【VBA＆VBS】備忘録

目次

はじめに

■整理

- VB（Visual Basic）
  - 開発環境にてコンパイルしてexeファイルを作成してから実行する。
  - 実行には別途dllが必要となる場合あり。
- VBScript / VBS（Visual Basic Script）
  - Visual Basicの簡易版。Office製品やコンパイラ無しで実行が可能。
  - WSH（Windows Scripting Host）という環境で動作する。wscript.exe や cscript.exe を使用して実行する。（あるいはIE上でスクリプトとして動作させることも可能）
  - プログラムが記載されたソースファイル（テキストファイル）を直接実行可能。
- VBA（Visual Basic for Applications）
  - Access や Excel や Outlook などのOffice製品などの中で動作させる言語。
  - マクロ言語に分類される。

■諸注意

▶ ※ VBAやVBSでJSONファイルを扱うときは、Python経由でやるのがいいかもね。

VBScript（VBS）

■実行方法

▶ ※ 日本語を使う場合、**Shift_JIS 形式で編集・保存**しておかないと実行できない。

【VBA＆VBS】備忘録

目次

- はじめに
- VBScript (VBS)
- VBA（アプリケーション問わず）
- Excel VBA
- Word VBA
- PowerPoint VBA

はじめに

■整理

- VB（Visual Basic）
  - 開発環境にてコンパイルしてexeファイルを作成してから実行する。
  - 実行には別途dllが必要となる場合あり。
- VBScript / VBS（Visual Basic Script）
  - Visual Basicの簡易版。Office製品やコンパイラ無しで実行が可能。
  - WSH（Windows Scripting Host）という環境で動作する。wscript.exe や cscript.exe を使用して実行する。（あるいはIE上でスクリプトとして動作させることも可能）
  - プログラムが記載されたソースファイル（テキストファイル）を直接実行可能。
- VBA（Visual Basic for Applications）
  - Access や Excel や Outlook などのOffice製品などの中で動作させる言語。
  - マクロ言語に分類される。

■諸注意

▶ ※ VBAやVBSでJSONファイルを扱うときは、Python経由でやるのがいいかもね。

VBScript（VBS）

■実行方法

▶ ※ 日本語を使う場合、**Shift_JIS 形式で編集・保存**しておかないと実行できない。

- ▶ ☆ 方法０：ダブルクリック
- ▶ ☆ 方法１：直接コマンドラインから実行
- ▶ ☆ 方法２：BATファイルで実行

■基礎

- ▶ 強制終了
- ▶ ※ `ステートメントがありません` と言われたら、 `仮引数:=` を書かないで、 `実引数` だけ書くようにしよう。
- ▶ 変数の命名規則は lowerCamelCase で、関数の命名規則は UpperCamelCase 。

■標準入出力

- ▶ コマンドラインでの引数
- ▶ 標準出力

■Word や Excel を操作

- ▶ ☆ WordファイルやExcelファイルを開いて編集
- ▶ ☆ WordファイルやExcelファイル中のマクロを実行
- ▶ ※ WordファイルやExcelファイル中のマクロを実行時、 `型が一致しません` のエラーで悩んだら　→　VBAのプロシージャの引数の型を Variant にしてあげる。

■時間

- ▶ ある時間だけ待つ

■ファイル操作

- ▶ 実行中.vbsの**ディ**のパス

■よくある間違い

- ▶ ※ For～Next 文の Next のあとには何も要らない。
- ▶ ※ 変数宣言は必須ではない。したくても `Dim 変数名` だけ（ `As データ型` はつけない）。
- ▶ ※ 変数宣言は必須ではないが、一度記述された変数の型はプログラムの中で変わることがない。

■ VBA の機能で VBScript に含まれていない機能

- Select Case ステートメント　キーワード Is または比較演算子が含まれる式
  キーワード To を使う値の範囲の指定が含まれる式
- エラー処理　Erl 関数
  Error ステートメント

- ▶ ☆ 方法０：ダブルクリック
- ▶ ☆ 方法１：直接コマンドラインから実行
- ▶ ☆ 方法２：BATファイルで実行

■基礎

- ▶ 強制終了　WScript.Quit
- ▶ ※ `ステートメントがありません` と言われたら、 `仮引数:=` を書かないで、 `実引数` だけ書くようにしよう。
- ▶ 変数の命名規則は lowerCamelCase で、関数の命名規則は UpperCamelCase 。

■標準入出力

- ▶ コマンドラインでの引数　Wscript.Arguments(*n*)　※>wscript ～.vbs **第0引数**
- ▶ 標準出力　WScript.Echo "こんにちは"

■Word や Excel を操作

- ▶ ☆ WordファイルやExcelファイルを開いて編集
- ▶ ☆ WordファイルやExcelファイル中のマクロを実行
- ▶ ※ WordファイルやExcelファイル中のマクロを実行時、 `型が一致しません` のエラーで悩んだら　→　VBAのプロシージャの引数の型を Variant にしてあげる。

■時間

- ▶ ある時間だけ待つ　WScript.Sleep ミリ秒

■ファイル操作

- ▶ 実行中.vbsの**ディ**のパス

```
set fso = createObject("Scripting.FileSystemObject")
fso.GetFolder(".")
```


■よくある間違い

- ▶ ※ For～Next 文の Next のあとには何も要らない。
- ▶ ※ 変数宣言は必須ではない。したくても `Dim 変数名` だけ（ `As データ型` はつけない）。
- ▶ ※ 変数宣言は必須ではないが、一度記述された変数の型はプログラムの中で変わることがない。

■ VBA の機能で VBScript に含まれていない機能

- Select Case ステートメント　キーワード Is または比較演算子が含まれる式
  キーワード To を使う値の範囲の指定が含まれる式
- エラー処理　Erl 関数
  Error ステートメント

	Resume ステートメント、Resume Next ステートメント
• 演算子	Like 演算子
• オブジェクト	Clipboard オブジェクト Collection オブジェクト
• オブジェクトの使用	演算子を使用したコレクションへの参照 キーワード TypeOf
• コレクション	Add メソッド、Count プロパティ、 Item メソッド、Remove メソッド ! 演算子を使用したコレクションへの参照
• 財務処理	すべての財務処理関数
• 条件分岐	#Const ディレクティブ #If...Then...#Else ディレクティブ
• 制御構造	DoEvents 関数 GoSub...Return ステートメント、GoTo ステートメント On Error GoTo ステートメント On...GoSub ステートメント、On...GoTo ステートメント 行番号、行ラベル
• 宣言	Declare ステートメント (DLL 参照のための宣言) キーワード Optional キーワード ParamArray Static ステートメント
• ﾀﾞｲﾅﾐｯｸﾃﾞｰﾀｴｸｽﾁｪﾝｼﾞ(DDE)	LinkExecute メソッド、LinkPoke メソッド、 LinkRequest メソッド、LinkSend メソッド
• データ型	バリアント型 (Variant) を除くすべての組み込みデータ型 Type...End Type ステートメント
• デバッグ	Debug.Print End ステートメント、Stop ステートメント
• 配列	Option Base ステートメント 0 以外のインデックスの最小値を指定した配列の宣言
• 日付と時刻	Date ステートメント、Time ステートメント
• その他	Def*type*ステートメント Option Base ステートメント Option Compare ステートメント Option Private Module ステートメント
• ファイル入出力	すべてのファイルの入出力機能　→**FSOは使える！泣**

	Resume ステートメント、Resume Next ステートメント
• 演算子	Like 演算子
• オブジェクト	Clipboard オブジェクト Collection オブジェクト
• オブジェクトの使用	演算子を使用したコレクションへの参照 キーワード TypeOf
• コレクション	Add メソッド、Count プロパティ、 Item メソッド、Remove メソッド ! 演算子を使用したコレクションへの参照
• 財務処理	すべての財務処理関数
• 条件分岐	#Const ディレクティブ #If...Then...#Else ディレクティブ
• 制御構造	DoEvents 関数 GoSub...Return ステートメント、GoTo ステートメント On Error GoTo ステートメント On...GoSub ステートメント、On...GoTo ステートメント 行番号、行ラベル
• 宣言	Declare ステートメント (DLL 参照のための宣言) キーワード Optional キーワード ParamArray Static ステートメント
• ﾀﾞｲﾅﾐｯｸﾃﾞｰﾀｴｸｽﾁｪﾝｼﾞ(DDE)	LinkExecute メソッド、LinkPoke メソッド、 LinkRequest メソッド、LinkSend メソッド
• データ型	バリアント型 (Variant) を除くすべての組み込みデータ型 Type...End Type ステートメント
• デバッグ	Debug.Print End ステートメント、Stop ステートメント
• 配列	Option Base ステートメント 0 以外のインデックスの最小値を指定した配列の宣言
• 日付と時刻	Date ステートメント、Time ステートメント
• その他	Def*type*ステートメント Option Base ステートメント Option Compare ステートメント Option Private Module ステートメント
• ファイル入出力	すべてのファイルの入出力機能　→**FSOは使える！泣**

- 変換　　CVar 関数、CVDate 関数
  Str 関数、Val 関数
- 文字列　　固定長文字列
  LSet ステートメント、RSet ステートメント
  Mid ステートメント
  StrConv 関数
- あと、アプリの組み込み定数は当然使えない。（ `vb~` は使えるっぽい）

- 変換　　CVar 関数、CVDate 関数
  Str 関数、Val 関数
- 文字列　　固定長文字列
  LSet ステートメント、RSet ステートメント
  Mid ステートメント
  StrConv 関数
- あと、アプリの組み込み定数は当然使えない。（ `vb~` は使えるっぽい）

VBA（アプリケーション問わず）

■注意

- ▶ ※「Label:」に飛んだら、そこ以降のコードしか読み込まない(実行しない)。
- ▶ ※ Objectは最後つねに後始末しておこう
- ▶ ※ `コマンドが正常終了しなかった`　→　直前に `DoEvents`　（多分これでいける）
- ▶ ☆ VBE画面を見やすくする
- ▶ ※ **アドインのプロジェクトは編集したら必ず上書き保存ボタンを押すこと**。ただし、保存されるタイミングはアプリケーションを終了した瞬間。

■よくやる間違い

- Setを抜かす　　→　　Set オブ = オブ　　**Set オブ = Nothing**

■基礎

- ▶ 型を取得
- ▶ 大域脱出
- ▶ 現在の選択を取り消す
- ▶ 強制終了

■変数の適用範囲

- ▶ プロシージャ内で Dim, Const, Static
- ▶ 宣言セクションでの Dim, Const
- ▶ 宣言セクションでの Public

■標準入出力

- ▶ 入力
- ▶ ﾀﾞｲｱﾛｸﾞﾎﾞｯｸｽで出力

VBA（アプリケーション問わず）

■注意

- ▶ ※「Label:」に飛んだら、そこ以降のコードしか読み込まない(実行しない)。
- ▶ ※ Objectは最後つねに後始末しておこう
- ▶ ※ `コマンドが正常終了しなかった`　→　直前に `DoEvents`　（多分これでいける）
- ▶ ☆ VBE画面を見やすくする
- ▶ ※ **アドインのプロジェクトは編集したら必ず上書き保存ボタンを押すこと**。ただし、保存されるタイミングはアプリケーションを終了した瞬間。

■よくやる間違い

- Setを抜かす　　→　　Set オブ = オブ　　**Set オブ = Nothing**

■基礎

- ▶ 型を取得　　TypeName(式)
- ▶ 大域脱出　　ラベルとGoToを使うのが吉
- ▶ 現在の選択を取り消す　　‥.Selection.Unselect
- ▶ 強制終了　　End

■変数の適用範囲

- ▶ プロシージャ内で Dim, Const, Static　　そのプロシージャのみ
- ▶ 宣言セクションでの Dim, Const　　モジュール内の全プロシージャ　※自動で静的に
- ▶ 宣言セクションでの Public　　全モジュール　※自動で静的に

■標準入出力

- ▶ 入力　　InputBox(*prompt*[, *title*][, *defalt*])
- ▶ ﾀﾞｲｱﾛｸﾞﾎﾞｯｸｽで出力　　MsgBox *prompt*[, *buttons*] [, *title*] [, *helpfile*, *context*]

- ▶ ｲﾐﾃﾞｨｴｲﾄｳｨﾝﾄﾞｳに出力
- ▶ 出力内容中で改行する
- ▶ 音を鳴らす

■条件分岐

- ▶ 条件分岐
- ▶ 比較演算子
- ▶ 論理演算子
- ▶ switch 文
- ▶ 何もしない
- ▶ ※ `True` `False` はそれぞれ `-1` `0` と等価。

■繰り返し処理

- ▶ 配列に対してforeach
- ▶ シートに対してforeach
- ▶ 中断し、次へ・脱出

■プロシージャ

定義など

- ▶ プロシを定義
- ▶ 配列を返す
- ▶ ※ Functionプロシージャについて、実は返り値はなくてもOK。
- ▶ 他モでの呼出し禁ず
- ▶ 早期リターン

引数について

- ▶ デフォルト値を設定
- ▶ 参照渡し・値渡し

呼び出し

- ▶ 呼び出し
- ▶ ※ ユーザーフォームのパブリックプロシージャを呼び出すときは、 `UserForm .` をつける必要がある。

- ▶ ｲﾐﾃﾞｨｴｲﾄｳｨﾝﾄﾞｳに出力　Debug.Print 式　や　Debug.Print 式1 ; 式2
- ▶ 出力内容中で改行する　vbCrLf
- ▶ 音を鳴らす　Beep

■条件分岐

- ▶ 条件分岐　If ～ Then　ElseIf ～ Then　Else　End If
- ▶ 比較演算子　=　<>　>　<　>=　<=
- ▶ 論理演算子　And　Or　Not
- ▶ switch 文　Select Case 式　Case 値　処理　End Select
- ▶ 何もしない　'何もしない　※ つまり何も書かなくていい。
- ▶ ※ `True` `False` はそれぞれ `-1` `0` と等価。

■繰り返し処理

- ▶ 配列に対してforeach　For n = LBound(arr) To UBound(arr)　arr(n)
- ▶ シートに対してforeach　For each sht in wb.worksheets
- ▶ 中断し、次へ・脱出　ラベルを使う　・　Exit For　や　Exit Do

■プロシージャ

定義など

- ▶ プロシを定義　Sub HogeHoge() ‥ End Sub
  Function HogeHoge() As 型名※ ‥ End Function　※オブも可能
- ▶ 配列を返す　Function Hoge() As 型名**()**　Hoge = 配列
- ▶ ※ Functionプロシージャについて、実は返り値はなくてもOK。
- ▶ 他モでの呼出し禁ず　Private Sub ‥　Private Function ‥
- ▶ 早期リターン　Exit Sub　Exit Function　を使う

引数について

- ▶ デフォルト値を設定　Optional 引数 As 型名 = デフォルト値
- ▶ 参照渡し・値渡し　ByRef 引数 As ‥　・　ByVal 引数 As ‥　※ **書かなければ前者に**

呼び出し

- ▶ 呼び出し　Hoge *arg1*, ... '#　変数 = Hoge(*arg1*, ...)
- ▶ ※ ユーザーフォームのパブリックプロシージャを呼び出すときは、 `UserForm .` をつける必要がある。

- ▶ プライベートプロシを呼び出し
- ▶ 他ブックのプロシを呼び出し
- ▶ ※ ユーザーフォームのプライベートプロシージャを呼び出すことはできない。また、ユーザーフォームのプロシージャを他ブックから呼び出すことはできない。

■例外処理

- ▶ エラーを防ぐべく確認
- ▶ わざとエラーを起こす

■数値

- ▶ 算術演算子
- ▶ 0≤乱数<1 の生成
- ▶ *min*≤乱数<*max* ∈ℕ
- ▶ 小数点以下切り捨て

■文字列

- ▶ ○○が何文字目に登場するか
- ▶ あるパターンに合致するか
- ▶ 文字列の結合
- ▶ TEXTJOIN
- ▶ 特殊な文字を表現
- ▶ ☆ 環境依存文字を表現
- ▶ 改行
- ▶ 文字列の前後の空白を除去
- ▶ 大・小文字にする
- ▶ 置換
- ▶ *str2*登場までの文字列を抽出
- ▶ 〃(後ろから)
- ▶ 文字列が数値みたいか

- ▶ プライベートプロシを呼び出し　Application.Run "'Hoge'", *arg1*, ... '#
変数 = Application.Run("'Hoge'", *arg1*, ...)
- ▶ 他ブックのプロシを呼び出し　Application.Run "'Book1.xlsm'!'Hoge'", *arg1*, ... '#
変数 = Application.Run("'Book1.xlsm'!'Hoge'", *arg1*, ...)
- ▶ ※ ユーザーフォームのプライベートプロシージャを呼び出すことはできない。また、ユーザーフォームのプロシージャを他ブックから呼び出すことはできない。

■例外処理

- ▶ エラーを防ぐべく確認　Debug.Assert 真偽値　※AssersionErrorメッセージは出せない
- ▶ わざとエラーを起こす　Err.Raise Number:=番号, Description:="エラー発生！"
※ 番号には 513～65535 の数字を設定するのが無難。

■数値

- ▶ 算術演算子　+　-　*　/　¥　Mod　^
- ▶ 0≤乱数<1 の生成　Randomize '　Rnd
- ▶ *min*≤乱数<*max* ∈ℕ　Randomize '　Int((*max* - *min* + 1) * Rnd + *min*)
- ▶ 小数点以下切り捨て　Int(数値)　※なお `-2.4` → `-3`

■文字列

- ▶ ○○が何文字目に登場するか　InStr(*str*, ○○)　や　InStr(*str*, ○○, *start*)　や　InStrRev(〃)　や　InStrRev(〃)　※ 後ろから探す
※: `1` 以上の整数　登場しなければ `0`
- ▶ あるパターンに合致するか　*str* Like *pattern*※　※ `?` や `*` が使える
- ▶ 文字列の結合　&
- ▶ TEXTJOIN　Join(*strArr*, *delemiter*)　※ `strArr` は文字列以外NG
- ▶ 特殊な文字を表現　"" (ダブルクオートに)
- ▶ ☆ 環境依存文字を表現
- ▶ 改行　vbCrLf　か　vbLf　か　vbCr　※適切なのは状況で変わる
- ▶ 文字列の前後の空白を除去　Trim(*str*)　※ 全角スペースにも対応。
- ▶ 大・小文字にする　UCase(*str*)　・　LCase(*str*)
- ▶ 置換　Replace(*str*, *old*, *new*)
- ▶ *str2*登場までの文字列を抽出　Left(*str*, InStr(*str*, *str2*) - 1)
- ▶ 〃(後ろから)　Mid(*str*, InStrRev(*str*, *str2*) + Len(*str2*))
- ▶ 文字列が数値みたいか　IsNumeric(文字列)

- ▶ 文字列を整数に変換
- ▶ 文字列を実数に変換
- ▶ ※ 引用符 `"` は `Chr(35)` や `""` で表せる。
- ▶ ☆ キャメルケースに空白を加える

■配列

- ▶ 配列を定義
- ▶ 配列が空かどうか
- ▶ 配列を一気に初期化
- ▶ 配列変数を使い回すなら
- ▶ 配列に要素を追加

■色

- ▶ RGBである色に設定

■フォント

- ▶ フォント種
- ▶ フォントサイズ
- ▶ 色

■時間

- ▶ ある時間だけ待つ

■システム

- ▶ コンピュータ名

■ファイル操作

APIを使わない

- ▶ ☆ 1行目だけ読み込む
- ▶ ☆ 1行ずつ読み込む
- ▶ ※ 行単位で読み込む場合、改行はキャリッジリターンCR（またはCR+LF）でされている必要がある。
- ▶ ファイル検索
- ▶ ファイルを上書き
- ▶ ファイルに追記
- ▶ ファイルを閉じる

- ▶ 文字列を整数に変換　Int(文字列※)　※小数点含んでいてもOK！
- ▶ 文字列を実数に変換　Val(文字列)
- ▶ ※ 引用符 `"` は `Chr(35)` や `""` で表せる。
- ▶ ☆ キャメルケースに空白を加える

■配列

- ▶ 配列を定義　Dim 配列名() As 型名
- ▶ 配列が空かどうか　(Not a) = -1
- ▶ 配列を一気に初期化　a = Split(文字列, 区切り文字)
- ▶ 配列変数を使い回すなら　Erase a　を忘れないように！
- ▶ 配列に要素を追加　aSize = UBound(a) + 1　ReDim Preserve a(aSize)　a(aSize) = 値

■色

- ▶ RGBである色に設定　･･.Color = RGB(*R*, *G*, *B*)　など

■フォント

- ▶ フォント種　･･.Font.Name = "フォント名"
- ▶ フォントサイズ　･･.Font.Size = *size*
- ▶ 色　･･.Font.Color = RGB(*R*, *G*, *B*)　など

■時間

- ▶ ある時間だけ待つ　Application.Wait [Now()] + ﾐﾘ秒後 / 86400000

■システム

- ▶ コンピュータ名　Environ("COMPUTERNAME")

■ファイル操作

APIを使わない

- ▶ ☆ 1行目だけ読み込む
- ▶ ☆ 1行ずつ読み込む
- ▶ ※ 行単位で読み込む場合、改行はキャリッジリターンCR（またはCR+LF）でされている必要がある。
- ▶ ファイル検索　Dir(絶ﾊﾟ※) をつかう　※ﾜｲﾙﾄﾞｶｰﾄﾞOK。ディなら末尾 `\` に
- ▶ ファイルを上書き　Open “絶ﾊﾟ” For Output As #番号　Print #番号, “文字列”
- ▶ ファイルに追記　Open “絶ﾊﾟ” For Append As #番号　Print #番号, “文字列”
- ▶ ファイルを閉じる　Close #番号

- ▶ 空いてるファイル番号

- ▶ ホームディのパス

FileSystemObject（FSO）による

- ▶ ☆ FSOを作成（後始末も）
- ▶ ファの存在確認
- ▶ ファ作成
- ▶ ステム
- ▶ 拡張子
- ▶ ベースネーム
- ▶ テキストファを開く
- ▶ テキストファを閉じる
- ▶ ☆ １行ずつ読み込む
- ▶ 一気に読み込む
- ▶ 文字列を書き込み
- ▶ １行書き込み
- ▶ 改行を書き込み
- ▶ ファを削除
- ▶ ※ FSOでワイルドカードを使う場合にかぎり、`?` は「任意の1文字または0文字」を表すっぽい。
- ▶ 親ディ

■プログラムファイルまたはcmdコマンドを実行

- ▶ ☆ 非同期で実行
- ▶ ☆ 同期で実行

Pythonとの連携

- ▶ ☆ Pythonを実行（コマンドプロンプトから）
- ▶ ☆ 引数を交換

■メールを送信

- ▶ ☆ Outlookでメールを送信

- ▶ 空いてるファイル番号　FreeFile関数　※返:Ineger(1~255)

- ▶ ホームディのパス　Environ("UserProfile")

FileSystemObject（FSO）による

- ▶ ☆ FSOを作成（後始末も）
- ▶ ファの存在確認　fso.FileExists(絶パ)　※:Boolean
- ▶ ファ作成　fso.CreateTextFile 絶パ
- ▶ ステム　fso.GetBaseName(絶パ)
- ▶ 拡張子　"." & fso.GetExtensionName(絶パ)
- ▶ ベースネーム　fso.GetBaseName(絶パ) & "." & fso.GetExtensionName(絶パ)
- ▶ テキストファを開く　Set f = fso.OpenTextFile(絶パ, モード定数)　か
Set f = fso.GetFile(絶パ).OpenAsTextStream(モード定数)
※: TextStremオブジェクト
- ▶ テキストファを閉じる　TextStreamオブ.Close
- ▶ ☆ １行ずつ読み込む
- ▶ 一気に読み込む　TextStreamオブ.ReadAll
- ▶ 文字列を書き込み　TextStreamオブ.Write 文字列　※ 書かれるのは `文字列` だけ
- ▶ １行書き込み　TextStreamオブ.WriteLine 文字列　※ 〃 `文字列 & 改行`
- ▶ 改行を書き込み　TextStreamオブ.WriteBlankLines 数
- ▶ ファを削除　fso.DeleteFile(絶パ※, True)　※ワイカドで複数削除可能
- ▶ ※ FSOでワイルドカードを使う場合にかぎり、`?` は「任意の1文字または0文字」を表すっぽい。
- ▶ 親ディ　fso.GetParentFolderName(パ)

■プログラムファイルまたはcmdコマンドを実行

- ▶ ☆ 非同期で実行
- ▶ ☆ 同期で実行

Pythonとの連携

- ▶ ☆ Pythonを実行（コマンドプロンプトから）
- ▶ ☆ 引数を交換

■メールを送信

- ▶ ☆ Outlookでメールを送信

Excel VBA

■便利

- ▶ ☆ プロシージャの頭尾のテンプレ
- ▶ コピー後の点線の表示を消す

■ブック

- ▶ マクロの帰属先のブック
- ▶ ☆ 別のブックを開く
- ▶ ブックの名前
- ▶ ブックの属するディのパス
- ▶ ブックのフルパス
- ▶ ☆ OneDriveに乗っているかもしれないブックのフルパス
- ▶ マクロ有効ブックとして保存

■シート

- ▶ シートの削除（警告なし）
- ▶ シートを非表示に
- ▶ 特定のシートがActiveなら
- ▶ フィルターの解除
- ▶ シートを保護・保護解除
- ▶ シートが保護されているか
- ▶ ☆ 仮設の作業場シート（WA）をつくる

■セル範囲・セル

セル範囲の取得

- ▶ シート全体のセル
- ▶ セルの個数

Excel VBA

■便利

- ▶ ☆ プロシージャの頭尾のテンプレ
- ▶ コピー後の点線の表示を消す　Application.CutCopyMode = False

■ブック

- ▶ マクロの帰属先のブック　ThisWorkbook
- ▶ ☆ 別のブックを開く
- ▶ ブックの名前　wb.Name　※拡張子付き
- ▶ ブックの属するディのパス　wb.Path　※OneDriveに乗ってたら厄介
- ▶ ブックのフルパス　wb.FullName　※ 〃
- ▶ ☆ OneDriveに乗っているかもしれないブックのフルパス
- ▶ マクロ有効ブックとして保存

```
wb.SaveAs Filename:=Filename, _
  FileFormat:=xlOpenXMLWorkbookMacroEnabled
```

■シート

- ▶ シートの削除（警告なし）

```
Application.DisplayAlerts = False
ns.Delete
Application.DisplayAlerts = True
```

- ▶ シートを非表示に　Worksheets("Sheet2").Visible = False
- ▶ 特定のシートがActiveなら　If ActiveSheet Is Workbooks("B1").Worksheets("S1") Then
- ▶ フィルターの解除

```
On Error Resume Next
ThisWorkbook.Sheets("Sheet1").ShowAllData
On Error GoTo 0
```

- ▶ シートを保護・保護解除　sht.Protect　・　sht.Unprotect
- ▶ シートが保護されているか　sht.ProtectContents
- ▶ ☆ 仮設の作業場シート（WA）をつくる

■セル範囲・セル

セル範囲の取得

- ▶ シート全体のセル　sht.Cells　※.Range("1:" & Rows.Count)と同義
- ▶ セルの個数　rng.Cells.CountLarge　※大抵は `.Count` でもいい

- ▶ ☆ 空白でないセル
- ▶ セル範囲をインプット
- ▶ 移動・大きさ変更
- ▶ ☆ 検索

セル範囲のデザイン

- ▶ フォント
- ▶ 水平方向の中央揃え
- ▶ 垂直方向の中央揃え
- ▶ 塗りつぶし

セルの値

- ▶ セルの値
- ▶ セルでの実際の表示
- ▶ セルのなかでの改行

コピー&ペースト

- ▶ コピー
- ▶ コピー＆貼り付け
- ▶ 形式を選んで貼り付け

テーブルライクなセル範囲にたいして

- ▶ ☆ 擬表の１列を取得

■画像

- ▶ 画像を挿入

■テーブル

- ▶ ※ `x_idx` や `y_idx` はすべて 1 からね。
- ▶ 表を取得
- ▶ 表の見出しの１つのセル
- ▶ 特定の行のセル範囲
- ▶ 特定の列オブを取得
- ▶ 特定の列のセル範囲

- ▶ ☆ 空白でないセル
- ▶ セル範囲をインプット　Set r = Application.InputBox(*prompt*, Type:=8)
- ▶ 移動・大きさ変更　.Offset(*x*, *y*)　・　.Resize(*x*, *y*)
- ▶ ☆ 検索

セル範囲のデザイン

- ▶ フォント　.Font.‥
- ▶ 水平方向の中央揃え　.HorizontalAlignment = xlCenter
- ▶ 垂直方向の中央揃え　.VerticalAlignment = xlCenter
- ▶ 塗りつぶし　.Interior.Color = RGB(*R*, *G*, *B*)　など

セルの値

- ▶ セルの値　.Value
- ▶ セルでの実際の表示　.Text
- ▶ セルのなかでの改行　vbLf

コピー&ペースト

- ▶ コピー　.Copy
- ▶ コピー＆貼り付け　.Copt Destination:=貼り付け先のセル範囲
- ▶ 形式を選んで貼り付け　.PasteSpecial Paste:=定数

テーブルライクなセル範囲にたいして

- ▶ ☆ 擬表の１列を取得

■画像

- ▶ 画像を挿入　シート.Activate　セル.Select　ActiveSheet.Pictures.Insert(画像の絶パ)

■テーブル

- ▶ ※ `x_idx` や `y_idx` はすべて 1 からね。
- ▶ 表を取得　Set LO = 表内のセル範囲.ListObject
- ▶ 表の見出しの１つのセル　LO.ListColumns("列").Range(1)　か　LO.HeaderRowRange(*x_idx*)
- ▶ 特定の行のセル範囲　LO.ListRows(*y_idx*)　※ 見出しの1つ下の行が1行目。
- ▶ 特定の列オブを取得　LO.ListColumns("列")　か　LO.ListColumns(*x_idx*)
- ▶ 特定の列のセル範囲　列オブ.DataBodyRange

- ▶ 特定の行、列のセル
- ▶ ☆ フィルター解除　<まだ不確か！！！！！！！！>

■Word

- ▶ ☆ Word ファイルにあるマクロを実行

■よくやる間違い

- 行と列を逆にする
- `.Range(.Cells(1, 2), .Cells(3, 4))` でピリオドを抜かす。

■FAQ

- ▶ ☆ Q．Excelを開くたび `'～.xlam' が見つかりません。‥` と出てきて消えません。どうすればよいですか？

- ▶ 特定の行、列のセル　列オブ.DataBodyRange(*y_idx*)
- ▶ ☆ フィルター解除　<まだ不確か！！！！！！！！>

■Word

- ▶ ☆ Word ファイルにあるマクロを実行

■よくやる間違い

- 行と列を逆にする
- `.Range(.Cells(1, 2), .Cells(3, 4))` でピリオドを抜かす。

■FAQ

- ▶ ☆ Q．Excelを開くたび `'～.xlam' が見つかりません。‥` と出てきて消えません。どうすればよいですか？

Word VBA

■ドキュメント

- ▶ マクロの帰属先の文書
- ▶ 文書の属するディのパス
- ▶ 文書のフルパス
- ▶ ☆ OneDriveに乗っているかもしれない文書のフルパス

■Selection

- ▶ ※ Selection という名前だが、複数の文字列が選択されていない（網掛けができていない）なら、普通にカーソルの位置がそれに該当する。
- ▶ 位置を取得
- ▶ カーソル移動
- ▶ 特定の位置に選択を移動
- ▶ 文字を入力
- ▶ 改行する
- ▶ ☆ 検索

■Selection または Range

- ▶ ※ Selection については、Range オブジェクトと完全に交換可能ではない。Range オブジェクトに付属するメンバが Selection にある場合もあれば、Selection**.Range** にある場合もある

Word VBA

■ドキュメント

- ▶ マクロの帰属先の文書　ThisDocument
- ▶ 文書の属するディのパス　doc.Path　※OneDriveに乗ってたら厄介
- ▶ 文書のフルパス　doc.FullName　※ 〃
- ▶ ☆ OneDriveに乗っているかもしれない文書のフルパス

■Selection

- ▶ ※ Selection という名前だが、複数の文字列が選択されていない（網掛けができていない）なら、普通にカーソルの位置がそれに該当する。
- ▶ 位置を取得　.Start　や　.End
- ▶ カーソル移動　.Move *unit*, *count*　※ .MoveStart、.MoveEnd もあるよ
- ▶ 特定の位置に選択を移動　ActiveDocument.Range(*start*, *end*).Select
- ▶ 文字を入力　.typeText 文字列
.insertAfter 文字列　.insertBefore 文字列
- ▶ 改行する　.typeText 文字列 & vbCr
- ▶ ☆ 検索

■Selection または Range

- ▶ ※ Selection については、Range オブジェクトと完全に交換可能ではない。Range オブジェクトに付属するメンバが Selection にある場合もあれば、Selection**.Range** にある場合もある

（もちろん Selection にも Selection.Range にもない場合もある）。

Selection.Range と Range に共通

▶ 蛍光ペンで塗りつぶす

Selection と Rangeに共通

▶ 範囲を拡大

■テーブル

▶ ☆ 結合を解除

■画像

▶ ☆ 文書中の InlineShape をすべて削除

PowerPoint VBA

■プレゼンテーション

▶ ☆ マクロの帰属先のプレゼン（ThisPresentationが使えないので実装！）

▶ プレゼンの属するディのパス

▶ プレゼンのフルパス

▶ ☆ OneDriveに乗っているかもしれないプレゼンのフルパス

▶ 第１ウィンドウ

▶ アクティブにする

▶ 現在選択しているもの

▶ 現在の選択の状況

▶ PPTXとして新規保存

■スライド

全スライド

▶ 背景色を変更

▶ 幅・高さを変更

▶ ☆ 削除

あるスライド

（もちろん Selection にも Selection.Range にもない場合もある）。

Selection.Range と Range に共通

▶ 蛍光ペンで塗りつぶす .Range.HighlightColorIndex = *wdColorIndex*

Selection と Rangeに共通

▶ 範囲を拡大 .Expand Unit:=*unit*

■テーブル

▶ ☆ 結合を解除

■画像

▶ ☆ 文書中の InlineShape をすべて削除

PowerPoint VBA

■プレゼンテーション

▶ ☆ マクロの帰属先のプレゼン（ThisPresentationが使えないので実装！）

▶ プレゼンの属するディのパス pp.Path ※OneDriveに乗ってたら厄介

▶ プレゼンのフルパス pp.FullName ※ 〃

▶ ☆ OneDriveに乗っているかもしれないプレゼンのフルパス

▶ 第１ウィンドウ pp.Windows(1) ※: **DocumentWindow型**

▶ アクティブにする pp.Windows(1).Activate

▶ 現在選択しているもの pp.Windows(1).Selection

▶ 現在の選択の状況 win.Selection.Type

▶ PPTXとして新規保存 pp.SaveAs *newFilePath*, ppSaveAsDefault

■スライド

全スライド

▶ 背景色を変更 pp.SlideMaster.Background.Fill.ForeColor.RGB = RGB(*R*, *G*, B)

▶ 幅・高さを変更 pp.PageSetup.SlideWidth = *width* ・ .SlideHeight = *height*

▶ ☆ 削除

あるスライド

▶ 末尾に新規追加

▶ 背景画像

▶ プレゼン内で移動

■図形全般

▶ ※ 図形には、画像も含まれる。

Shape型にたいして

▶ 図形の種類

▶ グループ解除

▶ 現在の選択を拡張する形で選択

ShapeRange型にたいして

▶ ※ ShapeRange型は、Shape型やShapes型にたいしてRangeメンバを呼び出せば得られる。

▶ 図形の個数

▶ グループ化

▶ コピー＆ペースト

■テキストボックス

▶ テキボを挿入

▶ 中の文字列

▶ フォント

■画像

▶ 画像を挿入

▶ 縦横比を固定

▶ 幅・高さを変更

▶ 位置を変える

■その他

▶ Excelを使う

▶ 末尾に新規追加　set sld = pp.Slides.Add(pp.Slides.Count + 1, ppLayoutBlank)

▶ 背景画像　sld.FollowMasterBackground = msoFalse
sld.Background.Fill.UserPicture *imgPath*

▶ プレゼン内で移動　sld.MoveTo *n*

■図形全般

▶ ※ 図形には、画像も含まれる。

Shape型にたいして

▶ 図形の種類　shp.Type

▶ グループ解除　shp.Ungroup

▶ 現在の選択を拡張する形で選択　shp.Select Replace:=msoFalse

ShapeRange型にたいして

▶ ※ ShapeRange型は、Shape型やShapes型にたいしてRangeメンバを呼び出せば得られる。

▶ 図形の個数　shprng.Count

▶ グループ化　shprng.Group.Name = "グループ名"

▶ コピー＆ペースト　shprng.Copy　sld.Shapes.Paste

■テキストボックス

▶ テキボを挿入　sld.Shapes.AddTextbox(msoTextOrientationHorizontal, *left*, *top*, *w*, *h*)

▶ 中の文字列　shp.TextFrame.TextRange.Text

▶ フォント　shp.TextFrame.Font.‥

■画像

▶ 画像を挿入　sld.Shapes.AddPicture(FileName:=*filePath*, LinkToFile:=msoFalse,
SaveWithDocument:=msoTrue, Left:=*left*, Top:=*top*)

▶ 縦横比を固定　shp.LockAspectRatio = True

▶ 幅・高さを変更　shp.Width = *width*　・　.height = *height*

▶ 位置を変える　shp.Left = *left*　shp.Top = *top*

■その他

▶ Excelを使う　Set exApp = CreateObject("Excel.Application")