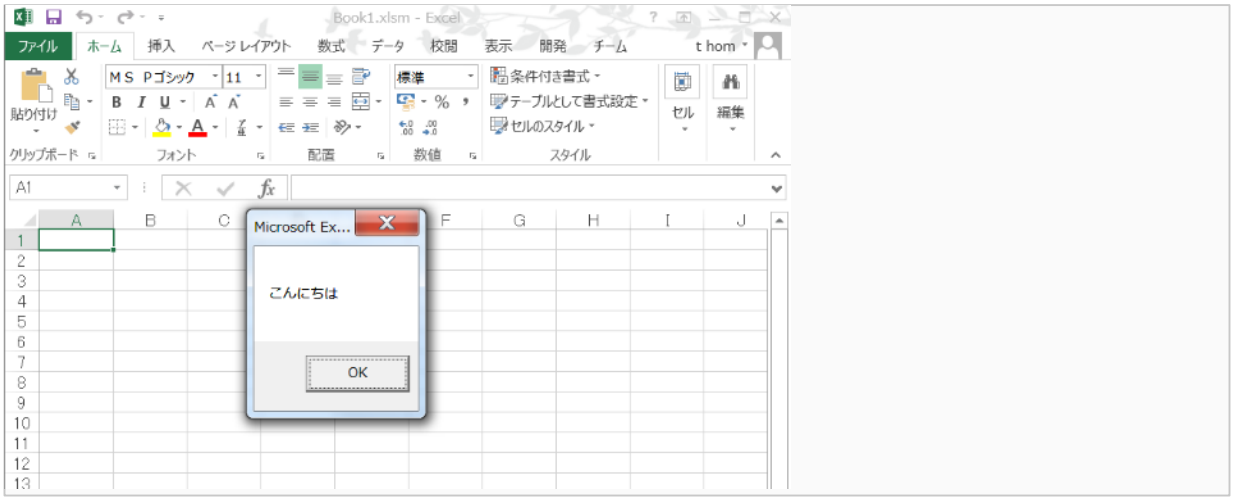
まずは、ファイルを保存し、念のため今マクロを書いている以外の Excel ブックは閉じておいてください。

そして、次のコードを記入します。

```
Sub 無限にあいさつ()
    Do While 1 = 1
        MsgBox "こんにちは"
    Loop
End Sub
```

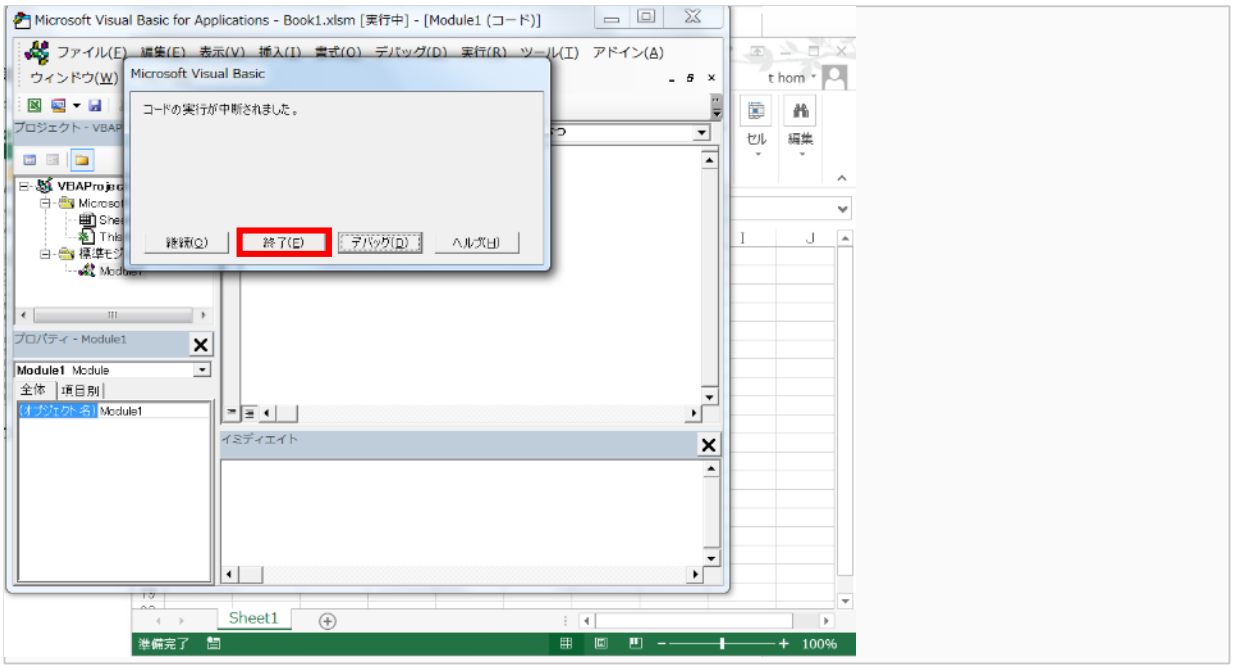
このコードは無限ループします。While 条件に「1 が 1 のとき」を書きますが、1 は常に 1 なので、常にループと同じ意味ですね。

実行すると次のようにメッセージが表示され、何度 OK を押しても無限終了できなくなります。また、Excel の×ボタンも押せません。



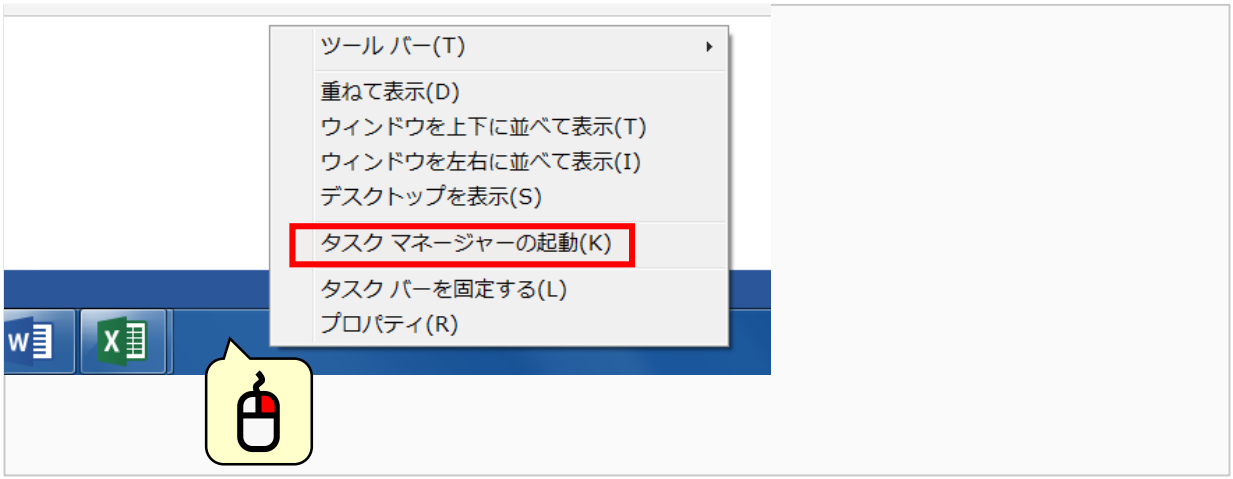
このような場合は、Ctrl キーを押しながら Pause/Break キーを押してください。Pause/Break は通常キーボードの右上のほうにあります。(ノート PC などではキー配置が異なったり、Fn キーなどと同時に押すケースがあります。キーが見つからない場合は後述するタスクマネージャーでの強制終了を行ってください。)

すると、次のメッセージが表示されるので、終了をクリックすると強制的に終了させることができます。

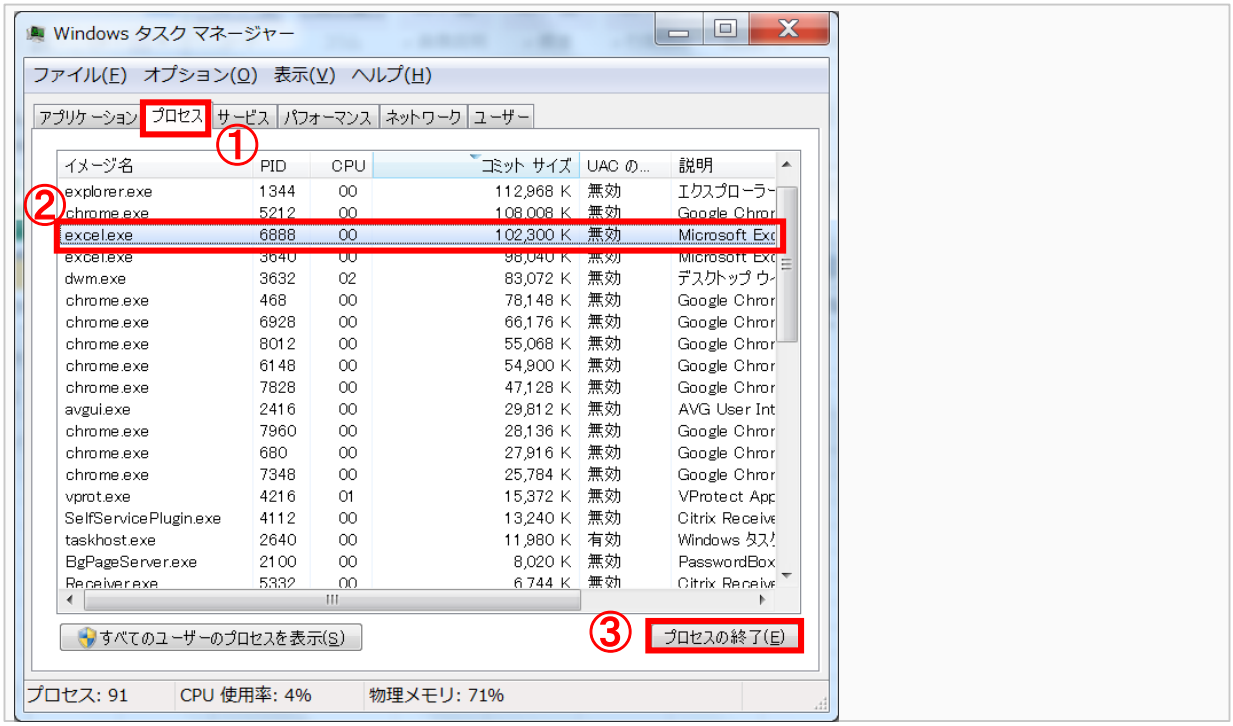


上記は大抵のケースで成功しますが、万が一うまくいかない場合は、タスクマネージャーを使用して Excel を強制終了することになります。念のためこちらの方法も紹介します。

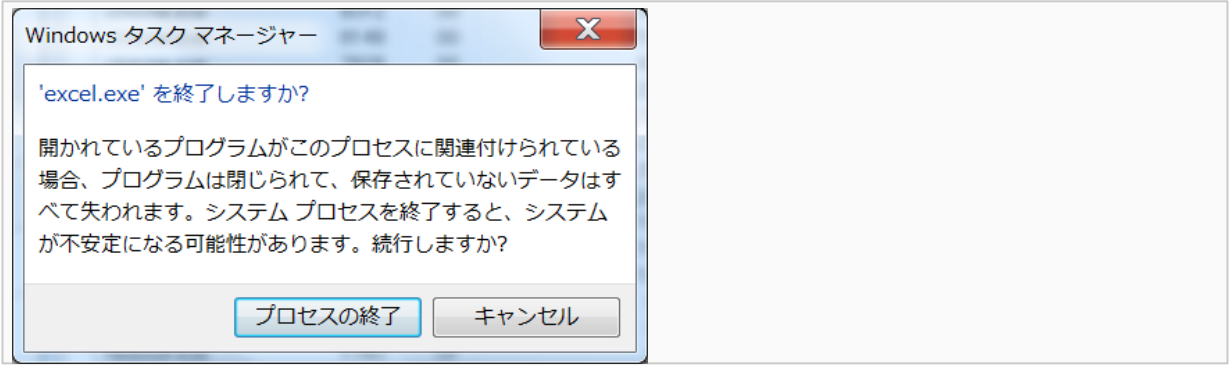
タスクバーを右クリックし、開いたポップアップメニューからタスクマネージャーの起動をクリックします。うまくいかない場合は、Ctrl と Shift と Esc を同時に押す方法でもタスクマネージャを起動することができます。



プロセスタブから excel.exe を探して選択し、プロセスの終了をクリックしてください。

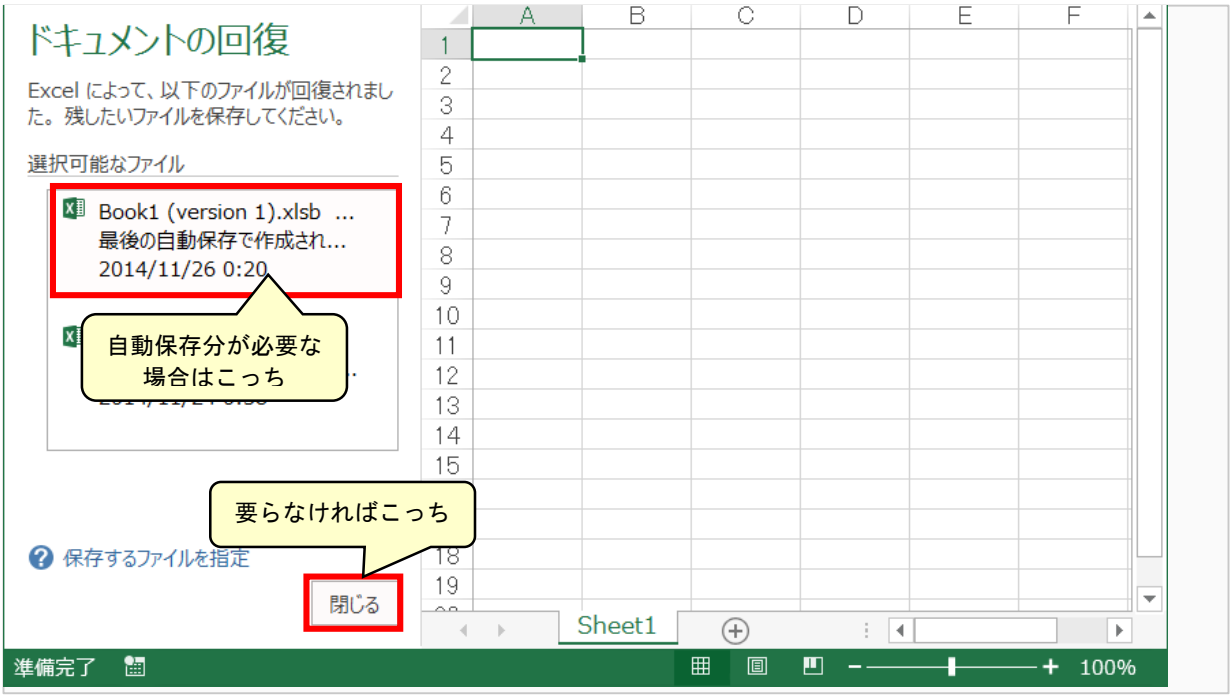


警告が出ますがそのまま終了します。



プロセスタブに excel.exe が複数ある場合はどちらかが本体なので、片方終了させてまだ残っている場合はもう片方も終了させます。

再度 Excel を起動した際に、ドキュメントの回復画面が出ている場合がありますが、前回自分で保存できていれば閉じるをクリックし、編集途中で保存できていなかった場合は「最後の自動保存で作成され…」の方を開いて保存してください。



無限ループへの対処は以上となります。

データ型について

プログラミングで扱うデータには、文字・数値・日付など様々な種類があります。人間にとって"1"も 1 も同じイチですが、コンピュータから見るとこれらは別物です。コンピュータはデータを文字型、数値型、日付型などの型に当てはめて解釈しており、これらをデータ型といいます。

試しに次のコードを入力して実行してください。

```
Sub 足し算()
    MsgBox 1 + 1
    MsgBox "1" + "1"
End Sub
```

最初のメッセージは 2 になりますが、次のメッセージは文字としての 1 がくっついて 11 になりました。では、これらなどうでしょうか。

```
Sub 足し算()
    MsgBox 1 + "1"
End Sub
```

今度は 2 になったと思います。数字だと認識したようです。次は数字にアルファベットを足してみましょう。

```
Sub 足し算()
    MsgBox 1 + "a"
End Sub
```

今度はエラーが発生してしまいました。環境によって表示が若干異なりますが、OK か終了でエラーメッセージを閉じてください。

「+」記号は、与えられたものを出来るだけ数値として足そうとするため、数値１と文字１を足し合わせる際に文字１も数値であると解釈します。文字１と文字１を足す場合は素直に文字同士の連結であると解釈してくれるようです。

ただし、VBA では数値１と文字 a は足すことができず、エラーになります。

次の場合はどうでしょうか。

```
Sub 結合()
    MsgBox 1 & 1
End Sub
```

「&」記号は、与えられたものを文字として解釈しますので、数値１と数値１が結合されて「11」になります。"1" + "1"と "1" & "1"はどちらも"11"になりますが、文字として結合する場合はなるべく後者を利用してください。

さて、次に、下記のプログラムを入力して実行してください。

```
Sub どっちが大きいか()
    x = "2"
    y = 3
    If x < y Then
        MsgBox "異常なし"
    ElseIf x > y Then
        MsgBox "そんなばかな"
    End If
End Sub
```

このプログラムは、x の方が大きいという結果になります。

コンピューターの内部で文字はコードとして扱われており、文字コードと呼ばれます。どうやら文字"2"の文字コードは 50 であるため、3 より"2"(50)の方が大きいという判断がなされているようです。

実際のプログラミングでわざわざ数値をダブルクォーテーションで囲む人は居ないと思いますが、次のようにユーザーに入力させる場合はどうしても文字列になってしまいます。

```
Sub 入力値との比較()
    x = InputBox("一桁の数字を入力してください。")
    y = 3
    If x < y Then
        MsgBox "入力値" & x & "は3より小さい"
    ElseIf x > y Then
        MsgBox "入力値" & x & "は3より大きい"
    End If
End Sub
```

InputBox に 1 を入れても、「入力値 1 は 3 より大きい」表示されてしまいます。

このように、プログラムを作成する際はデータ型を意識しておかないと、意図せずバグを生み出してしまう危険性があります。

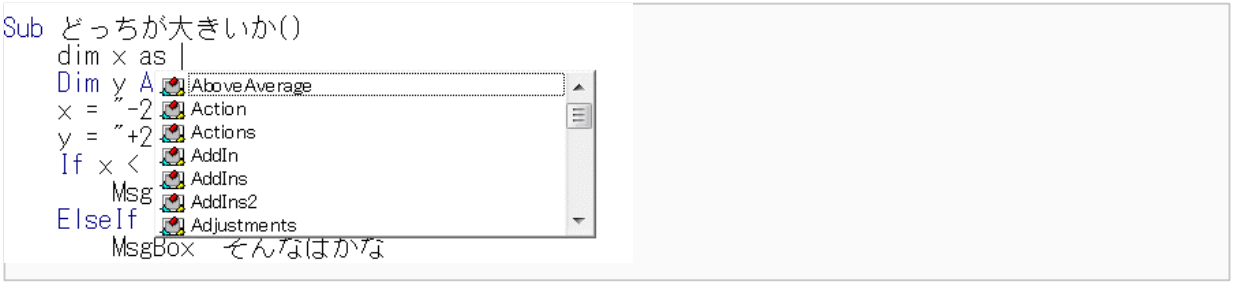
そうした混乱を避けるためには次項で説明する変数宣言を使用してください。

変数宣言

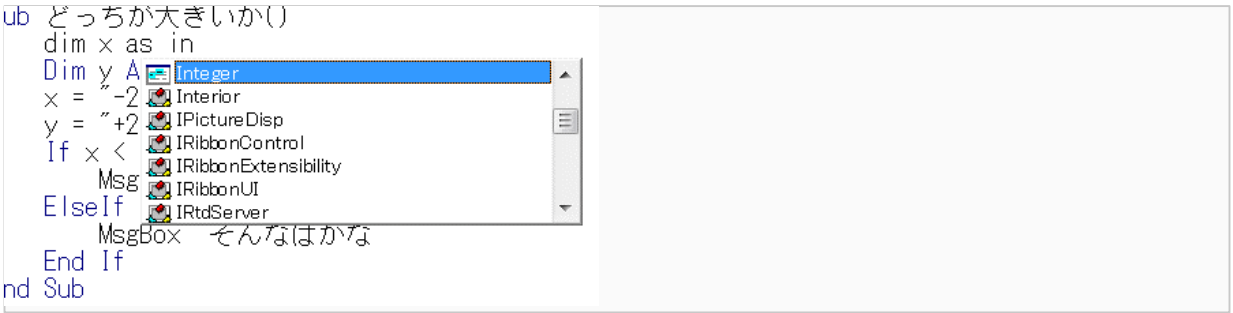
これまで変数には、いきなり値を代入していましたが、本来変数は宣言してから使うものです。前項のコードを次のように書き換えてください。Dim～を 2 行追加したほか、変数に代入する値も変えていますので注意してください。

```
Sub どっちが大きいか()
    Dim x As Integer
    Dim y As Integer
    x = "-2"
    y = "+2"
    If x < y Then
        MsgBox "異常なし"
    ElseIf x > y Then
        MsgBox "そんなばかな"
    End If
End Sub
```

dim x as の後にスペースを入力した時点で、次のような小窓が開いたかと思います。



これは入力候補を提示してくれるもので、マウスやカーソルキーでも選択できますが、まだ触らずに続けて integer の in まで入力すると、次のように Integer がヒットします。



ここで Enter キーを押すと Integer が入力されて次の行に移ります。小窓が消えてしまった場合は as の後ろを消してスペースを打ち直してください。最後まで手打ちしても同じ結果が得られま

すが、入力補完機能を使いこなすことではるかにコード入力が速くなりますので活用してください。

さて、コードを打ち終えたら実行してみましょう。結果は異常なしと出るはずです。

追加した下記の部分は x・y それぞれを Integer(整数)型として使用することを宣言しています。

```
Dim x As Integer
Dim y As Integer
```

整数とは 1、2、3 などの数字です。マイナスの整数とプラスの整数がありますが、少数や分数は含みません。普通、プラスの整数には+記号をつけませんが、この後の説明を分りやすくするためにあえてつけています。

次のコードは、x に文字列"-2"を代入していますが、x は Integer 型として宣言されているため、代入された時点で数値の-2 と判定されます。

```
x = "-2"
```

y も同じく Integer 型として宣言されているため、文字列"+2"は、数字の 2 と判定されます。したがって、当然次のコードは異常なしとなります。

```
If x < y Then
```

では、Integer を String に書き換えてみてください。

```
Dim x As String
Dim y As String
```

今度は結果が逆になります。

文字列「-2」と「+2」を比較した場合、先頭の文字から順番に文字コードとして比較されます。「-」の文字コードは 45、「+」の文字コードは「43」なので、文字列としては「+2」のほうが小さいと判断されます。

このように、型はプログラミングをする上でとても重要なものです。コンピューターの判断に任せるのではなく、変数を宣言することで明確に型を指定するようにしましょう。

変数宣言には他にも非常に重要な役割があります。

次のコードを入力し、実行してみてください。

```
Sub 計算ミスのサンプル()
    kihon = InputBox("基本給を入力してください")
    jikangai = InputBox("時間外手当を入力してください")
    kotsuhi = InputBox("通勤手当を入力してください")
    ikuji = InputBox("育児手当を入力してください")
    yakusyoku = InputBox("役職手当を入力してください")
    gokei = 0 + kihon + jikangai + kotsuhi + ikuji + yakushoku
    MsgBox "支払い額は" & gokei & "円です。〞
End Sub
```

gokei の最初に 0 を足しているのは、以後の変数を数値として認識させるためです。これが無いと全部文字としてつながってしまい、とんでもない金額になります。

さて、たとえば、すべて 1 と入力すると、5 円になるはずですが、4 円になってしまいます。

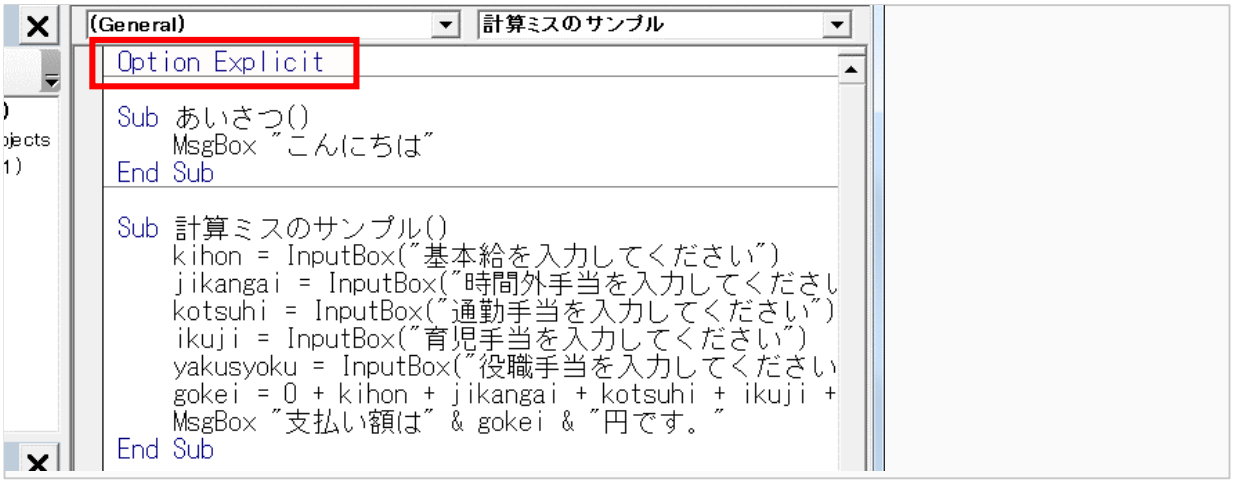
これはよくあるミスで、変数名の打ち間違いを再現しています。

あなたが入力した値は変数 yakushoku に入っていますので、yakusyoku を参照しても何もありません。したがって合計金額に役職手当が含まれないことになります。

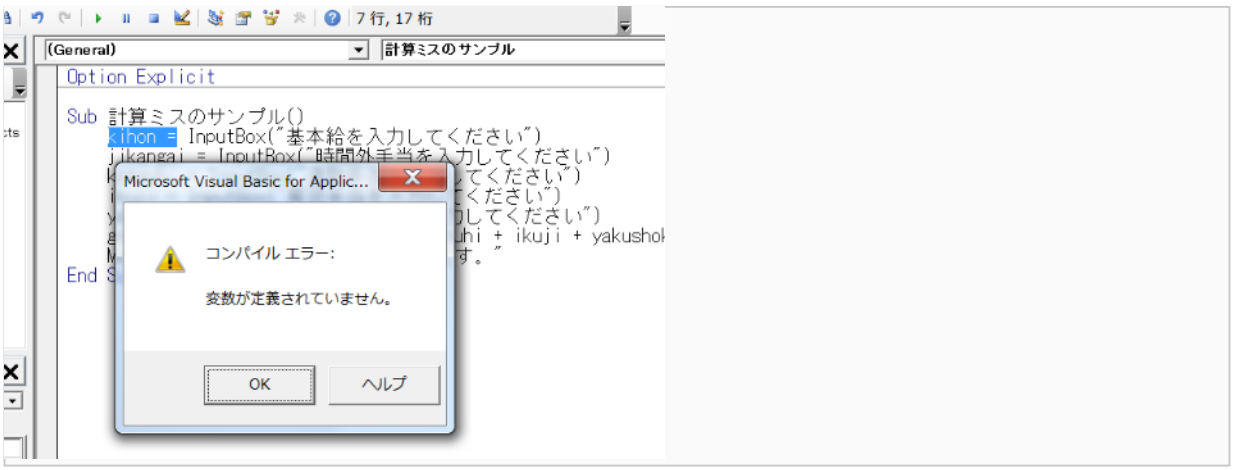
こんなプログラムを給与計算に使われたら、社員はたまったものじゃありません。

このようなミスは変数宣言を強制させることで防止できます。

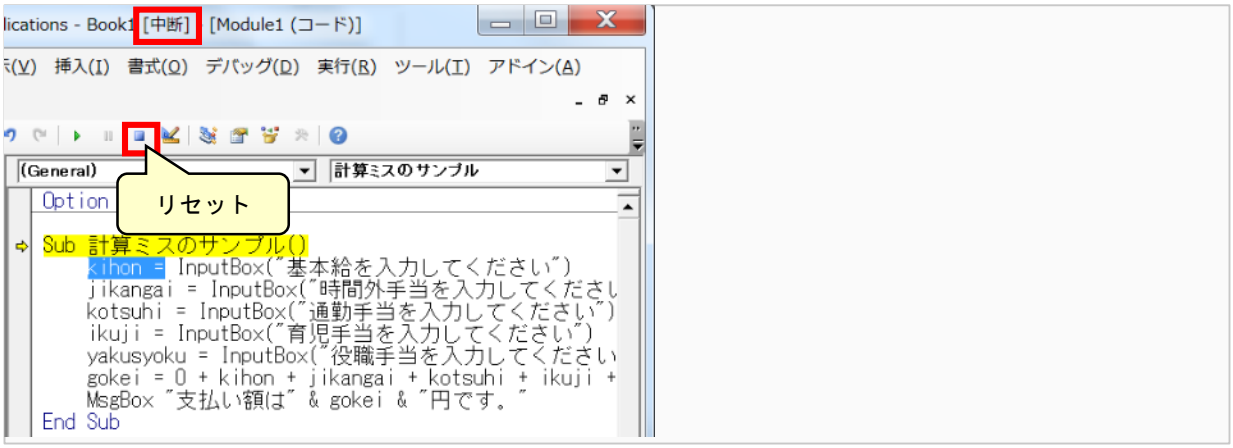
次のようにモジュールの先頭に Option Explicit と書くことで、そのモジュールでの変数宣言を強制させることができます。



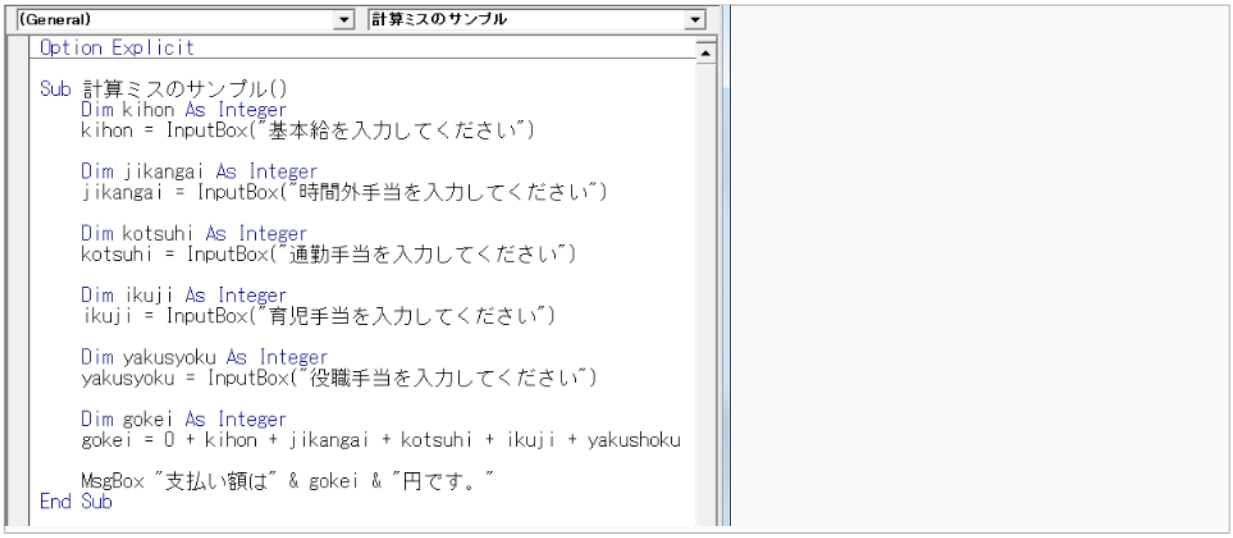
変数宣言を強制したうえで、先ほどのプログラムを実行すると、次のようにエラーになります。



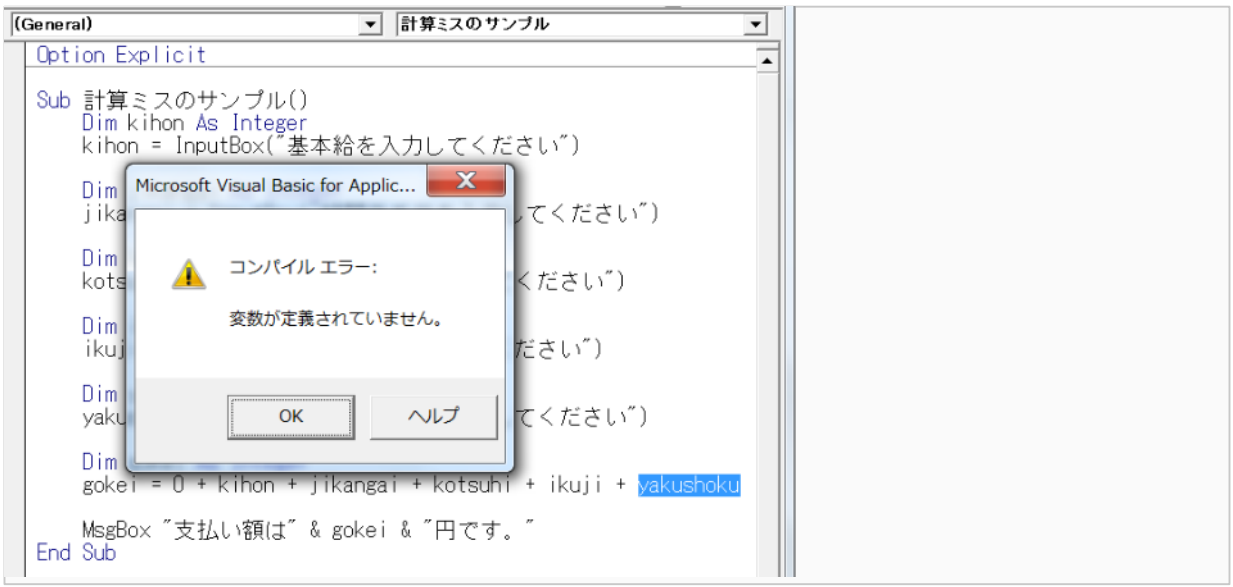
OK ボタンを押してエラーを閉じると次のように中断モードになりますので、リセットボタンで実行を止めてください。



では順に変数を宣言してみましょう。



これを実行すると、打ち間違えた変数 yakushoku が選択され、定義されていないというエラーが発生します。



これで役職手当が貰えないといったおかしなプログラムを作らずにすみます。エラーというのは一種の安全装置ですので、プログラム作成中にエラーが出ても、慌てずにじっくり原因を考えてください。

普段パソコンを使ううえではエラーなんて見たくありませんが、金額を間違えて平気で動いているプログラムよりは、エラーで停止してしまうプログラムの方がよっぽどマシだと言えます。

では、先ほどと同じようにエラーを閉じた後、リセットボタンで停止してくださ。

変数宣言を行って型を明示するメリットはもうひとつあります。計算が速くなり、メモリの消費が少なくなることです。型を明示しない場合、計算のたびに型を判定するので遅いです。また、型が分らない場合はとりあえず何でも入る大きなハコを用意しておく必要があるため、メモリを多く使いますが、整数が入ると分っていれば専用の小さなハコを用意すれば良いためにメモリを節約できます。

まぁ、遅い速いといっても現代のコンピュータは高性能なので、人間が感じられるほどの差はありません。パフォーマンス上のメリットについてはあまり神経質になる必要はありませんが、豆知識として抑えておいてください。

コラム　他言語と変数宣言

VBA と他のプログラミング言語では変数の扱いについてのルールが違います。

例えば C 言語では、そもそも変数は宣言しないと使用できませんし、アルファベットの大文字と小文字も区別します。また、宣言の方法も Dim ではなく、型名を先に書きます。C 言語では日本語の変数は使用できません。

下記は VBA の変数宣言を C 言語に変換した例です。

例) Dim x As Integer → int x

逆に PHP 言語ではそもそも変数宣言の機能が存在せず、変数名は常にドル記号から始まります。

VBA では変数宣言をするかしないかの判断はプログラマーに委ねられています。計算ミスが実務上の致命的な不具合につながるようなマクロなら当然宣言するべきですが、そういったリクスが無く、自分しか使わないような小規模なマクロなら必ずしも宣言する必要はないでしょう。

配列

連続した同じ種類のデータを変数に代入したいときは、データの数だけ変数が必要になります。これだと管理も宣言も大変なので、変数+番号で管理できるようにしたものが**配列**です。

配列は次のように宣言します。

```
Dim a(1 to 3) As Integer
```

これで整数型の配列 a が 1 番から 3 番まで作成されます。とりあえず配列名を a としましたが、特になんでも構いません。変数と同じように、日本語も使用できます。

では、値の代入と取り出しもやってみましょう。

```
Sub 配列テスト()
    Dim a(1 to 3) As Integer
    a(1) = 60
    a(2) = 200
    a(3) = 10
    MsgBox a(1) + a(2) + a(3)
End Sub
```

配列の番号指定には変数も使えるため、配列はループと相性が良く、次のようにループでそれぞれ異なる相手へのメッセージを表示させることができます。

```
Sub 連続あいさつ()
    Dim お客様(1 to 3) As String
    お客様(1) = "鈴木"
    お客様(2) = "山田"
    お客様(3) = "佐藤"
    Dim x As Integer
    x = 1
    Do While x <= 3
        MsgBox お客様(x) & "さん、こんにちは"
        x = x + 1
    Loop
End Sub
```

x = x + 1 という奇妙な式が出てきますが、これは x の中身に 1 を足した結果を x に再度代入する操作です。その結果、x がひとつ増えます。

変数が 1 で初期化され、ひとつずつ増えながら条件に達するまで繰り返すというのはよく使われるパターンで、このために専用の繰り返し文があります。連続あいさつマクロを次のように変更してみてください。

```
Sub 連続あいさつ()
    Dim お客様(1 to 3) As String
    お客様(1) = "鈴木"
    お客様(2) = "山田"
    お客様(3) = "佐藤"
    Dim i As Integer
    For i = 1 to 3 Step 1
        MsgBox お客様(i) & "さん、こんにちは"
    Next i
End Sub
```

先ほどと同じように動作したはずです。For～Next がループになります。For の後の i = 1 は、i の初期化です。to 3 が「3 まで」という条件で、Step 1 はひとつずつという意味になります。Next i は For に戻って繰り返すという意味です。Next の変数記号 i は省略できます。

逆順に佐藤さんからあいさつする場合は、次のように書きます。

```
    For i = 3 to 1 Step -1
        MsgBox お客様(i) & "さん、こんにちは"
    Next i
```

コラム　よく見かける 1 文字の変数名

プログラミングでよく見かける 1 文字の変数名は、a b c x y z n m i j k です。

最初の 6 つはおそらく数学の影響です。数学では通常、既に判明している数に a,b,c を使用し、これから計算でもとめる数を x,y,z とします。

その次の n は数を意味する Number の頭文字です。次が o ではなく m なのは、o がゼロとまぎらわしいからだそうです。

また、i j k は For 文の変数名によく使用されます。繰り返しを意味する英単語 Itarate の頭文字 i から始まってアルファベット順に使用されます。

プログラムを入力する段階ではアルファベットひと文字は短くて便利ですが、最終的には単語として読める変数名に直しておかないと読み返したときに意味が分からなくなります。ただし、繰り返しに使用する i j k だけは普通そのままにしておきます。

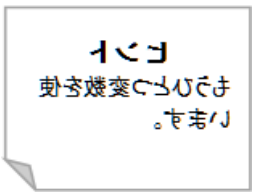
練習問題

では、文法の総仕上げとしていくつか練習問題をやってみましょう。巻末に解答例を用意していますが、一度は自分で考えてみて、コードを入力してください。

問題１）変数の入れ替え

変数 A と変数 B を用意し、適当な値を代入してください。そして、A と B の中身を入れ替えた後、メッセージボックスで次のように表示させてください。

「A の中身は○○。B の中身は○○。」

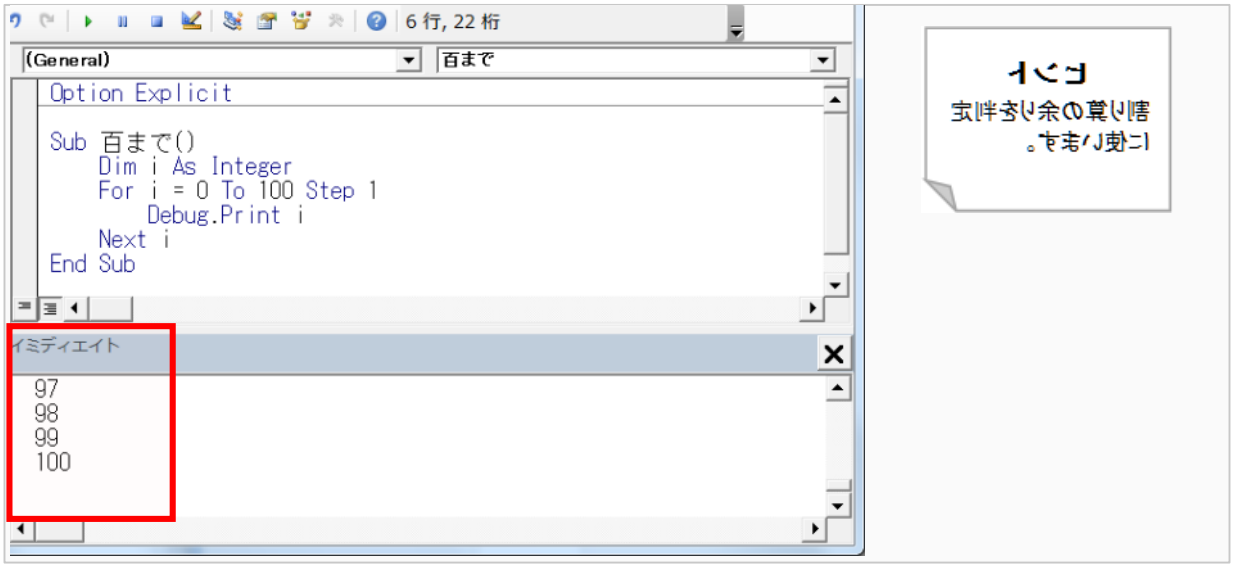


問題２）フィズバズ

英語圏の言葉遊びに、FizzBuzz というものがあります。1 から順に数字を言っていきますが、3 で割り切れる場合は Fizz、5 で割り切れる場合は Buzz、両方で割り切れる場合は FizzBuzz をそれぞれ数字の代わりに言います。

例) 1、2、Fizz、4、Buzz、6、7、8、Fizz、Buzz、11、12、Fizz、14、FizzBuzz、16、…

1～100 までの FizzBuzz を出力するプログラムを作成してください。このとき、MsgBox の代わりに Debug.Print と書くと、下図のようにイミディエイトウインドウに出力できますので、わざわざ OK ボタンを押す手間が省けます。

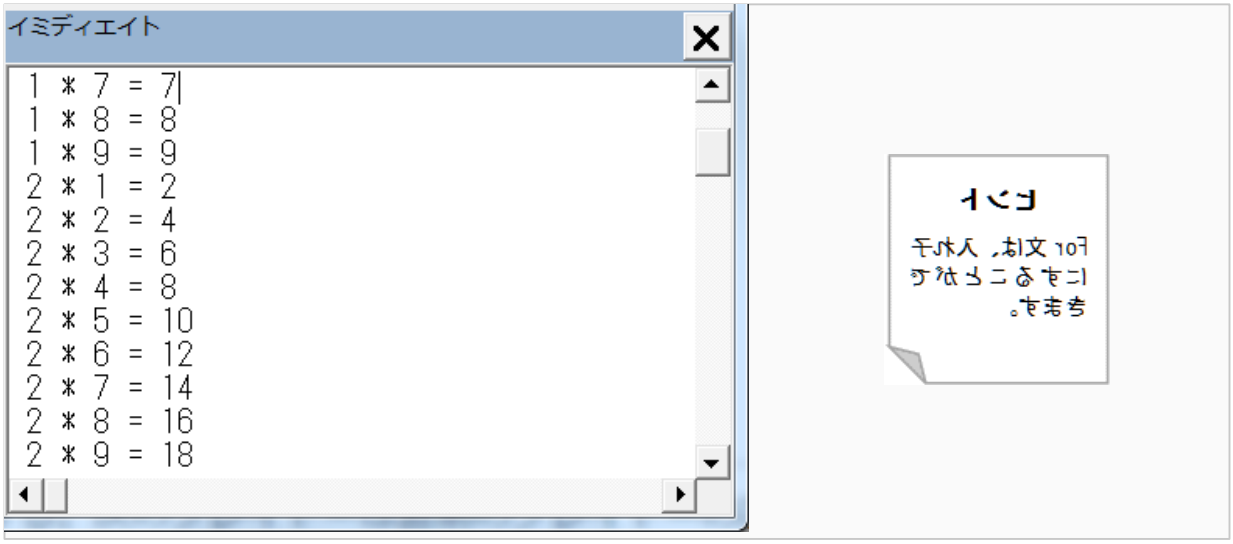


イミディエイトウインドウが表示されていない場合は、表示メニューから表示させてください。

出力内容を消すには、イミディエイトウインドウ内の任意の場所を選択して Ctrl+A で文字を全選択し、Delete キーを押します。

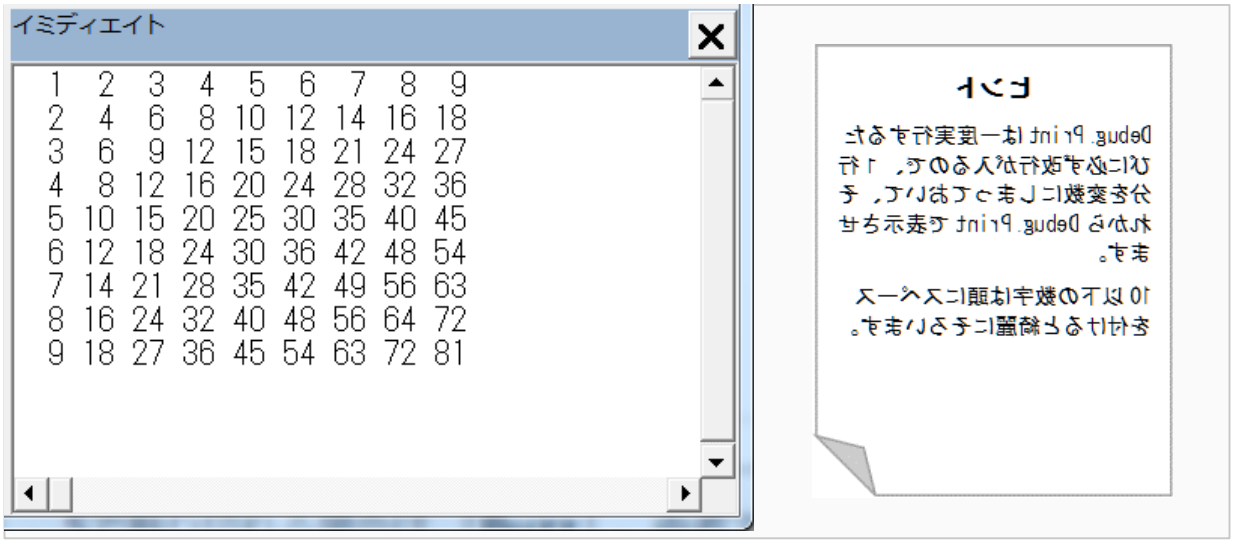
問題３）九九

下図の例のように、イミディエイトウインドウに九九を表示させてください。



エキストラ問題）九九の整形表示

これまでの問題が簡単すぎたという方は、九九の整形表示に挑戦してみてください。



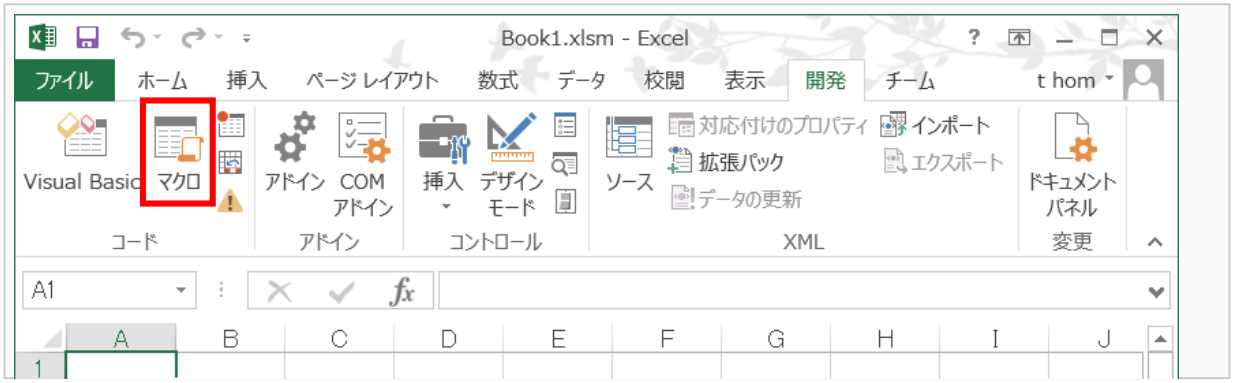
さて、いかがでしたか。1 問も解けなかったとしても、落ち込む必要はありません。巻末の答えを見て、なぜそうなるのか考えてみましょう。

文法の学習は以上で一旦終了です。他に便利な文法が 2 つほど残っていますが、これは Excel のシート上で実践した方がわかりやすいので後に説明します。

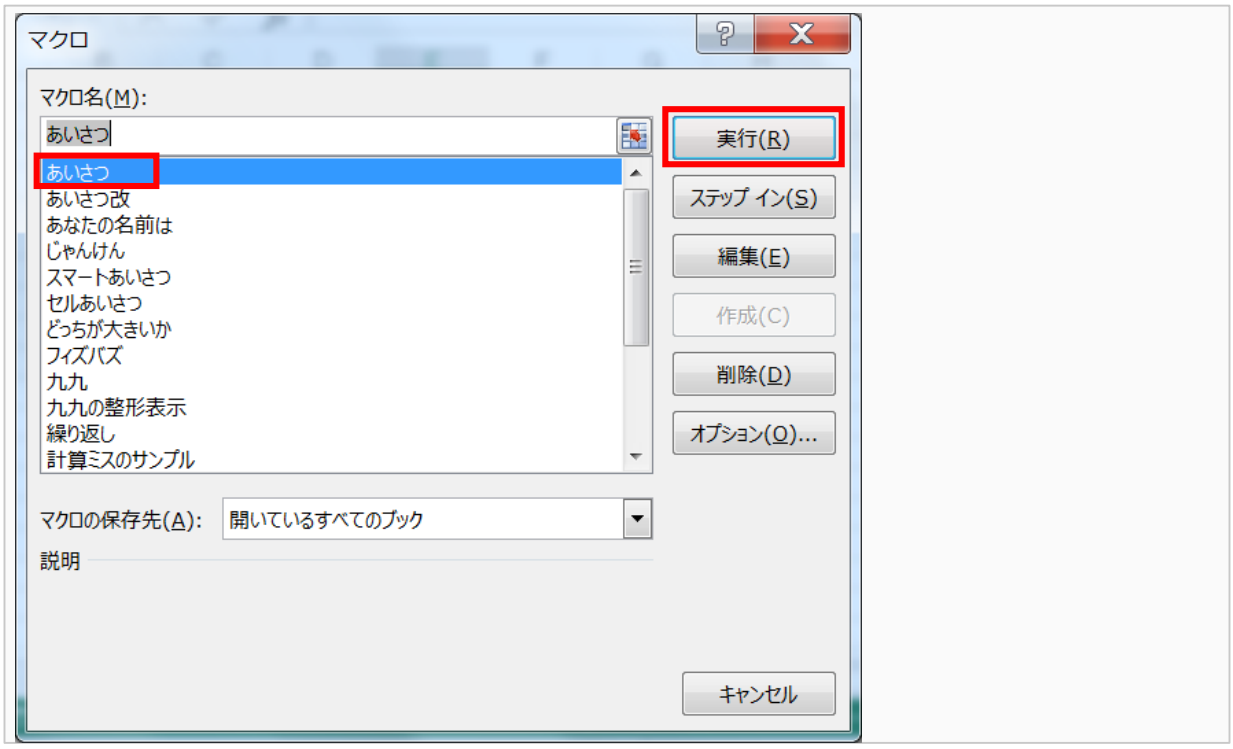
Excel 操作に入る前に、次項でブックからマクロを実行する方法を押さえておきましょう。

ブックからマクロを実行する

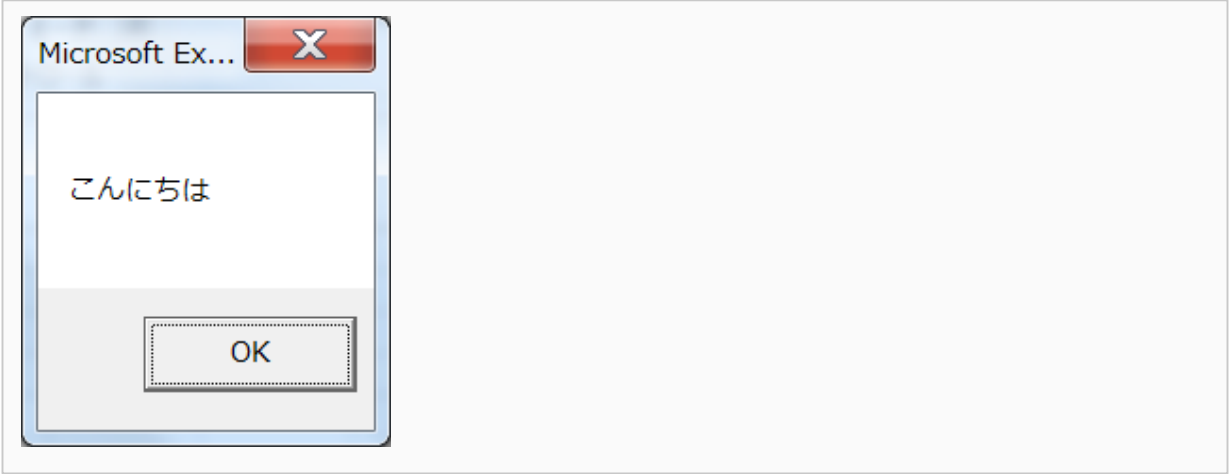
ブックの開発タブを選択し、マクロボタンを押してください。



次のような画面が表示されますので、項目から適当なマクロをマウスで選択し、実行ボタンを押します。



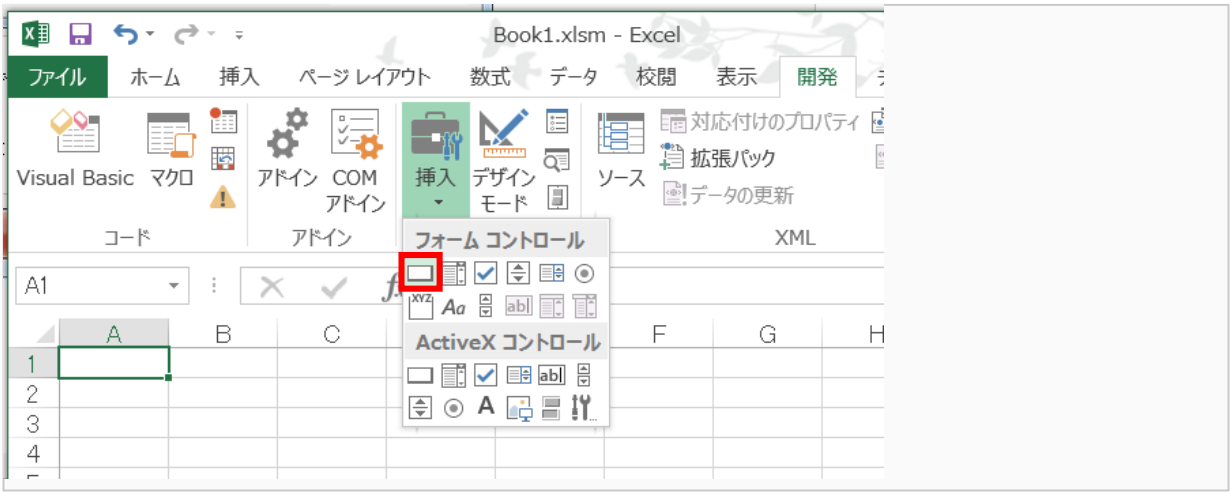
すると、選択したマクロが実行されます。



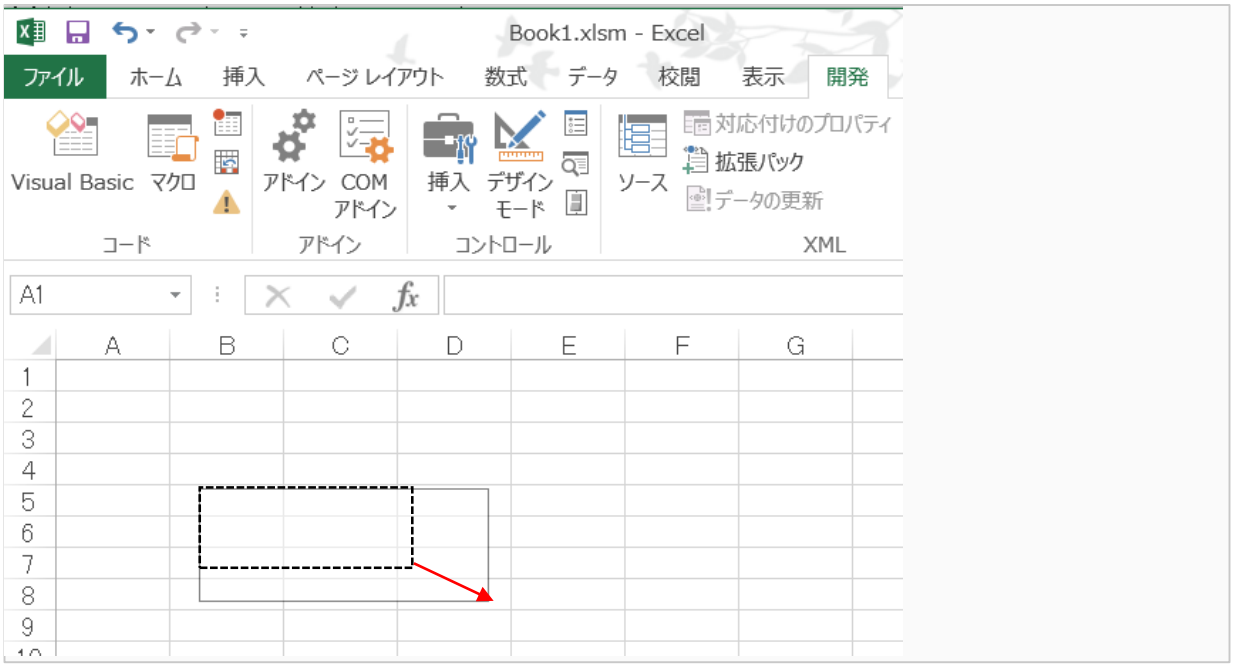
これがブックからマクロを実行させる方法の一つです。マクロの実行画面は、Alt+F8 キーでも呼び出すことができます。

次に、シート上にマクロを実行するためのボタンを配置してみます。

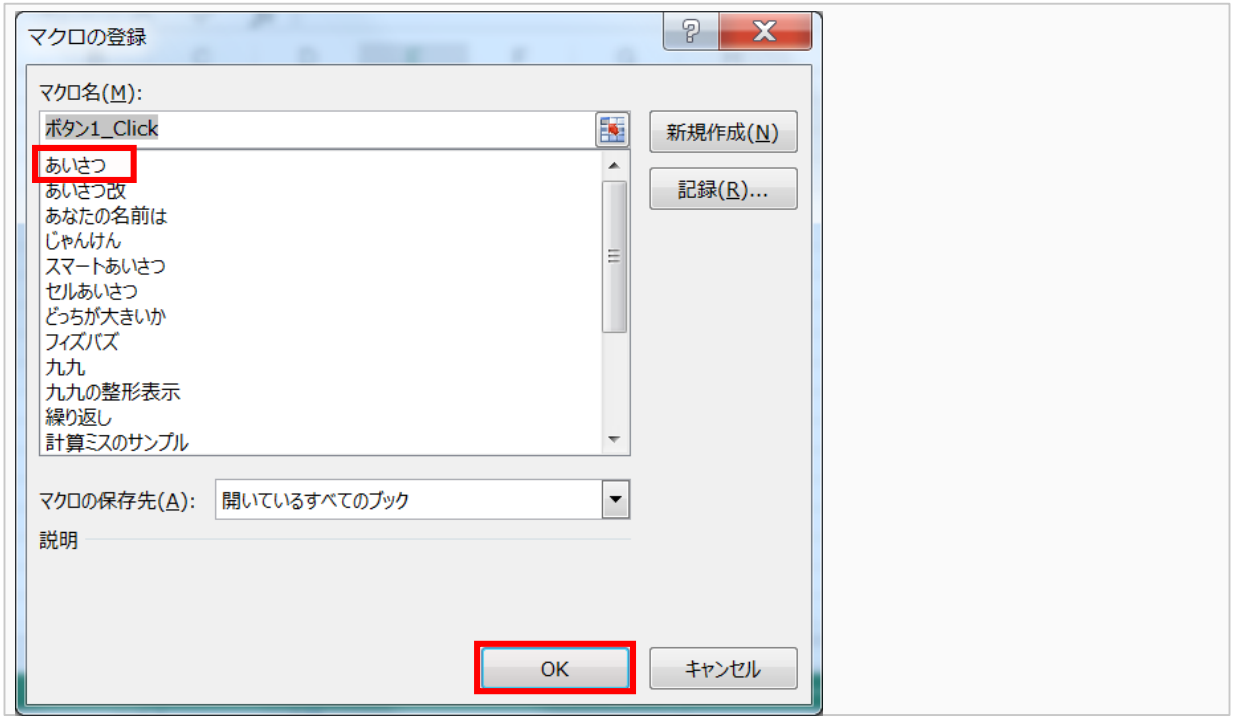
開発タブの**挿入**をクリックするとポップアップが表示されるので、フォームコントロールの左上にあるボタンのアイコンをクリックします。



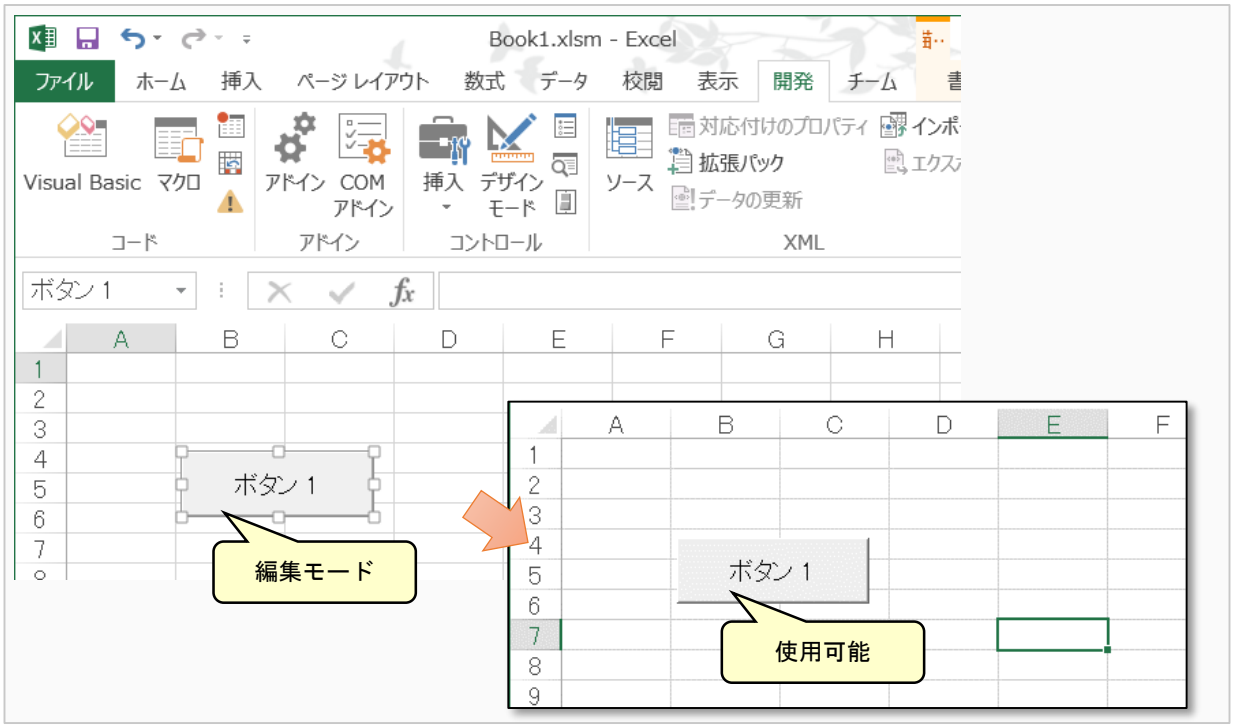
カーソルが十字型に変わるので、シート上でドラッグしてください。ドロップした位置でボタンのサイズが決まります。



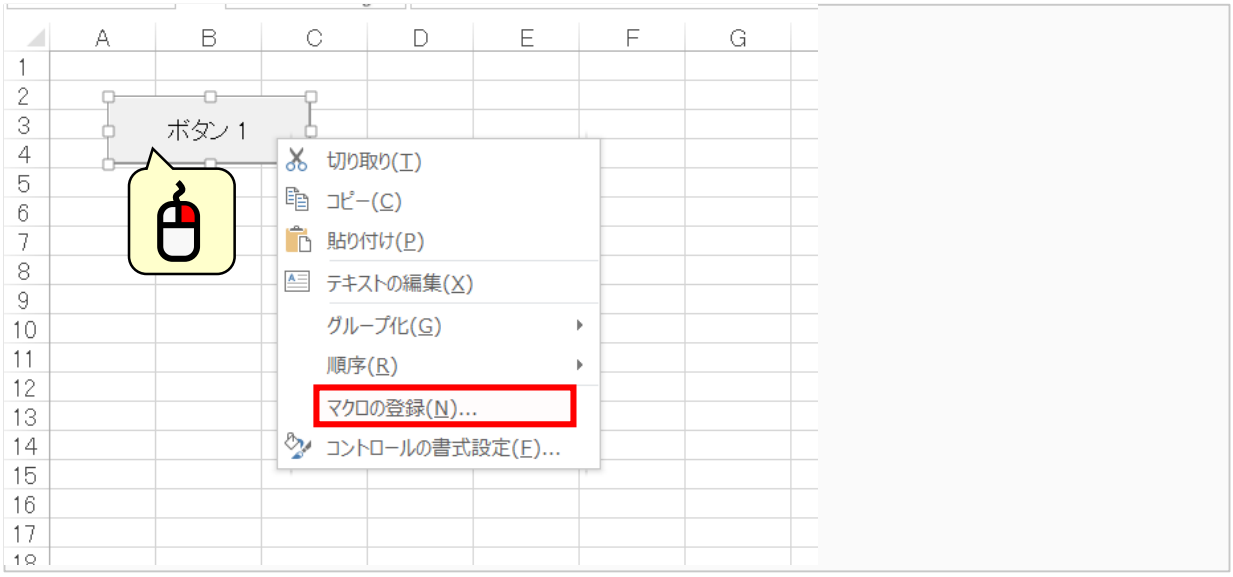
ドロップするとすぐにマクロ登録ウインドウが表示されますので、適当なマクロを選択して OK をクリックしてください。



ボタンが配置された直後は編集モードのためこのまま押すことはできません。一旦シート上のセルを選択すると押せるようになります。



間違えてマクロの登録をキャンセルしてしまったり、違うマクロを選んでしまった際は、右クリックすることでマクロの登録をやり直すことができます。



またボタン表面のテキストも右クリックメニューにありますので、分かりやすい名前に変更してください。

ボタンを移動させたりサイズを変更したいときは、右クリックしてからポップアップメニューを Esc キーでキャンセルしてください。するとボタンが編集モードのまま残るので、あとはオート

シェイプと同じ要領でサイズの変更や移動ができます。編集モードで Delete キーを押すとボタンを削除できます。

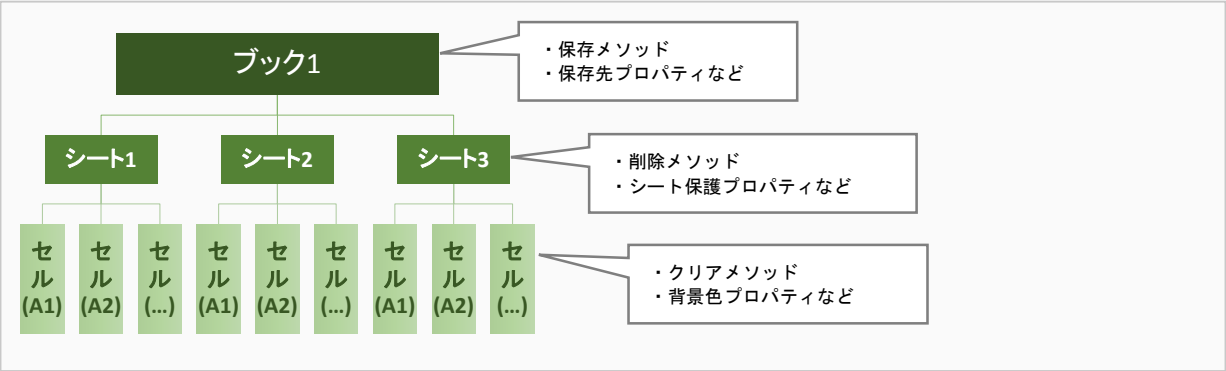
実はオートシェイプも右クリックすることでマクロを登録できるようになっています。
※Excel 2003 では残念ながらできません。

ボタンと違い押しても凹まないので使い道は限られますが、工夫次第で面白い使い方が出来るかもしれません。

Excel オブジェクトの操作

さて、ここからはお待ちかねの Excel オブジェクトの操作に入ります。Excel オブジェクトにはワークブック、ワークシート、セルなどがあります。他にもオートシェイプやグラフなどがありますが、これらはちょっと扱いが難しいので、まずは先に挙げた３つのオブジェクトの操作を覚えてください。

これらのオブジェクトは並列ではなく、次のような階層構造になっており、それぞれのオブジェクトがメソッド(機能)とプロパティ(性質)を持っています。



VBA で特定のセルに何か操作をするには、ブック.シート.セル.指示という風にドットでつないでいきます。では実際にコードを書いてみましょう。

```
Sub セルあいさつ()
    ThisWorkbook.Sheets("Sheet1").Range("A1").Value = "こんにちは"
End Sub
```

ドットは、日本語の「の」だと思ってください。つまり、以下のような意味です。
ThisWorkbook の Sheets("Sheet1")の Range("A1")の Value = "こんにちは"

さらに訳すとすれば、次のようになります。
このワークブックのシート("Sheet1")のセル範囲("A1")の値="こんにちは"

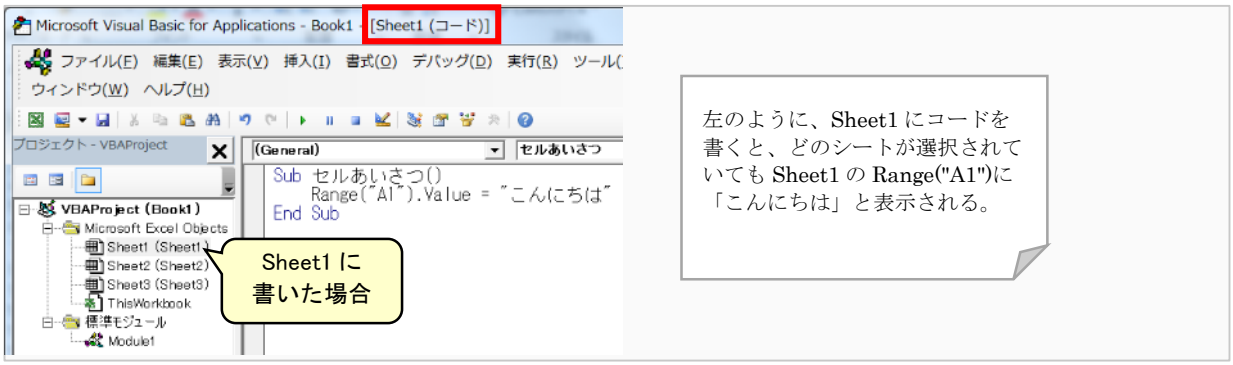
今選択されているシートの A1 セルに文字を入れたかったら、次のようにします。

```
Sub セルあいさつ()
    ActiveWorkbook.ActiveSheet.Range("A1").Value = "こんにちは"
End Sub
```

標準モジュールにマクロを書く場合、次のように ActiveWorkbook と ActiveSheet を省略しても同じ動作をします。

```
Sub セルあいさつ()
    Range("A1").Value = "こんにちは"
End Sub
```

もしシートオブジェクトにコードを書いた場合、ActiveSheet を省略するとマクロを書いたシートの Range("A1")という意味になりますので注意してください。

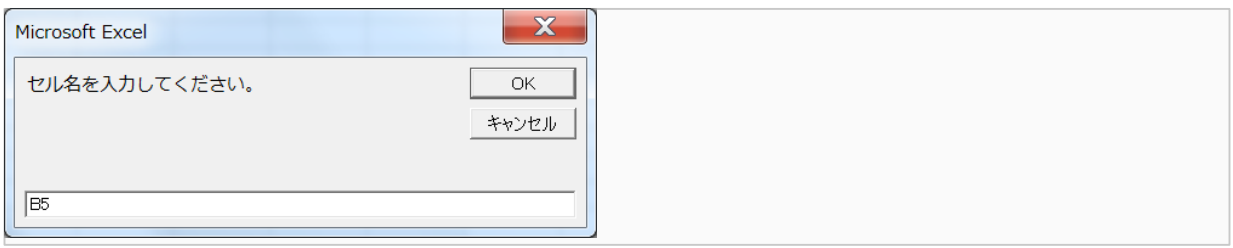


このあともう少し色々な処理を見ていきたいと思いますが、VBE からのマクロ実行ではアクティブセルの変化が見づらいので、前項で学習したブックからのマクロ実行を活用してください。

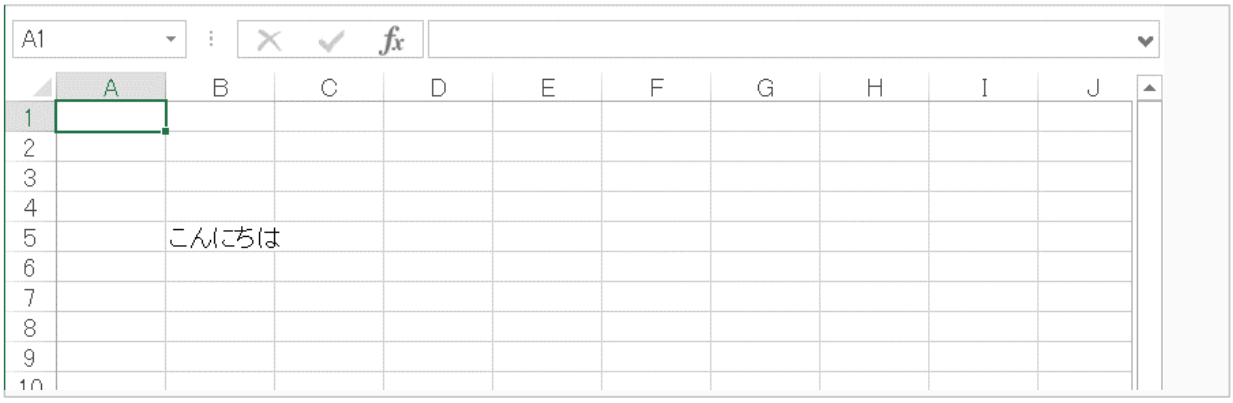
では、次のコードを入力してください。

```
Sub レンジ指定()
    Dim r As String
    r = InputBox("セル名を入力してください。")
    Range(r).Value = "こんにちは"
End Sub
```

ブックから実行すると、入力ボックスが表示されるので、試しに B5 と入力してみましょう。



すると、次のように B5 セルに「こんにちは」が表示されます。



もう一度実行し、今度は入力ボックスにコロン区切りで A1:C3 と入力してください。すると、次のように表示されます。

先ほど Range("A1").Value = "こんにちは"というコードを入力しましたが、この"A1"は文字列でセルの番地を表しているため、InputBox に入力された値を変数で活用することができるわけです。

文法の学習で、文字列と数字を&で結合しましたが、あれと同じことを Range の中でやってみましょう。次のコードを入力してください。

```
Sub 行指定あいさつ()
    Dim x As Integer
    x = 10
    Range("A" & x).Value = "こんにちは"
End Sub
```

これは、文字"A"と整数 10 をくっつけていますから、A10 セルに「こんにちは」が入ります。これができるということは、For 文を使ったループも可能です。

次のコードを入力して実行してください。

```
Sub 連続セルあいさつ()
    Dim i As Integer
    For i = 1 to 30 Step 1
        Range("B" & i).Value = "こんにちは"
    Next i
End Sub
```

Step を 2 にすれば、ひとつ飛ばしで入力することもできます。

次に、横方向に連続で処理する方法を伝えます。これは縦方向にも使えますが、Range の代わりに Cells を使用します。次のコードを入力して実行してください。

```
Sub 連続セルあいさつヨコ()
    Dim i As Integer
    For i = 1 to 30 Step 1
        Cells(1, i).Value = "こんにちは"
    Next i
End Sub
```

Cells のカッコには、(タテ, ヨコ)を数値で入れます。両方 i にすればナナメに進んでいきます。

さて、文字ばかりではつまらないので、一つ飛ばしにセルの色を変えてみましょう。これも簡単です。次のコードを入力してください。

```
Sub セルの色変更()
    Dim i As Integer
    For i = 1 To 30 Step 2
        Range("D" & i).Interior.Color = vbYellow
    Next i
End Sub
```

Interior は、インテリアのことです。部屋の内装などを指して使いますね。訳すと以下のような意味になります。
セル範囲("D" & i")の内装の色 = 黄色

vbYellow の他には、vbRed、vbBlue、vbGreen などがあります。

次に、選択されたセルの操作をやってみましょう。次のコードを入力してください。

```
Sub 選択セルの色を変更()
    Selection.Interior.Color = vbYellow
End Sub
```

選択されたセルを表すのは、Selection というオブジェクトです。ActiveWorkbook や ActiveSheet は書かず、いきなり Selection で始めます。これも簡単ですね。

セルが範囲選択されている場合は、それら全てが黄色になります。また、セルではなく、オートシェイプが選択されている場合はそのシェイプの背景色が黄色になります。

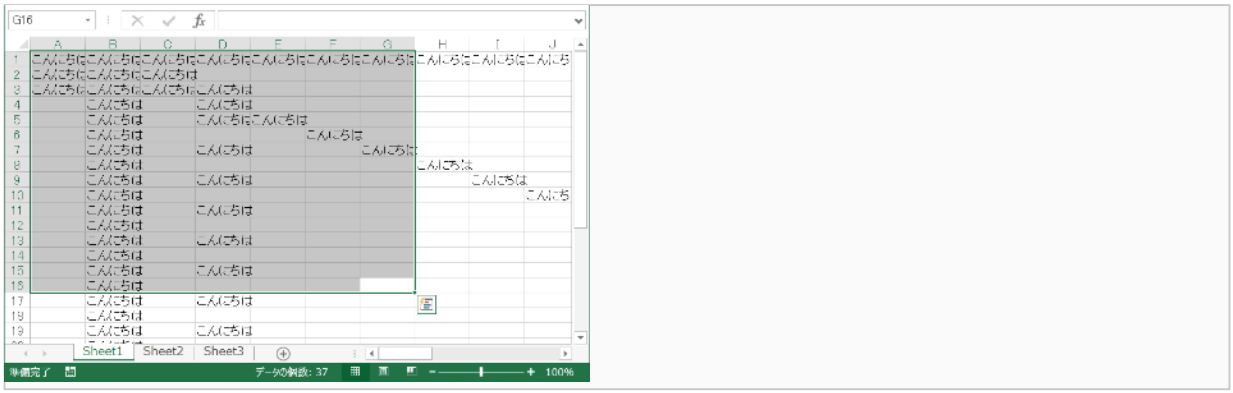
応用マクロ

さて、ではセルの操作を応用してもう少し実用的なマクロを作ってみましょう。ここで新しい文法が登場しますが、非常に便利なので是非覚えてください。

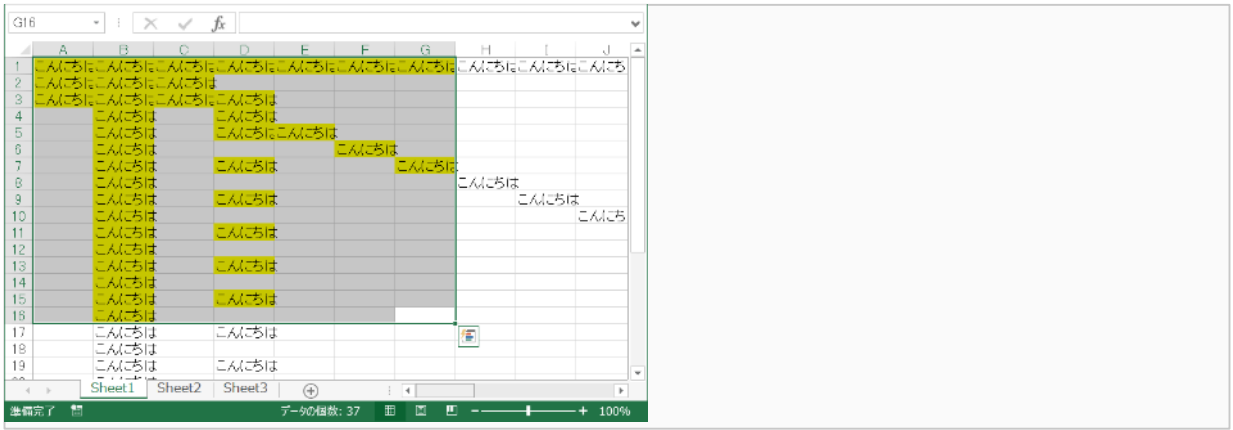
次のコードを入力してください。

```
Sub 条件付き色塗り()
    Dim x As Range
    For Each x In Selection
        If x.Value <> "" Then
            x.Interior.Color = vbYellow
        End If
    Next x
End Sub
```

そして、適当に値が入ったシートで、適当な範囲を選択して実行してみてください。



すると、値の入ったセルだけが黄色く塗りつぶされるのが分かります。

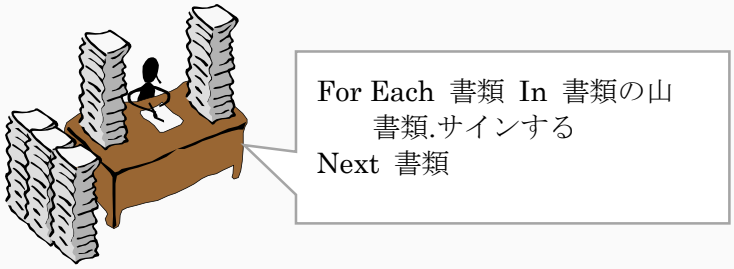


なぜこうなるのか、文法を説明します。まず以下の文を見てください。

```
    For Each x In Selection
```

これは For 文の一種ですが、Selection からひとつ取り出して x に代入しています。

For Each は数値の指定ではなく、与えられた集合の中からひとつずつ処理を適用していきます。例えるなら、書類の山から一枚ずつ処理をするイメージです。



次が For Each 文の中身ですが、ここで x には取り出されたひとつのセルが入っているので、Value が空かどうか判断し、背景色を変えることができます。

```
        If x.Value <> "" Then
            x.Interior.Color = vbYellow
        End If
```

変数宣言で Dim x As Range とありますが、これは Range オブジェクト型の変数です。他にシート型変数、ブック型変数などもありますが、全部覚える必要はありません。

これを応用すれば、特定の数値以下のセルの色を変えたり、特定の文字を含むセルの値を置き換えたりと実用的なマクロが作れます。

ただし、For Each は他のループと違って処理の順番が保証されていません。今のところは決まった順に処理されるようですが、マイクロソフトがエクセルをバージョンアップした時にどうなるか分からないので、処理順が重要なマクロの場合は For や Do～Loop を使いましょう。

コラム　For Each の処理順

For Each の処理順が保証されないのには理由があります。たとえば書類の山を一人で処理する場合は上から順に処理していきますよね。でももう一人手伝ってくれる人がいたらどうでしょうか。二人の処理スピードが同じとは限らないので、書類の順番はバラバラになります。最近の CPU ではこれと同じことが起きています。今のところ Excel VBA はそれを生かし切れていないため、一つの CPU(コア)で処理を行っていますが、将来的に複数のコアで並列処理するようになると、結果の順番もバラバラになる可能性があるのです。

With 句

With という文法について解説します。最後に説明するマクロの記録でも出てきますのでしっかり押さえてください。簡単です。

まずは With を使わない例を見てみましょう。次のコードを入力してください。

```
Sub セルの飾り付け ver1()
    ThisWorkbook.Sheets(1).Range("a1").Interior.Color = vbYellow
    ThisWorkbook.Sheets(1).Range("a1").Borders.LineStyle = xlSolid
    ThisWorkbook.Sheets(1).Range("a1").Font.Name = "メイリオ"
    ThisWorkbook.Sheets(1).Range("a1").Font.Color = vbRed
    ThisWorkbook.Sheets(1).Range("a1").Font.Bold = True
End Sub
```

Sheets(1)というのは新しく学ぶ指定方法ですが、シートは名前だけでなく何番目かという数値でも指定できます。

さて、上記のコードは同じ Range("A1")を 5 回も繰り返すので入力が面倒です。コピー&ペーストしても良いのですがコードが煩雑で見た目にも美しくないです。

これは、次のように書き直すことができます。

```
Sub セルの飾り付け ver1()
    With ThisWorkbook.Sheets(1).Range("a1")
        .Interior.Color = vbYellow
        .Borders.LineStyle = xlContinuous
        .Font.Name = "メイリオ"
        .Font.Color = vbRed
        .Font.Bold = True
    End With
End Sub
```

全部の行に書かれていた ThisWorkbook.Sheets(1).Range("a1")が一カ所にまとまって、ずいぶんすっきりしました。

このように共通の親項目に With 句を使用すると、親項目を省略してドットから記入することができます。

フォントの指定にも With を使ってみましょう。
With 句は以下のように入れ子にすることもできます。

```
Sub セルの飾り付け ver1()
    With ThisWorkbook.Sheets(1).Range("a1")
        .Interior.Color = vbYellow
        .Borders.LineStyle = xlContinuous
        With .Font
            .Name = "メイリオ"
            .Color = vbRed
            .Bold = True
        End With
    End With
End Sub
```

これで無駄な繰り返しがなくなりました。With はこのように、オブジェクトやプロパティの階層構造を省略するのに使用します。また、体感できるほどではありませんが、With を使うと、そうでない場合よりも実行スピードが速くなります。

マクロの記録を使用すると With 句も記録されることがありますので、覚えておいてください。

オブジェクト型変数

セルやシートなどのオブジェクトを格納する変数を、オブジェクト型変数と言います。オブジェクト型変数は**固有オブジェクト型変数**と**汎用オブジェクト型変数**に分けられます。

要するに、セルしか入らないセル型変数、シートしか入らないシート型変数などは固有オブジェクト型変数で、セルもシートもブックもなんでも入るのが汎用オブジェクト型変数です。

ではなんでも入る汎用オブジェクト型変数の方が良いかというと、そうでもありません。試しに次のコードを入力してみてください。

```
Sub 汎用オブジェクト変数テスト()
    Dim obj As Object
    Set obj = ThisWorkbook.Sheets(1)
    obj.Range("A1").Interior.Color = vbCyan
    Set obj = Nothing
End Sub
```

正しく動作しますが、obj.と入力しても入力候補が表示されず、行の最後まで手打ちすることになりましたね。ちなみに、Set と書いたのは、オブジェクト変数にオブジェクトを代入する際の決まり事で、これが無いとエラーになります。ただし、For Each の時だけは Set は不要です。また、最後に Nothing を代入するのも今のところは決まり事として覚えておきましょう。

では固有オブジェクト型を試してみましょう。

```
Sub 固有オブジェクト変数テスト()
    Dim obj As Worksheet
    Set obj = ThisWorkbook.Sheets(1)
    obj.Range("A1").Interior.Color = vbCyan
    Set obj = Nothing
End Sub
```

今度は obj.を入力した時点で入力候補が現れます。このように固有オブジェクト型を利用することでコード入力が楽になります。

ちなみに、ThisWorkbook.Sheets(1).と入力しても入力候補が出ないので、ワークシートは Worksheet 型の変数に入れてから使った方が便利です。あまり知られていませんが、シートにはワークシートの他にもグラフシートが存在するため、Sheets(1)だけでは Range プロパティが存在するかどうか分からないためです。ちなみにグラフシートの変数は、Chart 型です。

マクロの記録

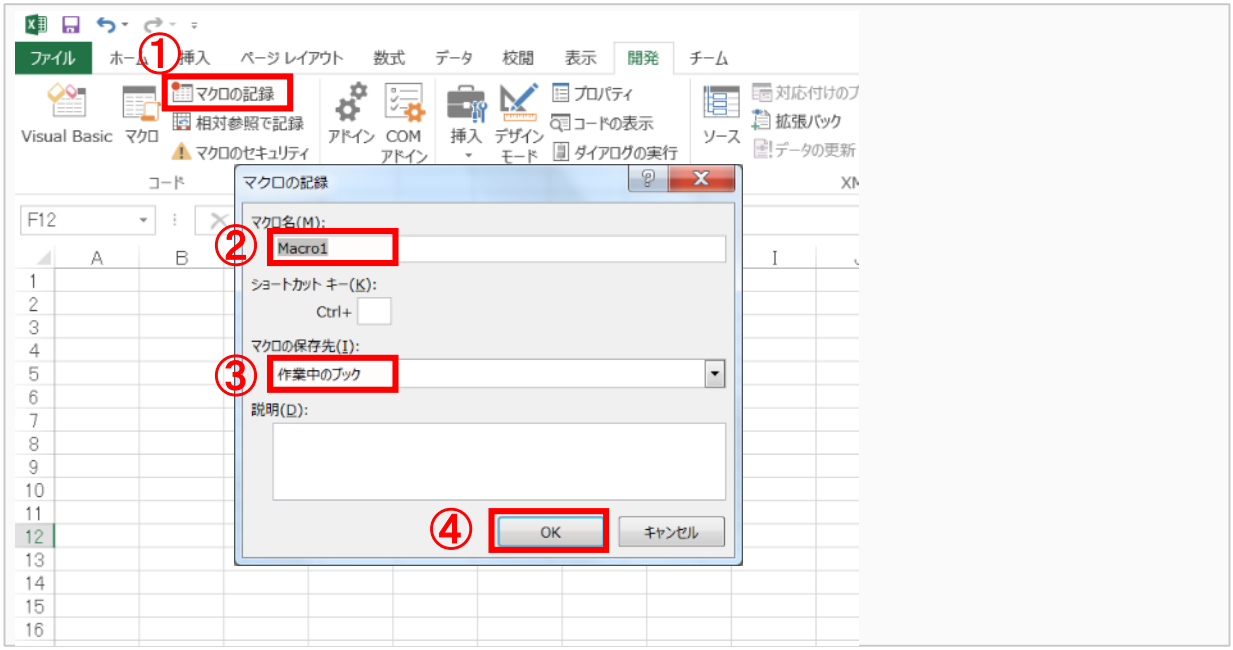
ようやくここまで来ました。これが最後の項目になります。

冒頭でマクロの自動記録だけでは役に立つプログラムは作れないと書きました。ではマクロの記録が何のために存在するかというと、Excel オブジェクトを操作するコードを調べるためです。

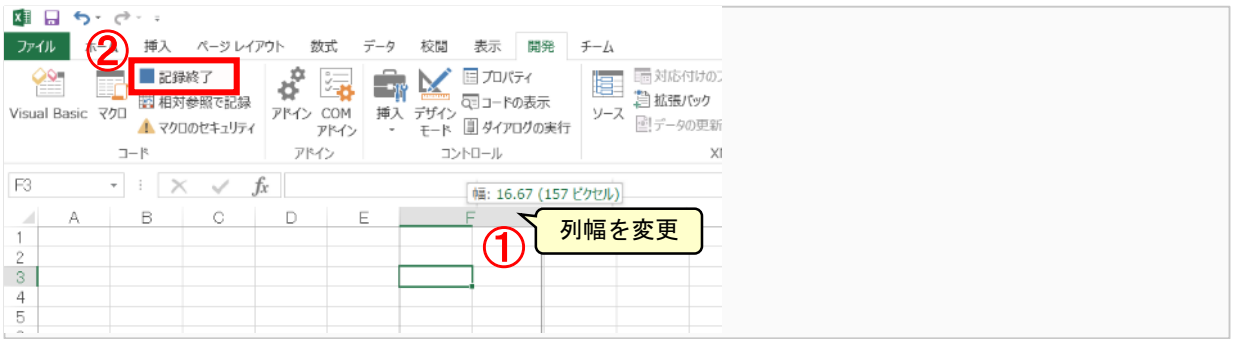
VBA のコードでシートを選択する方法が分からなかったり、セルの幅の広げ方が分からないとき、マクロの自動記録機能を使って操作に対応する VBA コードを調べます。

ではやってみましょう。

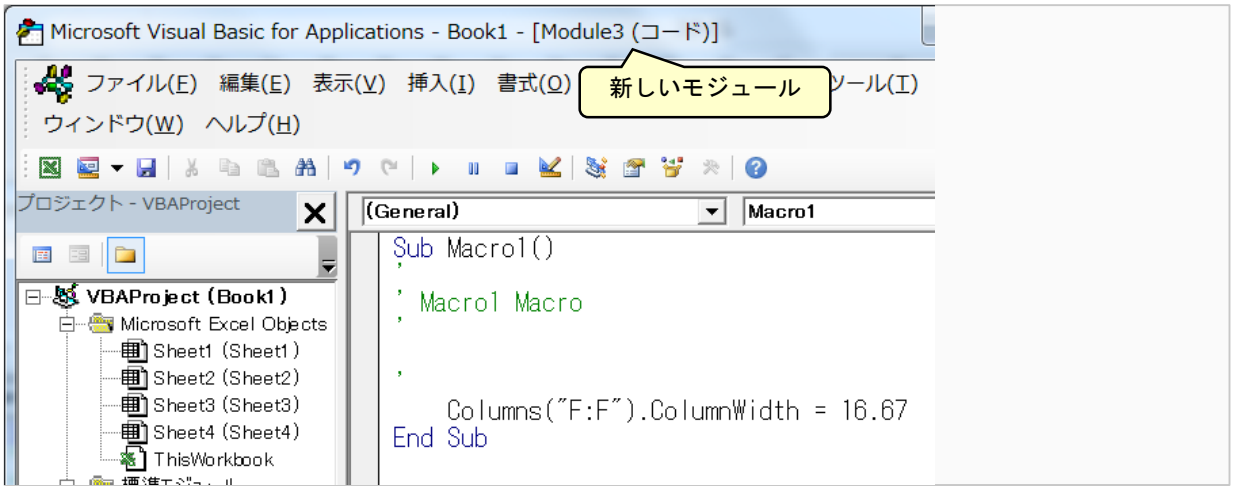
開発タブからマクロの記録ボタンをクリックしすると、次のようなマクロの記録ウインドウが表示されます。適当にマクロ名を決めて、作業中のブックに保存を選んだら、OK ボタンをクリックしてください。



OK をクリックするとマクロの記録がスタートしていますので、適当な列の幅を変えてみましょう。列幅の変更がおわったら、記録終了ボタンを押してください。



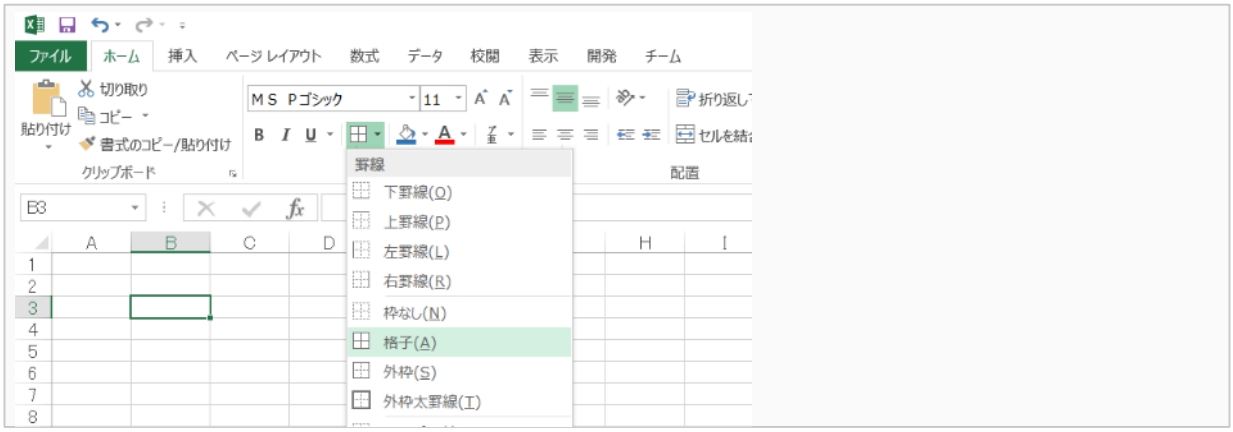
VB エディタを開くと新しいモジュールが追加されており、先ほどの操作が記録されています。



これで、列の幅を変える方法が分かりました。あとは好きなセルや数値に改変して使い回すことができます。慣れないうちは一度に記録するとどの操作がどのコードに対応するのか判別が難しいので、単体の操作をすこしずつ記録してコードを取り出すと良いです。

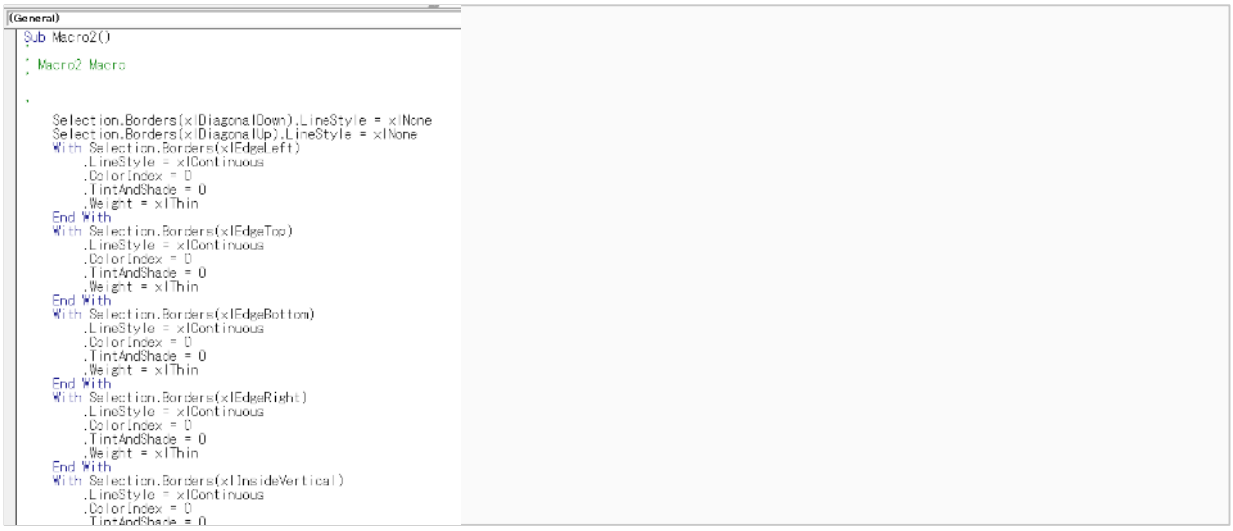
マクロの記録だけで手に負えないことも多々あります。たとえば罫線を引く処理を記録してみましょう。

マクロの記録を開始してからホームタブに戻り、適当なセルを選んで罫線を格子を設定してください。



その後、開発タブを開いて記録を終了し、VBA コードを見てみます。

たった一つ罫線を設定しただけなのに、なんと 45 行ものコードが記録されてしまいました。



```
(General)
Sub Macro2()
'
' Macro2 Macro
'

    Selection.Borders(xlDiagonalDown).LineStyle = xlNone
    Selection.Borders(xlDiagonalUp).LineStyle = xlNone
    With Selection.Borders(xlEdgeLeft)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
    End With
    With Selection.Borders(xlEdgeTop)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
    End With
    With Selection.Borders(xlEdgeBottom)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
    End With
    With Selection.Borders(xlEdgeRight)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
    End With
    With Selection.Borders(xlInsideVertical)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
```

まえがきで、自動記録されたコードは入門者には手に負えないと書きましたが、こういうことです。

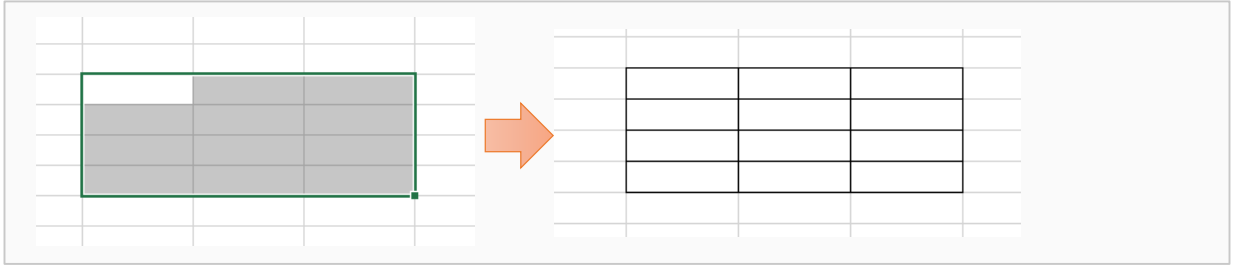
実はこのマクロ、次のように書き換えてもほぼ同じ動作をします。

```
Sub Macro2 シンプル化()
    Selection.Borders.LineStyle = xlContinuous
End Sub
```

今度はたったの 3 行です。マクロの記録が長いコードを出力するのは、それがコードを調べる機能だからです。本来指定が不要なデフォルト値も含めて出力されます。

自動記録の長いコードからシンプルなコードに編集するまでの流れを追っていきましょぅ。

Borders のカッコの中身はそれぞれの辺を表していますが、辺を指定せずに直接 Borders にプロパティを設定すれば、それは上下左右の四辺と選択範囲内の内部罫線に設定されます。斜めの線は個別に設定しない限り設定されません。



したがって、まず斜めに xlNone を指定している次の 2 行は削除します。xlNone は罫線なしのスタイルです。他の使い方としては、セルの背景色に xlNone を設定して塗りつぶしを消去することができます。

```
    Selection.Borders(xlDiagonalDown).LineStyle = xlNone
    Selection.Borders(xlDiagonalUp).LineStyle = xlNone
```

そして、他の Borders の処理は中身が一緒なので次のようにまとめることができます。

```
(General)
Sub Macro2_シンプル1()
    With Selection.Borders
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
    End With
End Sub
```

LineStyle 以外の値はデフォルト値なので、特に変更する必要はありません。

したがって With も外してしまって、先ほどの 3 行で完成です。

ただし、あらかじめセルの罫線が赤色などに設定されている場合は赤色のまま罫線が設定されますので、ColorIndex を 0 に設定してやると黒に戻ります。ColorIndex は Excel2003 以前の指定方法で、Excel の内部で管理しているカラーパレット番号を指します。

.Color = vbBlack でも同じ動作をしますので、こちらに書き換えるのがオススメです。

最初はマクロの記録だけでコードを調べると苦労しますので、本格的に学ぼうと思ったら VBA の逆引き辞典を 1 冊用意してください。「〇〇したい」といった目的からコードを引ける本がいくつか書店で売られていますので、気に入ったものを選ぶとよいでしょう。

インターネットで情報を集めるのも良いですが、基礎知識が付くまでは情報を探そうにもキーワードが分からないと思うので、入門段階では紙の本がオススメです。

以上でこの入門書での学習はおしまいです。いかがでしたでしょうか。一度読んだだけで理解するのは難しいと思います。不思議なもので、一晩寝て朝起きたら分からなかった箇所が分かるようになったり、時間をおいて読み返したらなぜか簡単に謎が解けたりするものです。

今ひとつ理解できなかった箇所はもう一度読み直したり、コードを入力したり、工夫しながら復習してみてください。

練習問題の解答例

以下に練習問題の解答例を掲載します。あくまで一例ですので、これ以外の方法でも問題の条件を満たしているなら正解です。

解答 1

```
Sub 変数の入れ替え()
    Dim A As Integer
    Dim B As Integer
    Dim 保管 As Integer
    A = 10
    B = 30

    保管 = A
    A = B
    B = 保管

    MsgBox "A の中身は" & A & "。B の中身は" & B & "。"
End Sub
```

解答 2

```
Sub フィズバズ()
    Dim i As Integer
    For i = 1 To 100 Step 1
        If i Mod 3 = 0 And i Mod 5 = 0 Then
            Debug.Print "FizzBuzz"
        ElseIf i Mod 3 = 0 Then
            Debug.Print "Fizz"
        ElseIf i Mod 5 = 0 Then
            Debug.Print "Buzz"
        Else
            Debug.Print i
        End If
    Next i
End Sub
```

解答３

```
Sub 九九()
    Dim i As Integer
    Dim j As Integer
    Dim 列 As String
    For i = 1 To 9 Step 1
        For j = 1 To 9: Debug.Print i & " * " & j & " = " & i *
j: Next
    Next i
End Sub
```

エキストラ問題の解答

```
Sub 九九の整形表示()
    Dim i As Integer
    Dim j As Integer
    Dim 行 As String
    For i = 1 To 9 Step 1
        行 = ""
        For j = 1 To 9
            If i * j < 10 Then
                行= 行 & " "
            End If
            行 = 行 & i * j & " "
        Next j
        Debug.Print 列
    Next i
End Sub
```

オススメの書籍

以下に、私がオススメする市販の入門書を列挙しておきます。市販の入門書は最初にマクロの記録を紹介していますが、文法をしっかり身につけた後で読む分には必要な情報が網羅されており非常に有用です。

かんたんプログラミング Excel 2010 VBA 基礎編

あくまで私の主観による評価ですが、入門書としてはこの本がいちばん分かりやすく整理されているように思います。Amazon での評価も上々でしたので、一度書店で手にとってみてください。

基礎編、応用編、コントロール・関数編の 3 種類が出版されています。他に、Excel 2007 版もあります。

Excel 2013 と Excel 2010 の違いはあまり気にする必要はありませんが、2007 を使っている場合は多少の違いがありますので、2007 版を購入するのが良いでしょう。

Excel VBA 逆引き辞典パーフェクト 2013/2010/2007/2003 対応

こちらはやりたいことから逆引きするための辞典です。Kindle 版も出ていますが、検索しにくいと不評ですので、紙の本をオススメします。

辞典タイプの本は他にもいくつか出版されていますが、網羅している内容は似たり寄ったりなので、説明を読んでみて分かりやすいと思ったものを買うと良いと思います。

クレジット

このドキュメントのオリジナル版は下記の URL でダウンロードできます。

http://www.thom.jp/

タイトル　Excel VBA 入門教材 急がば回れ！文法から覚えるやさしい VBA 入門

2014 年 1 月 4 日　公開

2015 年 6 月 7 日　改定

著者　本村（ほんむら）　貴之（たかゆき）

以上