

Visual Basic for Applications



Basics of programming in VBA

fundamental concepts

object-oriented model of the application

introduction to VBA in Excell

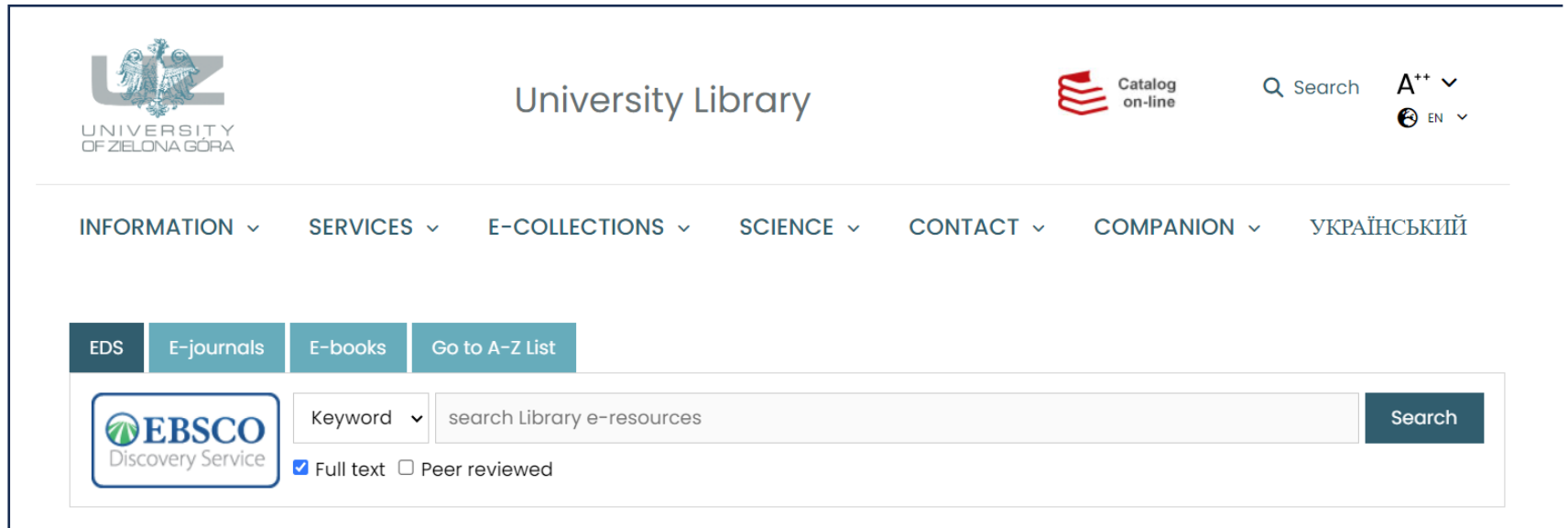
Materials

<http://staff.uz.zgora.pl/ipajak>

<http://staff.uz.zgora.pl/gpajak>

Literature

EBSCO search browser (<http://www.bu.uz.zgora.pl/>)



The screenshot displays the University Library website. At the top left is the University of Zielona Góra logo. The main header features the text "University Library". To the right, there is a "Catalog on-line" icon, a search bar with a magnifying glass icon and the text "Search", and language selection options "A++" and "EN". Below the header is a navigation menu with links: "INFORMATION", "SERVICES", "E-COLLECTIONS", "SCIENCE", "CONTACT", "COMPANION", and "УКРАЇНСЬКИЙ". The main content area has a row of buttons: "EDS", "E-journals", "E-books", and "Go to A-Z List". Below this is the EBSCO Discovery Service search interface, which includes a search bar with a dropdown menu set to "Keyword", the text "search Library e-resources", and a "Search" button. There are also checkboxes for "Full text" (checked) and "Peer reviewed" (unchecked).

Web browser configuration: [server-proxy-cache](#)

Literature

- ❑ DeMarco Jim, *Pro Excel 2007 VBA* , Apress 2008, ISBN: 978-1-59059-957-0; 978-1-4302-0580-7 
- ❑ Kofler Michael, *Definitive Guide to Excel VBA*, Second Edition, Apress 2003, ISBN: 978-1-59059-103-1; 978-1-4302-0666-8 
- ❑ Walkenbach John, *Excel VBA Programming For Dummies, Second edition*, John Wiley & Sons Incorporated, 2010, ISBN: 978-0-470-50369-0; 978-0-470-63275-8 
- ❑ Walkenbach John, *Excel 2010 Power Programming with VBA*, John Wiley & Sons Incorporated, 2010, ISBN: 978-0-470-47535-5; 978-0-470-62548-4 
- ❑ Dixon Helen, *Excel 2007. Beyond the Manual*, Apress 2007, ISBN: 978-1-59059-798-9; 978-1-4302-0389-6 
- ❑ Scott Kendall, *Fast Track UML 2.0*, Apress 2004, ISBN: 978-1-59059-320-2; 978-1-4302-0720-7 



e-book download



access via a web browser

Fundamental concepts

Basic concepts

Algorithm – a set of steps leading to the solution of a specific task; a set of commands specifying the method of processing a dataset with the order of their execution.

Programming language – a formalized language used to write computer-executed algorithms.

High-level programming language – programming language independent of particular type of computer, close to human language, the execution of the program requires translation using an appropriate translator.

Programming – designing programs and preparing them for operation; coding of algorithms in a given programming language.

Program – an algorithm written in a specific programming language with the data structures on which it operates; "recipe" expressed in the appropriate notation according to which the computer or other device performs the actions included in the algorithm.

BASIC programming language

BASIC (*Beginner's All-purpose Symbolic Instruction Code*) – a high-level programming language designed by John Kemeny and Thomas Kurtz based on Fortran and Algol-60 in 1964. In the original version it included only elementary instructions, the program was executed step by step (pipelined), no structured or object programming possible.

Visual Basic (VB) – a high-level programming language designed by Microsoft (1991). The syntax based on original BASIC, but it is modernized and extended. Partially object-oriented language, it does not support all object-oriented programming mechanisms. Development finished in 2008 (VB 6.0).

Visual Basic .NET – successor of VB 6.0, designed for .NET Framework.

Visual Basic for Applications (VBA) – simplified version of Visual Basic implemented as language of macros in Microsoft Office package (also available in AutoCAD, WordPerfect, Statistica). VBA does not allow to create standalone programs, program code is embedded in a document of a specific application (e.g. "docm" or "xlsm" file), it operates in environment of this application.

VBA application in MS Excel

- ❑ Recording macros to automate repetitive tasks.
- ❑ Creating macros to perform complex activities that require interaction with the user.
- ❑ Creating new worksheet functions, not available in Excel, to use in formulas.
- ❑ Creating user interface elements (e.g. buttons, lists) placed in worksheets.
- ❑ Creating new application functions performing operations not available in Excel.
- ❑ Creating dialog windows as interface of new operations.
- ❑ Creating add-ins extending capabilities of application.

Object-oriented programming

Object-oriented programming (OOP) – programming, usually in object-oriented programming language, based on defining objects, their properties and methods. Each object is treated as independent fragment of code, operation of the program is based on cooperation between objects.

Object – abstract entity representing certain element or concept existing in the modeled system. Object is distinguishable from other objects, has a name and well defined boundaries.

Property (attribute, field) – important feature of the object from the model point of view.

Method – operation assigned to object, action that the object can perform.

Class – definition of the structure of the certain group of objects.

Examples

Object: *John Smith*, class: *Student*

Properties: *first name*, *last name*, *date of birth*, *student group*, *student grading list*.

Methods: *calculate age*, *calculate average of student grades*.

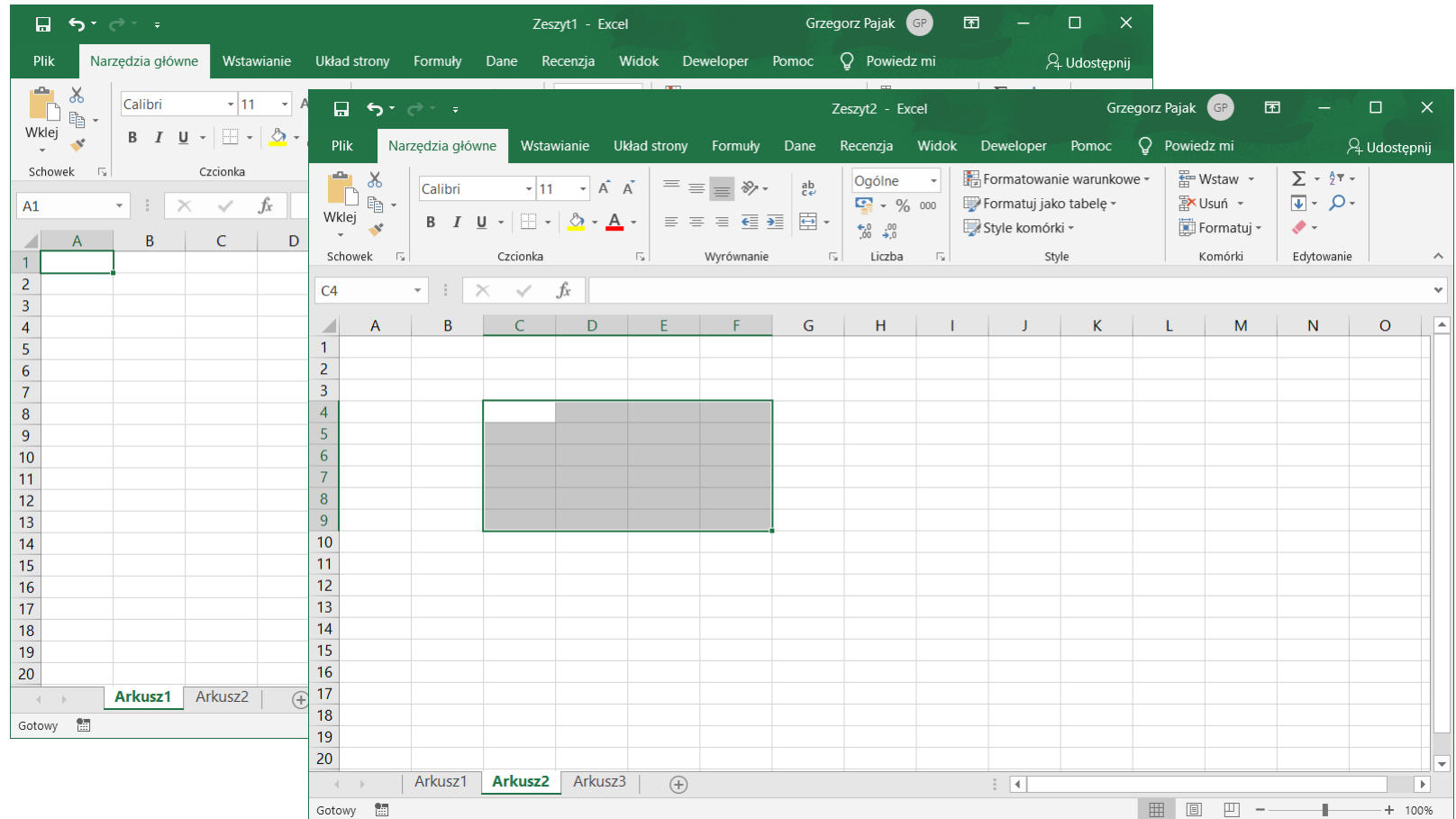
Object: *Excel document*, class: *Workbook*

Properties: *name*, *sets of cells*, *rows*, *columns*, *selected range*, *etc.*.

Methods: *add new worksheet*, *activate workhit*, *calculate formula*, *save document*, *etc.*

Object-oriented model of Excell

Object-oriented model of Excel



Two documents: *Zeszyt1*, *Zeszyt2*; active document: *Zeszyt2*

Document *Zeszyt1*: two worksheets: *Arkusz1*, *Arkusz2*

Document *Zeszyt2*: three worksheets: *Arkusz1*, *Arkusz2*, *Arkusz3*; active worksheet: *Arkusz2*

Active cell in *Arkusz2*: *C4*, selected range (i.e. selection) in *Arkusz2*: *C4:F9*

Object-oriented model of Excel

Classes of objects

- Application (properties: *user name*, *active cell*, *selected range*, etc.),
- Workbook (properties: *document name*, *file path*, *active sheet*, etc.),
- Worksheet (properties: *worksheet name*, *used range of cells*, etc.),
- Range (properties: *address*, *value*, *number of rows and columns*, etc.).

Objects

- Application – Excel,
- Workbook – each document (*Zeszyt1* and *Zeszyt2*),
- Worksheet – each worksheet (*Arkusz1-2* in *Zeszyt1* and *Arkusz1-3* in *Zeszyt2*),
- Range – single cell.

Objects collections

- Workbooks – collection of opened documents,
- Worksheets – collection of worksheets in certain document,
- Range – collection of cells (e.g. selected range, column or row, etc.).

Object-oriented model of Excel

APPLICATION (EXCELL)

WORKBOOKS (COLLECTION OF OPENED DOCUMENTS)

WORKBOOK (ZESZYT1)

WORKSHEETS

(COLLECTION OF ALL WORKSHEETS IN ZESZYT1)

WORKSHEET (ARKUSZ1)

WORKSHEET (ARKUSZ2)

WORKBOOK (ZESZYT2, ACTIVE)

WORKSHEETS

(COLLECTION OF ALL WORKSHEETS IN ZESZYT2)

WORKSHEET (ARKUSZ1)

WORKSHEET (ARKUSZ2, ACTIVE)

WORKSHEET (ARKUSZ3)

Note: The above model is not complete and use some simplifications.
It corresponds to example presented in s.8.

Object Application

Application represents whole application (Excel in this case), it exists since program is launched, always in one instance.

Selected properties

- **ActiveCell** – class Range, active cell (only one possible),
- **ActiveSheet** – class Worksheet, active worksheet (only one possible),
- **ActiveWorkbook** – class Workbook, active workbook (only one possible),
- **Selection** – class Range, selected range of cells (in active worksheet),
- **UserName** – string, application user name,
- **Worksheets** – class Worksheets, collection of worksheets in active workbook.

Selected methods

- **Calculate** – recalculates all open workbooks,
- **InputBox(text)** – displays a dialog box for user input,
- **Quit** – quits application.

Collection Workbooks

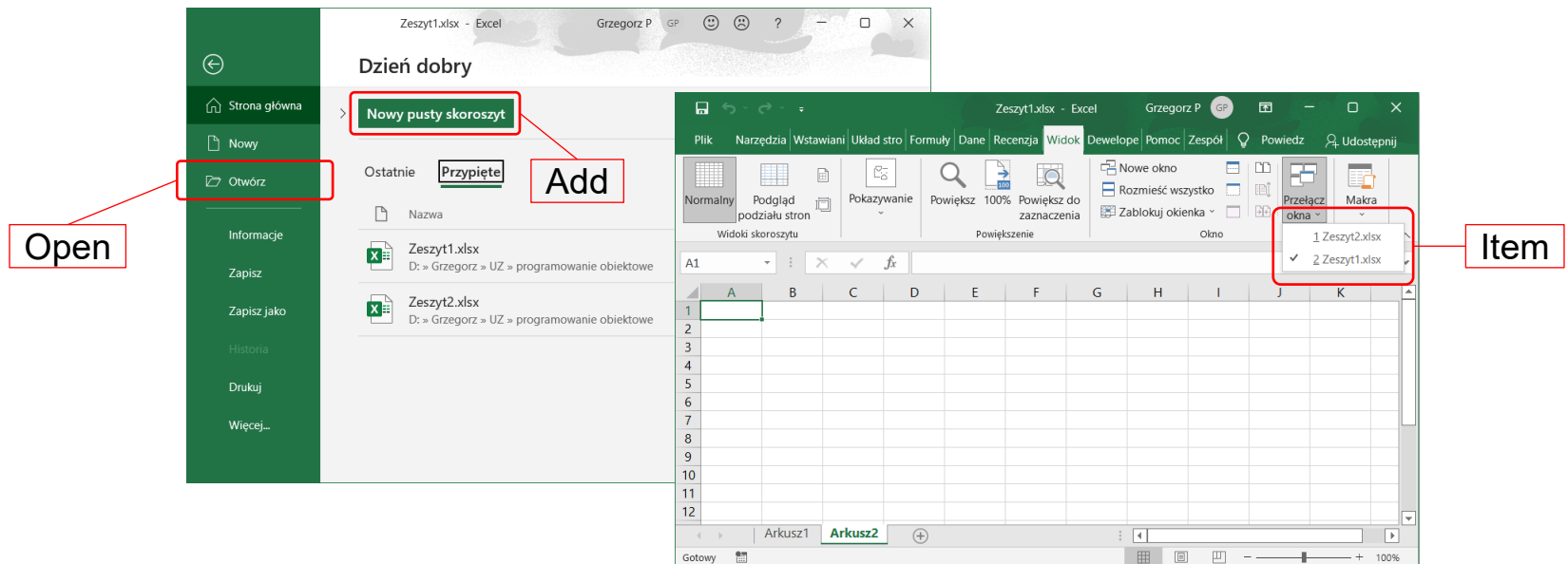
Workbooks represents collection of all opened documents.

Selected properties

- **Count** – Long, number of objects in collection (number of opened documents),
- **Item**(*name|index*) – class Workbook, single object in collection (by name or index).

Selected methods

- **Add** – adds new, empty document to collection,
- **Open**(*name*) – opens document specified by *name*.



Object Workbook

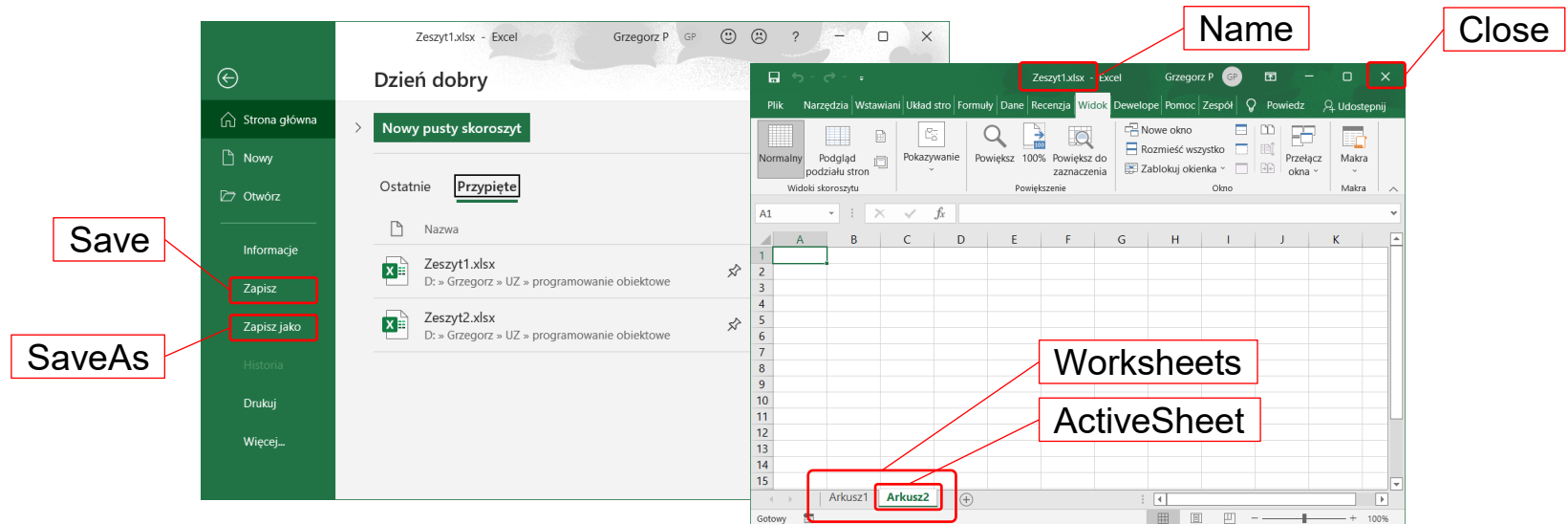
Workbook represents one document from collection Workbooks.

Selected properties

- **ActiveSheet** – class Worksheet, active worksheet,
- **Name, Path, FullName** – String, name of file, path and path+name,
- **Worksheets** – class Worksheets, collection of worksheets included in the workbook.

Selected methods

- **Activate** – activates Workbook,
- **Close** – closes Workbook,
- **Save, SaveAs(name)** – saves Workbook using current or new *name*.



Collection Worksheets

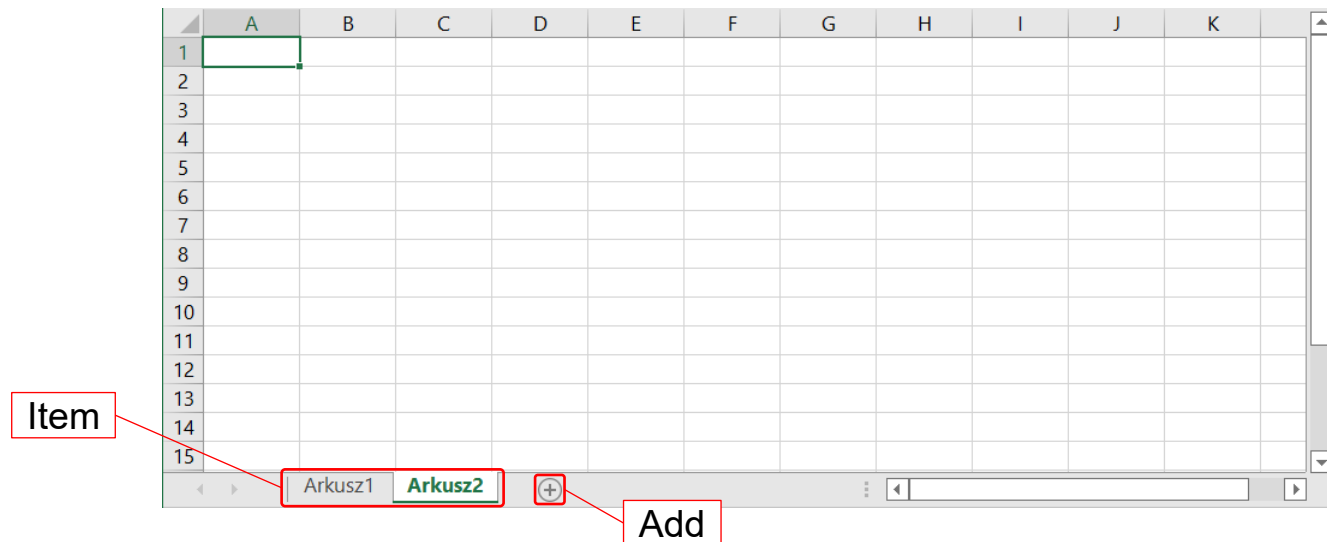
Worksheets represents collection of worksheets.

Selected properties

- **Count** – Long, number of objects in collection (number of worksheets),
- **Item**(*name|index*) – class Worksheet, single object in collection (by name or index).

Selected methods

- **Add**(*before, after*) – adds new, empty worksheet to collection *before* or *after* specified worksheet.



Object Worksheet

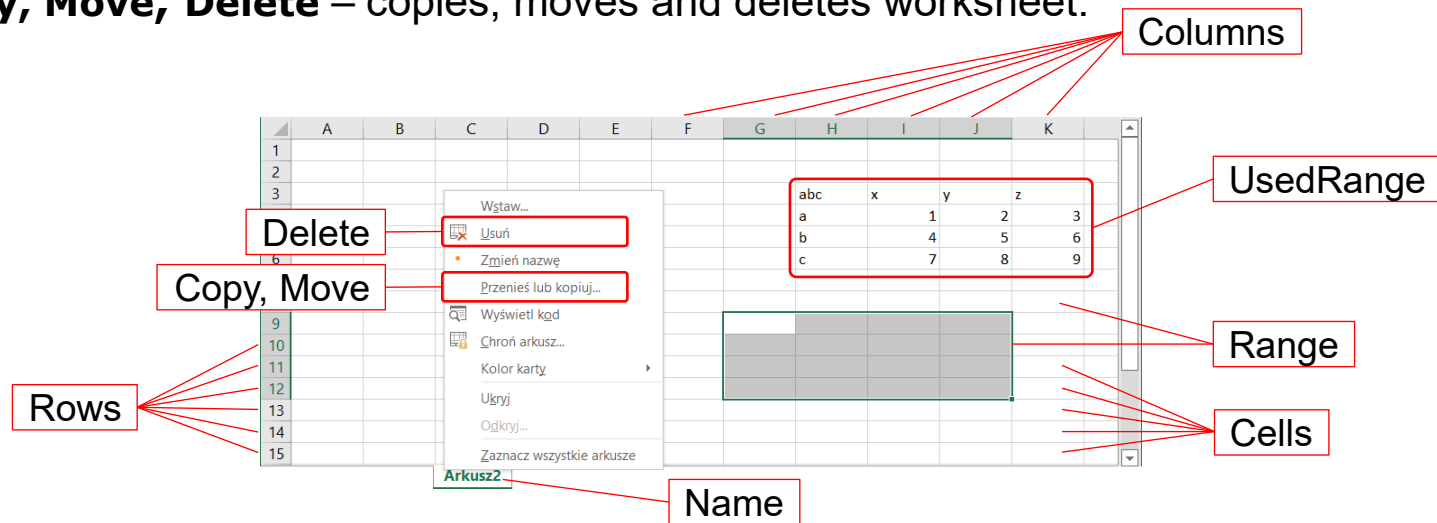
Worksheet represents one worksheet from collection Worksheets

Selected properties

- **Cells**(*index*) – class Range, collection of cells of worksheet,
- **Columns**, **Rows** – class Range, collections of columns and rows,
- **Name**, **Index** – name (String) or number (Long) of worksheet,
- **Range**(*rng*) – class Range, collection of cells in any range,
- **UsedRange** – class Range, used range of worksheet.

Selected methods

- **Activate** – activates worksheet,
- **Copy, Move, Delete** – copies, moves and deletes worksheet.



Object Range – selected properties

Range represents any range of cells (whole worksheet, selected columns or rows, selected range, etc.).

Selected properties

- **Address** – String, address of range (e.g. C3:F5),
- **Cells** – class Range, collection of cells in the range,
- **Columns, Rows** – class Range, collections of columns and rows included in range,
- **Count, Column, Row** – Long, number of cells, number of first column and row,
- **End(*dir*)** – cell at the beginning/end of the range (xlDown, xlToLeft, xlToRight, xlUp),
- **EntireRow** – class Range, entire row(s) containing the range,
- **Font** – font used in range (object, contains name, color, size, etc.),
- **Formula, FormulaR1C1** – formula contained in range in A1 or R1C1 notation,
- **Offset(*r,c*)** – class Range, range shifted by *r* rows and *c* columns,
- **Resize(*r,c*)** – class Range, resized range containing *r* rows and *c* columns,
- **Text** – String, content of range as text,
- **Value** – Variant, value entered into the cell.

Object Range – selected methods

Range represents any range of cells (whole worksheet, selected columns or rows, selected range, etc.).

Selected methods

- **Activate** – activates range (used to select one cell in worksheet),
- **Clear** – clears range (removes values, formulas, formats),
- **ClearContents** – clears content of range (only values and formulas),
- **Copy** *rng*, **Cut** *rng*, **PasteSpecial** – copies/cuts/pastes one range to other range (specified by *rng*) or to clipboard if range is not specified,
- **Delete** *dir* – removes range, the remaining cells are moved in direction specified by *dir* (xlShiftToLeft, xlShiftUp),
- **FunctionWizard** – displays the function wizard dialog,
- **Insert** *dir* – inserts range, the remaining cells are moved in direction specified by *dir* (xlShiftToRight, xlShiftDown),
- **Merge** – merges the cells in range,
- **Select** – selects range (used to select group of cells in worksheet),
- **Speak** – speaks the content of the cells in range.

Object Browser

Object Browser – the tool accessible in VBA, allows to browse classes, their properties and methods.

The screenshot shows the VBA Object Browser window. The 'Range' object is selected in the 'Library' list. The 'Search Results' table lists the following:

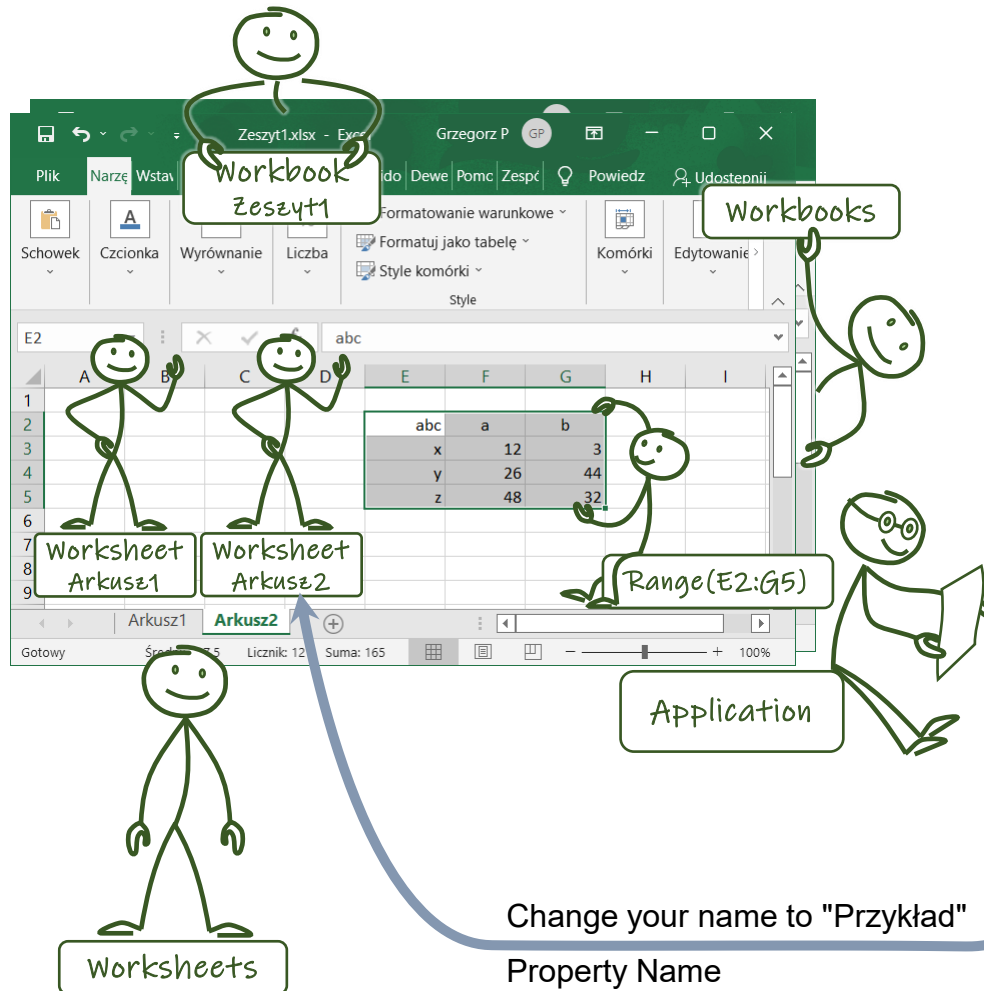
Library	Class	Member
Excel	AllowEditRange	
Excel	AllowEditRanges	
Excel	Range	
Excel	Ranges	
Excel	ShapeRange	

The 'Classes' list on the left includes: PublishObject, PublishObjects, Queries, QueryTable, QueryTables, QuickAnalysis, Range, Ranges, RecentFile, RecentFiles, RectangularGrac, ReflectionForma, Research, RoutingSlip, RTD, and Ruler2. The 'Members of 'Range'' list includes: Addindent, Address, AddressLocal, AllowEdit, Application, Areas, Borders, Cells, Characters, Column, Columns, ColumnWidth, Comment, CommentThreaded, Count, and Countl arae. The 'Property Cells As Range' section shows 'read-only' and 'Member of Excel.Range'. A context menu is open over the 'Cells' member, showing options like Copy, View Definition, Find Whole Word Only, Group Members, Show Hidden Members, References..., Properties..., Help, Dockable, and Hide. The 'Help' option is highlighted.

Annotations and icons:

- Searching element:** Points to the 'Range' dropdown in the 'Library' list.
- Search results:** Points to the 'Search Results' table.
- Element list (classes, modules, types):** Points to the 'Classes' list.
- global objects:** Represented by a globe icon.
- classes:** Represented by a cube icon.
- modules:** Represented by a document icon.
- types:** Represented by a folder icon.
- Content of selected element (properties, methods, etc.):** Points to the 'Members of 'Range'' list.
- properties:** Represented by a document icon.
- methods:** Represented by a document icon.
- events:** Represented by a lightning bolt icon.
- elements of types:** Represented by a folder icon.
- Syntax:** Points to the 'Help' option in the context menu.

The idea of working with objects – example I

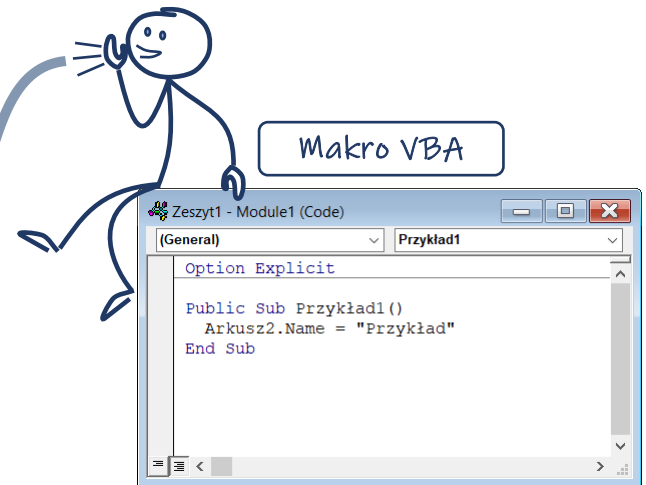


Task: Rename worksheet "Arkusz2" to "Przykład".

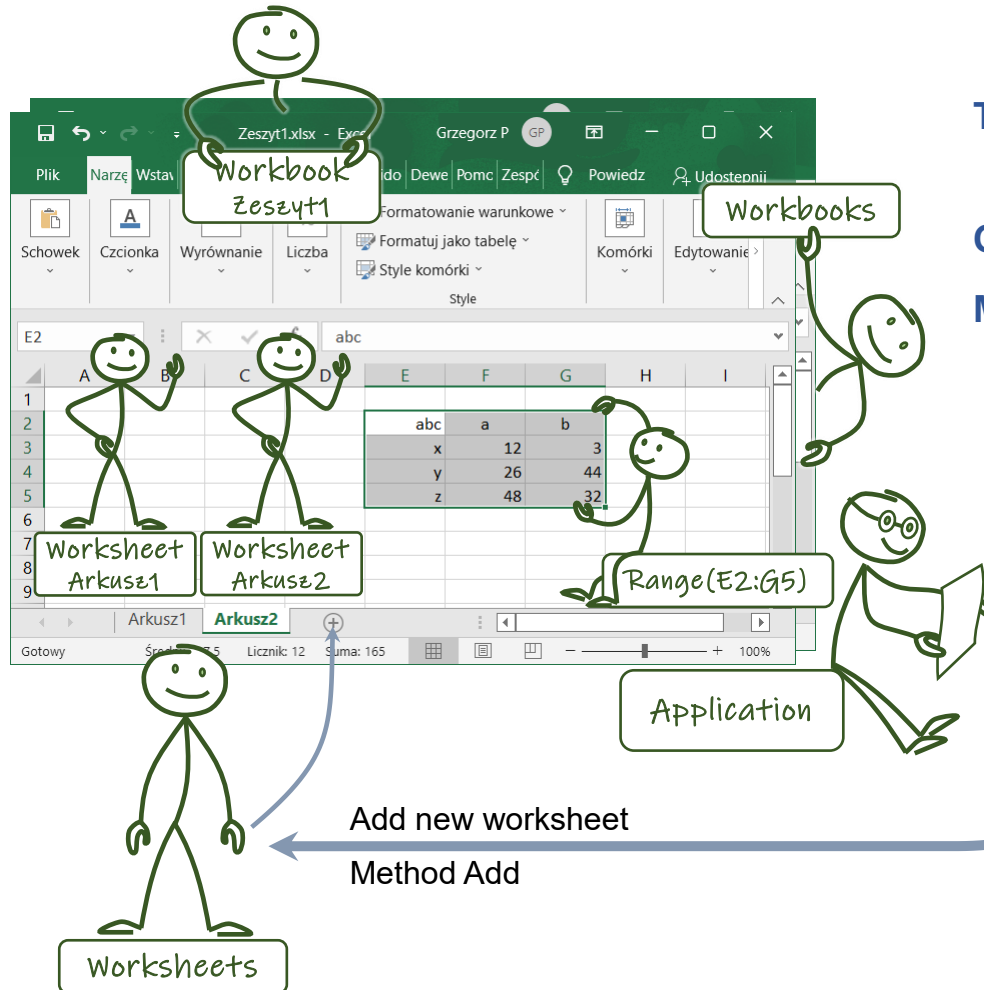
Object: Worksheet Arkusz2.

Property: Name.

Change your name to "Przykład"
Property Name



The idea of working with objects – example II



Task: Add new worksheet to workbook "Zeszyt1".

Object: Worksheets from "Zeszyt1"

Method: Add

Add new worksheet
Method Add

Makro VBA

```
Option Explicit

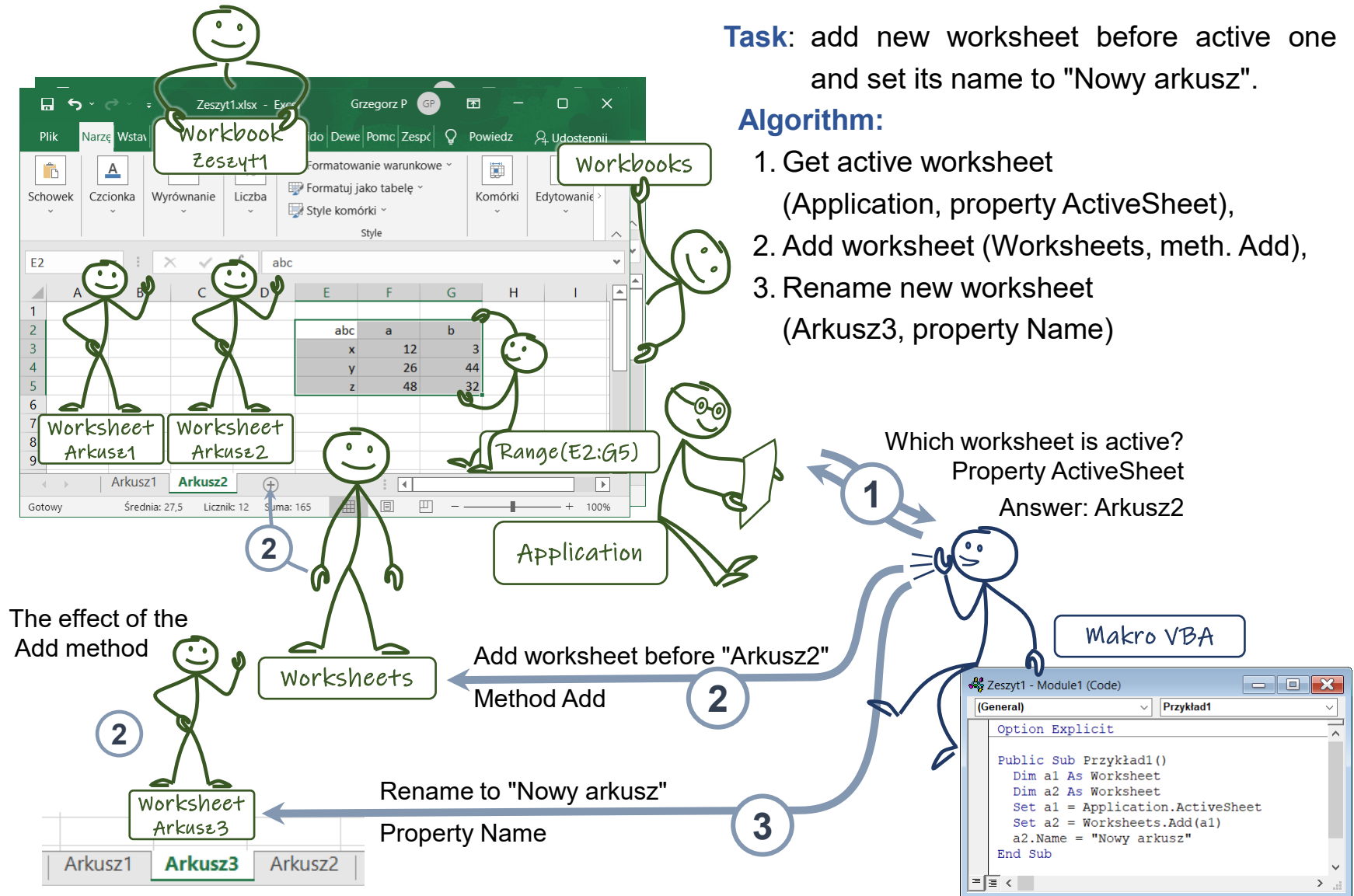
Public Sub Przykład1()
    Workbooks("Zeszyt1.xls").Worksheets.Add
End Sub
```

The idea of working with objects – example III

Task: add new worksheet before active one and set its name to "Nowy arkusz".

Algorithm:

1. Get active worksheet
(Application, property ActiveSheet),
2. Add worksheet (Worksheets, meth. Add),
3. Rename new worksheet
(Arkusz3, property Name)



Introduction to VBA in Excell

Basic principles of programming in VBA

- ❑ VB is case insensitive (additionally, text editor automatically corrects source code).
- ❑ The names defined by user must start with a letter, it can not contain special chars (space , * ; . = etc.).
- ❑ A single instruction should be placed on one program line.
- ❑ In order to split instruction into several program lines continuation symbol „ _” (space followed by underscore) should be used.
- ❑ If several instructions are placed in one program line they should be separated by „:” (reduces the readability of the code, it should be used only in special cases).
- ❑ Empty lines are omitted, they can be used to divide the code into smaller blocks (improves program readability).
- ❑ Language keywords (name of instructions, elements of macro headers, etc.) are reserved and cannot be used as names of elements defined by user.
- ❑ Text that begins with an apostrophe is a comment, it does not affect the execution of the program.

Using objects

Reference to property and method

`object_name.property_name`

`object_name.method_name`

Examples

`Application.ActiveSheet` – reference to active worksheet

`Application.ActiveSheet.Name` – reference to name of active worksheet

`Application.Worksheets.Add` – call the method Add (creates new worksheet)

Reference to collection

`collection_name(index)`

`index` describes element in collection, its form depends on collection kind (usually number and/or text).

Examples

`Application.Workbooks(1)` – reference to the first open document

`Application.Workbooks("Z1.xlsm")` – reference to document „Z1.xlsm”

`Application.Workbooks("Z2.xlsm").Worksheets("Arkusz1")` – reference to worksheet „Arkusz1” in document „Z2.xlsm”

Default objects

Default objects can be omitted in references to their properties of object types.

Default objects in VBA

- ❑ Object **Application** is always is default and it can be omitted in most references, so references in the form:

```
Application.Selection
```

```
Application.Workbooks("Zeszyt1.xlsm")
```

may be shortened to:

```
Selection
```

```
Workbooks("Zeszyt1.xlsm")
```

- ❑ Objects **Workbook** and **Worksheet** can be omitted if reference applies to active document or worksheet, so references in the form:

```
ActiveWorkbook.Worksheets("Arkusz2")
```

```
ActiveSheet.Range("A5:C7")
```

may be shortened to:

```
Worksheets("Arkusz2")
```

```
Range("A5:C7")
```

Data types

Data type defines the range of values that a given program element can store (e.g. property of an object).

Predefined data types in VBA

- **Byte** – small integer 0 .. 255
- ➔ ▪ **Integer** – integer -32 768 .. 32 767
- **Long** – long integer -2 147 483 648 .. 2 147 483 647
- ➔ ▪ **Single** – real number (single precision, 8 digits) 1.40e-45 .. 3.40e38
- **Double** – real number (double precision, 16 digits) 4.94e-324 .. 1.79e308
- **Currency** – number with 4 decimal places
-922 337 203 685 477.5808 .. 922 337 203 685 477.5807
- ➔ ▪ **Boolean** – logical with values: True and False
- **Date** – date/time 1 January 100 .. 31 December 9999
- ➔ ▪ **String** – sequence of characters, values in quotation marks, max. 2³¹ chars.
- **Object** – reference to any object
- ➔ ▪ **Variant** – any type, can store any of the above values

Note: 1.756e02=1.756·10²=175.6, 2.5e-03=2.5·10⁻³=0.0025

Data types – examples

127	– Integer (Byte , Integer or Long)
5786	– Integer (Integer or Long)
"127"	– sequence of characters (String)
12.576	– real number (Single , Double , or Currency)
True	– logical value (Boolean)
"False"	– sequence of characters (String)
253,8	– incorrect value (a comma instead of a dot)
25.39e-11	– real number (Single or Double)
#20/5/2019 8:20:00 PM#	– date and time (Date)
ActiveCell	– Application property, object (Object , precisely Range)
Selection	– Application property, object (Object , precisely Range)
Worksheets(1)	– Application property, object (Object , precisely Worksheet)
ActiveCell.Value	– type depends on the cell content, Variant

Assignment statement

Assignment statement

element_name = value

An assignment statement sets a value of specific item of program (such as an object property). The type of the assigned value must match the type of the item.

Example 1 – setting a value in the active cell of the worksheet

```
ActiveCell.Value = 125
```

```
ActiveCell.Value = "Sum"
```

Example 2 – setting font properties in cell "C7"

```
Range("C7").Font.Color = RGB(255, 0, 0)
```

```
Range("C7").Font.Color = vbRed
```

```
Range("C7").Font.Name = "Courier New"
```

Note 1: Font is an object property, it contains properties Color, Name, Size, etc.

Note 2: Color can be specified using the function `RGB` giving as parameters the saturation of the red, green and blue components or using predefined constants `vb<color_name>`.

Macro examples

Modifying active cell: setting value to 257, changing font to Courier New, blue, 14pt.

```
Public Sub Example1()
```

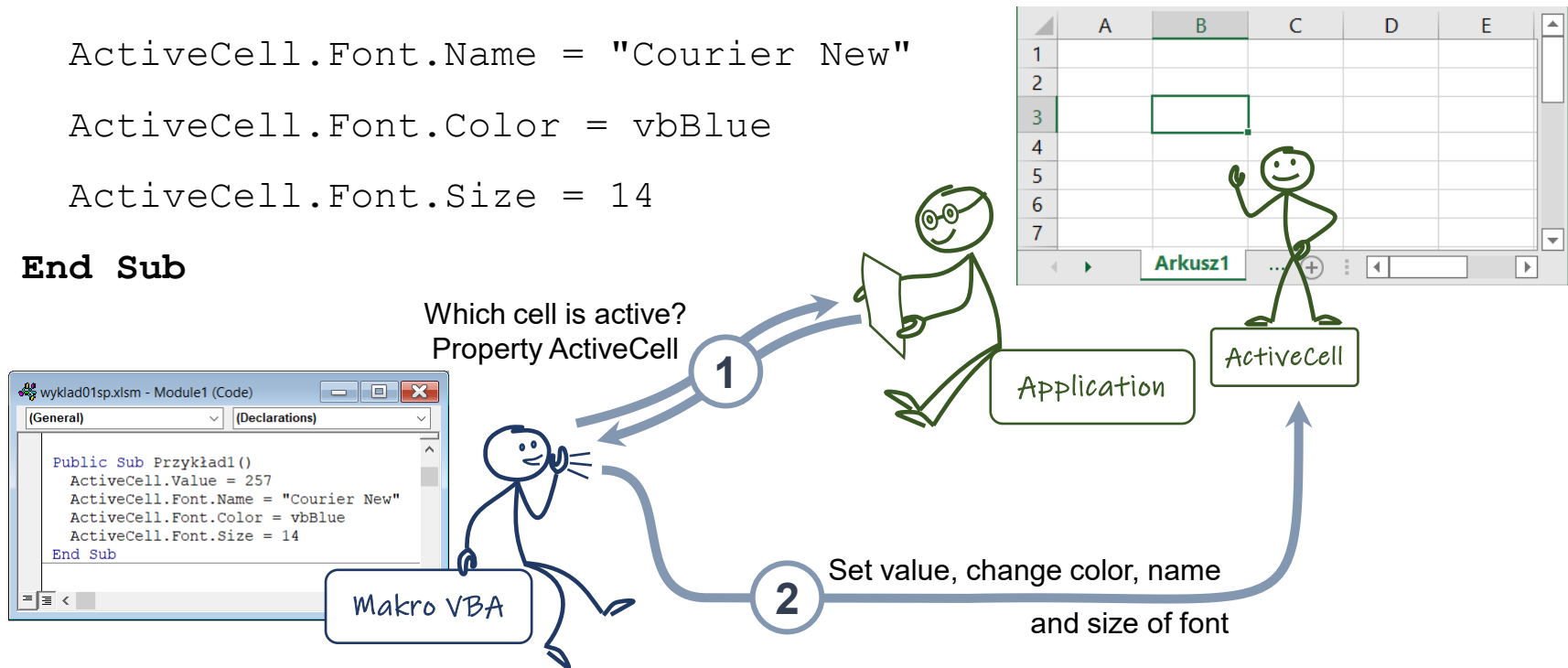
```
    ActiveCell.Value = 257
```

```
    ActiveCell.Font.Name = "Courier New"
```

```
    ActiveCell.Font.Color = vbBlue
```

```
    ActiveCell.Font.Size = 14
```

```
End Sub
```



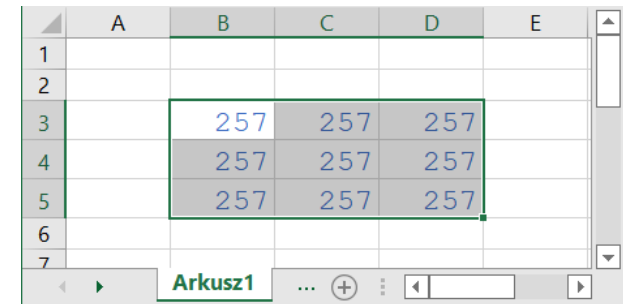
Note: ActiveCell is a property of the Application object, because it is the default object in the reference its name may be omitted.

Source codes are available on the website

Macro examples

Modifying all cells in the currently selected range : setting value to 257, changing font to Courier New, blue, 14pt.

```
Public Sub Example2()  
    Selection.Value = 257  
    Selection.Font.Name = "Courier New"  
    Selection.Font.Color = vbBlue  
    Selection.Font.Size = 14  
  
End Sub
```



Note: Changing the properties of a Range object changes the corresponding properties of all cells in that range.

Macro examples

Writing the contents of the range A1:A5 from Arkusz1 to the range C3:E7 in Arkusz2, then activating Arkusz2.

```
Public Sub Example3()  
    Worksheets("Arkusz2").Range("C3:E7").Value = _  
    Worksheets("Arkusz1").Range("A1:A5").Value  
    Worksheets("Arkusz2").Activate
```

End Sub

Note: one instruction is written on two lines, continuation symbol is needed: " _" (space followed by underscore).

Writing the contents of selected range two columns further to the right

```
Public Sub Example4()  
    Selection.Offset(0, 2).Value = Selection.Value
```

End Sub

Source codes are available on the website

With statement

With <object>

references to properties and methods

End With

Inside the `With` statement, all references to an *object* properties and methods can be written as:

`.property_name`

`.method_name`

Example

Both macros set value of active cell to 257 and change its font (color and size).

```
Public Sub Example1a()  
    ActiveCell.Value = 257  
    ActiveCell.Font.Color = vbBlue  
    ActiveCell.Font.Size = 14  
End Sub
```

```
Public Sub Example1b()  
    With ActiveCell  
        .Value = 257  
        .Font.Color = vbBlue  
        .Font.Size = 14  
    End With  
End Sub
```

Message box

To display message on screen the following standard procedure can be used

```
MsgBox <text>
```

where <text> is a displayed message.

Examples

Simple message

```
Public Sub Example5()  
    MsgBox "This is a message box"  
End Sub
```

User application name

```
Public Sub Example6()  
    MsgBox Application.UserName  
End Sub
```



Note: UserName is not an object property, so in this case object Application can not be omitted.