作って学べる VBA 講座（マクロを作って実感しよう）

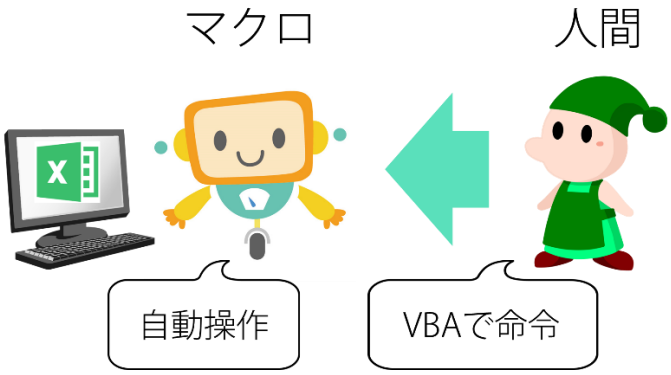
内容

【はじめに】この講座はどんなもの？

１つのマクロ（Excel プログラム）を作りながら、VBA とはどんなものか？マクロとはどんなものか？それらを体験的に理解することができる講座です。

【第０章】準備編

【0-1】マクロと VBA って？



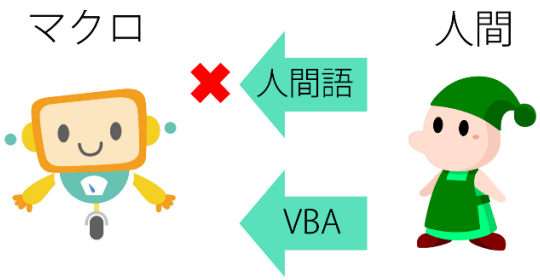
この講座では、VBA を使って請求書を作成するマクロを作ります。

…と、その前に。マクロと VBA とは何でしょうか？

何となく聞いたことはあるけれど、「説明してみて下さい」と言われるとハッキリと答えられないかもしれません。この問いに答えられるのが上記の図です。

- マクロは、Excel を自動操作してくれる仕組み（プログラム）のことです。
- 人間は、マクロにどう働いてもらうのかを「VBA」という言葉で命令します。

ところで、なぜわざわざ「VBA」という言葉を使うのでしょうか？



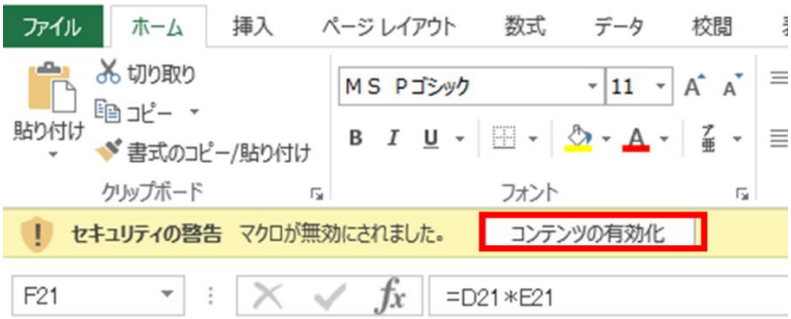
それは、マクロは人間語を理解できないからです。そこで、より機械の言葉に近い「VBA」という言葉を使ってマクロに命令を伝えるのです。

【0-2】お手本マクロを試してみよう

本講座で作成するマクロ「１タッチで請求書」がどんなものか、実際に動かして確認してみましょう。

VBA課題お手本.xlsm

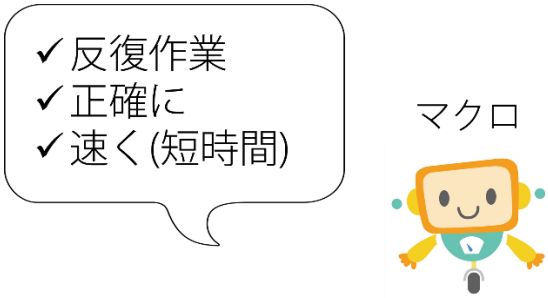
①ファイル「VBA 課題お手本.xlsm」を開きます。
②このような画面が表示された場合、「コンテンツの有効化」



③それぞれボタンをクリックしてみましょう。
（「請求書作成」ボタン、「クリアボタン」、「PDF 出力」ボタンなど）

※お手本マクロを閉じる：
次の講座のために、お手本マクロは閉じておきましょう。

【0-3】マクロの利点



- 反復作業…何十回、何百回と反復する作業でも、自動的に行ってくれます。
- 正確に作業…大量のデータを扱う作業でも、正確に行ってくれます。
- 速く（短時間）…人間が行ったなら時間がかかる作業を、とても速く行ってくれます。

【0-4】「開発タブ」の準備

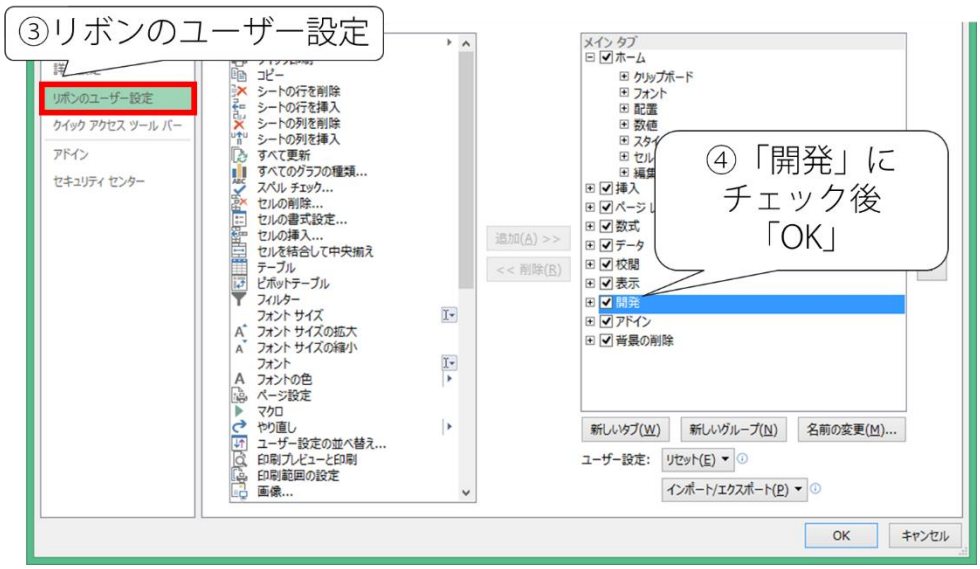
VBA がどんなものか理解するために、とにかく始めてみることが大切です。
まずは準備をしていきましょう。

Excel で新しいブックを作成しておいて下さい。

…と、その前に、VBA を始めるためには、あなたの Excel に「開発タブ」が表示されている必要があります。
このようなタブが表示されているかご確認下さい。

もし表示されていない場合は、以下の設定をして下さい。
①「ファイル」をクリック
②「オプション」をクリック

③「リボンのユーザー設定」をクリック
④「開発」にチェックを入れて「OK」をクリック



【0-5】VBE を使ってみよう

「開発」タブが表示されたら、いよいよ VBA を入力していきます。

「開発」タブをクリックして、一番左の「Visual Basic」をクリックして下さい。

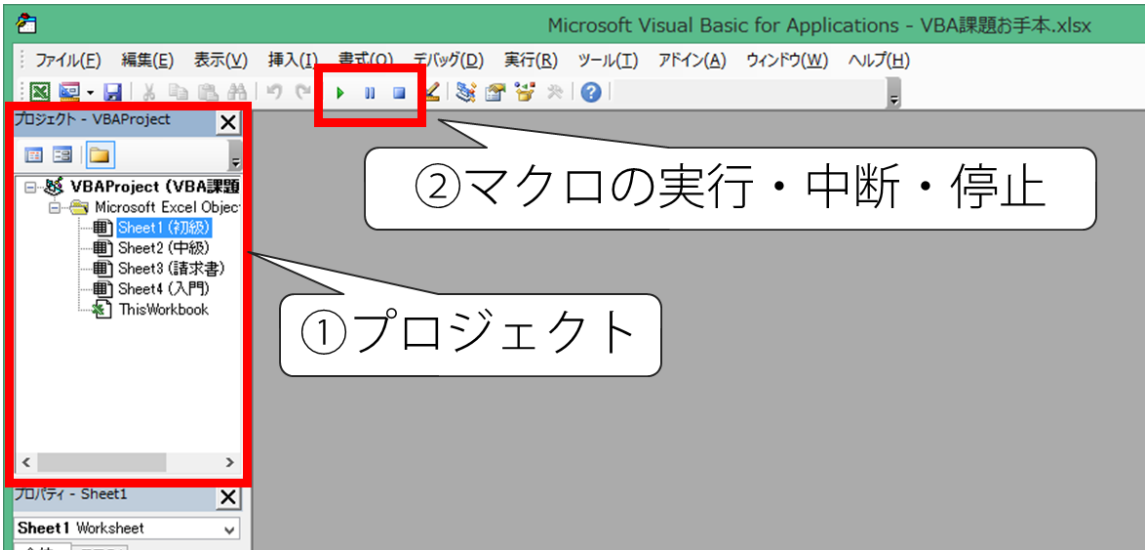
このような画面が表示されたかと思います。

これは、VBA を編集するための「VBE」と呼ばれる画面です。

ひとまず入門者が覚えるべき場所は、２箇所です。

①プロジェクトの要素一覧

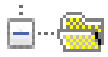
②マクロの実行・中断・停止ボタン



【0-6】標準モジュールを追加

もう少しで、準備は終わりです。

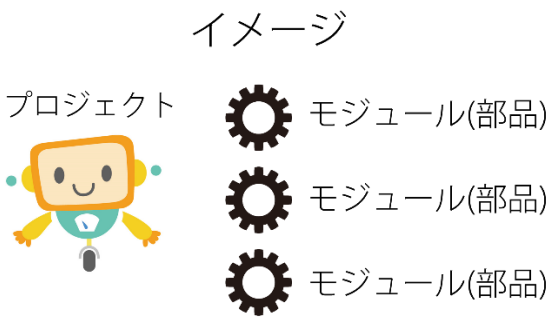
VBA のコードを入力し始めるために、「標準モジュール」というものを追加しましょう。

①  を右クリック

②「挿入」>「標準モジュール」をクリック

すると、「標準モジュール　Module1」という物が追加され、右側には真っ白な欄が表示されます。ここが、VBA のコードを入力するための欄という事です。

【補足】モジュールって何？



ところで、モジュールとは何でしょうか？

VBA では、ブックのマクロ全体を「プロジェクト」という単位で呼びますが、その一つ一つの構成要素（部品のようなもの）を「モジュール」と呼びます。

【0-7】機能を作ろう（Sub プロシージャ）

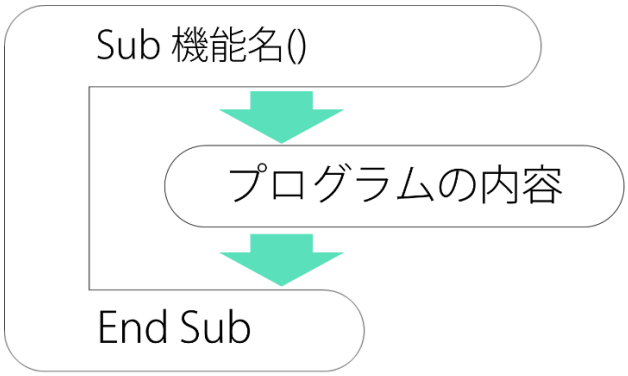
それでは、以下のようにコードを入力してみましょう。

✎【新しく入力してみよう】
Sub Seikyu() Enter キー

すると、このように自動的に表示されたはずです。

```
Sub Seikyu()

End Sub
```

VBA では、新しい機能を作るときには「Sub 機能名() ～ End Sub」でくくる約束があります。
（Sub 以外にも記述方法は色々ありますが、今は置いておきましょう。）



ですので、「Sub 機能名()」を入力して Enter キーを押すと、自動的に「End Sub」が挿入されるのです。

今回は「Sub Seikyu(請求)()」という機能名を私が名づけました。（今日は請求書作成マクロを作るからです。）このように、機能名は、その機能の内容に応じた名前を付けるとよいでしょう。
VBA 用語ですが、単一の機能のことを「プロシージャ」と呼びます。

【0-8】最初の体験「あ」（MsgBox 関数）

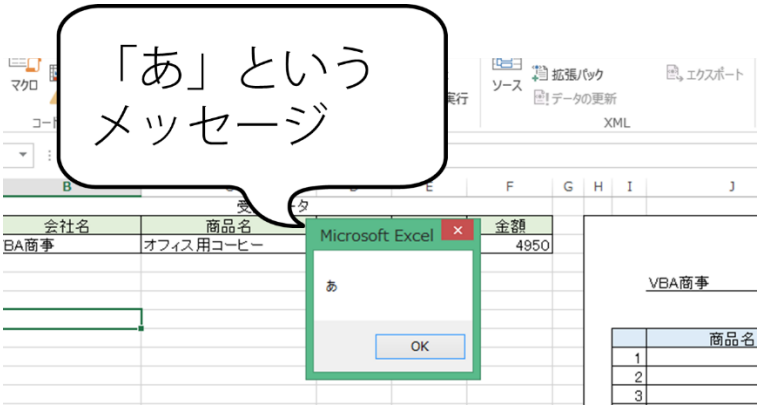
それでは、いよいよプログラムの中身を入力していきます。以下のように入力して実行してみてください。

✎【入力してみよう】

```
Sub Seikyu()
    MsgBox "あ"
```

```
End Sub

【実行】
```

入力できたら、  の左にある  実行ボタンを押してみましょう。



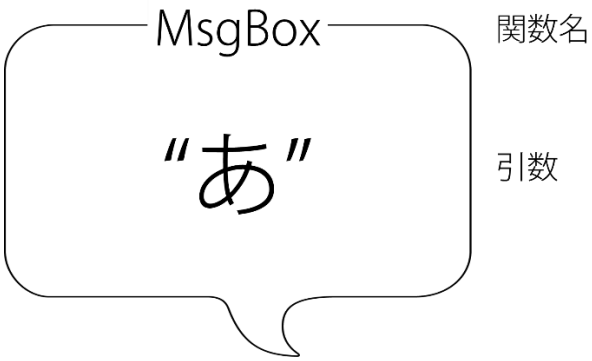
いかがでしょうか？このようなメッセージが表示されれば成功です。
（確認できたら、「OK」をクリックしておいて下さい。）

【0-9】関数名と引数

何となく、感覚的に理解できたでしょうか？

```
    MsgBox ″あ″
```

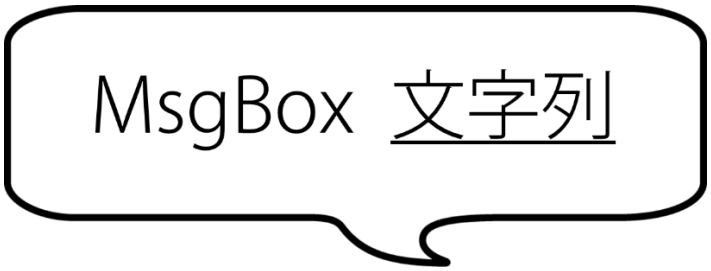
この一文は、『メッセージボックスで「あ」と出力させなさい』ということですね。



「MsgBox」のような命令を「関数」と呼びます。「”あ”」のような要素を「引数」と呼びます。

```
MsgBox  “あ”

関数名  引数（ひきすう）
```

このように、VBA で命令を記述する時には関数名と引数を組み合わせて使うことが多くあります。MsgBox 関数はこれから多く使用するので、覚えておきましょう。



【✍覚えておこう】
MsgBox 文字列
メッセージボックスで文字列を表示させる。

【0-10】「こんにちは」と表示させてみよう
それでは、次のようにコードを変えてみましょう。

✎【入力してみよう】

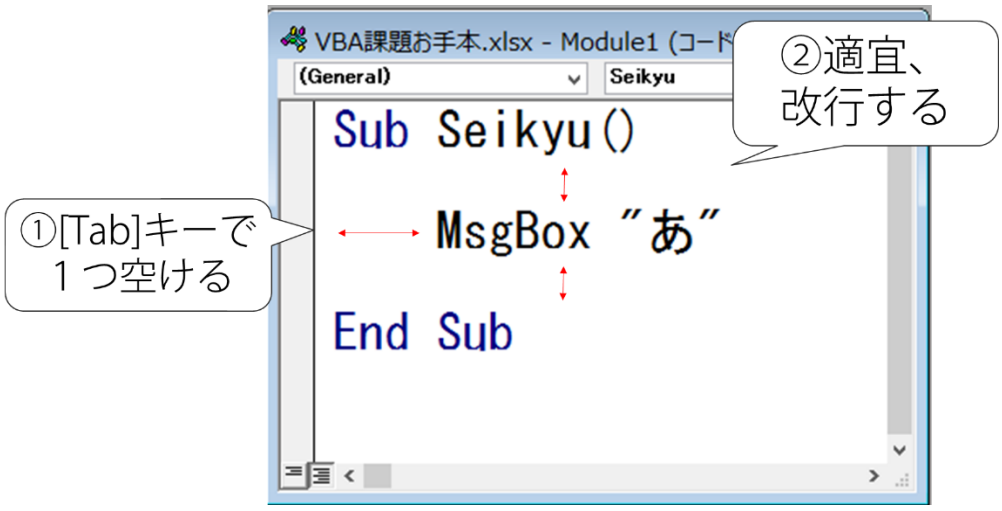
```
Sub Seikyu()
    MsgBox ″こんにちは″
End Sub
```

【実行】

いかがでしょうか。今度は「こんにちは」と表示されます。命令は同じ「MsgBox 関数」ですが、引数を変えるだけで出力される内容が変わることがわかりますね。

補足：「"」（ダブルクォーテーション）を使おう
このように、文字列を使う際には「″（ダブルクォーテーション）」で囲いましょう。これは、コンピュータに「これは数字ではなく文字列ですよ」と分かるように伝えるためです。

補足：コードを綺麗に書こう
VBA のコードを書く時には、後で見やすいように以下のことを意識しましょう。
①Sub～End Sub の間に書き込む命令は、[TAB]キーで１つ分空けて書く。
②適宜、改行をして行間を空ける。



【0-11】変数を使ってみよう

次に、変数という概念を使ってみます。
ここからは「変数」という概念を使っていきます。
次のようにコードを変更してみましょう。

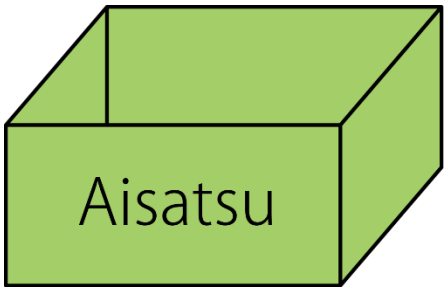
✎【入力してみよう】

```
Sub Seikyu()
    Dim Aisatsu As String
    Aisatsu = "こんにちは"
    MsgBox Aisatsu
End Sub
```

【実行】

さて、いかがでしょうか？
さっきと同じように「こんにちは」と表示されましたね。
これは、「変数」という仕組みを使ったものです。

変数というものは、値を入れておく「箱」のようなものです。
変数という「箱」に、数字や文字を入れておくことで、プログラムは様々な処理をすることができます。今回は、「Aisatsu」という名前の箱を作りました。




```
    Dim Aisatsu As String
```

この一文は、「Aisatsu」という名前の「変数の箱」を用意する、という意味です。

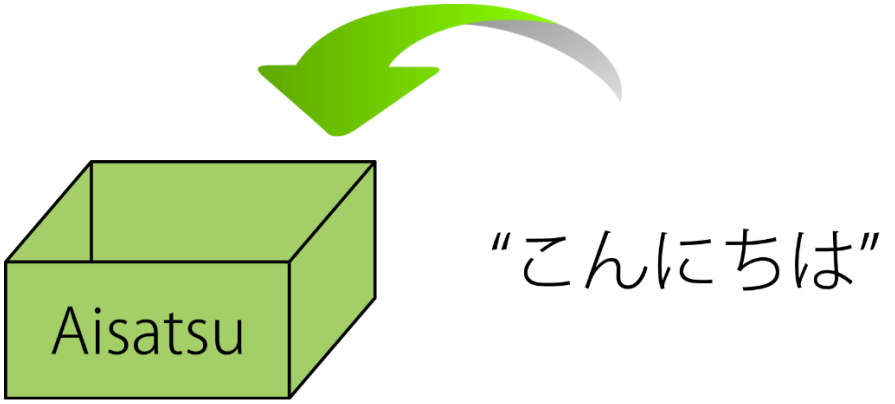
- Dim　定義する。
- Aisatsu　「Aisatsu」という名前の
- As String　（これは「文字列型の」、という意味ですが、後で詳しく解説しますので、今はまだ「そういう決まり文句だ」と思って置いておいてください。）

この一行目によって、「Aisatsu」という名前の箱が作られた、ということです。（これを箱「Aisatsu」と呼ぶことにします）

しかし、まだこの箱「Aisatsu」には、何も入っていません。空っぽの状態です。
次の二行目で、箱の中に文字を入れ込んでいきます。

```
    Aisatsu = "こんにちは"
```

これは、先ほど作った箱「Aisatsu」の中に、「こんにちは」という文字列を入れ込んだという意味の一文です。



つまり、この一文によって、箱のなかに「こんにちは」という文字が入ったということになります。

【補足】「＝」はイコールじゃない！？

違和感を覚える方もいるかもしれませんが、プログラミング言語では、変数の箱に何かを入れる（代入する）ことを「=」の記号であらわします。

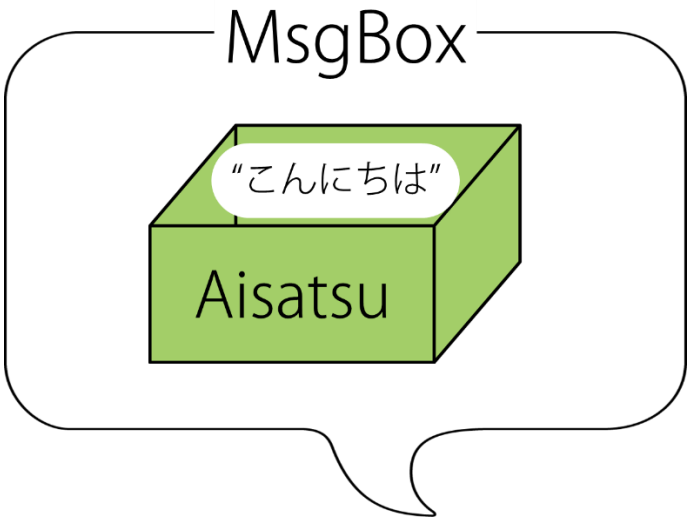
Aisatsu　＝〝こんにちは〟　…という一文の意味するところは、

Aisatsu　←〝こんにちは〟　…このような意味合いがあります。

（個人的には、この記載方法の方がわかりやすいと思うのですが、いろいろな理由から「=」の記号が使われているのだと思われます。）

```
    MsgBox Aisatsu
```

この一文は、箱「Aisatsu」の中に入っている文字を、メッセージボックスで出力するという意味です。



この一文で、いよいよ箱「Aisatsu」の中に入っている文字が表に出てくることになります。

【補足】変数とは…

いま「箱」と表現したものを、プログラミングでは「変数」と呼びます。

変数には以下のような性質があります。

・自由に名前をつけて定義する（作り出す）ことができる。

・様々なデータを代入（入れ込む）することができる。

・代入されたデータを出力して（引き出して）利用することができる。

先ほどの３行の文では、

①箱「Aisatsu」を定義する。

②箱「Aisatsu」の中に、データ「こんにちは」を入れ込む。

③箱「Aisatsu」のデータを引き出して、メッセージボックスで表示させる。
ということを行いました。

【0-12】なぜ、わざわざ変数を使うの？

さて先ほどのように、わざわざ変数を使って記載したことには何の理由があるのでしょうか。

その理由は、下記の３つです。

（１）あとで修正・変更が楽になる。
（２）何度も反復して使うことができる。
（３）その他の理由（後述）

例を用いて説明します。このような例をご覧ください。

（例）

```
MsgBox ″田中さん、こんにちは″
MsgBox ″佐藤さん、こんにちは″
MsgBox ″鈴木さん、こんにちは″
```

同じような文言を３行にわたって書く場合です。このような場合、入力が大変ですね。

第一に、「こんにちは」と３回も書くことはとても面倒ですね。
３回くらいならまだマシかもしれませんが、もし同様のことを 100 回行うようなマクロを作らなければならなかったら？100 回も「こんにちは」と打ち込まなければならないのは、いかにも面倒です。

第二に、あとで変更する際に大変です。
例えば、このように変更しなければならないとしたらどうでしょうか。

【修正】
「こんにちは」をすべて、→「こんばんは」に変更しなければならない。

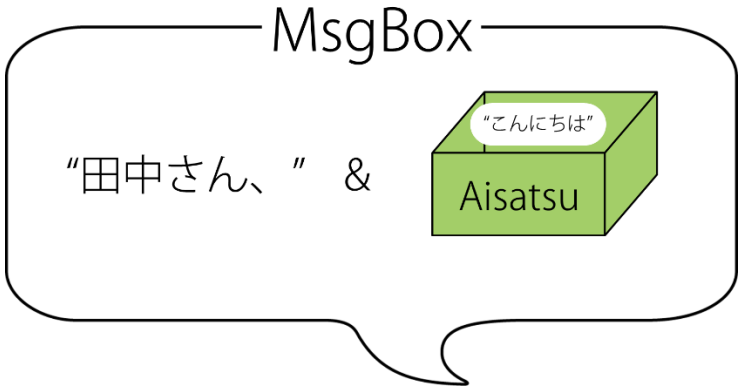
このような修正を後から行うときには、一つ一つの「こんにちは」を削除して、「こんばんは」に書き換えなければならず、とても面倒です。

【変数を使った場合は？】
それでは、変数を使って「例」を書き換えた場合、どのように記載すればよいでしょうか。

（変数を使って書き換えたもの）

```
Dim Aisatsu As String
Aisatsu = ″こんにちは″
MsgBox ″田中さん、″ & Aisatsu
MsgBox ″佐藤さん、″ & Aisatsu
MsgBox ″鈴木さん、″ & Aisatsu
```

このように書き換えることができます。
変数の箱「Aisatsu」の中には「こんにちは」という文字列が入っているので、それをメッセージボックスで出力しているのです。



【補足】「&」という記号について
VBA では、文字列を結合させる（くっつける）場合には「&」の記号を用います。
例えば、 ″田中さん、″ と 箱「Aisatsu」の中身をくっつける場合には、
″田中さん、″ & Aisatsu のように記述します。

覚えておこう
(文字列をつなげる)

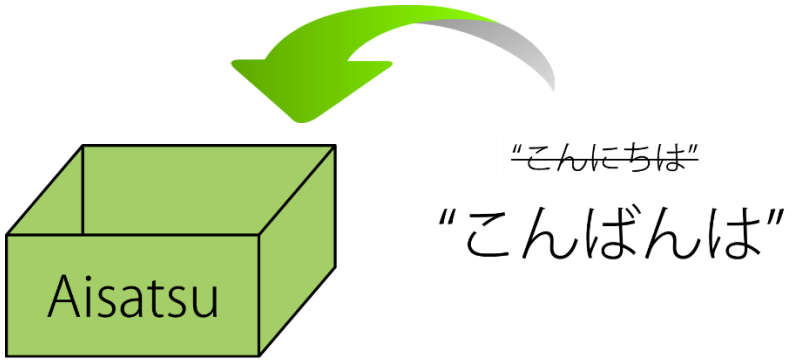
✓ 文字列　＆　文字列
✓ 文字列　＆　変数名

それでは、以下のように修正をしなければならなくなった場合、どう変更すれば良いでしょうか？

> 【修正】
> 「こんにちは」をすべて、→「**こんばんは**」に変更しなければならない。

いかがでしょうか。今回の場合は、一行だけ変更すれば作業が済みます。

> （変更前）
> Aisatsu = ″こんにちは″
>
> （変更後）
> Aisatsu = ″こんばんは″



上記のように、変数を用いておけば、たった一行を修正するだけで全てのプログラムが修正されることと同じことになります。とても便利ですね。

他にも変数を用いるメリットはとてもたくさんあるのですが、今は割愛して後述いたします。
とにかく今の段階では、「何度も同じことを記述しなければならない文がある場合は、変数を使うと便利になることがあるんだ」というように何となく理解しておいていただければ結構です。

【補足】変数は「"」で囲わなくていいの？

ところで、「″」を使って Aisatsu を囲い、「″Aisatsu″」と記載しなくてもいいのか？
という疑問を持たれた方もいるかもしれません。
箱「Aisatsu」を表す場合、「″」で囲ってはいけません。

```
Sub Seikyu()
    Dim Aisatsu As String
    Aisatsu = ″こんばんは″
    MsgBox Aisatsu      ‘（正）
    MsgBox ″Aisatsu″    ‘（誤）
End Sub
```

上記の（正）と（誤）には意味の違いがあります。

実際にマクロを実行するとわかりますが、
（正）の場合では、メッセージボックスに「こんばんは」と出力されます。
（誤）の場合では、メッセージボックスに「Aisatsu」とそのまま出力されます。

「″」で囲ってしまうと、エクセルは『「Aisatsu」は変数の箱ではなくて、ただの文字なんだな』として認識してしまうので、メッセージボックスでもそのまま「Aisatsu」と出力されてしまうのです。

変数を用いる場合は、「″」で囲わないようにする、という風に覚えておいてください。

【0-13】プログラミングで大切な３ポイント

以下、プログラミングを覚えておきたい大切なポイントです。

- 変数　　今回勉強しました。
- 条件分岐　初級で勉強します。
- 反復処理　初級で勉強します。

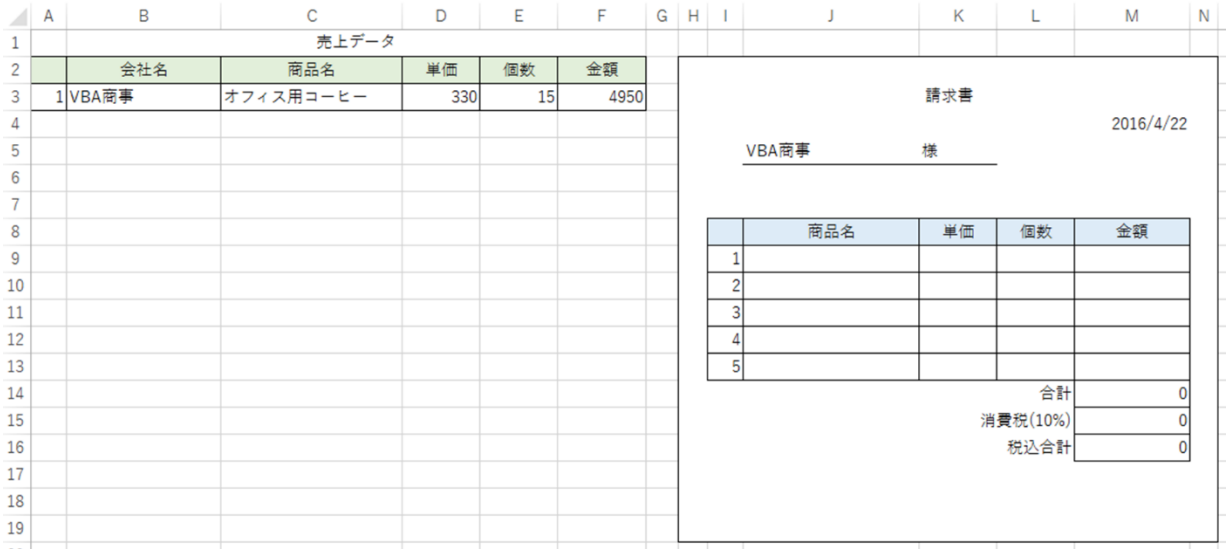
【第１章】入門編「かわいい請求書マクロ」を作ろう
この講座では、レベルに合わせて３種類の「１タッチで請求書」を作成しながらマクロの基礎を学びます。

・入門編「かわいい請求書マクロ」
・初級編「やさしい請求書マクロ」
・中級編「使える請求書マクロ」

第１章入門編では、最もシンプルな請求書作成マクロを作りながら、VBA の基本的な扱い方を学んでいきましょう。

<u>【1-1】課題ファイルを開こう</u>
下記のファイルを開いて下さい。

①「VBA 課題.xlsx」を開いて下さい。
②「入門」シートを開いて下さい。



	会社名	商品名	単価	個数	金額
1	VBA商事	オフィス用コーヒー	330	15	4950

請求書

2016/4/22

VBA商事　様

	商品名	単価	個数	金額
1				
2				
3				
4				
5				

合計	0
消費税(10%)	0
税込合計	0

【どんな作業をするためのマクロを作るの？】
今回の課題で作成するものは、「請求書」を作るためのマクロです。

このシートの左側には注文の一覧が書かれています。
そして、右側には請求書のフォーマットが書かれています。

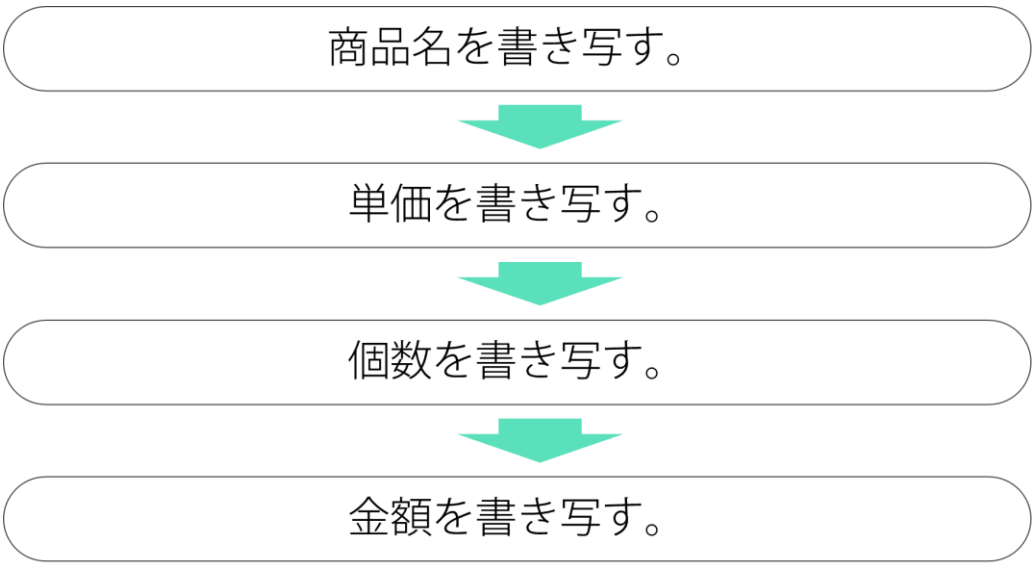
つまり、左側の「注文一覧」から情報を抜き出して、「請求書」に書き出すということで請求書を作成することができます。

【1-2】マクロなしで考えてみよう

マクロを作る前に大切なことは、いきなりプログラムを書かないことです。
まずはマクロなしで（人の手で）この請求書を作る場合、どんな操作が必要なのか？
これを考えてみることが大切です。

もしもマクロなしで、通常どおりに人の手で Excel を操作してこの請求書を作るとしたら、どうするでしょうか？

売上データから請求書に、「商品名」を書き写す。
売上データから請求書に、「単価」を書き写す。
売上データから請求書に、「個数」を書き写す。
売上データから請求書に、「金額」を書き写す。



このような操作が必要になるとわかります。

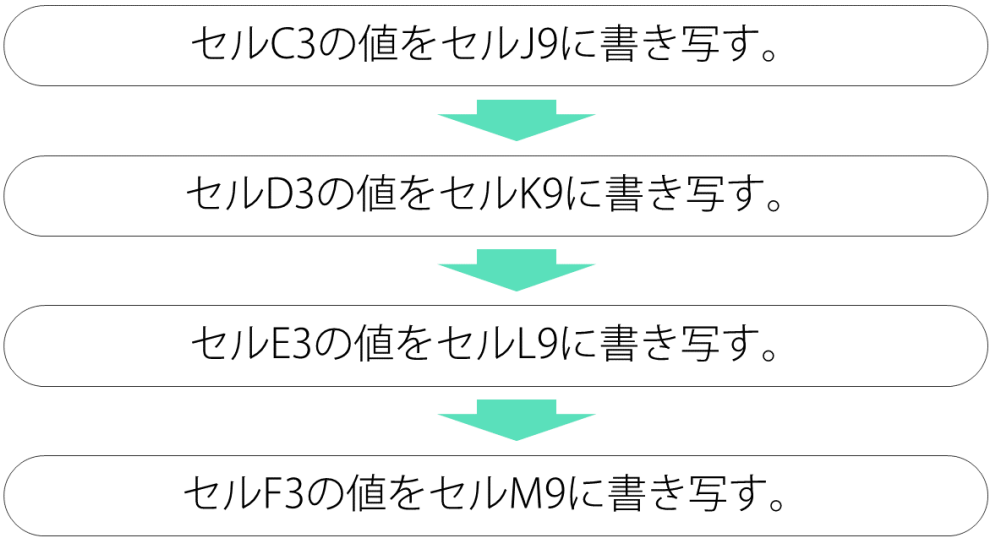
もう少し具体的に、Excel シートのセル番号を使って考えてみます。
例えば、「売上データ」の表において商品名（「わかりそうなエクセル」という商品）は、セル C3 に記載されています。それを請求書に書き写すのですから、セル J９に書き写すということになります。

つまり、上記の操作をセル番地でいうと

セル C3 の内容をセル J9 に書き写す。
セル D3 の内容をセル K9 に書き写す。

セル E3 の内容をセル L9 に書き写す。
セル F3 の内容をセル M9 に書き写す。

このような操作が必要だということがわかります。



この考えが大切です。
Excel のマクロという機能は、このように人間が手動で行う単純作業を Excel に自動的にやらせるということだからです。

VBE の画面を表示する：
次の操作のために、開発>Visual Basic　をクリックしておきましょう。

【1-3】セルの値を参照（Cells().Value）

それでは、いよいよ請求書作成マクロを作っていきます。
まずはウォーミングアップです。VBE ではこのように記載して下さい。

✎【入力してみよう】

```
Sub Seikyu()
    MsgBox (Cells(3, 4).Value)
End Sub
```

【実行】

コードの内容を見てみましょう。

```
MsgBox (Cells(3, 4).Value)
```

この関数は、先ほどから何度か使っていますね。
引数の内容をメッセージボックスとして出力させるという命令です。

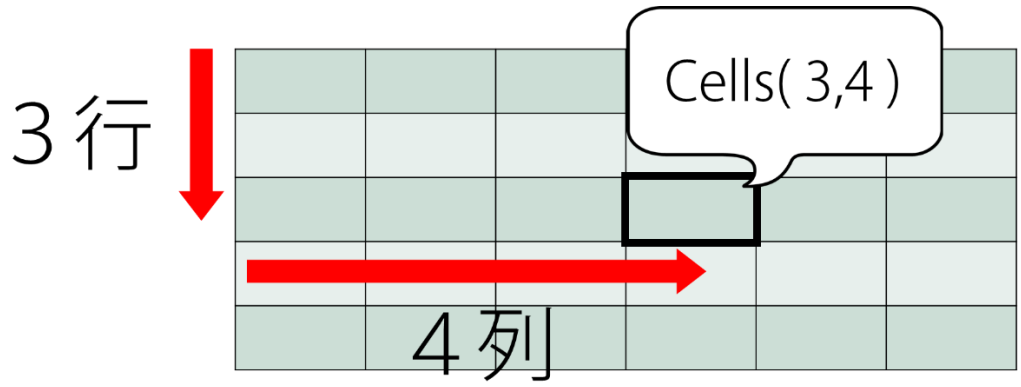
では、中に入っている
`Cells(3,4).Value`
とは何でしょうか？

これは VBA の機能ではとても大切なことなので覚えておきましょう。

Cells(3,4)	Excel の 3 行目 4 列目のセルの…
.Value	数値を

という意味です。

これは、表にするとわかりやすくなります。

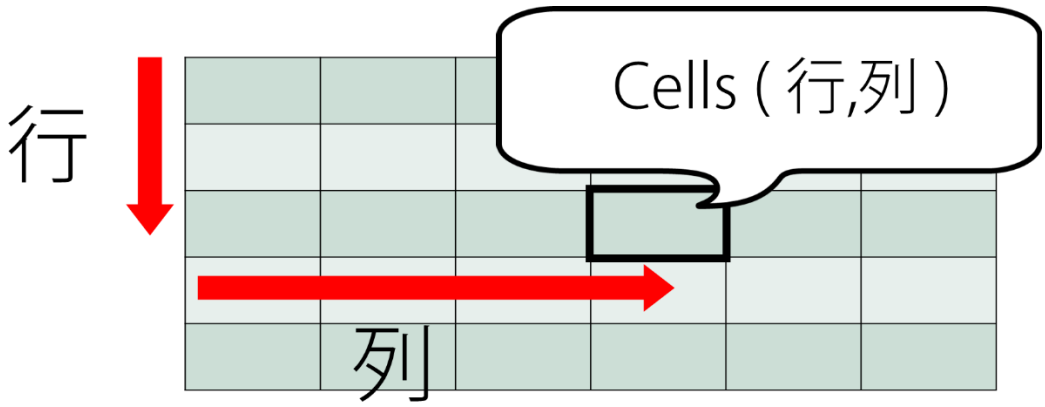


エクセルの表は、「行」と「列」でマス目を数えます。

Cells(3,4)とは、3 行目の 4 列目を意味しますので、
上から 3 段目の左から 4 番目と言い換えることもできます。

✍【覚えておこう】

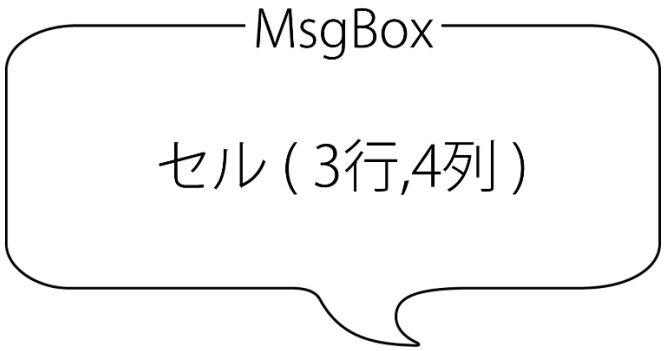
Cells（ 行 , 列 ）



それではコードに戻りましょう。

```
MsgBox (Cells(3, 4).Value)
```

この一文の意味することは、
上から 3 段目の左から 4 列目にあるセルの数値を、メッセージボックスに表示させなさい。
ということになります。



ですから、この命令を実行させると、メッセージボックスで
「わかりそうなエクセル」と表示されたことが確認できたでしょうか。

✎【試してみよう】
試しに、このように書き換えて実行してみましょう。
①MsgBox(Cells(3,5).Value)
②MsgBox(Cells(3,6).Value)
③MsgBox(Cells(2,3).Value)
【実行してみよう】

「Cells」の使い方がわかってきたでしょうか？
VBA でマクロを作る際には、とても重要な記述方法ですので、これから使いながら覚えていきましょう。

【1-4】請求書作成ボタン Lv.1（セルを書き写そう）

それでは、いよいよ本題である請求書マクロの制作にとりかかっていきます。
コードを書き変えて、このように記述して下さい。

✎【入力してみよう】

```
Sub Seikyu()
    Cells(9, 10).Value = Cells(3, 3).Value
End Sub
```

【実行】

シートの右側にある請求書のフォーマットに、「オフィス用コーヒー」と書き移されれば、成功です！

では、この一文は何を意味しているでしょうか。考えてみます。

Cells(9,10).Value
上から９段目の左から１０番目のセルの数値
つまり、請求書のフォーマットにあるセル「J9」を意味しています。

Cell(3,3).Value
上から 3 段目の左から 3 番目のセルの数値
つまり、請求書のフォーマットにあるセル「C3」を意味しています。

「=」は値を代入することを意味するので、次のような意味となります。

Cells(9, 10).Value	=	Cells(3, 3).Value
セル J9 の値に	代入して下さい、	セル C3 の値を。

【補足】「＝」は「←」！？

日本語とは逆の順番の言い回しになって、遠回しのように見えますが、これは慣れていくしかありません。

このように記載した方が、感覚的にはわかりやすいかもしれません。

Cells(9, 10).Value　　←　　Cells(3, 3).Value セル J9 の値に　　代入して下さい、　セル C3 の値を。

何となく、感覚的に理解できたでしょうか？

セルからセルに値を書き写すとき、「=」を使うと覚えておきましょう。

【補足】Delete で削除してもう一度

ところで、このマクロを何度も実行しても、一度目と結果は変わりません。

なぜかというと、すでにセル J9 には「わかりそうなエクセル」と書き込まれているからです。

すでに書き込まれているセルに、同じものを何度書き写しても見た目は変わりませんね。

では、セル J9 をクリックして「Delete」キーを押して、値を削除して下さい。

それから、マクロを実行してみて下さい。

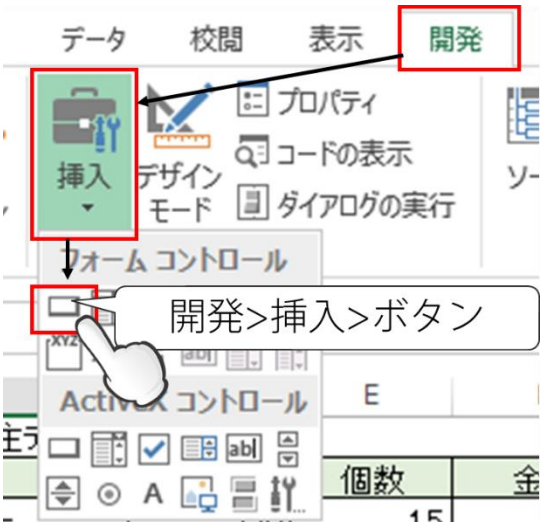
そのようにセルの値を削除したり、マクロを実行したりを繰り返せば、マクロの動作を何度も確認することができます。

【1-5】ボタンで実行されるようにしよう

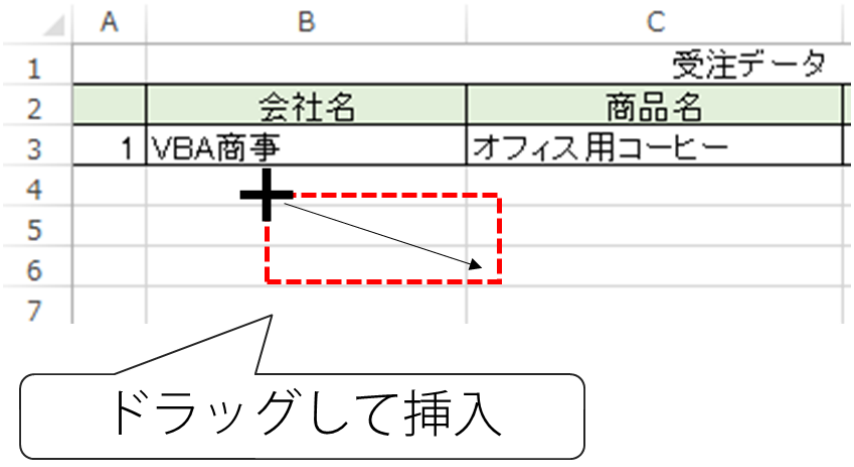
お手本のように、Excel のシート上で「ボタン」をクリックしたらマクロが実行されるように設定をします。これが出来ると、非常にかっこ良くなります。

※Excel の画面を表示させる：
次の操作のために、Excel の画面に戻っておきましょう。

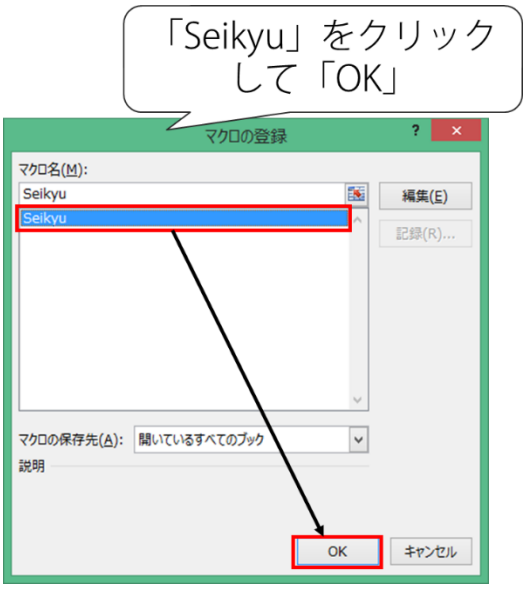
①「開発」タブ>「挿入」ボタン>「ボタン」をクリック



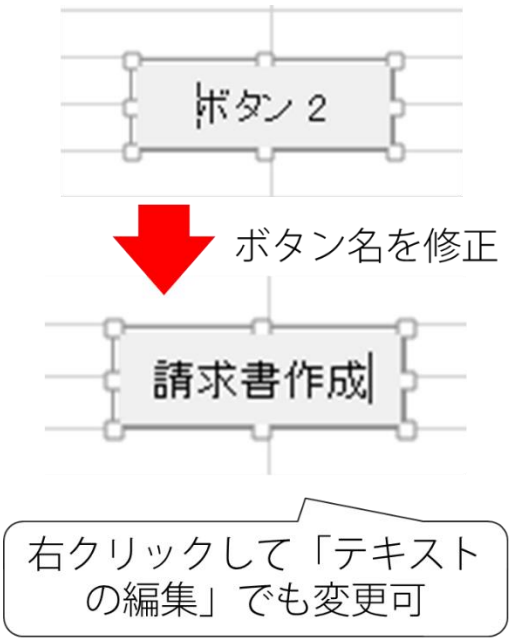
②マウスをドラッグして、ボタンを挿入



③「Seikyu」をクリックして「OK」



④ボタン名を「請求書作成」に変更



これで、ボタンをクリックしたらマクロ「請求書作成」が実行されるようになります。

【1-6】クリアボタン Lv.1（""（空白）を書き込む）

次に、クリアボタンを作っておきましょう。

ここでは、一度請求書作成ボタンで作成した請求書をクリアする機能を作ります。
先ほどのように、毎回請求書をクリックして「Delete」キーを押していたのでは大変です。
そこで、ボタン１つを押せばセルの値をクリアしてくれるようなボタンを作っておきます。

✎【入力してみよう】

```
Sub Seikyu()
    Cells(9, 10).Value = Cells(3, 3).Value
End Sub

Sub Clear()
    Cells(9, 10).Value = ""
End Sub
```

【実行】

一つずつコードを読み解いていきましょう。

```
Sub Clear()
```

ここでは、新しく関数を作りだす宣言をしています。
請求書作成ボタンを作るときは、「Sub Seikyu()」でしたね。
今回はクリアボタンですので、「Sub Clear()」という名前を付けておきます。
（終わりは、「End Sub」で締めます。）

```
    Cells(9, 10).Value = ""
```

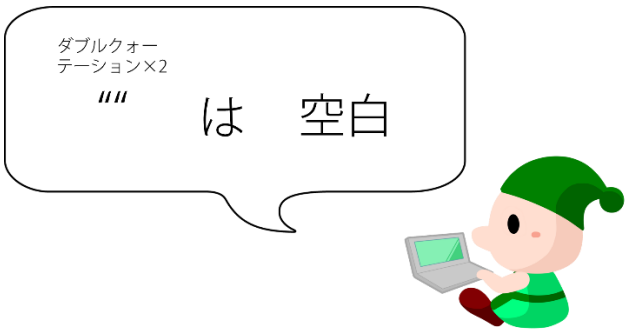
上から 9 行目、左から 10 列目のセル（つまり J9）の値に「""」（空白）を書き込む。

こういった意味になります。

「""」のようにダブルクォーテーションを２つ続ける場合、それは「空白」を意味します。
VBA では、セルの値を空白にする際にこのような方法をとることがよくあります。

数値をクリアする　≒　空白を書き込む

これらは、ほぼ同じような意味を持つということです。覚えておきましょう。

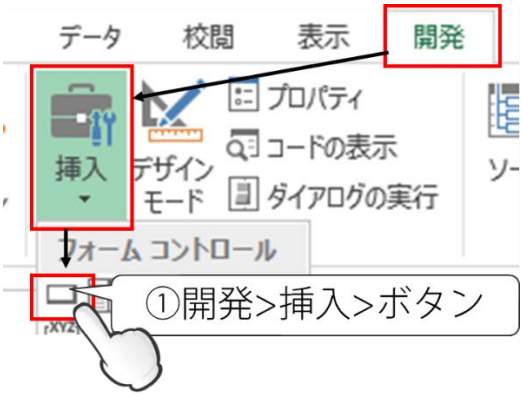


マクロを実行してみると、J9 の値がクリアされるはずです。

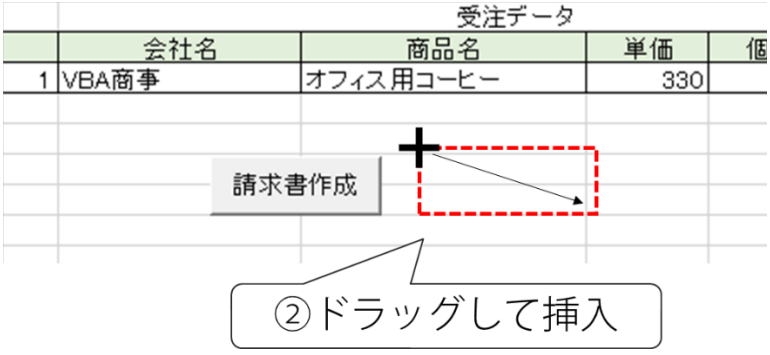
【補足】ボタンで実行されるようにしておこう

クリアボタンの機能も、Excel シート上でボタンをクリックしたら実行されるように設定しておきましょう。

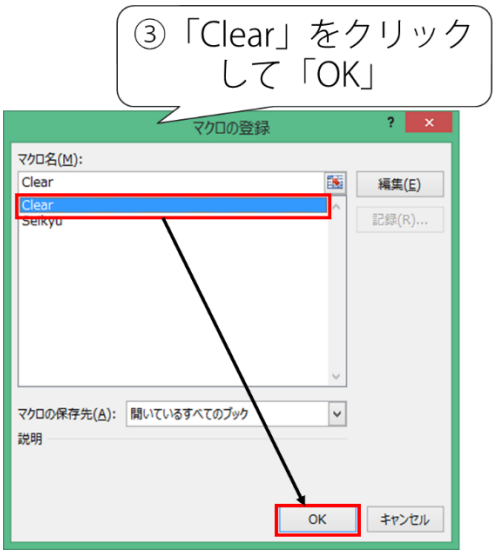
①「開発」タブ＞「挿入」＞「ボタン」



②マウスをドラッグして、請求書の横にボタンを挿入



③「Clear」を選択して「OK」ボタン



④ボタン名を「クリアボタン」に修正

【1-7】請求書作成ボタン Lv.2（複数のセルを書き移す）

さて、この命令文が理解できてしまえば、「請求書マクロ」の課題を作る上での重要なキーをマスターしたと言えます。

先ほどの実行項目のリストに戻ってみましょう。

セル C3 の内容をセル J9 に書き写す。
セル C4 の内容をセル J10 に書き写す。
セル C3 の内容をセル J11 に書き写す。
セル C3 の内容をセル J12 に書き写す。

これらを Excel に実行させる命令をすべて記述すれば、このマクロは完成するということです。
一つ目の「セル C3 の内容をセル J9 に書き移す。」については、前述の方法で記述しました。

```
Cells(9, 10).Value = Cells(3, 3).Value
```

でしたね。

それでは、2 行目、3 行目、4 行目の命令も同じように連続して記述すれば良いということになります。

このように記述してみましょう。

✎【入力してみよう】

```
Sub Seikyu()
    Cells(9, 10).Value = Cells(3, 3).Value
    Cells(9, 11).Value = Cells(3, 4).Value
    Cells(9, 12).Value = Cells(3, 5).Value
    Cells(9, 13).Value = Cells(3, 6).Value
End Sub
```

【実行】

いかがでしょうか。
実行してみると、「売上データ」の商品名・単価・個数・金額がすべて「請求書」のフォーマット１行目に書き移されたはずです。

このように、プログラムの記述方法は単純です。
人間が行うべき作業を、１行ずつコンピュータに対する命令文として記述していく。
すると、４行の命令文を記述すればコンピュータは上から順番に４つの命令を実行してくれるのです。

【1-8】クリアボタン Lv.2（複数のセルをクリアする）

さて、先ほどの「クリアボタン」を使って、請求書をクリアしてみましょう。
…おや？と思ったかもしれません。現状のクリアボタンの機能では、セル J9 の「商品名」しかクリアされませんね。

なぜでしょうか？

```
Cells(9, 10).Value = ""
```

そうですね、このままではセル（9 段目の 10 列目）（J9）の値をクリアする命令しか記述できていません。

本当は、J9、J10、J11、J12 のすべてにおいて値をクリアしてほしいのですが…

それでは、どうすればいいでしょうか？
このように修正しましょう。

✎【入力してみよう】

```
Sub Clear()
    Cells(9, 10).Value = ""
    Cells(9, 11).Value = ""
    Cells(9, 12).Value = ""
    Cells(9, 13).Value = ""
End Sub
```

【実行】

クリアボタンをクリックすれば、すぐに請求書の１行目がクリアされるはずです。

【1-9】クリアボタン Lv.3（複数のセルを Range で指定する）

さて、もうすぐ入門編「かわいい請求書」のプログラミングは完成です。

ここで、大切な記述方法をお伝えしておきます。

「クリアボタン」における記述を見直してみましょう。

```
Cells(9, 10).Value = ""
Cells(9, 11).Value = ""
Cells(9, 12).Value = ""
Cells(9, 13).Value = ""
```

これらの命令は、４つの隣り合ったセルへ、上から順番に空白(””)を書き込んでいくという単純なプログラムです。

これはシンプルで分かりやすいのですが、しかし、なんだか無駄が多いように見えます。
（４つの隣り合ったセルを扱うのだから、いっぺんに１度で命令してしまえないのだろうか？）
と疑問に思えます。

ましてや、今回はセルが４つだから良いものを、もしも 100 個のセルがあったら大変です。
（同じような命令を 100 個記述しなければならないかと思うと…ぞっとします。）

そこで、ここではRangeという記述方法を使って、たった一行で書き換えてみます。

```
Range(Cells(9, 10), Cells(9, 13)).Value = ""
```

✎【入力してみよう】

```
Sub Clear()
    Range(Cells(9, 10), Cells(9, 13)).Value = ""
End Sub
```

【実行】

このように書き直して、実行してみて下さい。

【補足】使わないけど残しておきたいコードは「コメント化」

「せっかく今まで書いていた４行の命令文も残しておきたい…」と思う方もいるかもしれません。そこで、残しておきたい命令文の行頭に「‘」を記述しておいて下さい。

（残しておきたいコードの行頭に「’（クォーテーション）」

```
‘    Cells(9, 10).Value = ""
‘    Cells(9, 11).Value = ""
‘    Cells(9, 12).Value = ""
‘    Cells(9, 13).Value = ""
```

いかがでしょうか？行頭に「‘」を書き込んだ行は、色が変わって少し薄くなりました。
これは、「コメント化」という機能です。
行頭に「’」をつけてコメント化された文は、プログラムとしては無視されます。
つまり、実行させたくない記述や、しばらく保留しておきたい記述があったら、行頭に「’」をつけてコメント化しておけば、その一行はプログラムとして実行されないということです。

（コラム）コメント化の機能は、本来の使い方としては「書籍における注釈文」のように使用されることが主です。書いてあるプログラム文書について、「’ これはこういう意味がある一文だ」というように、注釈をつけておくことによく使われます。
その理由は、
・後で自分がプログラムを見返したときに、何を書いたのか忘れてしまわないように（備忘録）
・チームで開発をしている際、自分以外のチームメンバーが読んだ際に意味が伝わるように（注釈）
といった理由があります。
プログラミングをする場合は、適宜、コメントを上手に加えてきましょう。自分にも周りの人にも分かりやすいプログラムが出来上がります。

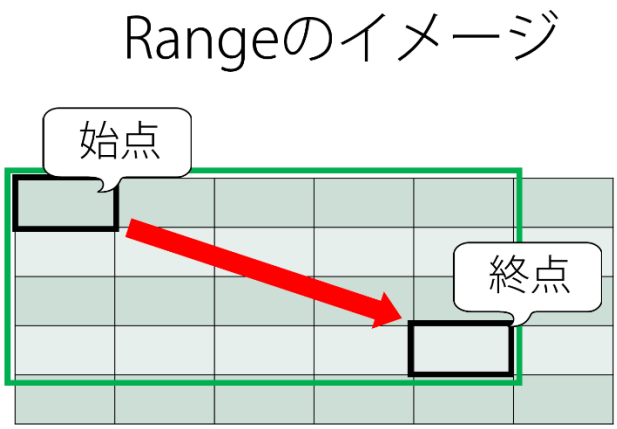
さて、先ほどの話に戻ります。

```
Range(Cells(9, 10), Cells(9, 13)).Value = ""
```

新たに Range( ) という記述が出てきました。

Range( )は、「セルの範囲」を指しています。
つまり、複数の隣り合ったセルを、一度に「セル範囲」として指定することができるものです。

【覚えておこう】
Range(始点のセル,終点のセル)



私たちが Excel でセル範囲を選択するときには、マウスを使って範囲をドラッグして選択することがよくありますね。

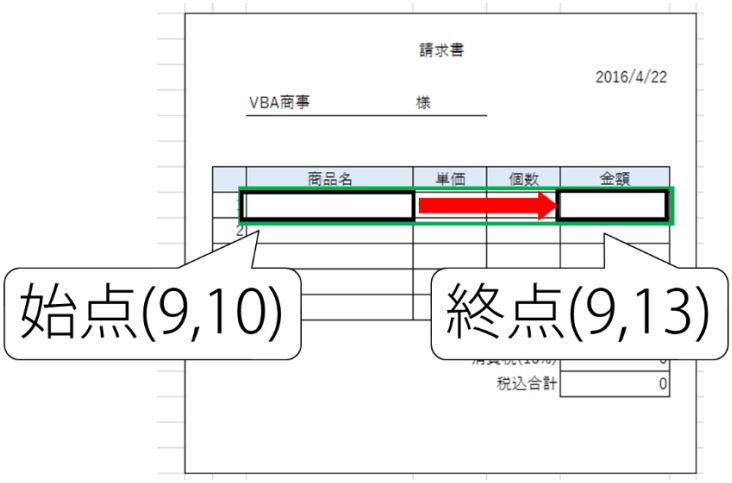
（人間が操作する場合）

始点のセル Cells(9,10)　～　終点のセル Cells(9,13)　までをドラッグする

これと同じようなイメージで、Excel マクロに同様のセルを選択させる場合には

Range（Cells(9,10)，Cells(9,13)）
範囲（ 始点のセル ， 終点のセル ）

このように選択することができるということです。



つまり、今回の一文では、セル範囲の値に空白を書き込むということを意味しています。

Range(Cells(9, 10), Cells(9, 13)).Value = ""
範囲（ 始点のセル ， 終点のセル).数値　← 空白を書き込む

イメージは理解できたでしょうか？

※実際にコードを記述をする場合、「(」や「)」（かっこ）をいくつも入れ子のように重ねて打ち込むことになりますので、タイプミスがないように気をつけて下さい。

いかがだったでしょうか。
最初、４行にわたって記述していた命令が、たった１行で済んでしまいました。
このようにセル範囲を Range( )でまとめて選択することが VBA ではよくありますので、覚えておきましょう。

【1-10】請求書作成ボタン Lv.3（複数のセルを Range で指定する）

前項目で「Range(　)」を使った記述を学びましたが、ここで気づいた方もいるかもしれません。

Range( )は、クリアボタンだけじゃなく、請求書作成ボタンにも使えるのでは？

実際にその通りです。先ほどの「請求書作成ボタン」でも同様に、Range( )を使って書き直せば、たった１行で書き直すことができます。

現状の「請求書作成ボタン」をもう一度見てみましょう。

```
Cells(9, 10).Value = Cells(3, 3).Value
Cells(9, 11).Value = Cells(3, 4).Value
Cells(9, 12).Value = Cells(3, 5).Value
Cells(9, 13).Value = Cells(3, 6).Value
```

これらを、Range( )を使った記述方法に書き換えるにはどうすれば良いでしょうか？
考えてみましょう。

※先ほどのコードを消さずに残しておきたい、という場合は、行頭に「'」をつけてコメント化しておいて下さい。

こちらが正解の記述方法です。

✎【入力してみよう】

```
Sub Seikyu()
    Range(Cells(9, 10), Cells(9, 13)).Value （改行せず,続けて入力して下さい）
                                    = Range(Cells(3, 3), Cells(3, 6)).Value
End Sub
```

【実行】

コードを１つずつ読み解いていきます。

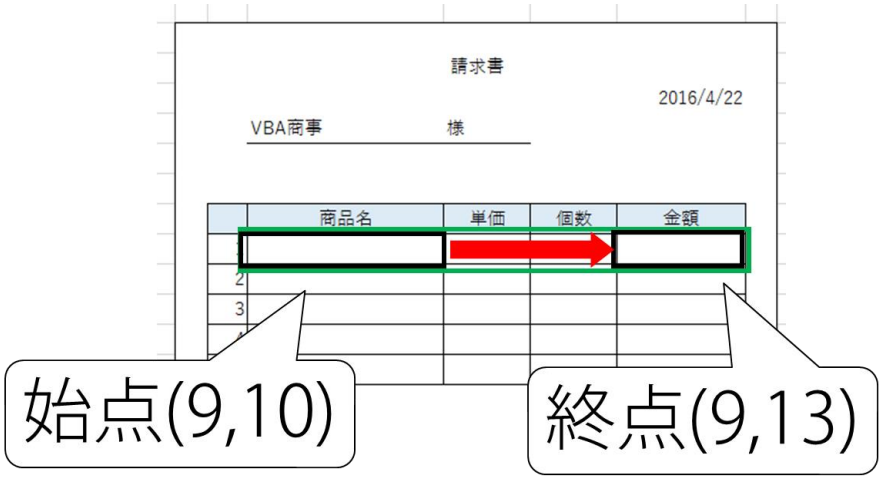
```
Range(Cells(9, 10), Cells(9, 13)).Value = Range(Cells(3, 3), Cells(3, 6)).Value
```

少し一行が長くなってしまったので、難解に見えるかもしれません。
単純化して考えてみましょう。

【左辺（「=」の左側）】

Range(Cells(9, 10), Cells(9, 13)).Value

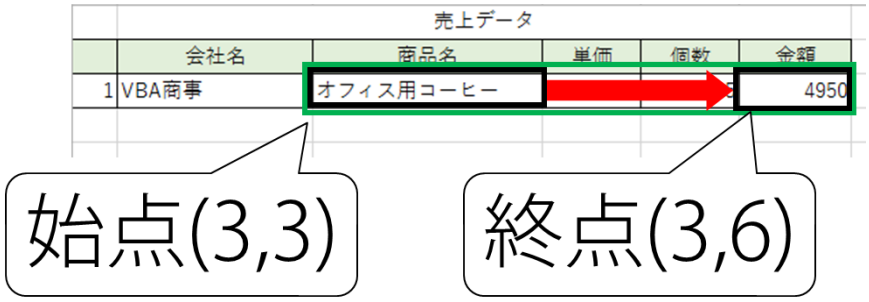
範囲（ セル(9,10)からセル(9,13) まで).の数値



【右辺（「=」より右側）】

Range(Cells(3, 3), Cells(3, 6)).Value

範囲（ セル(3,3)からセル(3,6) まで).の数値

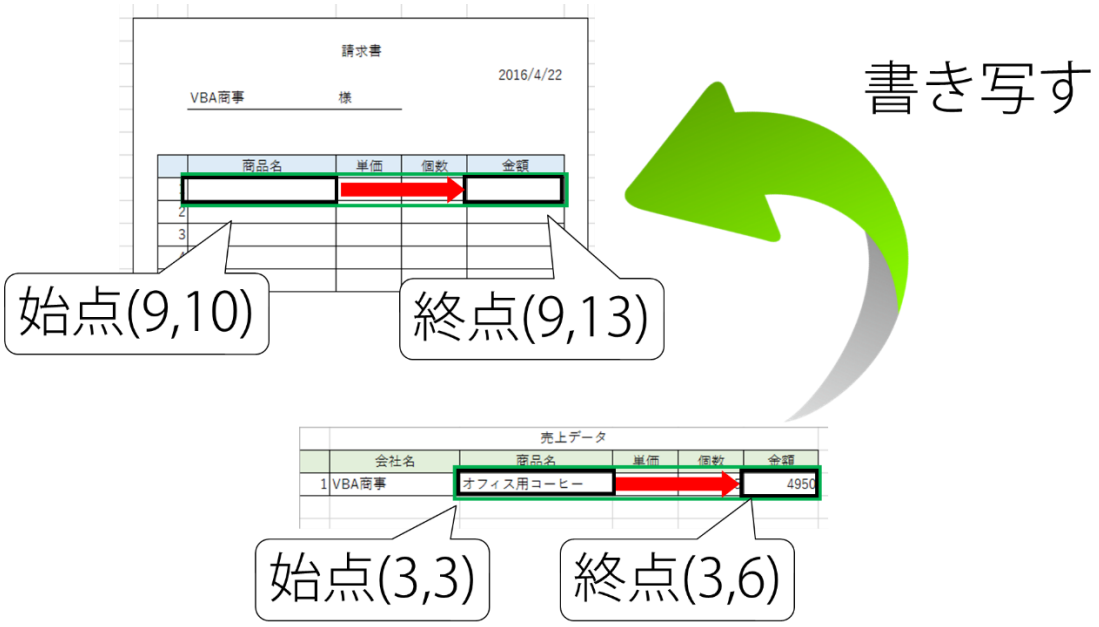


つまり、この一文の意味するところは、下記のようなイメージとなります。

Range(Cells(9, 10), Cells(9, 13)).Value = Range(Cells(3, 3), Cells(3, 6)).Value

範囲（ セル(9,10)からセル(9,13) まで).の数値に、

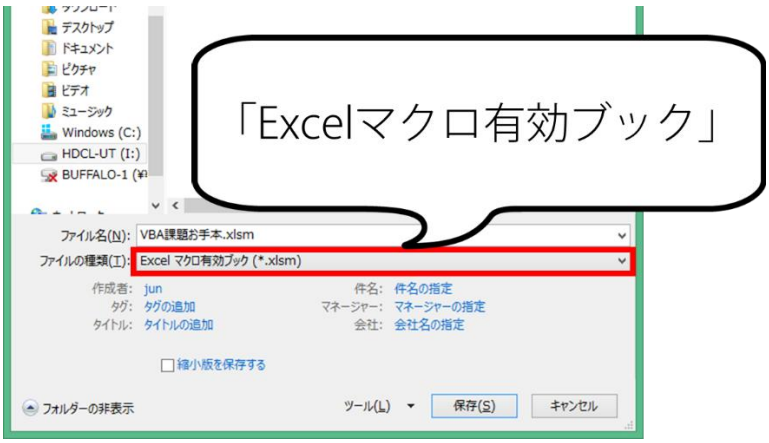
範囲（ セル(3,3)からセル(3,6) まで).の数値を書き込んで下さい。



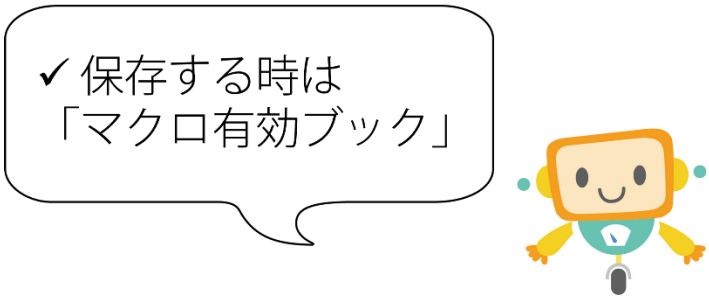
いかがでしょうか。
一見すると複雑で難解な記述に見えた一文でしたが、一つずつを読み解いていくと意味がわかってきます。このように、「Range」によってセルを指定する方法は、VBA プログラミングではこの先も重宝しますので、覚えておきましょう。

【1-11】マクロ有効ブックとして保存（.xlsm）

ここまで制作したマクロは、Excel のブックとして保存しておきましょう。
[ファイル]＞[名前をつけて保存]＞ブック名「VBA 課題」などと名前をつけて保存します。
ここで、**ファイルの種類は「Excel マクロ有効ブック」を選択して下さい。**
マクロを記述したブックは、この形式で保存しなければ、せっかく作ったマクロが消えてしまいます。



覚えておこう



入門編の課題、お疲れ様でした！

ここまでが入門編「かわいい請求書」の課題制作でした。いかがだったでしょうか。
VBA で記述するプログラムというものは、ほとんどが、私たち人間が Excel を普段から操作している手順をそのままプログラム言語に訳したものです。ここまでの課題を理解していくことで、「プログラムってこういうものなんだ」「VBA ってこういうものか」という感覚やイメージを取り入れていただければ幸いです。

【次への導入】入門レベル課題「かわいい請求書」の欠点！？

入門レベルの課題では、ひとまず請求書のフォーマットに沿った文書を１ボタンで制作できる「かわいい請求書」を制作しました。
しかし、この請求書マクロには欠点があります。

・受注データが１つだけの場合にしか使えない。

これでは、実用性に欠けています。
そこで、「初級レベル課題」では、さらに実用的な請求書マクロを作っていきます。

【第２章】初級編「やさしい請求書マクロ」を作ろう

シート「初級」を開いて下さい。

受注データ

	会社名	商品名	単価	個数	金額
1	VBA商事	オフィス用コーヒー	330	15	4950
2	エクセル運輸	オフィス用カフェラテ	360	21	7560
3	VBA商事	コーヒー豆100g	300	10	3000
4	関数カンパニー	オフィス用コーヒー	330	18	5940
5	エクセル運輸	コーヒー豆100g	360	14	5040

請求書

2016/4/23

VBA商事　　様

	商品名	単価	個数	金額
1				
2				
3				
4				
5				
			合計	0
			消費税(10%)	0
			税込合計	0

今回はこのようなデータになっています。

「受注データ」
・５件の売上データがある。（データが複数）
・複数の会社と取引している。

１章の「かわいい請求書作成マクロ」とは違って、**データが複数個あることが最大の変更点**です。

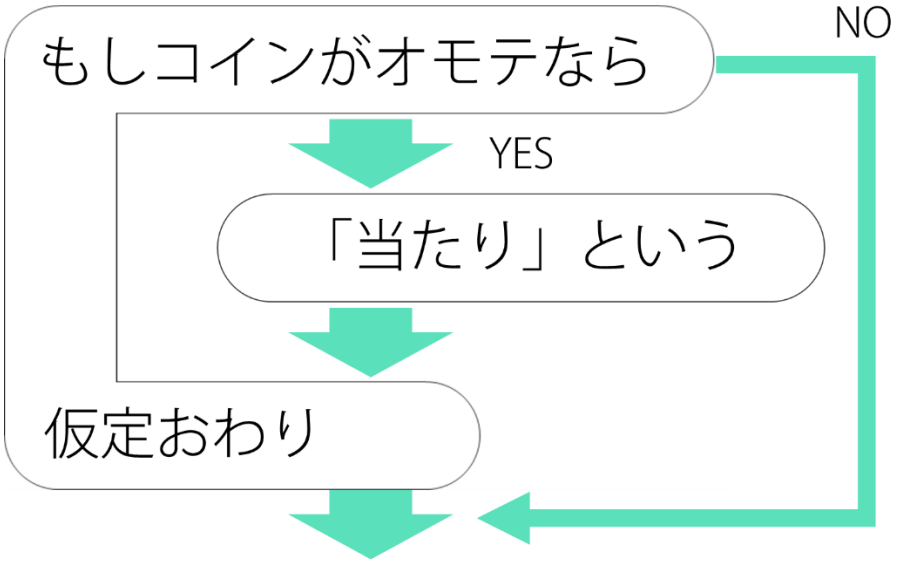
そこで、請求書を作るために、新たに「条件判断」という概念を学んでいきましょう。

【2-1】もしも（条件判断）について学ぼう（IF 文）

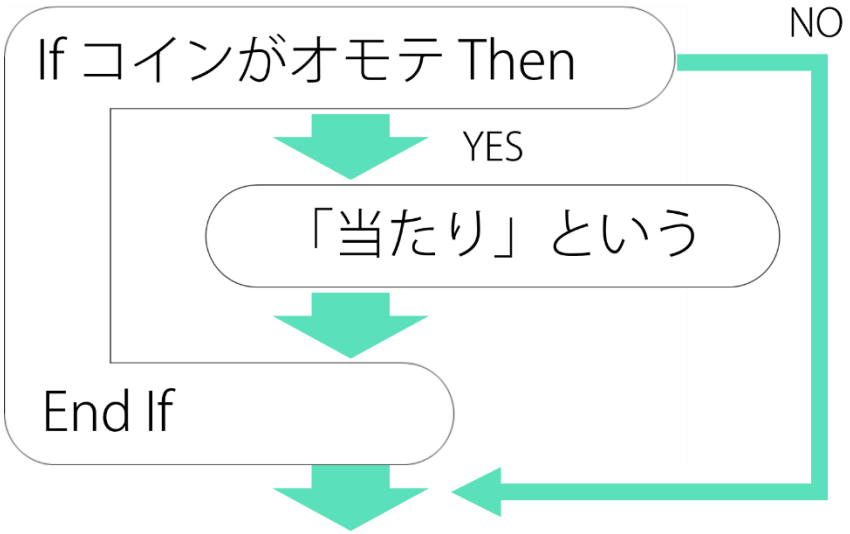
コインを投げて、オモテが出たのかウラが出たのかを判断する例を考えてみましょう。

次のような場合を考えてみましょう。

コインを投げて、オモテが出たら「当たり！」と言う。



この例をプログラミング的に書き表すと、以下のように表せます。



プログラミングにおいては、「もし…」という条件判断の際には「If」という書き方で表します。

```
If 条件 Then
    命令
End If
```

覚えておこう(IF文)

【2-2】If 文で条件判断を使ってみる(1)

前項目で「条件判断をする」という表現をしましたが、今回の課題マクロ作成にとっても非常に大切な要素です。

さて、実際のコードで理解していきますので、このように記述して実行してみましょう。

✎【入力してみよう】
※前回から改行して続けます： これまで入力していた「Sub Seikyu()」や「Sub Clear()」から数行空けて、その下に以下を入力していきます。

Sub Seikyu2()
　If Cells(3, 2).Value = "VBA 商事" Then
　　MsgBox ("これは VBA 商事です")
　End If
End Sub
 	
【実行】

実行してみると、メッセージボックスには
「これは VBA 商事です」と表示されるはずです。

If 条件 Then

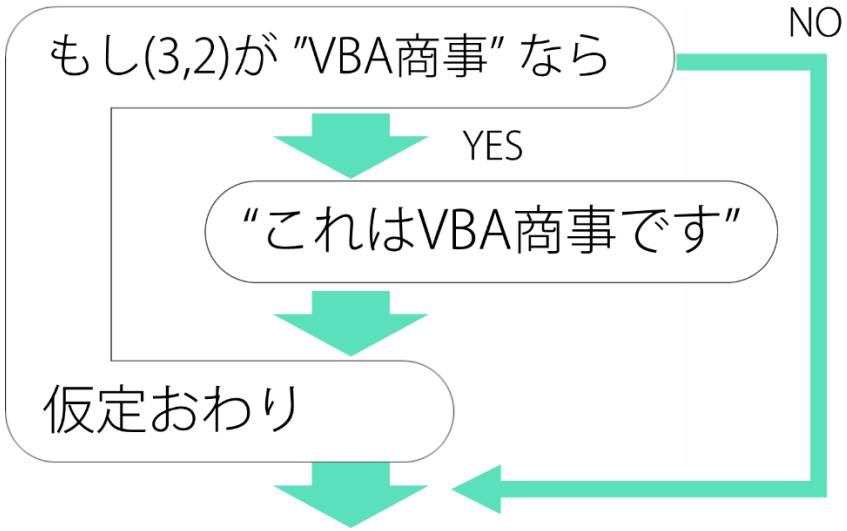
```
    （　命令　）
End If
```

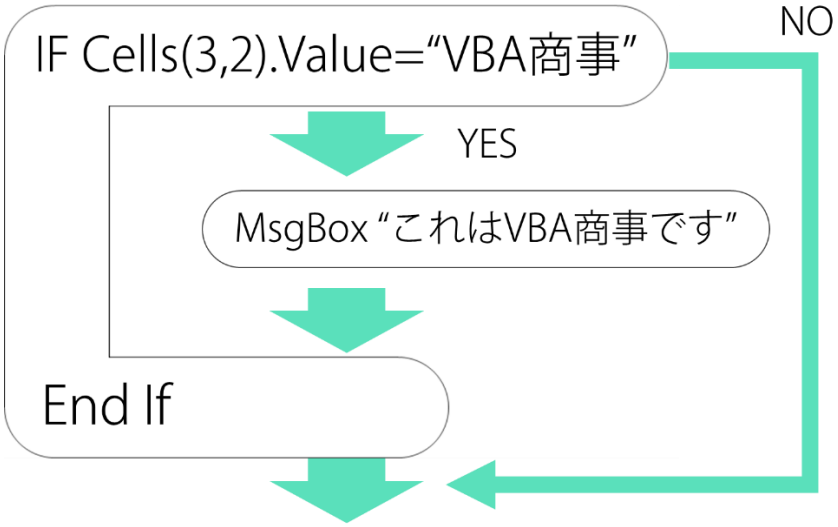
今回の「条件」は、このような条件として書かれています。

```
If Cells(3, 2).Value = "VBA 商事" Then
```
もし、セル(3,2).の値が　= "VBA 商事"　ならば…

という意味です。

この場合の「=」の記号は、算数と同じで、「～と等しい」という意味を指しますのでご注意下さい。





【試してみよう】
①セル B3 の値を、Excel のシート上で他の値に書き換えてみてください。
つまり B3 のセルを「VBA 商事」ではなく他の値（例えば「エクセル運輸」など）に書き換えてみます。
②請求書作成マクロを実行してみましょう。
③メッセージボックスが表示されないことを確認します。
④**※B3 の値を「VBA 商事」に戻しておきましょう。**

メッセージボックスは何も表示されなかったはずです。
つまり、マクロでは「もし B3 のセルが「VBA 商事」ならば…」という条件に合わなかったので、メッセージボックスを表示させなかったということです。

条件に合わなければ、命令は実行されません。

【2-3】If 文で条件判断を使ってみる(2)

それでは先ほどのコードを、次のように書き直してみて下さい。

✎【入力してみよう】

```
Sub Seikyu2()
    If Cells(3, 2).Value = Cells(5, 10).Value Then
       MsgBox "これは"& Cells(5, 10).Value
    End If
End Sub
```

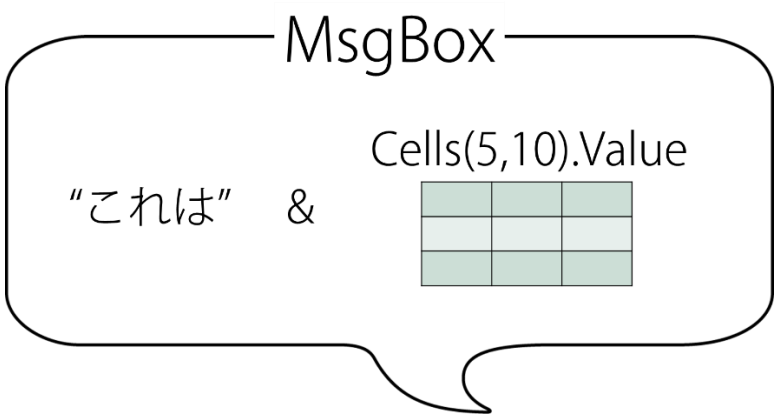
【実行】

いかがでしょうか。

実行してみると、もしセル(3,2)（つまりセル B3）に入力されている値が、セル(5,10)（つまりセル J5）に入力されている値と一致する場合、
メッセージボックスで「これは「VBA 商事」」と表示されるようになっています。

```
MsgBox ("これは" & Cells(5, 10).Value )
```

ここでは、セル(5,10)（つまりセル J5）に入力されている値を抜き出して、メッセージボックスに出力しています。



いかがでしょうか。これが条件判断（IF 文）の構文でした。
今回の実習では、条件にあった場合に単にメッセージボックスを出すだけの処理になっていますが、次からは、これを活用して条件判断しながら請求書を自動作成させていきます。

【2-4】請求書ボタン Lv.1（IF 文で条件判断）
それでは、次のようにコードを修正してみて下さい。

✎【入力してみよう】

```
Sub Seikyu2()
   If Cells(3, 2).Value = Cells(5, 10).Value Then
      Range(Cells(9, 10), Cells(9, 13)).Value （改行せず続けて入力して下さい）
                                    = Range(Cells(3, 3), Cells(3, 6)).Value
   End If
End Sub
```

【実行】

いかがでしょうか。
条件設定は同じですが、「命令」の方を修正しています。

```
Range(Cells(9, 10), Cells(9, 13)).Value = Range(Cells(3, 3), Cells(3, 6)).Value
```

範囲（(9,10) ～ (9,13)の値に ← 範囲（(3,3)～(3,6)の値を書き移す。

このような意味を持っています。

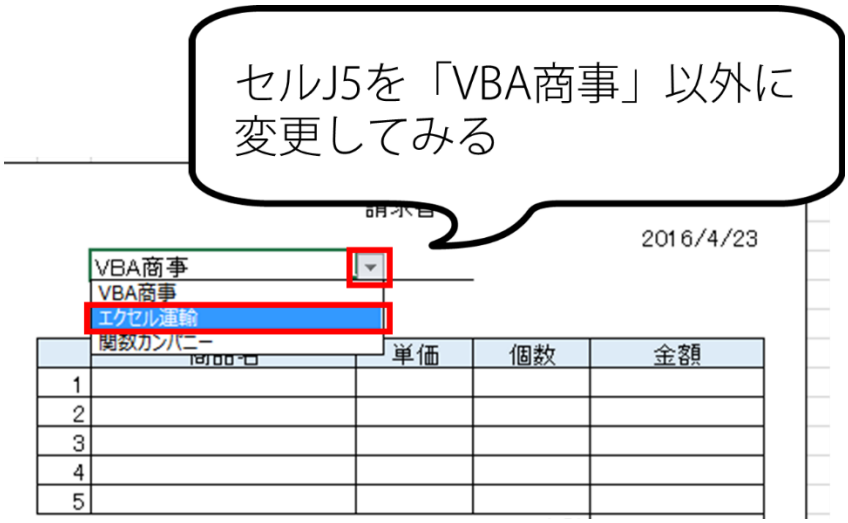
つまり、条件に合致した場合にだけ、注文データから請求書にデータを書き移してください。
という記述内容です。

【試してみよう】

セル J5 を「VBA 商事」以外に変更してみよう。

・（「エクセル運輸」に変更してマクロを実行）

・（「関数カンパニー」に変更してマクロを実行）



【2-5】請求書ボタン Lv.2（IF 文を 2 回）

前セクションでは、売上データ一覧表の一番始めの行だけを条件判断して、もしも「VBA 商事」ならばそのデータを請求書のフォーマットに書き移すという記述をしました。
しかし、売上データは 1 行だけではなく、**5 行あります。**ですから、５行とも同じように条件判断をする必要があります。
では、２行目も同様に記述してみましょう。

✎【入力してみよう】

```
Sub Seikyu2()
    If Cells(3, 2).Value = Cells(5, 10).Value Then
        Range(Cells(9, 10), Cells(9, 13)).Value（改行せず続けて入力して下さい）
                                        = Range(Cells(3, 3), Cells(3, 6)).Value
    End If

    If Cells(4, 2).Value = Cells(5, 10).Value Then
        Range(Cells(10, 10), Cells(10, 13)).Value（改行せず続けて入力して下さい）
                                        = Range(Cells(4, 3), Cells(4, 6)).Value
    End If
End Sub
```

【実行】

今回は、２度目の「IF～End If」を追加しました。
これによって、「売上データ」の２行目についても条件判断をし、もしも「VBA 商事」ならばデータを請求書に書き移す、という処理を行わせています。

【2-6】請求書ボタン Lv.3（IF 文を５回）

前セクションができたら、いよいよ売上データの５行すべてに条件判断をさせるようコードを記述しましょう。

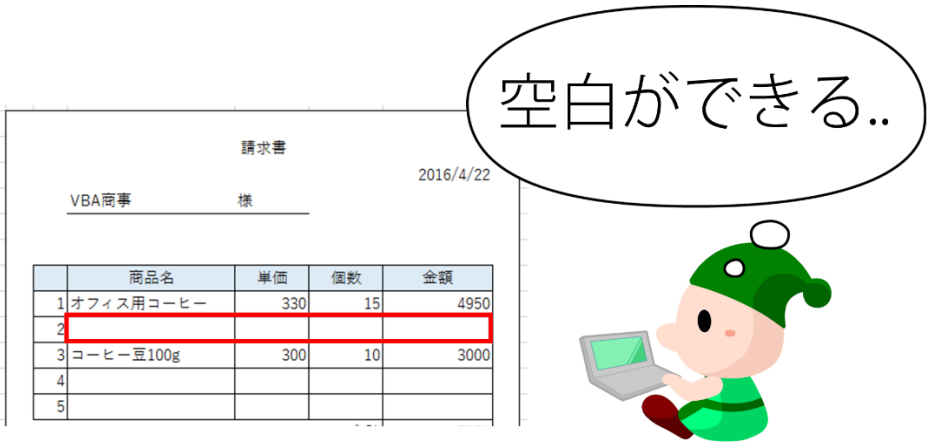
✎【入力してみよう】

```
Sub Seikyu2()
    If Cells(3, 2).Value = Cells(5, 10).Value Then
        Range(Cells(9, 10), Cells(9, 13)).Value
                                    = Range(Cells(3, 3), Cells(3, 6)).Value
    End If

    If Cells(4, 2).Value = Cells(5, 10).Value Then
        Range(Cells(10, 10), Cells(10, 13)).Value
                                     = Range(Cells(4, 3), Cells(4, 6)).Value
    End If

    If Cells(5, 2).Value = Cells(5, 10).Value Then
       Range(Cells(11, 10), Cells(11, 13)).Value（改行せず続けて入力して下さい）
                                     = Range(Cells(5, 3), Cells(5, 6)).Value
    End If

   If Cells(6, 2).Value = Cells(5, 10).Value Then
       Range(Cells(12, 10), Cells(12, 13)).Value（改行せず続けて入力して下さい）
                                    = Range(Cells(6, 3), Cells(6, 6)).Value
    End If

   If Cells(7, 2).Value = Cells(5, 10).Value Then
       Range(Cells(13, 10), Cells(13, 13)).Value（改行せず続けて入力して下さい）
                                    = Range(Cells(7, 3), Cells(7, 6)).Value
   End If
End Sub
```

【実行】

実行してみると、売上データのうち会社名が「VBA 商事」のデータだけが請求書に書き移されます。かなり請求書作成マクロとしての機能がしっかりしてきました。

【2-7】このマクロの欠点！？



ここまでのプログラミングで、かなり請求書作成マクロとしてそれらしい機能が作られてきたと思います。「これで完成」といっても構いませんが、ちょっとお待ち下さい。
現状のマクロでは、欠点がいくつかあるのです。

・請求書に空白の行ができてしまうことがある。

例えば、「VBA 商事」の請求書を作りたい場合。売上データにおける「VBA 商事」のデータが、行を空けて飛び飛びに存在していた場合、請求書にもそのまま「飛び飛び」で書き移されてしまうのです。

これは、ビジネス上の文書としては見た目上、非常によろしくない欠点ですね。
もちろん、「あとは人間が手作業で Excel を操作して、穴がなくなるように仕上げをする…」ということでもいいですが、せっかくなら最後の仕上げまでマクロに自動的にやって欲しいところです。

では、「請求書に穴がないよう（上詰めで）書き移す」というマクロを記述することは可能でしょうか？

可能です。
ただし、この命令を記述するためには、「ある概念」を利用する必要があります。
それが、**カウンタ変数**というものです。

【2-8】カウンタ変数について学ぼう

ここからは、「カウンタ変数」 という概念を覚えていきます。
字面を見てみると全く新しい概念のように感じるかもしれませんが、そんなに難しいことではないのでご安心下さい。

【例え話で理解しよう】
小さい子供が、ジャンプを５回する場面を思い浮かべてみましょう。
「今まで何回ジャンプをしたか？」
小さい子は、数を覚えておくことが難しいため、**指をつかって数えることでしょう。**

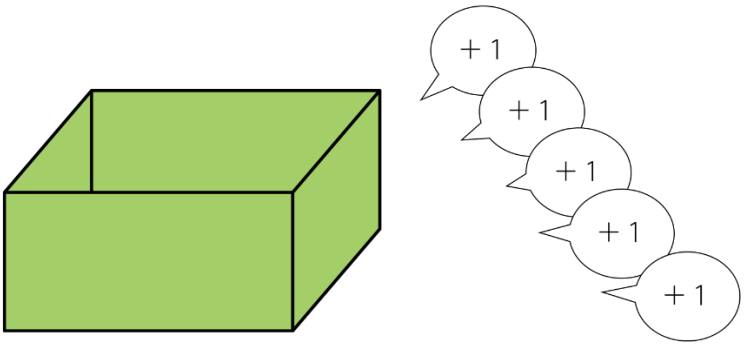
ジャンプをする（１回目）　→指の本数を＋１　（指は２本になった）
ジャンプをする（２回目）　→指の本数を＋１　（指は２本になった）
ジャンプをする（３回目）　→指の本数を＋１　（指は３本になった）
ジャンプをする（４回目）　→指の本数を＋１　（指は４本になった）
ジャンプをする（５回目）　→指の本数を＋１　（指は５本になった）

おそらく、このように回数を数えるのではないでしょうか？
指の本数を１つずつ増やしていくことで、「ジャンプをした回数」をカウントしておくということですね。

この「指」の役割をするのが、今回扱う「カウンタ変数」というものです。

カウンタ変数　＝　回数を数えるために、１つずつ増やしていくもの

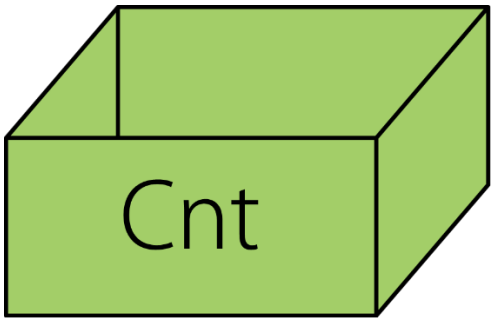
カウンタ変数



「変数」については前述しましたが、もう一度復習しておきましょう。
変数とは、数値を入れ込んでおくための「箱」だと説明をしました。

この箱には名前をつけることができますが、今回は「カウント」を意味する「Count（数える）」の単語にちなんで、「Cnt」という変数名にします。

Cnt という変数の「箱」を作りました。



【補足】
なぜ Count でなく Cnt と名付けるのか？プログラミングでは、このように元の英単語から母音（a,e,i,o,u）を抜いた省略形を用いることが多いためです。その理由は、何度も打ち込むことが多いため、あまり長い変数名だと煩わしく、コードが長ったらしくなってしまい分かりにくくなってしまうためです。そのため、変数名は短いものが好ましく、このような省略形を用いられることがよくあります。

さて、もしもジャンプの回数をこの「Cnt」の箱をつかって数えるのなら、このように数えることになります。

箱「Cnt」を用意する。
箱「Cnt」には、数字「０」が入っている。

ジャンプをする（１回目）　→箱「Cnt」の数字を＋１　（現在１）
ジャンプをする（２回目）　→箱「Cnt」の数字を＋１　（現在２）
ジャンプをする（３回目）　→箱「Cnt」の数字を＋１　（現在３）
ジャンプをする（４回目）　→箱「Cnt」の数字を＋１　（現在４）
ジャンプをする（５回目）　→箱「Cnt」の数字を＋１　（現在５）

このような流れで、現在の回数をカウントしていくことになります。
箱「Cnt」の数字は、０から始まって「１、２、３、４、５」と増えていくということですね。
何となく意味がわかったでしょうか。プログラミングでは、何かの回数をカウントするために、このようにカウンタ変数という「箱」を用意して用います。

それでは、実際に VBA のコードではどのように記述すればいいでしょうか。

【2-9】カウンタ変数を使ってみよう

✎【新しく入力してみよう】

```
Sub Count()
    Dim Cnt As Integer
    Cnt = 0
    MsgBox (Cnt)

    Cnt = Cnt+1
    MsgBox (Cnt)

    Cnt = Cnt+1
    MsgBox (Cnt)
End Sub
```

【実行】

いかがでしょうか。
メッセージボックスでは順番に、「0」「1」「2」と表示されたはずです。

一つ一つのコードの意味を説明します。

Dim Cnt As Integer

「Cnt」という変数の「箱」を用意する…数値型として。

「As Integer」というものは、「数値型として」という意味で、「この箱には数字が入ります」という用途を示します。

（補足：）
「この箱には、数字が入ります」という用途を示したので、この箱には数値しか入りません。逆に、「文字列（「あ」や「a」など）」を入れることはできません。数値以外のものを入れようとするとエラーを起こします。※このように、プログラミングのルールでは、変数の箱を用意する際に「用途」を示します。それは、コンピュータが予想外のデータを箱に入れられて不具合を起こすことを防ぐためだとお考え下さい。）

Cnt = 0

用意したカウンタ変数「Cnt」の中に、0という数字を入れ込むという意味を指します。

箱「Cnt」　←　0を入れ込む

MsgBox (Cnt)

メッセージボックスで、箱「Cnt」の中に入っているものを表示させます。

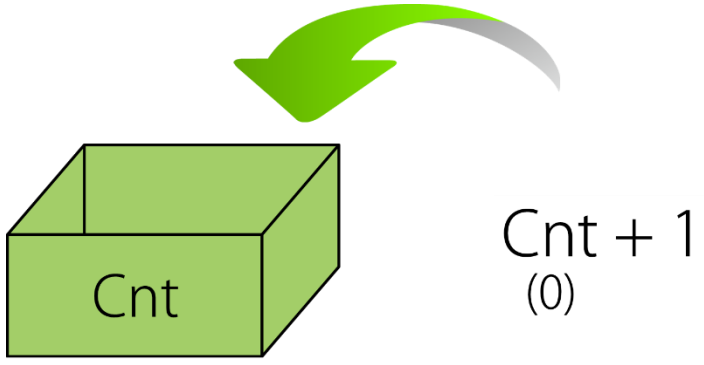
箱「Cnt」に入っているのはまだ「0」ですので、メッセージボックスには「0」と表示されます。

Cnt = Cnt+1

この記述は、慣れるまでは分かり難いと感じるかもしれません。

箱「Cnt」の中には、これまで「Cnt」に入っていた数字に＋１した数字を入れ込むという意味です。

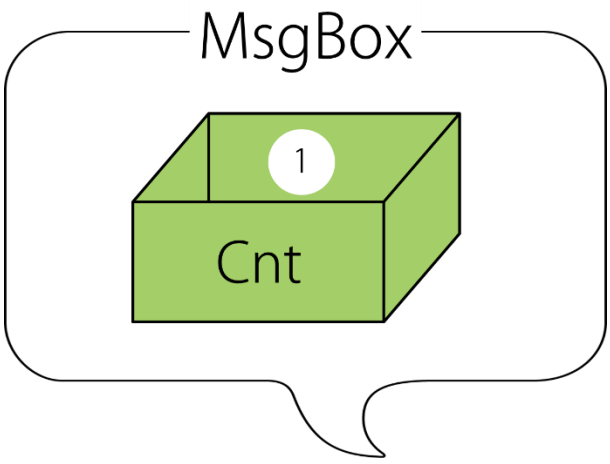
箱「Cnt」　←　「Cnt」に＋１した数値を入れ込む



このようなイメージです。

※【補足】おかしな式？
数学の方程式のように考えると、この記述は矛盾したものに思えます。
x = x +1
なんて、方程式では誤った等式ですよね。
しかし、プログラミングにおいて変数を扱う際には、この記述方法で矛盾になりません。
プログラミングで変数を扱う時には、「=」は「←」のような意味を指します。
（←　右の内容を、左の内容に「入れ込む」という意味合い）

MsgBox（Cnt）
もう一度、箱「Cnt」の中に入っている数値をメッセージボックスで出力させます。
この時点で箱「Cnt」の中には、前述のように＋１された値が入っているため、メッセージボックスでは「１」と表示されることになります。



（以下、同様に繰り返し）

このようにして、箱「Cnt」に入っている数字を順番に＋１ずつ増やしていきながら、メッセージボックスで出力させているのが今回のマクロです。

このような「カウンタ変数」の扱い方は、VBA やプログラミングにおいてよく使われる重要なテクニックですので、覚えておきましょう。

【2-10】請求書ボタン Lv.4（Cnt を使って上詰めにする）

さて、準備はできましたので、本題の「請求書作成」に戻っていきます。

いよいよ請求書作成マクロに変数「Cnt」を登場させます。

✎【入力してみよう】

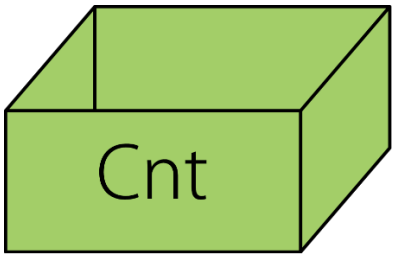
```
Sub Seikyu2()
    Dim Cnt As Integer
    Cnt = 9
    If Cells(3, 2).Value = Cells(5, 10).Value Then
        Range(Cells(Cnt, 10), Cells(Cnt, 13)).Value
                                    = Range(Cells(3, 3), Cells(3, 6)).Value
        Cnt = Cnt + 1
    End If

    If Cells(4, 2).Value = Cells(5, 10).Value Then
        Range(Cells(Cnt, 10), Cells(Cnt, 13)).Value
                                    = Range(Cells(4, 3), Cells(4, 6)).Value
        Cnt = Cnt + 1
    End If

    If Cells(5, 2).Value = Cells(5, 10).Value Then
       Range(Cells(Cnt, 10), Cells(Cnt, 13)).Value
                                    = Range(Cells(5, 3), Cells(5, 6)).Value
        Cnt = Cnt + 1
    End If

   If Cells(6, 2).Value = Cells(5, 10).Value Then
       Range(Cells(Cnt, 10), Cells(Cnt, 13)).Value
                                    = Range(Cells(6, 3), Cells(6, 6)).Value
        Cnt = Cnt + 1
    End If

   If Cells(7, 2).Value = Cells(5, 10).Value Then
       Range(Cells(Cnt, 10), Cells(Cnt, 13)).Value
                                    = Range(Cells(7, 3), Cells(7, 6)).Value
        Cnt = Cnt + 1
   End If
End Sub
```

【実行】

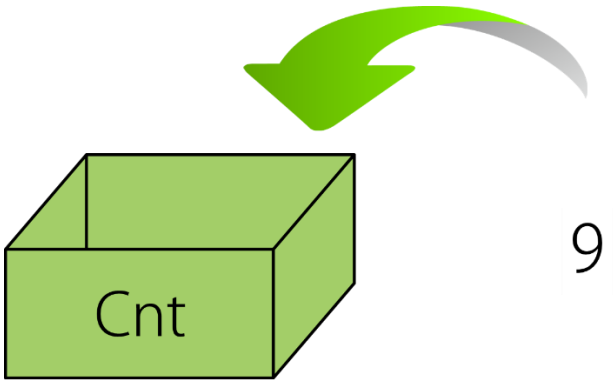
今までは請求書に穴が空いてしまっていたのに、今回の変更を加えると、穴が空かずにしっかりと上詰めで書き移されたことがわかります。
このような改善ができたのは、カウンタ変数を使ったからです。

一行ずつ見ていきましょう。

Dim Cnt As Integer
カウンタ変数である「Cnt」を用意する一文です。

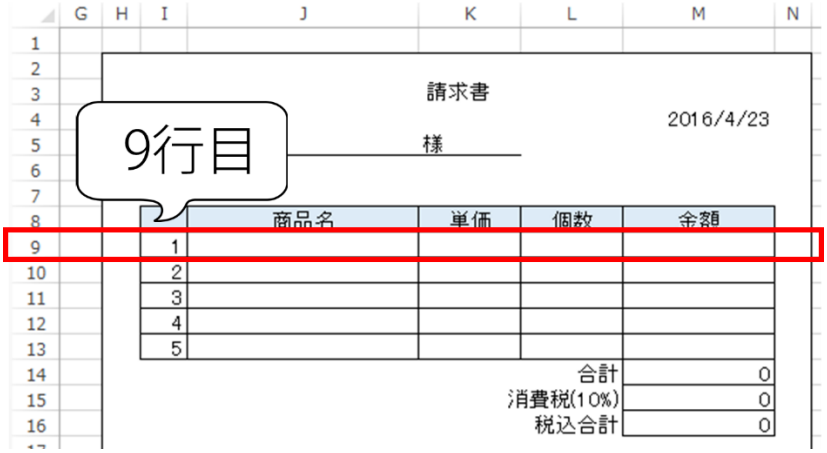


Cnt = 9
箱「Cnt」　←　9　という数字を入れ込んでいます。



なぜ「9」を入れ込むのでしょうか？それは、最初に請求書へ書き移すべき「行数」が 9 であるためです。

図をごらんください。



最初に請求書フォーマットへ書き移すべき行は「9 行目」にあります。
そのため、「まずは 9 行目に書き移させたい！」という記憶をしておくために、「9」という数値を入れ込んでおいたということです。

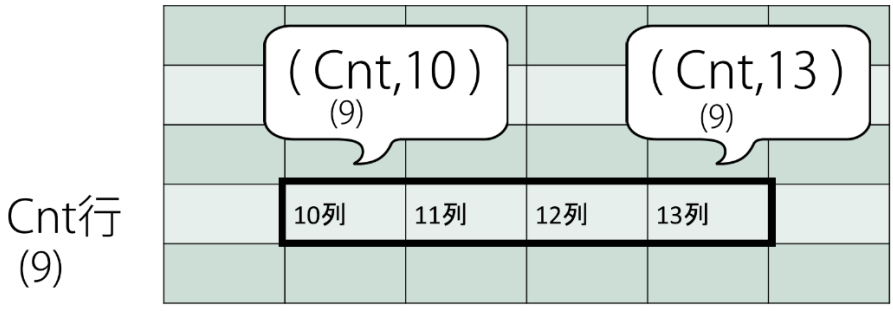
```
Range(Cells(Cnt, 10), Cells(Cnt, 13)).Value = Range(Cells(3, 3), Cells(3, 6)).Value
```

ここは少々複雑な表現になります。

まず左辺にある以下の記述について。

```
Range(Cells(Cnt, 10), Cells(Cnt, 13)).Value
```

範囲(セル(Cnt 行の 10 列)～セル(Cnt 行の 13 列) .の値



この時点では、「Cnt」には「9」という数字が入っていますので、Cnt は「9」と置き換えて考えると理解できます。

これらは請求書のフォーマットの最初の行を意味しています。
一番はじめは、最初の行に書き移させたいという事ですね。

次に、右辺を見てみましょう。

```
= Range(Cells(3, 3), Cells(3, 6)).Value
```

← 範囲(セル(3 行 3 列) ～セル(3 行 6 列) .の数値を　（入れ込む）

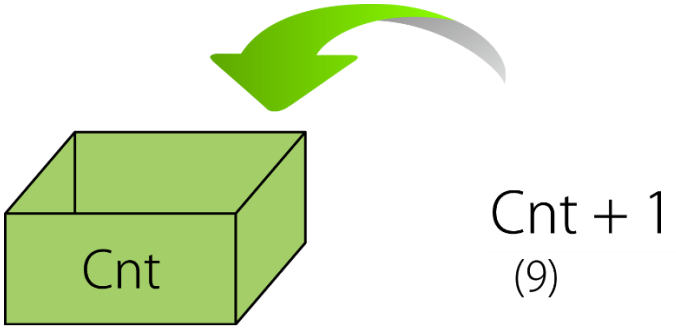
ここは前回から変化しません。
売上データの 1 行目にある[会社名、商品名、単価、個数、金額]のセル範囲を書き移させる処理を記述しています。

以上の 1 行で、請求書フォーマットの Cnt 行目（現在は 9 行目）にデータを書き移すという処理を記述してあります。

そして、次の一文が重要です。

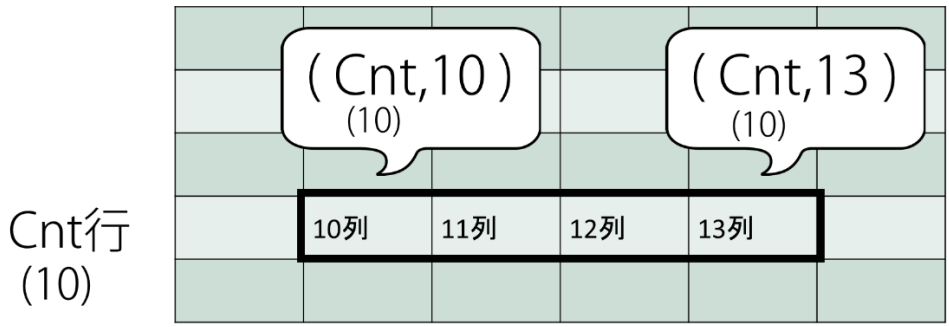
```
Cnt = Cnt + 1
```

箱 Cnt　←　箱 Cnt に＋１した数値を入れ込む。



前述のように、Cnt には「9」が入っていました。そこに＋１した数値を入れ込むということは、Cnt に「10」が入るということになります。

ということは、どうなったでしょうか。
「次は、請求書の 10 行目にデータを書き移しましょう」、ということになります。



このように、データの書き移しをした後で Cnt を＋１増やしていくことで、次にデータを書き移すべき行番号を更新しています。

流れを表すと、

箱「Cnt」を用意する。
箱「Cnt」に 9 を入れ込む。
売上データを 9 行目に書き移す。　→箱「Cnt」の数字を＋１　（現在 10）
売上データを 10 行目に書き移す　→箱「Cnt」の数字を＋１　（現在 11）
売上データを 11 行目に書き移す　→箱「Cnt」の数字を＋１　（現在 12）
売上データを 12 行目に書き移す　→箱「Cnt」の数字を＋１　（現在 13）
売上データを 13 行目に書き移す

このように、売上データを書き移すたびに Cnt を＋１ずつ増やしています。

ただし、今回のマクロでは「IF 文」を使用していますので、条件判断が入ってきます。
もし、会社名が「VBA 商事」ではなかった場合には、データの書き移しが実行されません。すると、Cnt の中身も＋１加算されないまま次の判断に移りますので、空白行が空かないように請求書が出来上がっていくのです。

【2-11】このマクロの欠点！？

さて、このようにして、請求書作成マクロはかなり完成されてきました。
空白行を作ることなく、きれいに上詰めにして書き移してくれる…このようなマクロを作れるようになれば、VBA 初心者の基本を十分に達成しています。

しかし、現状のマクロには少々の欠点があります。

「コードが長くなってしまう」という問題です。

現状のマクロでは、同じようなコードを５回続けて書いているため、とても冗長になっています。
また、今回の課題では注文一覧に５つしかデータが無かったからよかったものの、仮にもし 100 個、200 個とデータがあったらどうでしょうか？

同じようなコードを 100 回入力しなければいけないとしたら、とても大変です。

（作業を効率化するために VBA マクロを作っているのに、これでは本末転倒ですね。）

問題点

✓ コードが長い..

✓ 拡張性に欠ける

そこで、このように同じようなコードを何回も記述しなければならない場合のために、「反復（ループ）」という仕組みを使っていきます。

【2-12】例え話でわかる「反復（ループ）」

このセクションからは、新しく「反復（ループ）」という概念を理解していきます。
先ほどのように、例え話で理解していきましょう。

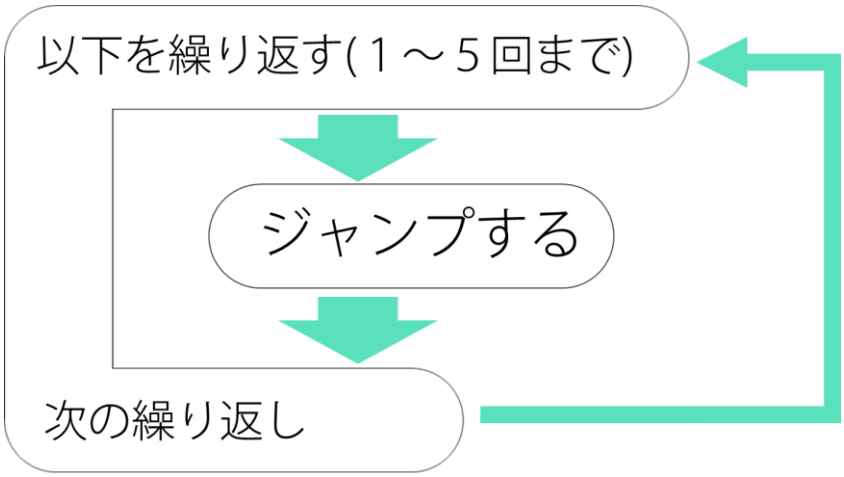
小さい子供が、ジャンプを５回行う場合。
単純にいうと、このように表すことができます。

ジャンプをする。（１回目）
ジャンプをする。（２回目）
ジャンプをする。（３回目）
ジャンプをする。（４回目）
ジャンプをする。（５回目）

同じ「ジャンプをする」を５回も書いていますが、これを書き換えてみましょう。

以下を繰り返す（１～５回目まで）
　ジャンプをする
次の繰り返し

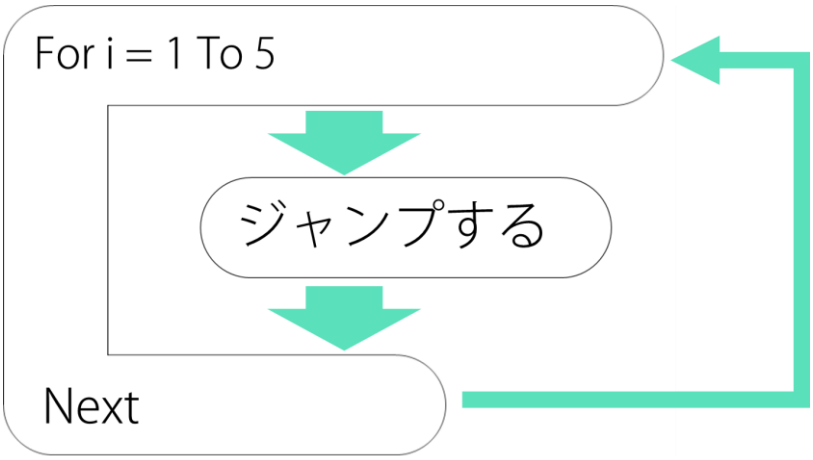
このように表すことができます。
（プログラミング的に書きあわらしているので、やや形式ばった言い方になっていますが）



この記述方法が、プログラミングでいう「反復（ループ）」というものです。
見た目上にもスッキリした記述になったことがわかります。
また、この記述方法ならば、100 回繰り返すとしても最初に「100 回目まで」と書き換えてしまえば、簡単に変更することができます。

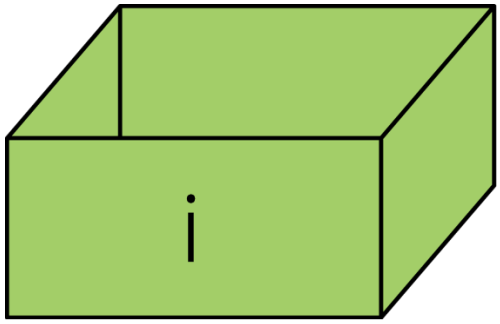
さて、「プログラミング的に書きあわらしている」と先述しましたが、VBA の書き方で本当にこの反復（ループ）を書き表した場合、どのように書き表わせば良いでしょうか。

```
For i = 1 To 5
  [ジャンプをする命令]
Next
```



このように記述します。
先ほど日本語では「以下を繰り返します。」～「次の繰り返し」と書き表した枠取りを、
「For」～「Next」と書き換えます。

では、「i」とは何でしょうか？
実はこれもカウント変数なのです。
カウント変数とは、回数などをカウントするために用意する変数の「箱」のことですね。



小さい子が、回数を数える時に「指」をつかって回数を数えるときに似ていると説明しました。

「For～」のループの場合は、１回反復したらカウンタ変数の「i」を１つ増やす、ということが行われます。

先頭の以下の表現についてですが、

　For i = 1 To 5

　箱「i」が 1 から始まり、5 になったら反復を終了。

という意味になります。

１回目のループ

```
For i = 1 To 5                【i は 1 からスタート】
  [ジャンプをする]
Next                          【i は+1 増加】【i は 2 になった】
```

２回目のループ

```
For i = 1 To 5
  [ジャンプをする]
Next                          【i は+1 増加】【i は 3 になった】
```

３回目のループ

```
For i = 1 To 5
  [ジャンプをする]
Next                          【i は+1 増加】【i は 4 になった】
```

このようにして、i が 5 になるまでループを繰り返し、i が 5 になった周に「Next」にたどり着いたら、このループを終わります。

少し混乱するかもしれませんが、VBA のコードを実際に入力しながら慣れていくものなので、今は例文をそのまま真似して慣れていきましょう。

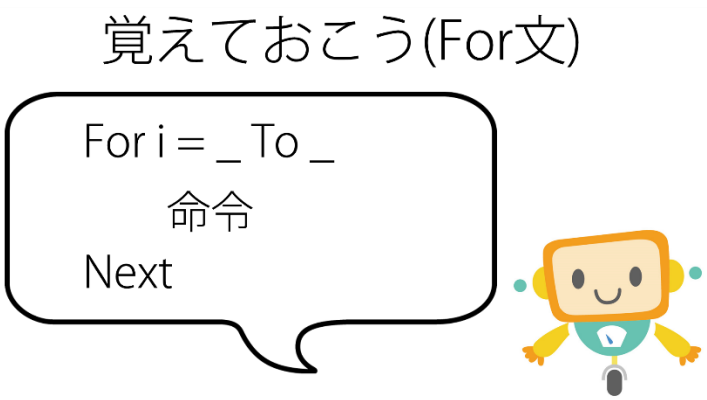
【覚えておこう】

反復（ループ）の構文

For ｉ = 始点 To 終点

［命令］

Next



・For 文によるループ

【2-13】反復（ループ）を使ってみよう

✎【新しく入力してみよう

```
Sub Routine()
    For ｉ = 1 To 3
        MsgBox（"ジャンプ"）
    Next
End Sub
```

【実行】

「ジャンプ」というメッセージボックスが 3 回連続で表示されたはずです。

これは、カウンタ変数「i」が 1 から始まり、1,2,3 と増えながら 3 回命令が反復されたためです。

補足：なぜカウンタ変数は「i」というの？

これは、整数をあらわす「Integer」という単語の頭文字をとって慣習的に「i」とされているものと言われています。しかし、実はカウンタ変数はなにも「i」でなくても構いません。「Int」や「Num」などとあらわす人もいます。VBA やその他のプログラミング言語でも、For 構文に用いるカウンタ変数はどんな名前でも構いません（ただし他の変数名と重複しないようにすること）。

【試してみよう】

For の反復回数指定や変数を変えてみましょう。

・「For i = 1 To 5」に変更してみる。

・「For i = 3 To 7」に変更してみる。

・「For a = 0 To 3」に変更してみる。

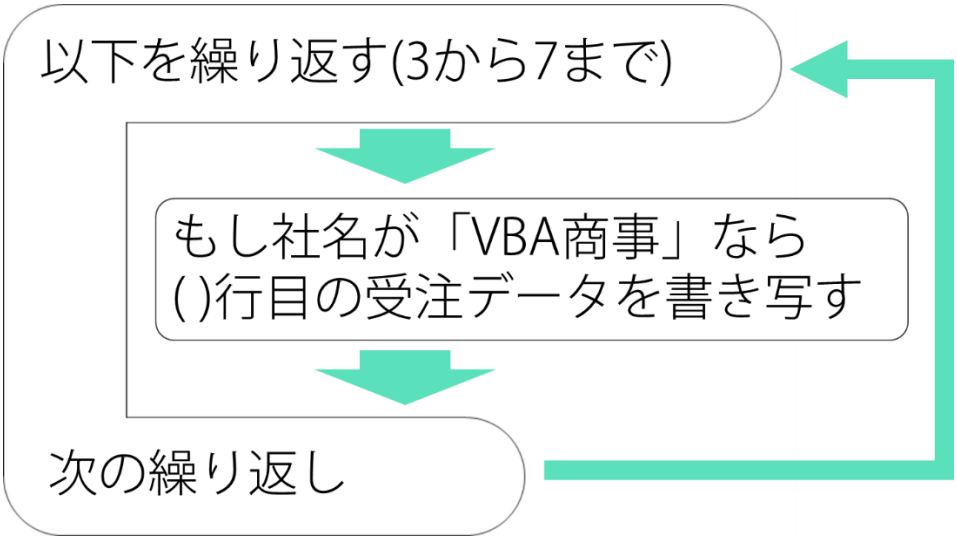
次の変更も試してみましょう。

```
✎【変更してみよう】
Sub Routine()
    For i = 1 To 3
        MsgBox (i)
    Next
End Sub
【実行】
```

メッセージボックスで「1」「2」「3」と順番に出力されたはずです。
このループでは、カウンタ変数 i が「1」から始まり、反復するたびに「1」「2」「3」と＋１ずつ増加しているために、メッセージボックスで出力される内容も i に合わせて「1」「2」「3」と１つずつ増加していることがわかります。

【2-14】請求書作成マクロ Lv.5（For 文で繰り返し）

それでは、請求書作成マクロに戻ります。
前セクションまで作成してきた「請求書作成マクロ」では、注文データを 5 行読み込むために、同じ VBA の命令を 5 回打ち込んで実行させていました。
しかし、それではとても冗長ですし、もし注文データが 100 行など膨大にあった場合には対処しきれません。そこで、For 構文を使ったループ（反復）による表記に変更したいと思います。



「( ) 行目」には、3 から 7 までの数字が入ります。

受注データを上から読んでいく場合、最初のデータは Excel シートの 3 行目から始まるので、まずは「(3)行目」から始まります。そして、1 度目のループを終えたら「(4)行目」にうつり、次のループを終えたら「(5)行目」にうつり…（繰り返し）最終的に「(7)行目」にうつって命令を実行したら、この反復を終えます。

実際にソースを書き込んで実行しながら確かめてみましょう。

✎【修正してみよう】

```
Sub Seikyu2()
    Dim Cnt As Integer
    Cnt = 9

    For i = 3 To 7
        If Cells(i, 2).Value = Cells(5, 10).Value Then
          Range(Cells(Cnt, 10), Cells(Cnt, 13)).Value(改行せず続けて入力して下さい)
                                        = Range(Cells(i, 3), Cells(i, 6)).Value
          Cnt = Cnt + 1
        End If
    Next
End Sub
```

【実行】

前セクションまでよりも随分、文章がすっきりしたことがわかりますね。

少し構文が複雑になってきますので、タイプミスなどでエラーが起こらないように注意して入力して下さい。
問題なく入力できていれば、マクロを実行すると請求書の書き込みが行われたはずです。

それでは、一文ずつ見ていきます。

For i = 3 To 7
カウント変数 i が、「3」から始まって「7」になるまで＋１ずつ増加しながらループすることを意味します。

If Cells(i, 2).Value = Cells(5, 10).Value Then
もしセル(i 行の 2 列）の値が、　セル(5 行の 10 列)の値に等しいならば…
という意味です。

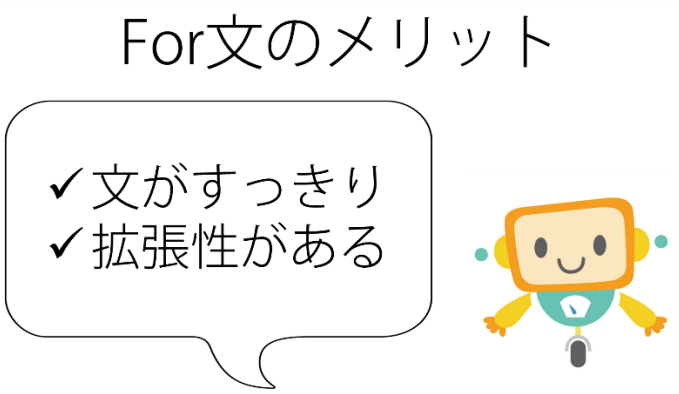
Range(Cells(Cnt, 10), Cells(Cnt, 13)).Value = Range(Cells(i, 3), Cells(i, 6)).Value
これは、i が入っている右辺から見てみましょう。

= Range(Cells(i, 3), Cells(i, 6)).Value
範囲（セル（i 行の 3 列）～セル（i 行の 6 列））の値を…（請求書に書き移す）
という意味です。

i は反復するたびに＋１ずつ増えていきますので、
１回目のループならば「3 行目の…」という意味になりますし、
２回目のループならば「4 行目の…」という意味になり、
３回目のループならば「5 行目の…」という意味になります。
（繰り返し）
５回目のループならば「7 行目の…」という意味になり、反復が終了します。

【補足】繰り返しを用いるメリット
このように繰り返し（ループ）を用いることで、以下のようなメリットがあります。
・コードがすっきりする。（同じ命令文を何度も繰り返し記述する必要がない）
・データが何 10 行、何 100 行に増えたとしても対応できる。

（「For～」の条件だけを変更すれば、すぐに対応できます）



どちらかというと、後者のメリットのために反復（ループ）を使うことが VBA プログラミングでは多いと言えます。

【2-15】この請求書マクロの欠点！？

お疲れ様でした。ここまでを制作できれば、VBA マクロの初級編としては十分といえる知識の骨組みが備わったと言えます。
ただし、ここまで制作した「請求書マクロ」には欠点があります。

・データの行数が 5 行に限られている。
注文データがどんどん増加していった場合行数の増加に自動対応することができません。

・注文データと請求書が、同じ１つのシートに収まってしまっている。
請求書は別のシートに作ってプリントアウトしたいが、現状それができていません。

上記のような問題があります。
もうすでにマクロとしてかなり実用的な機能が作られているのですが、実務で使うとなるともう少し機能を詰めておきたい所ですね。

そこで、次の「中級編」では、上記のような問題を解決する機能を追加していきます。

【第３章】中級編「使える請求書マクロ」を作ろう

それでは、シート「中級」を開いて下さい。
いよいよ中級編の課題では、実務で使える形式の請求書作成マクロを作っていきます。

シート「中級」　　シート「請求書」

受注一覧

	会社名	商品名	単価	個数	金額
1	VBA商事	オフィス用コーヒー	330	15	4950
2	エクセル運輸	オフィス用カフェラテ	360	21	7560
3	VBA商事	コーヒー豆100g	300	10	3000
4	関数カンパニー	オフィス用コーヒー	330	18	5940
5	エクセル運輸	コーヒー豆100g	360	14	5040
6	表計算コーポ	オフィス用カフェラテ	360	11	3960
7	パワポ建設	オフィス用コーヒー	330	8	2640
8	VBA商事	オフィス用カフェラテ	360	3	1080
9	エクセル運輸	コーヒー豆100g	300	5	1500
10	パワポ建設	オフィス用カフェラテ	360	2	720
11	関数カンパニー	コーヒー豆100g	360	1	360
12	エクセル運輸	オフィス用コーヒー	330	7	2310
13	関数カンパニー	コーヒー豆100g	360	6	2160
14	VBA商事	コーヒー豆100g	300	12	3600
15	エクセル運輸	オフィス用コーヒー	330	14	4620
16	VBA商事	オフィス用コーヒー	330	3	990
17	表計算コーポ	オフィス用カフェラテ	360	9	3240
18	エクセル運輸	コーヒー豆100g	300	13	3900
19	VBA商事	コーヒー豆100g	360	11	3960
20	表計算コーポ	オフィス用カフェラテ	360	7	2520
21	関数カンパニー	オフィス用コーヒー	330	6	1980
22	VBA商事	オフィス用カフェラテ	360	5	1800

発行日：　2016/4/22
請求書番号：　A0001

請求書

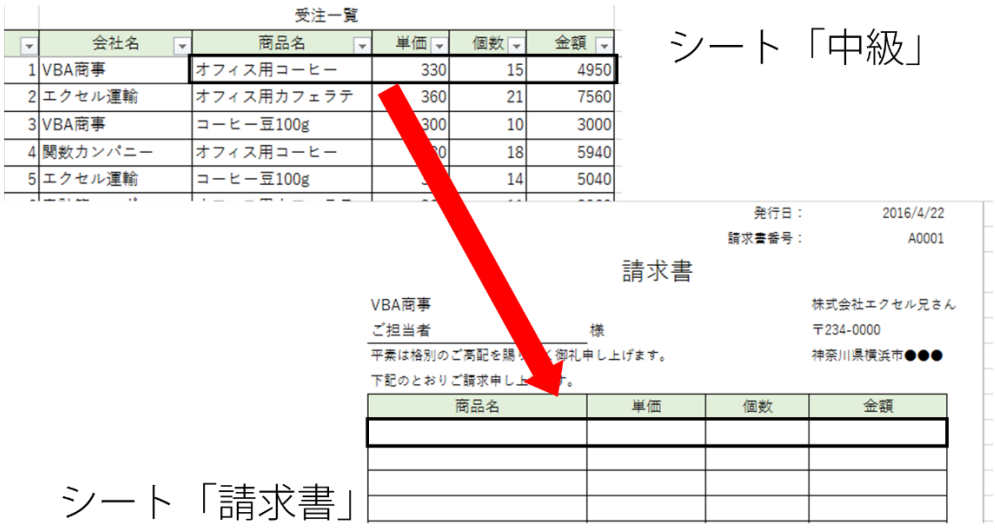
VBA商事　　株式会社エクセル兄さん
ご担当者　　様　　〒234-0000
平素は格別のご高配を賜り厚く御礼申し上げます。　　神奈川県横浜市●●●
下記のとおりご請求申し上げます。

商品名	単価	個数	金額

シート「中級」には受注データが 30 件記録されています。
シート「請求書」には、未記入の請求書フォーマットがあります。

本来、仕事で使うような Excel ブックはこのように２つのシートに分かれているはずです。
そこで、今回作るマクロは、１つのシートでマクロを実行して、その結果を２つ目のシートに出力するという内容にする必要があります。

シート「中級」　→（データを書き移す）　→シート「請求書」



受注一覧

	会社名	商品名	単価	個数	金額
1	VBA商事	オフィス用コーヒー	330	15	4950
2	エクセル運輸	オフィス用カフェラテ	360	21	7560
3	VBA商事	コーヒー豆100g	300	10	3000
4	関数カンパニー	オフィス用コーヒー	330	18	5940
5	エクセル運輸	コーヒー豆100g	360	14	5040

シート「中級」

発行日：　2016/4/22
請求書番号：　A0001

請求書

VBA商事　　株式会社エクセル兄さん
ご担当者　　様　　〒234-0000
平素は格別のご高配を賜り厚く御礼申し上げます。　　神奈川県横浜市●●●
下記のとおりご請求申し上げます。

商品名	単価	個数	金額

シート「請求書」

【3-1】シートをまたいだセル指定（Worksheets）

さて、今回のように、「別のシートを操作する」という命令を記述するときにしなければいけないことは、シート名を（必ず）明記するということです。

✎【新しく入力してみよう】

```
Sub OtherSheet()
    MsgBox(Cells(8,2).Value)
    MsgBox(Worksheets("請求書").Cells(8,2).Value)
End Sub
```

【実行】

二回のメッセージボックスにはどんな違いがあったでしょうか？

MsgBox(Cells(8,2).Value)
これは前セクションまで記載してきた記述方法でした。

MsgBox(Worksheets("請求書").Cells(8,2).Value)
こちらは、Worksheets("請求書")と記載することで、「シート「請求書」の…」とシート名を明記しています。すると、指定したシートのセルを参照することができるのです。

補足：シート名は書かなくてもいいの？
前者のように、シート名を明記しなかった場合にはどうなるのでしょうか？
シート名を明記しなかったときには、「現在アクティブなシート」からセルが選ばれます。
「アクティブなシート」という意味は、現在「選択されている」ということだと理解しておいて下さい。

【覚えておこう】

シート名を明記してセルを参照する場合
Worksheets("シート名").Cells(,)
※シート名は""で囲うこと。
※シート名を明記しなかった場合は、アクティブな（現在選択されている）シートから参照される。

【3-2】請求書作成ボタン Lv.1（Worksheets を指定）

・まずは請求先の会社名を書き移させよう
それでは請求書マクロの制作に取り掛かっていきます。
といっても、入門～初級の課題を制作できたあなたなら、怖がることはありません。
これまでと基本的には同じような内容を記述していきます。ただし、今回は「シート名」を明記することを忘れずに記述していきます。

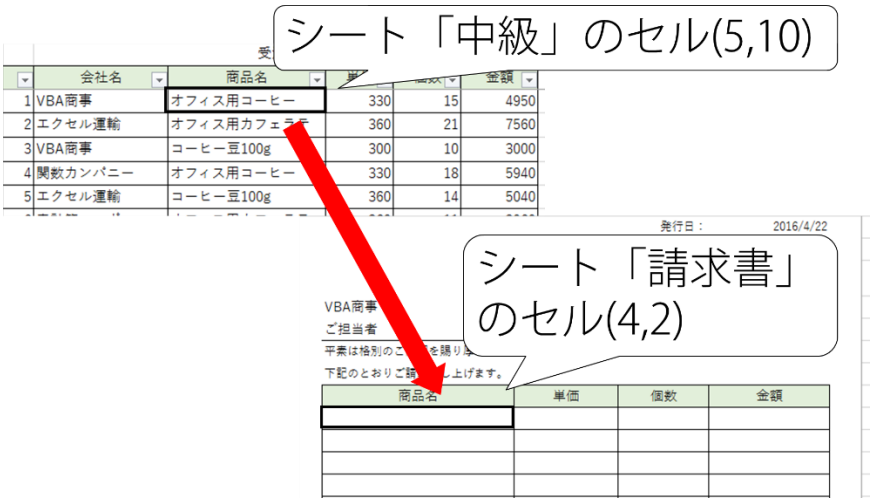
✎【新しく入力してみよう】

```
Sub Seikyu3()
    Worksheets("請求書").Cells(4, 2).Value    (改行せず続けて入力して下さい)
                                    = Worksheets("中級").Cells(5, 10).Value
End Sub
```

【実行】

いかがでしょうか。問題なく実行されれば、シート「中級」のセル J5 に入力されていた会社名が、シート「請求書」のセル B4 に書き移されているはずです。
文章の意味を読み解いてみましょう。

Worksheets("請求書").Cells(4, 2).Value = Worksheets("中級").Cells(5, 10).Value
シート「請求書」の　セル(4 行 2 列)の値に　←　シート「中級」のセル(5 行 10 列)の値を書き移す。



【3-3】クリアボタン Lv.1（Worksheets を指定）

ここまでの課題でも作成してきたお馴染みの「クリアボタン」を作っていきましょう。
今までと同じようなものですが、やはりワークシート名を明記することを忘れないようにしましょう。

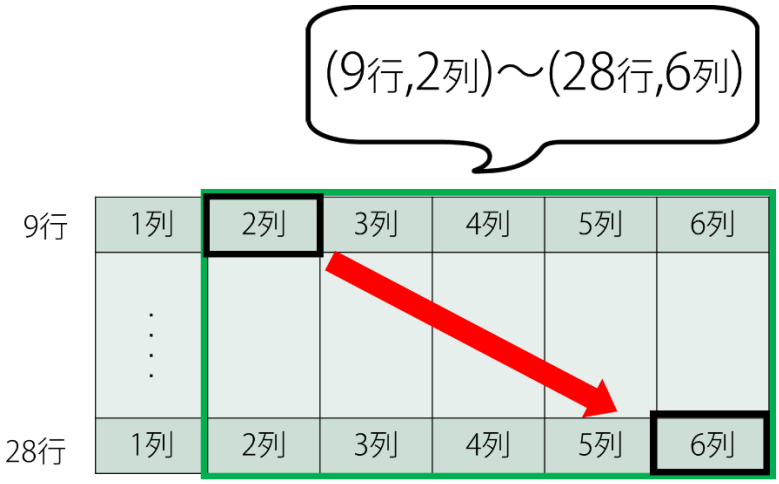
✎【新しく入力してみよう】

```
Sub Clear3()
    Worksheets("請求書").Range(Cells(9,2),Cells(28,5)).Value = ""
End Sub
```

【実行】

問題なく実行できれば、シート「請求書」の B9 から E28 の範囲がクリアされます。
上記のコードを一語ずつ読み解いてみましょう。

Worksheets("請求書").Range(Cells(9,2),Cells(28,5)).Value = ""
ワークシート「請求書」の　範囲(セル(9 行 2 列)～セル(28 行 5 列)の値に
空白を書き込む

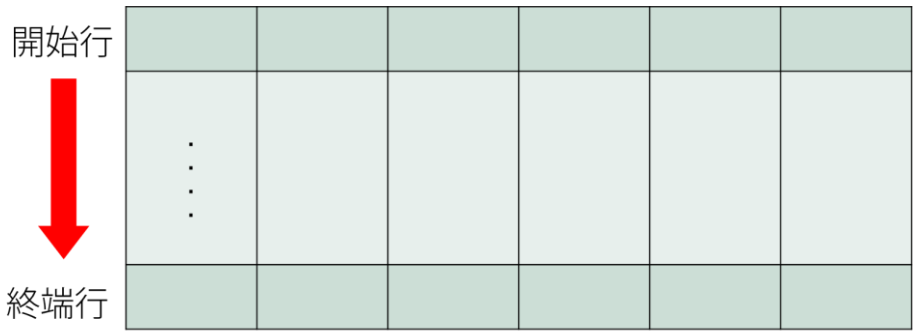


【3-4】終端行を取得する（End(xlDown)）

さて、ここからは「請求書」を作成するマクロの本筋となるコードを作っていきます。

その前に、**「終端行を取得する」**ということを行っておきます。
「終端行」というと何やら難しそうな響きの言葉ですが、何を意味しているか？

それは、「注文データの表の一番下（終端）は、何行目なのか？」を調べるということです。



シート「中級」に記載されている注文データの一覧表をご覧下さい。
この注文データは、現状は 30 件のデータが入力されているので終端行は「シートの 32 行目」であることがわかります。
しかし、ふつう注文データは日々、追加されていきます。
ですから、明日になったら 50 件に増えているかもしれないし、100 件に増えているかもしれません。そうなると、VBA のプログラムには「シートの 32 行目までのデータを書き移す」とだけ限定して記述してしまうと、増えた分のデータまでは書き移すことができないことになってしまいます。
では、どうすれば良いでしょうか。

マクロを実行した時点で、「いま現在の注文データの終端行は、何行目なのか？」を調べさせるように VBA で記述しておけば良いということです。

そうしておけば、終端行が 32 行目ならば「32 行目」、終端行が 52 行目ならば「52 行目」と、マクロが自分で判断をしてデータを書き移してくれることになります。これは便利ですね。

では、終端行を取得するための記述方法を次に示します。

✎【新しく入力してみよう】

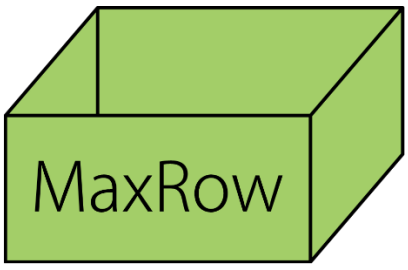
```
Sub MaxRow()
    Dim MaxRow As Integer
    MaxRow = Worksheets("中級").Cells(3,2) .End(xlDown).Row
    MsgBox MaxRow
End Sub
```

【実行】

```
Dim MaxRow As Integer
```

定義する 「MaxRow」 数値型として

まずは、変数「MaxRow」を宣言しています。（Row は行をあらわす単語です）
この変数の箱を用意しておいて、この中に終端行は何行目か？という数値を記憶させる目的のためです。




```
MaxRow = Worksheets("中級").Cells(3,2) .End(xlDown).Row
```

箱「MaxRow」　←シート「中級」のセル(3,2)と　連なっているデータの最終行

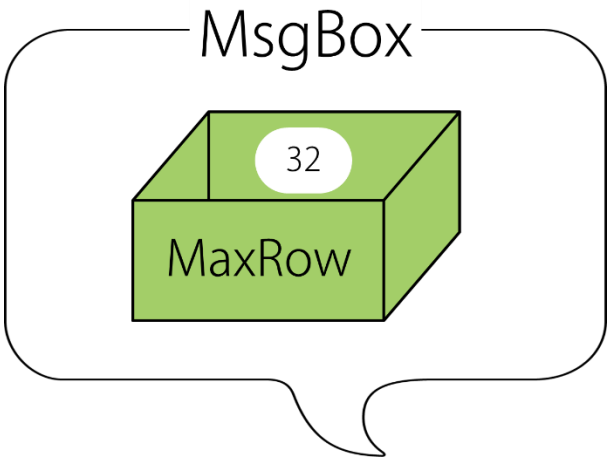
「Cells(3,2).End(xlDown).Row」
この記述は新しく出てきた記述ですね。

Cells(3,2).End(xlDown)は、セル(3 行 2 列)と連なっている（塊になっている）データの一番下のセルを指定する記述方法です。

3 行目	Cells(3,2)↓		
…	↓		
32 行目	Cells(3,2).End(xlDown)		

Cells(3,2).End(xlDown).Row
その.Row（行番号）を参照するという記述が上記のコードですので、「32（行目）」ということになります。

MsgBox MaxRow
メッセージ出力　「MaxRow」の中身を



この時点で変数の箱「MaxRow」の中には、32 が入れ込んでありますので、メッセージボックスには「32」と出力されることになります。

これで、終端行を調べる方法がわかりました。
終端行の行番号は、変数の箱「MaxRow」に入れ込むことができました。

それではいよいよ仕上げです。
請求書を作るコードを完成させましょう！

【3-5】請求書ボタン Lv.2（終端行を指定）

✎【修正してみよう】

```
Sub Seikyu3 ()

  Dim Cnt As Integer
  Dim MaxRow As Integer

  MaxRow = Worksheets("中級").Cells(3,2).End(xlDown).Row
  Cnt = 9

  For i = 3 To MaxRow
     If Worksheets("中級").Cells(i, 2).Value （※改行せずに１行で続けて入力して下さい。）
                                    = Worksheets("中級").Cells(5, 10).Value Then

        Range(Worksheets("請求書").Cells(Cnt, 2), Worksheets("請求書").Cells(Cnt,
        5)).Value = Range(Worksheets("中級").Cells(i, 3), Worksheets("中級").Cells(i,
        6)).Value （※改行せずに１行で続けて入力して下さい。）

       Cnt = Cnt + 1

    End If
  Next

  Worksheets("請求書").Cells(4, 2).Value = Worksheets("中級").Cells(5, 10).Value
  Worksheets("請求書").Activate

End Sub
```

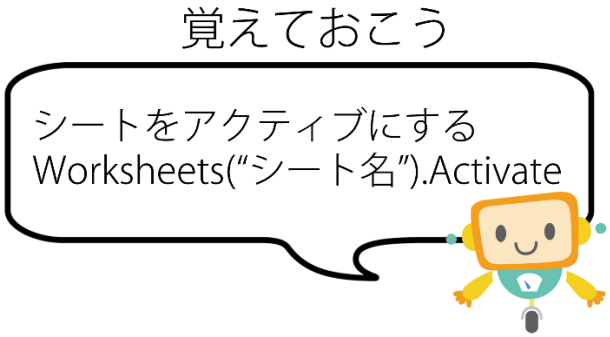
【実行】

```
For i = 3 To MaxRow
```

繰り返しのための「For～」文です。「i= 3 To MaxRow」は、受注データの最初の行番号（3 行目）から終端行の行番号（MaxRow）になるまで繰り返すことを意味しています。

```
Worksheets("請求書").Activate
```

ワークシート「請求書」を　アクティブにする一文です。
マクロの実行後に自動的にシート「請求書」に切り替わるようになっています。



【3-6】PDF 出力ボタンを作ろう

最後の仕上げとして、PDF ファイルを出力するボタンを作ります。

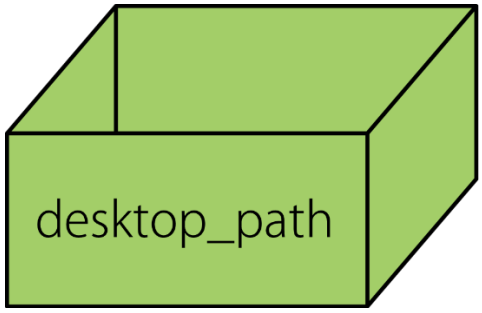
✎【新しく入力しよう】

```
Sub OutputPdf ()

    Dim desktop_path As String
    desktop_path = CreateObject("WScript.Shell").SpecialFolders.Item("Desktop")


    ActiveSheet.ExportAsFixedFormat Type:=xlTypePDF,  （※改行せずに 1 行で続けて入力）
                    Filename:=desktop_path & "¥" & Cells(3, 8).Value
End Sub
```

【実行】

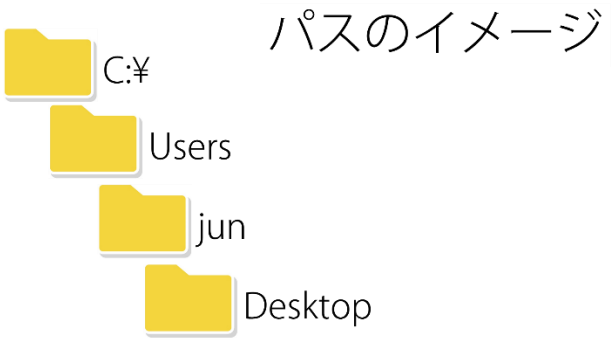
```
Dim desktop_path As String
```

PDF ファイルを出力するために、パソコンのデスクトップのパスを取得するための「箱」を用意しています。ここでは「desktop_path」という箱を用意しています。



【補足】「パス」って何？

パスとは、ファイルやフォルダがどの位置にあるのかを示す番地のようなものです。



Windows の場合ならば、

「C:¥」というフォルダの中に「Users」というフォルダがあって、

その中に「jun」というフォルダがあって、その中に「Desktop」がある（一例です）

…というような構造になっています。

それをパスとして表すと、以下のように「¥」で区切って表します。

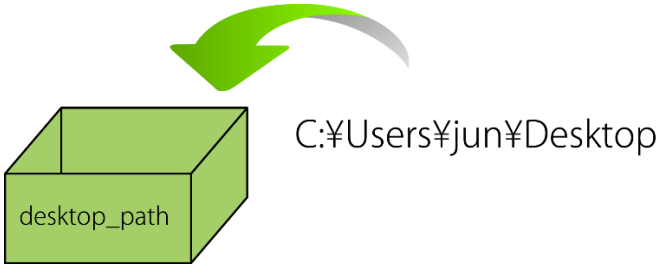
C:¥Users¥jun¥Desktop

```
desktop_path = CreateObject("WScript.Shell").SpecialFolders.Item("Desktop")
```

変数 desktop_path に、デスクトップのパスを入れ込む。

準備しておいた箱「desktp_path」に、先ほどのパスを入れ込んでいます。

ここの記述方法は、決まり文句のように覚えておいて下さい。

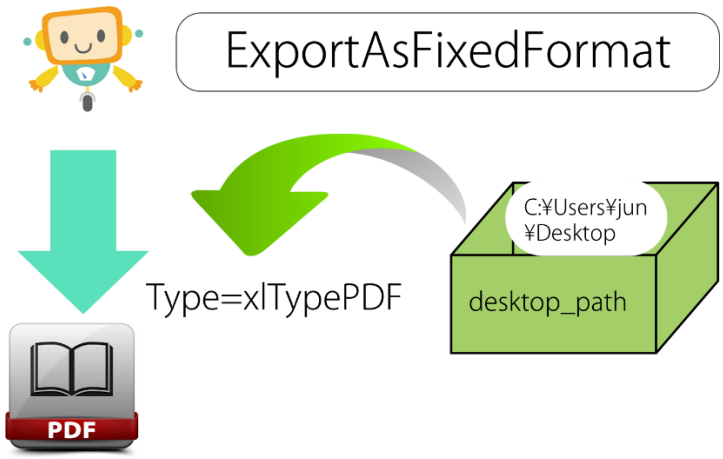


```
ActiveSheet.ExportAsFixedFormat Type:=xlTypePDF,　（※改行せずに１行で続けて入力）
                    Filename:=desktop_path & "¥" & Cells(3, 8).Value
```

少し長い一文ですが、ここで実際に PDF ファイルを出力しています。
全文を完全に理解する必要はありませんが、イメージとしてとらえておきましょう。

変数「desktop_path」に入っている値と、セル(3,8)に書いているファイル名を「&」で組み合わせています。

また、「ExportAsFixedFormat」では、あるフォーマットにしたがってファイルを出力しています。「Type=xlTypePDF」では、「PDF ファイルとして」とファイル形式を指定しています。



※シート「請求書」の右側に、「PDF 出力」ボタンも作っておきましょう。