# その他の基本的な知識・テクニック

## 乱数

　サイコロを振ったとき、1～6の何が出るかは分かりません。このような数を乱数と言い、乱数を並べたものを乱数列と言います。サイコロを振って生成される乱数列は、1～6の数が不規則に並び、各数値の発生確率は1/6です。このような数を人間が人為的に作るのは困難です。規則性が生じてしまったり、確率が等しくなくなったりします。身近な乱数としては、や*e*（自然対数の底）があります。

　学校現場では席替えや班分けなどのときに乱数が必要です。

　Excel VBAで乱数を使う方法以下の3通りがあります。

(1) ワークシート関数RandBetweenを使う

(2) ワークシート関数RANDを使う

(3) VBA関数Rnd()を使う

　ワークシート関数とはセルの中に「=SUM(A1:A3)」のように書き込める関数を指します。乱数系のワークシート関数は、どこかのセルの内容を更新するたびに、実行されます。

　乱数の質については和田維作氏による

<http://isaku-wada.my.coocan.jp/rand/rand.html>　良い乱数・悪い乱数

が非常に詳しくてわかりやすいです。このサイトによると、(1)(3) は質が良くないようですが、ここでは乱数の質はパスして次に進みます。

　3つの方法の中で一番わかりやすいのがRandBetween(a,b)を使う方法なので、RandBetweenを使う方法から話を始めます。

　シート「乱数1」を使います。

　以下のプログラムを実行すると、変数aに1～6の乱数が入ります。ワークシート関数をExcel VBAから利用するときは、WorksheetFunction.関数名　という形式で用います。ピリオドはオブジェクトの階層構造を表します。日本語の「の」に相当すると考えればよいです。この場合は「ワークシート関数」の「RandBetweenという名前の関数」と解釈してください。

 a = WorksheetFunction.RandBetween(1, 6)

　セルA1～A20に1～6の乱数を入れるプログラムは以下のようになります。

 For i = 1 To 20

 Range("A" & i) = WorksheetFunction.RandBetween(1, 6)

 Next i

　ExcelのWorksheetFunction.RandBetweenが作る乱数は良い乱数とは言えないように思われます。偏りがあります。

　C言語, pythonなど通常のプログラミング言語において、乱数を発生させる関数は、「0以上1未満」の値を返します。「0以上1未満」の数を利用して「1～6の整数」を得るには、以下のようにします。

|  |  |
| --- | --- |
| 数式 | 取り得る範囲 |
| Rnd() | 0～0.9999…… |
| Rnd()\*6 | 0～5.9999…… |
| Int(Rnd()\*6) | 0, 1, 2, 3, 4, 5 |
| Int(Rnd()\*6)+1 | 1, 2, 3, 4, 5, 6 |

　このように、1～nまでの整数の乱数を得るには、「0以上1未満」の乱数に対して、「nをかける」「整数部を取る」「1を足す」という操作をします。

 For i = 1 To 20

 Randomize

 Range("A" & i) = Int(Rnd() \* 6) + 1

 Next i

　Rnd() 関数はあらかじめ定められた乱数テーブルに従って乱数を返します。Randomize はシステムタイマーから取得した現在の時刻を元にして、使用する乱数テーブルを指定する。上記のRandmizeを省略してもプログラムとしては問題ないが、毎回Randomizeすることで、乱数の質を良くすることができる。

【発展】

　和田氏のページによると、ワークシート関数のRAND() は良質な乱数を提供すると書いてあります。しかし、なぜかRAND()はWorkSheetFunction.RAND() と書いても使えない。以下のように書く必要がある。

 a = Application.Evaluate("RAND()")

 b = Int(a \* 6) + 1

　シート「乱数2」に切り替えて下さい。A列にA～Yまで25名の名前があります。もとの名簿を保存するため、いったんB列にコピーし、B列を操作することにします。掃除当番を決めたいと思います。名前をランダムに並べ直してから、1班3人としてE, F, G列に並べます。

方法１

1. 25人から1人を選び、その人をB1に入れる。元々B1に入っていた名前は、選ばれた人の場所に入れる。
2. 24人から1人を選び、その人をB2に入れる。元々B2に入っていた名前は、選ばれた人の場所に入れる。
3. 以下、この繰り返し。

　この考えをプログラムにすると、以下のようになります。

 For i = 1 To 24

 ' i 行目を確定させる

 a = WorksheetFunction.RandBetween(i, 25) ' aはi～25 の乱数

 If a <> i Then

 tmp = Range("B" & i) ' i行目とa行目を

 Range("B" & i) = Range("B" & a) ' 入れ替える

 Range("B" & a) = tmp

 End If

 Next i

　3.1節で学習したように、値を入れ替えるときは、一時的に値を保持する変数（ここではtmp）が必要です。

　上記のサンプルではa <> iのときに入れ替えていました。a = iの場合に、Ifの中を実行してもセルに変化はありません。従って、

 For i = 1 To 24

 a = WorksheetFunction.RandBetween(i, 25) ' i～25 の乱数

 tmp = Range("B" & i) ' 入れ替え

 Range("B" & i) = Range("B" & a)

 Range("B" & a) = tmp

 Next i

と書いても動作は同じです。筆者はIfがあった方が分かりやすいと思うのですが、ここは好みの問題かも知れません。

　ランダムに名前を並べ替えるとき、多くの人が最初に思いつくのは以下の方法だと思われます。

方法2

 2人を選び、その2人の場所を入れ替える

 これを何回か繰り返す

　この考えをプログラムにすると以下のようになります。先述のように、Ifはなくてもかまいません。

 For i = 1 To 30

 a = WorksheetFunction.RandBetween(1, 25)

 b = WorksheetFunction.RandBetween(1, 25)

 If a <> b Then

 tmp = Range("B" & a)

 Range("B" & a) = Range("B" & b)

 Range("B" & b) = tmp

 End If

 Next i

　この方法は良いとは言えません。ここでは30回繰り返していますが、何回繰り返したら十分ランダムになるかの基準が明確ではないからです。

　以上2つの方法を示しました。このようにしてB列にランダムに並べた名前を3人1班としてFGH列に並べるプログラムは既に学習しました。以下のようになります。

 gyou = 1

 retu = 6

 For i = 1 To 25

 Cells(gyou, retu) = Range("B" & i)

 retu = retu + 1

 If retu = 9 Then

 retu = 6

 gyou = gyou + 1

 End If

 Next i

　最後の人は、一人でやるのはかわいそうなので、手作業で8班に入れてあげましょう。

【課題1】

　A列から2人選んで、C1, C2に入れます。

　まず、A列をB列にコピーしなさい。B列について作業をします。

　B1～B14から1人選んでC1に書き込みなさい。たとえばB8を選んだ場合、B9～B14をB8～B13にコピーし、B14は空にしなさい（B9～B15をB8～B14にコピーするのと同じです）。すなわち選んだ人を消去し、残りの人を1行ずつ前に詰めます。

　B1～B13から1人選んでC2に書き込みなさい。そして選んだ人を消去し、残りの人を1行ずつ前に詰める操作をしなさい。

　可能であれば、以上の操作をn回（nを変数にする）できるようなプログラムにしなさい。

【例題】（100マス計算）

　100マス計算は陰山英男先生が考案された学習法です。このシートを作ってみましょう。まず、A2～A10に1～9の数字を入れ、B1～J1に1～9の数字を入れます。

 For i = 1 To 9

 Cells(1 + i, 1) = i ' A1～A9 に 1～9 を入れる

 Cells(1, 1 + i) = i ' A2～I2 に 1～9 を入れる

 Next i

　次にA2～A10, B1～J1のセルの数値をランダムに並べ替えます。

 For i = 2 To 10

 a = WorksheetFunction.RandBetween(i, 10) ' 1列目の

 tmp = Cells(i, 1) ' a 行目と

 Cells(i, 1) = Cells(a, 1) ' i 行目を入れ替える

 Cells(a, 1) = tmp

 Next i

 For i = 2 To 10

 a = WorksheetFunction.RandBetween(i, 10) ' 1行目の

 tmp = Cells(1, i) ' a 列目と

 Cells(1, i) = Cells(1, a) ' i 列目を入れ替える

 Cells(1, a) = tmp

 Next i

　これで100マス計算のシートができました。

　次に2重ループを利用して、答えを書き込んでみましょう。

 For i = 2 To 10

 For j = 2 To 10

 Cells(i, j) = Cells(i, 1) + Cells(1, j)

 Next j

 Next i

【ビンゴゲーム】

　シート「ビンゴゲーム」にビンゴゲーム用のシートを作ります。1列目は1～15, 2列目は16～30, 3列目は31～45, 4列目は46～60, 5列目は61～75が入ります。ただし、真ん中のセルCells(3, 3) には "FREE" という文字を入れます。

　次のようなアルゴリズムで、シートを作ります。

1. retu = 1 To 5 としてループを組む。

2. F列を作業領域として使う。F1～F15に列に対応する数字を入れる。

3. F列をランダムに入れ替える。

4. F列の1～5行目をretu列目の1～5行目にコピーする。

5 Cells(3, 3) に "FREE" と入れる。

 a = 0

 For retu = 1 To 5

 For i = 1 To 15

 Range("F" & i) = a + i ' F列に数値を入れる

 Next i ' a+1 ～ a+15

 a = a + 15

 For i = 1 To 15

 r = WorksheetFunction.RandBetween(i, 15)

 tmp = Range("F" & i) ' i 行と ｒ 行を

 Range("F" & i) = Range("F" & r) ' 入れ替える

 Range("F" & r) = tmp

 Next i

 For i = 1 To 5 ' F列目の 1～5行目を

 Cells(i, retu) = Range("F" & i) ' retu列の 1～5行目に

 Next i ' コピーする

 Next retu

 Cells(3, 3) = "FREE" ' 中央の場所は FREE

## ボタン

　これまでの方法では、プログラムを実行するには「開発」タブを表示させてVisual Basic Editorを起動し、「実行」を命令していました。

　Excel VBAの使い方を知らない人にもプログラムを使ってもらえるようにしましょう。

　「開発」→「コントロール：挿入」→「フォームコントロール：ボタン」を選択すると、ボタンを作成するモードになります。ドラッグするとボタンを作成します。altを押しながらドラッグすると、ボタンの境界をセルの境目に合わせます。「マクロの登録」ウィンドウが開きます。「マクロ」とは「SubからEnd Subまでの1つのプログラム」を意味します。そのボタンに結びつけるプログラムを選びます。

　後から「ボタンの上で右クリック」→「マクロの登録」でそのボタンに結びつけるプログラムを変更することができます。

　ボタン上に表示するテキストの設定は「右クリック」→「テキストの編集」です。

　ボタンの消去は「右クリック」→「切り取り」です。

【例題】

　カウンタを作ります。A1にはあらかじめ数値が入っていることを仮定します。

ボタンは2つあります。ボタンAを押すと、1増加するようにしなさい。ボタンBを押すと、1減少するようにしなさい。ただし0以下にはならないようにしてください。

　ボタンAの表面には「カウントアップ」、ボタンBの表面には「カウントダウン」と表示しなさい。

【解答】

　ボタンAを押したときに呼ぶ関数をcount\_up、ボタンBのときはcount\_downとするとき、以下のようになる。

 Sub count\_up()

 Range("A1") = Range("A1") + 1

 End Sub

 Sub count\_down()

 If Range("A1") >= 1 Then

 Range("A1") = Range("A1") - 1

 End If

 End Sub

【課題2】

　暗算ゲームを作ります。2つボタンを設置し、ボタンA, Bと呼びます。ボタンAを押したらA1とA2に10～99までの乱数を表示しなさい。A1とA2の和がA3に入っていることを仮定します。ボタンBを押したら、答えが正しければB1に「正解」そうでなければ「誤り」と表示するプログラムを作成しなさい。

## マクロの記録

　例えば、「行高を指定したい。どのような命令を書けばよいか？」を知りたいとき、

以下の方法があります。

1. Webで調べる。「Excel VBA　セル　行高」としてgoogleで検索すると該当ページが見つかるでしょう。
2. 書籍で調べる。「できる大事典　Excel VBA」のような辞書的な本を用いて「セルの高さ」の項を見ると書いてあります。ネットが使えない環境では書籍が必要でしょう。
3. 自分で調べる。今から説明します。

マクロの記録

　以下のように操作して下さい。

1. 「開発」→「コード：マクロの記録」をクリックします。その場所のアイコンが「記録終了」に変わります。
2. どこかの行を選択し、行高を変更します。
3. 「記録終了」を押します。一連の作業が1つのプログラムとして記録されます。

　VBAエディタのオブジェクトエクスプローラの欄（左上の「プロジェクト - VBAProjectと書いてあるウィンドウ）を見て下さい。プロジェクトエクスプローラのウィンドウが見当たらない場合は「表示」→「プロジェクトエクスプローラ」で開いて下さい。新しいモジュールが一つ追加されています。ダブルクリックして中を見て下さい。6行目のセル高さを50.25にしたのなら、以下のようになっていると思います。

Sub Macro1()

'

' Macro1 Macro

'

'

 Rows("6:6").RowHeight = 50.25

End Sub

　Macro1()を実行すると、6行目のセル高さを50.25にします。50.25を40に書き換えて実行したり、"6:6" の部分を "2:4" に変更して実行したりみて、動作を確認して下さい。

　このことから、n行目～m行目の高さをxにするには、以下のように書けば良いことが分かります。

 Rows(n & ":" & m).RowHeight = x

（注意）

　Rows(n の直後に半角空白を入れないと、エラーになる。

　文字列を連結するときに & を使うと、数値は自動的に文字列に変換されます。連結に + を使うときは、以下のようにCStrを使って数値を文字列に明示的に変換する必要があります。

 Rows(CStr(n) + ":" + CStr(m)).RowHeight = x

【例題】

　シート「名前記入」は名前記入用の用紙の作成途上です。図4.1のような体裁にしなさい。ふりがなの行の高さは20、名前の行は40です。



図4.1　名前書き込み用シート（上から2段を表示）

【解答1】

　「奇数行と偶数行で高さを変える」という考えをプログラムにすると、以下のようになります。i Mod 2はiを2で割ったときの余りを返します。余りが1のときは奇数行、0のときは偶数行です。

 For i = 1 To 16

 If i Mod 2 = 1 Then

 Rows(i & ":" & i).RowHeight = 20

 Else

 Rows(i & ":" & i).RowHeight = 40

 End If

 Next i

【解答2】

　行高を変数hに入れます。「行高20の次は行高40、その次は行高20・・・」という考えをプログラムで表すと、以下のようになります。

 h = 20

 For i = 1 To 16

 Rows(i & ":" & i).RowHeight = h

 If h = 20

 h = 40

 Else

 h = 20

 End If

 Next i

　h = 40とh = 20の部分は、それぞれh = h + 20, h = h - 20としても良いです。

　セルに罫線を引く方法も調べてみます。「セルB6の周囲に罫線を引く」という動作を「マクロの記録」で記録すると、以下のようになります。

Sub Macro2()

'

' Macro2 Macro

'

'

 Range("B6").Select

 Selection.Borders(xlDiagonalDown).LineStyle = xlNone

 Selection.Borders(xlDiagonalUp).LineStyle = xlNone

 With Selection.Borders(xlEdgeLeft)

 .LineStyle = xlContinuous

 .ColorIndex = xlAutomatic

 .TintAndShade = 0

 .Weight = xlThin

 End With

 With Selection.Borders(xlEdgeTop)

 .LineStyle = xlContinuous

 .ColorIndex = xlAutomatic

 .TintAndShade = 0

 .Weight = xlThin

 End With

 With Selection.Borders(xlEdgeBottom)

 .LineStyle = xlContinuous

 .ColorIndex = xlAutomatic

 .TintAndShade = 0

 .Weight = xlThin

 End With

 With Selection.Borders(xlEdgeRight)

 .LineStyle = xlContinuous

 .ColorIndex = xlAutomatic

 .TintAndShade = 0

 .Weight = xlThin

 End With

 Selection.Borders(xlInsideVertical).LineStyle = xlNone

 Selection.Borders(xlInsideHorizontal).LineStyle = xlNone

End Sub

　今度はかなり複雑です。想像力を働かせて解読しましょう。

 Range("B6").Select

 Selection.Borders(xlDiagonalDown).LineStyle = xlNone

の部分はB6を選択しなくてよいなら、以下のように書き換え可能です。この方が分かりやすいでしょう。

 Range("B6").Borders(xlDiagonalDown).LineStyle = xlNone

　次に以下の部分を解読しましょう。

 Range("B6").Select

 （略）

 With Selection.Borders(xlEdgeLeft)

 .LineStyle = xlContinuous

 （略）

 End With

　先ほどと同様にB6を選択しなくてよいなら以下のように書き換え可能です。

 Range("B6").Borders(xlEdgeLeft).LineStyle = xlContinuous

　上記の命令で、セルの左端の罫線を実線に設定することが推測されます。セルをB6からB8に変えてみてください。セルB8の左側に罫線が引かれたと思います。

　このように「マクロの記録」で得たプログラムのうち、必要な部分を抽出して使いましょう。

　これ以外の部分について説明します。以下は罫線の色を指定する部分です。

 Range("B6").Borders(xlEdgeLeft).ColorIndex = xlAutomatic

　色番号（整数）を指定しています。何番がどういう色なのかを調べるのは面倒なので、色の指定は以下の方法がよいでしょう。

 Range("B6").Borders(xlEdgeLeft).Color = RGB(0, 0, 0)

　LineStyleについてはxlContinuous(実線) 以外に xlDot(点線), xlDashDot(一点鎖線), xlDouble(二重線)などが指定可能です。

　Weightは罫線の太さを指定する項目でxlThin, xlMedium, xlThickなどが指定可能です。

　それ以外の部分は滅多に使わないと思いますが、何を指定する項目なのかを説明しておきます。

* Borders(xlDiagonalDown).LineStyle …… 左上から右下へ引く罫線の属性
* Borders(xlDiagonalUp).LineStyle …… 右上から左下へ引く罫線の属性
* Borders(xlInsideVertical).LineStyle …… 複数をセルを選択したとき、内部に引く縦罫線の属性
* Borders(xlInsideHorizontal).LineStyle …… 複数のセルを選択したとき、内部に引く横罫線の属性
* Borders(xlDegeLeft).TintAndShade …… -1～+1の範囲で明度を指定。-1のとき黒, +1のとき白, 0のとき指定した色です。

## セルのクリア・色つけ

　Cellsは全セルを表します。イミディエイトウィンドウで

 Cells.Select

と打ってみて下さい。シート上の全セルを選択します。

　セルのクリアは3通りの方法があります。

Cells.ClearContents セルの内容を消去しますが、高さや幅、罫線や塗りつぶし色はそのままです。

Cells.Clear セルの内容、罫線、塗りつぶしなどを削除しますが、列幅、行高はそのままです。

Cells.Delete 罫線、塗りつぶし、列幅、行高さなど全てをデフォルト値に戻します（シートを新規作成したのと同等）。

　セルA2の背景色をつける方法は、以下の通りです。

 Range("A2").Interior.Color = RGB(255, 255, 200)

 あるいは

 Cells(2, 1).Interior.Color = RGB(255, 255, 200)

　RangeあるいはCellsのプロパティであるInterior.Colorに色を表す整数を代入します。RGB(r, g, b) は色を表す整数を計算する関数です。r, g, bには0～255の整数を代入します。関数RGB(r, g, b) の戻り値は以下の値です。

 r + ( g \* 256 ) + ( b \* 256 \* 256 )

　「色なし」に設定するときは以下のように書きます（以下ではRangeを使う方法を記します）。

 Range("A2"). Interior.Pattern = xlNone

　xlNoneはExcel VBAがデフォルトで用意している定数で、Debug.Print xlNoneで値を見ると、-4142という数値が入っています。

　Interior.Patternではなく、以下のようにInterior.Colorを使っても「色なし」に設定できます。

 Range("A2"). Interior.Color = xlNone

　Colorは通常はRGB値を表す整数を与えますが、試したところ負の値なら何を与えても「色なし」に設定されるようです。

【例題】

　シート「市松模様」のA1～G8の範囲において、セルの色を市松模様につけなさい。色はRGB(255, 255, 220)とRGB(220, 255, 255)を使い、左上はRGB(255, 255, 220)を使いなさい。

　愚直にプログラムを書くと以下のようになります。ここでcolorという変数名を使っています。本書ではモジュールの先頭にOption Explicitと書き、全ての変数宣言を強要することを推奨しています。本例のようにcolorを変数宣言する場合は問題は生じませんが、Option Explicitを用いず、変数宣言なしでcolorという変数に代入しようとすると、Excelがデフォルトで保持している定数colorを変更することになり、定数は変更できないのでエラーが発生します。

 Dim i, j, color, left\_color

 left\_color = 0 ' 一番左の列の色 : 0 か 1 をとる

 For i = 1 To 8

 color = left\_color ' その行の左端のセルの色

 left\_color = left\_color + 1

 If left\_color = 2 Then

 left\_color = 0

 End If

 For j = 1 To 7

 If color = 0 Then

 Cells(i, j).Interior.color = RGB(255, 255, 220)

 Else

 Cells(i, j).Interior.color = RGB(220, 255, 255)

 End If

 color = color + 1

 If color = 2 Then

 color = 0

 End If

 Next j

 Next i

　プログラム中、left\_colorやcolは0→1→0→1→0....と変化します。

　この数列を得る別の方法として、1ずつ増加する数列を2で割った余りとして得る方法があります。Modを使ってスマートに書き直すと、以下のようになります。

 For i = 1 To 8

 left\_color = (i - 1) Mod 2

 For j = 1 To 7

 col = (left\_color + (j - 1)) Mod 2

 If col = 0 Then

 Cells(i, j).Interior.Color = RGB(255, 255, 220)

 Else

 Cells(i, j).Interior.Color = RGB(220, 255, 255)

 End If

 Next j

 Next i

　さらにスマートに書くと以下のようになります。

 For i = 1 To 8

 for j = 1 to 7

 If (i + j) Mod 2 = 0 Then

 Cells(i, j).Interior.Color = RGB(255, 255, 220)

 Else

 Cells(i, j).Interior.Color = RGB(220, 255, 220)

 End If

 Next i

## データが入っている最後の行を取り出す

　これまでのプログラムではFor i = 2 To 11 のように書きました。11はデータの個数によって変わります。データの個数が変わると、11の部分を書き換える必要があります。プログラム中に11と書いてある部分が何か所もある場合、書き換え忘れミスが起こるかもしれません。この11のような数をマジックナンバーと言い、プログラムの中に書いてはいけないとされています。

　これを回避するには、変数を1つ導入します。例えば、プログラムの最初で

 last\_row = 11

とし、プログラム中で11という数が必要なときはlast\_rowを使います。こうすれば、11という数値を変更する場合、書き換える箇所は1か所で済み、誤りが混入しにくくなります。

　さらにlast\_rowを自動的に取得する方法があります。A列でデータが入っている最後の行を取り出すには、以下のように書きます。

 last\_row = Range("A" & Rows.Count).End(xlUp).Row

　Rows.Countはシートの行数を表します。ゆえにRange("A" & Rows.Count) はRange("A1048576")と同等です。B列でデータが入っている最終行を取り出すには、"A" の部分を "B" に置き換えて下さい。

　A列で最初に現れる空セルの1つ前の行を取得するには次のように書きます。

 last\_row = Range("A1").End(xlDown).Row

　ただしこの方法はA1のみにデータ入っている場合には使えません。last\_row = 1048576となります。Range( ) で指定するセルとその下のセルの最低2個のセルにデータが入っている必要があります。

　本節で学習した

 Range("A" & Rows.Count).End(xlUp).Row

 Range("A1").End(xlDown).Row

は頻繁に使う命令ですが、暗記してタイプできるかというと、実は筆者も覚えていません。コピペします。例えばデスクトップにExcel VBA虎の巻ファイル（txtファイル）を置いておき、よく使うパターンはそこからコピペすれば良いでしょう。

## Forにおける注意

　以下のようなことをしてはいけません。

 For i = 1 To 10

 i = 3

 Next i

　For文として使うループ変数に値を代入しています。このFor 文においてiの値は1→2→3→4→……→8→9→10となり、10回目のNext iを実行してループを抜けた後のiの値は11になります。上のように、ループの中でi = 3として、ループ変数iの値を強制的に変更してしまうと、永遠に終了条件を満たさなくなってしまい、無限ループになってしまいます。場合によっては、Excelがハングアップします。

　原則として、Forループの中でループ変数を変更してはいけません。

　Forが終わった後、変数iの値はどうなっているでしょうか？　次のプログラムを実行してみて下さい。

 For i = 1 To 10

 Next i

 Debug.Print i

　実行してみると、iの値は11になっています。この例のように、iが1ずつ増えていく場合、Forループ終了時点でのループ変数iの値は「終了値 + 1」です。ループ終了後、ループ変数の値を使うことはないと思いますが、知っておいて損はないと思います。

## シートの切り替え・異なるシートのデータの操作

　本節で学習するプログラムは「標準モジュール」の中に書いて下さい。シートの中やThisWorkbookの中に書いても正しく実行されません。

　これまでは操作したいシートを選んだ状態で、プログラムを実行しました。例えばRange("A1") = "abc"とすると、現在選択しているシートのセルA1に文字列 "abc" が入ります。それ以外のシートのセルを操作するには以下の方法があります。

1. Sheets("シート名").Range("A1") のようにシート名も書く
2. シートを切り替える

　1.の方法について説明します。以下のようにシートを指定することで、異なるシートのセルを操作することができます。

　読み出し例：

 a = Sheets("シート名").Range("A1")

書き込み例：

 Sheets("シート名").Range("A1") = 10

　シートを指定する方法として、シートの通し番号を指定する方法があります。シートには通し番号が1から順番に付いています。一番左端のシートが1番です。通し番号3のシートのセルA1に10を入れるには以下のように書きます。

 Sheets(3).Range("A1") = 10

　この方法を使うと、シートの番号を変数を使って指定することができます。シートの個数を変数n\_sheetに入れるには以下のように書きます。

 n\_sheet = Sheets.Count

　次に2. の方法について説明します。Excelではどれか1つのシートを選択した状態になっています。選択されたシートをアクティブなシートと呼びます。アクティブなシートの名前を変数sheet\_nameに入れるには以下のように書きます。

 sheet\_name = ActiveSheet.name

　アクティブなシートを切り替えるには以下のように書きます。

 Sheets("シート名").Select

 あるいは

 Sheets(3).Select ' 3 はシート番号

【課題3】

　「1組」「2組」「3組」のシートについて、セル幅を10、高さを20に設定しなさい。シート番号1の全てのセルの横幅と高さを設定するには以下のように書きます。

 Sheets(1).Cells.ColumnWidth = 8

 Sheets(1).Cells.RowHeight = 20

　シート名を使う場合は以下のように書きます。

 Sheets(i & "組").Cells.ColumnWidth = 8

 Sheets(i & "組").Cells.RowHeight = 20

　上記の命令中Cellsという部分があります。Cellsはそのシート中の全てのセルを表します。

【課題4】

　1組, 2組, 3組のシートに名簿があります。各クラスの人数は10人です。補習対象者に1が記入されています。補習対象者をピックアップし、シート「補習対象者」に並べるプログラムを作成しなさい。

【課題5】

　通常パソコン室では先生が生徒を指導するためのシステムが導入されており、生徒の出欠をファイルに記録することができます。ファイルsyusseki.csvが作成されると仮定します。syusseki.csvはCSV (Comma Separated Value) 形式と呼ばれるファイルです。データをコンマで区切った形式で、表を表すときに使われます。Excelで作った表をxlsx形式で保存すると、Excelからしか読むことはできませんが、CSV形式で保存すると、CSVファイルはテキストファイルの一種なので、色々なアプリで読み書きすることができます。

　syusseki.csvをシート「csv読み込み用」に読み込みなさい。PC番号、学生の学籍番号、名前、ログイン時刻が記録されています。

　Office 2016の場合は「データ」→「外部データの取り込み：テキストファイル」として、インポートし「カンマやタブなどの区切り文字によってフィールドごとに区切られたデータ」を選択し、「区切り文字」として「カンマ」にチェックを入れて読み込んで下さい。

　Microsoft 365の場合は、「ファイル」→「オプション」→「データ」で「レガシデータインポートウィザードの表示」で「テキストから（レガシ）」にチェックを入れておき、「データ」→「データの取得と変換：従来のウィザード」で以下は同じです。

　csvファイルをダブルクリックするとExcelが起動してcsvファイルを開きます。この状態でコピペする方法もあります。

　このファイルを利用して、生徒の出欠欄を埋めなさい。出席したとき "○", 欠席のとき "×" を入れます。

## メッセージボックス

　処理中に何か表示したいときは、Debug.Printでイミディエイトウィンドウに表示するか、どこかのセルに表示するという方法を今まで学びました。

　それ以外の方法として、メッセージボックスを表示する方法があります。例を示します。

 MsgBox "これがメッセージです"

　これはOKだけが出るメッセージボックスを表示します。メッセージボックスにはYes/No, OK/Cancelなどいくつかのパターンがあります。

 MsgBox "これがメッセージです", vbYesNo

とすると、Yes/Noの2つのボタンを持つメッセージボックスを表示します。押したボタンの値を取得するときは、次のように括弧付きで書きます。

 a = MsgBox("これがメッセージです", vbYesNo)

　aにはYesのとき6, Noのとき7が入ります。この例のvbYesNoの部分にはvbOKonly, vbOKCancel, vbYesNoCancelなどが入ります。適切なものを選んで下さい。

　メッセージボックスに表示する文字列をその時々によって変えたいときは、以下のように書きます。

 i = 2

 moji = "i = " & i & " です。"

 a = MsgBox(moji)

## Do Whileループの一般形

　本書では2章でForループを勉強し、3章で無限ループを学習しました。無限ループではDo While Trueという表現を学習しましたが、これはDo While ループを組むときの一つの特殊な形式です。Do While ループは本来は以下の形式をしています。

 Do While 条件式

 条件が成立したときの処理

 条件が成立したときの処理

 ......

 Loop

　Do While ～ Loopまでがループであり、先頭に条件式を置きます。条件式が成立する場合は、Loopまでの処理を行い、成立しない場合はLoopの次の行から実行します。

　例を示します。

 i = 1

 Do While i <= 10

 Range("A" & i) = i

 i = i + 1

 Loop

　ループ先頭に i <= 10 という条件式があります。成立しているなら、Loopまでの区間を実行します。ループを1回実行するたびにiの値は1増えますから、A1～A10のセルに1～10の数値を書き込みます。

　3章で学んだ無限ループは、条件式の部分にTrueと書いています。これは条件が常に成立することを意味するので、結果として無限ループとなります。

　筆者は1979年頃にBasicからプログラミングに入門しました。当時のBasicにはループとしてFor文しかありませんでした。無限ループやDo WhileループはIfとGotoで実現していました[[1]](#footnote-1)。

　Do Whileループは「条件が成立している間はループを実行」です。「無限ループとIf, Exit Doの組み合わせ」は「条件が成立したらループから脱出」です。条件の与え方が逆です。筆者はDo Whileループがない時代にプログラミングに入門したため、未だにDo Whileループに違和感があります。「無限ループとIf, Exit Doによる脱出」の方が分かりやすいと判断し、本書では3章でそのパターンを最初に提示しました。これは、一般のプログラミングの教科書とは異なった教え方です[[2]](#footnote-2)。

　Do Whileの直後に条件を与えるループの場合、条件を与える場所はループの先頭1箇所です。それに対して筆者が推奨する「無限ループ＋Exit Do」は、(1) 脱出条件を書く場所がループ中のどこにあるのか分からない　(2) ループから脱出する箇所を複数設定できてしまう、という問題があります。

　プログラムを分かりやすくするため、関数の出口 (C言語の場合はreturn) は1箇所にするべきという考え方があります。ループからの脱出箇所も1箇所だけにしないといけないのかも知れません。

　Do Whileループの説明として、風呂に水を入れる例えで説明する方法があります。風呂に水が満杯になるまで水を入れ続けるのは、「満杯になるという条件が成立するまで水を入れ続ける」ことです。もしかしたら、最初にこの例えを出して、Do Whileループを学習する方が良かったかも知れません。

1. Goto命令は諸悪の根源とされ、現在はほとんど使いません。Excel VBAにもGoto命令はありますが、本書では扱いません。

　嫌われ者のGoto命令ですが、コンピューターはIfとGotoで動いています。CPUが理解できるのは機械語だけです。機械語にはFor, Do Whileなどの命令はありません。あるのはIfとGotoだけです。 [↑](#footnote-ref-1)
2. 筆者と同じ考えの人は他にもます。<https://www.wareko.jp/blog/best-loop-structure-for-programming>の執筆者も同じことを書いています。2021.10.8 アクセス。「ワレコの講座 for/while/do-while/do-untilのどれが良い？ ループ処理」で検索すると見つかる。 [↑](#footnote-ref-2)