

# Techniki programowania



## Podstawy programowania w VBA

pojęcia podstawowe  
obiekty model aplikacji  
podstawowe obiekty i ich właściwości  
typy danych, zmienne i stałe

- ❑ Alexander M., Kusleika R. *Excel 2016 PL. Programowanie w VBA*, Helion, Gliwice 2016
  - ❑ Kuciński K., *Visual Basic dla Excela w przykładach*, Wydawnictwo Witanet 2015
  - ❑ Lewandowski M., *VBA dla Excela 2010. Leksykon kieszonkowy*, Helion, Gliwice 2012
  - ❑ Lewandowski M., *Tworzenie makr w VBA dla Excela 2010/2013 Ćwiczenia*, Helion, Gliwice 2014
  - ❑ McFedries P., *Microsoft Office 2007 PL język VBA i makra: usprawnij działanie najpopularniejszego pakietu biurowego*, Helion, Gliwice 2008
  - ❑ Walkenbach J., *Excel 2013 PL. Programowanie w VBA dla bystrzaków*, Helion, Gliwice 2014
- 
- ❑ Walkenbach J., *Excel 2016 PL. Biblia - Helion*, Gliwice 2016
  - ❑ Wrotek W., *VBA dla Excela 2016 PL: 222 praktyczne przykłady*, Helion, Gliwice 2016
  - ❑ Baca J., *Excel 2016 i programowanie VBA. Kurs video. Poziom drugi. Zaawansowane techniki tworzenia makr*, Videopoint 2016
  - ❑ Jelen B., Syrstad T., *Excel 2016 VBA i makra*, PROMISE 2016.

**Algorytm** – przepis postępowania prowadzący do rozwiązania określonego zadania; zbiór poleceń określających sposób przetwarzania zbioru danych ze wskazaniem kolejności w jakiej mogą być wykonane.

**Język programowania** – sformalizowany język opisu algorytmów przeznaczonych do wykonywania przez komputer.

**Język wysokiego poziomu** – niezwiązany z określonym typem komputera, wykonanie programu wymaga wcześniejszego przetłumaczenia przy pomocy odpowiedniego translatora.

**Programowanie** – konstruowanie programów i przygotowanie ich do eksploatacji; kodowanie algorytmów w danym języku programowania.

**Program** – algorytm zapisany w określonym języku programowania wraz ze strukturami danych na których operuje. Program jest przepisem wyrażonym w odpowiedniej notacji według którego komputer lub inne urządzenie interpretujące wykonuje czynności przewidziane w algorytmie.

# Język programowania BASIC

---

**BASIC** (ang. *Beginner's All-purpose Symbolic Instruction Code* – uniwersalny kod instrukcji symbolicznych dla początkujących) to język programowania wysokiego poziomu opracowany przez Johna Kemeny'ego i Thomasa Kurtza w oparciu o Fortran i Algol-60 w 1964 roku. W pierwotnej wersji program wykonywany potokowo, brak możliwości programowania strukturalnego.

**Visual Basic (VB)** to język programowania wysokiego poziomu stworzony przez firmę Microsoft. Składna oparta o BASIC, unowocześniona, wprowadzono nowe elementy. Język częściowo obiektowy (nie wspiera wszystkich mechanizmów programowania obiektowego). Rozwój zakończony w 2008 roku (VB 6.0).

**Visual Basic .NET** – następca VB 6.0, przeznaczony do tworzenia aplikacji dla platformy .NET Framework.

**Visual Basic for Applications (VBA)** to uproszczona wersja Visual Basic'a zaimplementowana jako język makrodefinicji w aplikacjach pakietu Microsoft Office (dostępny także w AutoCAD, WordPerfect, Statistica). VBA nie pozwala na tworzenie samodzielnych programów, kod programu jest elementem dokumentu określonej aplikacji (np. pliku „docm”, lub „xlsm”) i działa wyłącznie w jej środowisku.

# Zastosowanie VBA w programie Excel

---

- ❑ Rejestrowanie makr automatyzujących powtarzalne czynności.
- ❑ Tworzenie makr realizujących złożone czynności, wymagające komunikacji z użytkownikiem.
- ❑ Tworzenie nowych funkcji arkuszowych, niedostępnych w programie Excel, które można wykorzystać w formułach.
- ❑ Tworzenie nowych funkcji aplikacji realizujących operacje, które nie są niedostępne w programie Excel.
- ❑ Tworzenie przycisków i innych elementów sterujących dostępnych w arkuszach.
- ❑ Tworzenie elementów interfejsu użytkownika (okna dialogowe).
- ❑ Tworzenie dodatków rozszerzających możliwości aplikacji.

**Programowanie obiektowe** (**OOP**, *object-oriented programming*) – programowanie, zazwyczaj w obiektowym języku programowania, które polega na wyodrębnieniu obiektów, ich własności oraz metod. Każdy obiekt traktowany jest jako samodzielny, zamknięty fragment kodu, a funkcjonowanie programu opiera się na współpracy pomiędzy obiektami.

**Obiekt** – abstrakcyjny byt reprezentujący pewną rzecz lub pojęcie obserwowane w modelowanym systemie. Obiekt jest odróżnialny od innych obiektów, ma nazwę i dobrze określone granice.

**Własność** (atrybut, pole) – istotna z punktu widzenia modelu cecha obiektu.

**Metoda** – operacja przypisana do obiektu, operacja, którą potrafi wykonać dany obiekt.

**Klasa** – definicja struktury pewnej grupy obiektów.

## Przykłady

Obiekt: *Jan Kowalski*, klasa: *Student*

Własności: nazwisko, imię, data urodzenia, grupa, lista ocen.

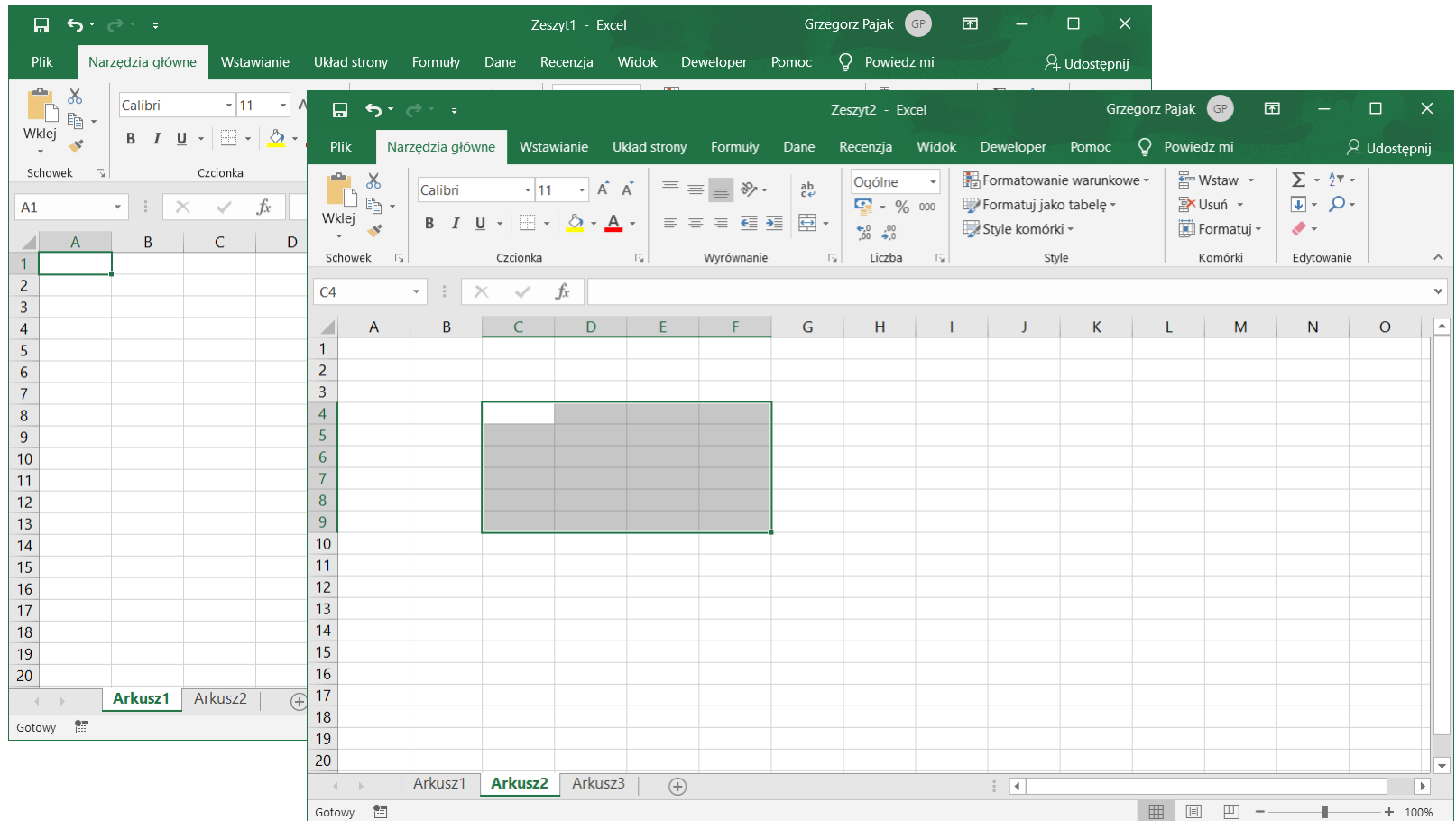
Metody: oblicz wiek, oblicz średnią, napisz kolokwium.

Arkusze programu Excel

Właściwości: nazwa, zbiór komórek, zbiór wierszy, wybrany zakres komórek

Metody: aktywuj, przelicz, usuń, zapisz.

# Obiektowy model Excel-a



Dwa dokumenty: *Zeszyt1*, *Zeszyt2*; dokument aktywny: *Zeszyt2*

Dokument *Zeszyt1*: dwa arkusze: *Arkusz1*, *Arkusz2*

Dokument *Zeszyt2*: trzy arkusze: *Arkusz1*, *Arkusz2*, *Arkusz3*; arkusz aktywny: *Arkusz2*

Aktywna komórka w *Arkusz2*: C4, wybrany zakres w *Arkusz2*: C4:F9

## Klasy obiektów

- Application (własności: użytkownik, aktywna komórka, wybrany zakres, itp.),
- Workbook (własności: nazwa dokumentu, lista arkuszy, arkusz aktywny, itp.),
- Worksheet (własności: nazwa arkusza, zbiór komórek, używany zakres komórek, itp.),
- Range (własności: adres, komórki, kolumny, wiersze, wprowadzone wartości, itp.).

## Obiekty

- Application – aplikacja (Excel),
- Workbook – pojedynczy dokument (*Zeszyt1* i *Zeszyt2*),
- Worksheet – pojedynczy arkusz (*Arkusz1-2* w *Zeszyt1* i *Arkusz1-3* w *Zeszyt2*),
- Range – pojedyncze komórki arkuszy.

## Zbiory obiektów (kolekcje)

- Workbooks – zbiór wszystkich otwartych dokumentów,
- Worksheets – zbiór wszystkich arkuszy danego dokumentu,
- Range – zbiór komórek (np. w zaznaczonym zakresie, w kolumnie lub wierszu).



## APPLICATION (EXCELL)

### WORKBOOKS (ZBIÓR ZESZYTÓW/DOKUMENTÓW)

#### WORKBOOK (ZESZYT1)

##### WORKSHEETS (ZBIÓR ARKUSZY)

###### WORKSHEET (ARKUSZ1)


###### WORKSHEET (ARKUSZ2)


#### WORKBOOK (ZESZYT2, AKTYWNY)

##### WORKSHEETS (ZBIÓR ARKUSZY)

###### WORKSHEET (ARKUSZ1)


###### WORKSHEET (ARKUSZ2, AKTYWNY)


###### WORKSHEET (ARKUSZ3)


*Uwaga: powyższy model jest niekompletny i zawiera pewne uproszczenia. Odpowiada przykładowi przedstawionemu na s. 7.*

**Application** reprezentuje całą aplikację, istnieje od chwili uruchomienia Excel-a, zawsze występuje tylko w jednym egzemplarzu.

## Wybrane własności

- **ActiveCell** – klasa Range, aktywna komórka (może być tylko jedna),
- **ActiveSheet** – klasa Worksheet, aktywny arkusz (Worksheet),
- **ActiveWorkbook** – klasa Workbook, aktywny dokument (Workbook),
- **Selection** – klasa Range, aktualnie wybrany zakres komórek,
- **UserName** – string, nazwa użytkownika,
- **Worksheets** – klasa Worksheets, kolekcja arkuszy aktualnego dokumentu.

## Wybrane metody

- **Calculate** – przelicza wszystkie otwarte dokumenty (Workbook),
- **InputBox(text)** – wyświetla okno z umożliwiające wprowadzenie wartości,
- **Quit** – zamyka program.

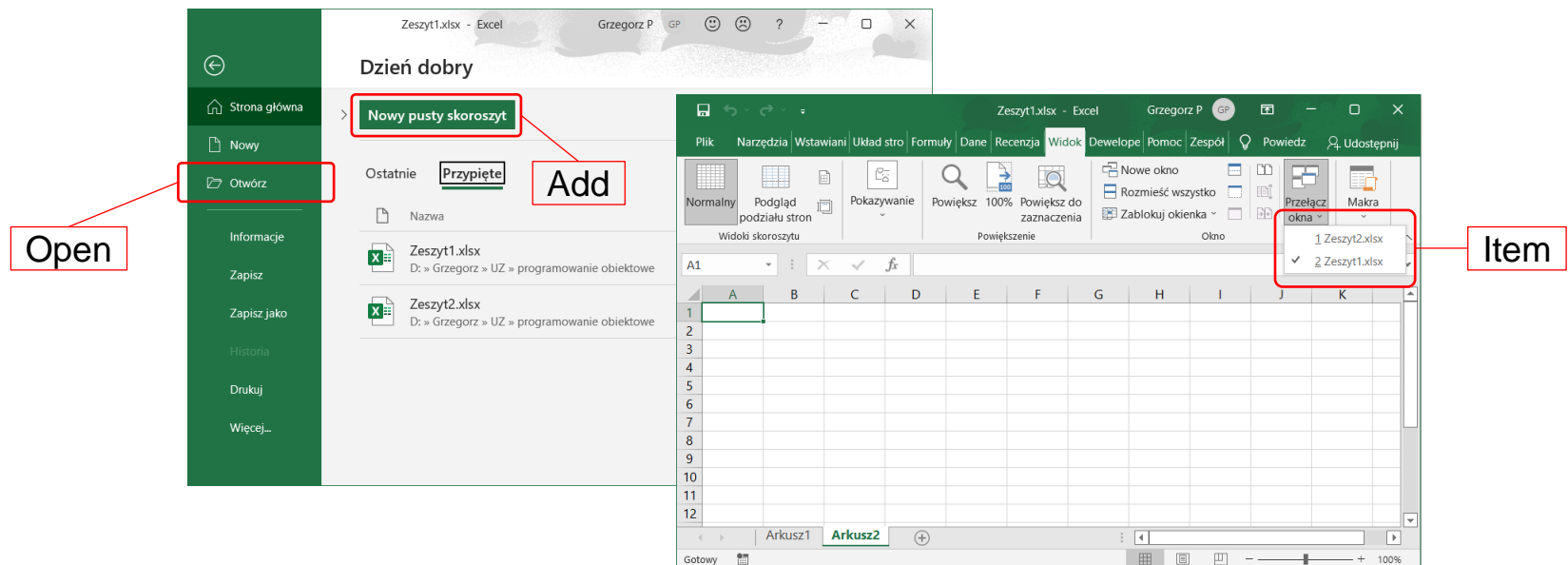
**Workbooks** reprezentuje kolekcję aktualnie otwartych dokumentów.

## Wybrane własności

- **Count** – Long, liczba otwartych dokumentów (elementów kolekcji),
- **Item(*nazwa*|*index*)** – klasa Workbook, dokumentu o określonej nazwie lub indeksie.

## Wybrane metody

- **Add** – dodaje nowy, pusty dokument (Workbook) do kolekcji,
- **Open(*nazwa*)** – otwiera dokument o określonej *nazwie*.



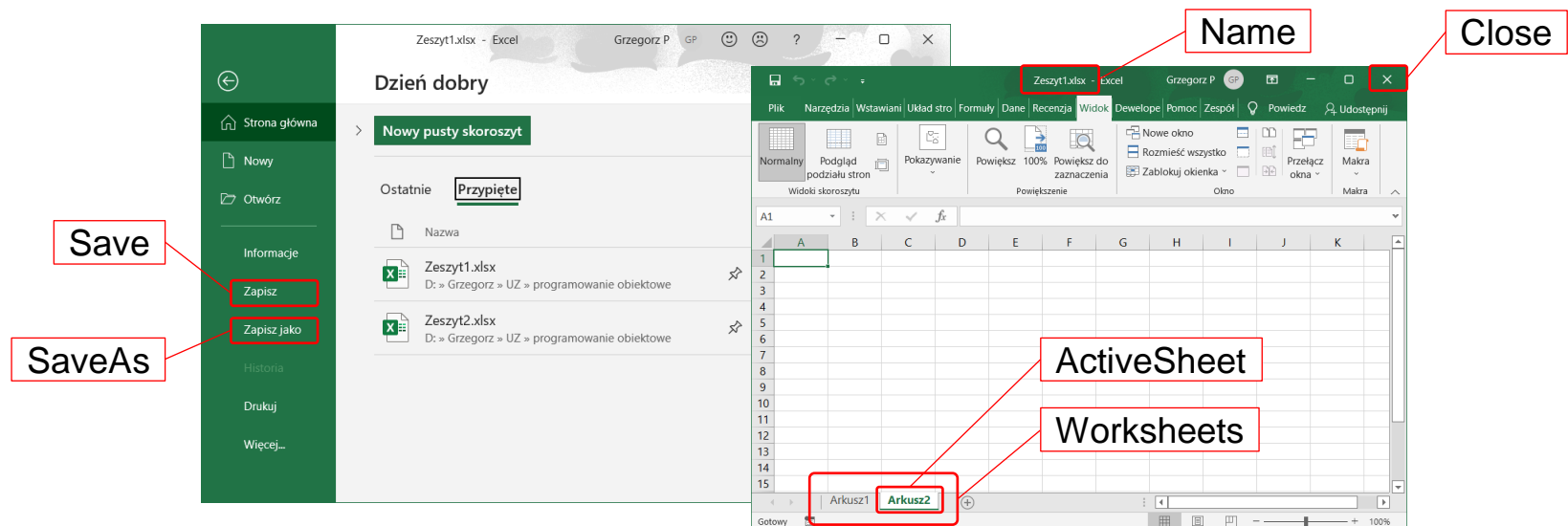
**Workbook** reprezentuje pojedynczy dokument w kolekcji Workbooks.

## Wybrane własności

- **ActiveSheet** – klasa Worksheet lub Chart, aktywny arkusz lub wykres,
- **Name, Path, FullName** – String, kolejno: nazwa, ścieżka, pełna nazwa dokumentu,
- **Worksheets** – klasa Worksheets, kolekcja arkuszy tworzących dokument.

## Wybrane metody

- **Activate** – aktywuje Workbook,
- **Close** – zamyka Workbook,
- **Save, SaveAs(nazwa)** – zapisuje Workbook pod dotychczasową lub nową nazwą.



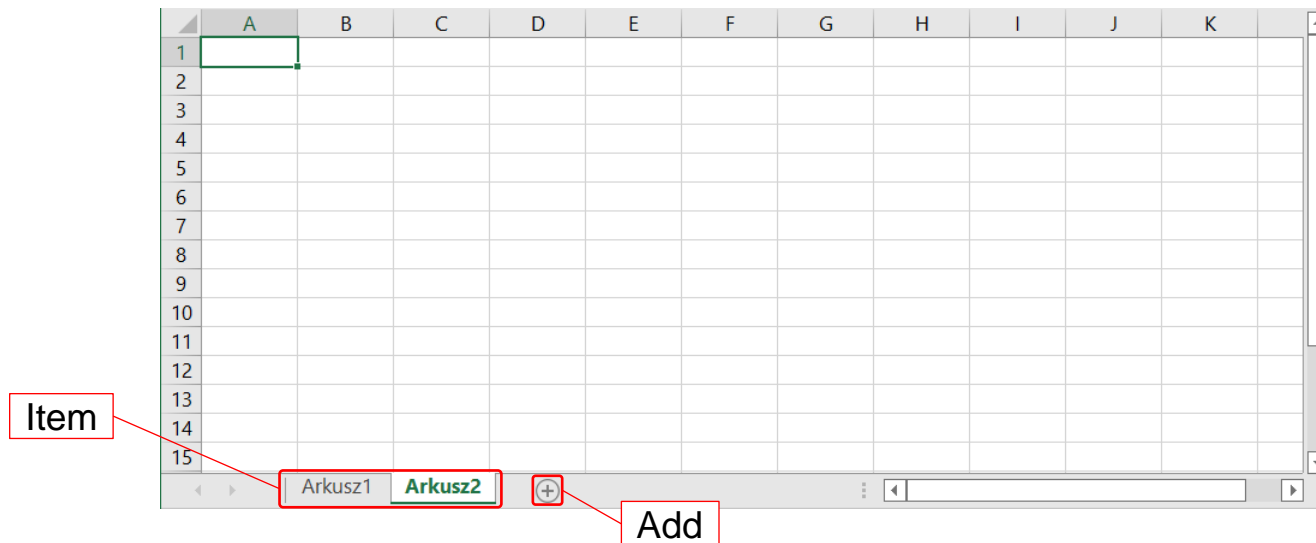
**Worksheets** reprezentuje kolekcję arkuszy lub wykresów.

## Wybrane własności

- **Count** – Long, liczba elementów kolekcji,
- **Item**(*nazwa|index*) – klasa Worksheet, element o określonej nazwie lub indeksie.

## Wybrane metody

- **Add**(*przed, po*) – dodaje nowy, pusty arkusz (Worksheet) do kolekcji *przed* lub *po* arkuszu istniejącym.



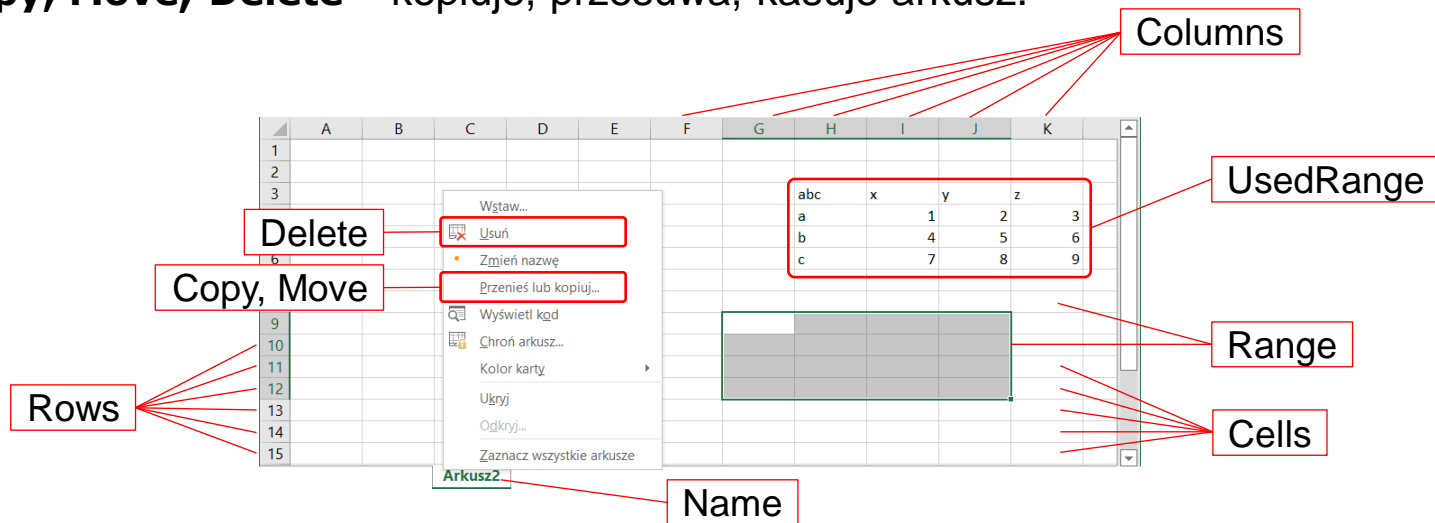
**Worksheet** reprezentuje pojedynczy arkusz w kolekcji Worksheets

## Wybrane własności

- **Cells**(*indeks*) – klasa Range, kolekcja komórek arkusza,
- **Columns, Rows** – klasa Range, kolekcje kolumn i wierszy,
- **Name, Index** – nazwa (String) lub numer (Long) arkusza w kolekcji,
- **Range**(*zakres*) – klasa Range, kolekcja komórek z wybranego zakresu,
- **UsedRange** – klasa Range, używany zakres arkusza.

## Wybrane metody

- **Activate** – aktywuje (wybiera) arkusz,
- **Copy, Move, Delete** – kopiuje, przesuwa, kasuje arkusz.



# Obiekt Range – własności

---

**Range** reprezentuje pewien zakres komórek (cały arkusz, wybrane kolumny, wybrane wiersze, wskazany zakres, itp.).

## Wybrane własności

- **Address** – String, adres zakresu,
- **Cells** – klasa Range, kolekcja komórek zakresu,
- **Columns, Rows** – klasa Range, kolekcje kolumn i wierszy w zakresie,
- **Count, Column, Row** – Long, liczba komórek, numer pierwszej kolumny i wiersza,
- **End(*kierunek*)** – komórka z początku/końca zakresu (*xlDown, xlToLeft, xlToRight, xlUp*),
- **EntireRow** – klasa Range, cały wiersz (wiersze) zawierające zakres,
- **Font** – czcionka używana w zakresie (obiekt, zawiera nazwę, kolor, rozmiar, itd.),
- **Formula, FormulaR1C1** – formuła zawarta w komórce w notacji A1 lub R1C1,
- **Offset(*w, k*)** – klasa Range, zakres przesunięty o *w* wierszy i *k* kolumn,
- **Resize(*w, k*)** – klasa Range, zakres o *w* wierszach i *k* kolumnach,
- **Text** – String, zawartość zakresu w postaci tekstowej uwzględniającej formatowanie,
- **Value** – Variant, wartość wprowadzona w zakresie.

*Uwaga: własności Formula..., Text i Value zwracają wartość pustą w przypadku zakresu obejmującego wiele komórek.*

**Range** reprezentuje pewien zakres komórek (cały arkusz, wybrane kolumny, wybrane wiersze, wskazany zakres, itp.).

## Wybrane metody

- **Activate** – aktywuje zakres (należy używać do wyboru komórki arkusza),
- **Clear** – czyści zakres (wartości, formuły, formaty),
- **ClearContents** – czyści zawartość zakresu (tylko wartości i formuły),
- **Copy(zakres), Cut(zakres), PasteSpecial** – kopiuje/wycina/wkleja zakres do innego zakresu lub schowka, jeżeli zakres docelowy nie został określony,
- **Delete(przesunięcie)** – usuwa zakres, pozostałe komórki są przenoszone w kierunku określonym przez *przesunięcie* (xlShiftToLeft, xlShiftUp),
- **FunctionWizard** – wyświetla okienko asystenta wstawiania funkcji,
- **Insert(przesunięcie)** – wstawia zakres, pozostałe komórki są przenoszone w kierunku określonym przez *przesunięcie* (xlShiftToRight, xlShiftDown),
- **Merge** – łączy komórki zakresu,
- **Select** – wybiera (zaznacza) zakres,
- **Speak** – odczytuje na głos zawartość komórek w zakresie.



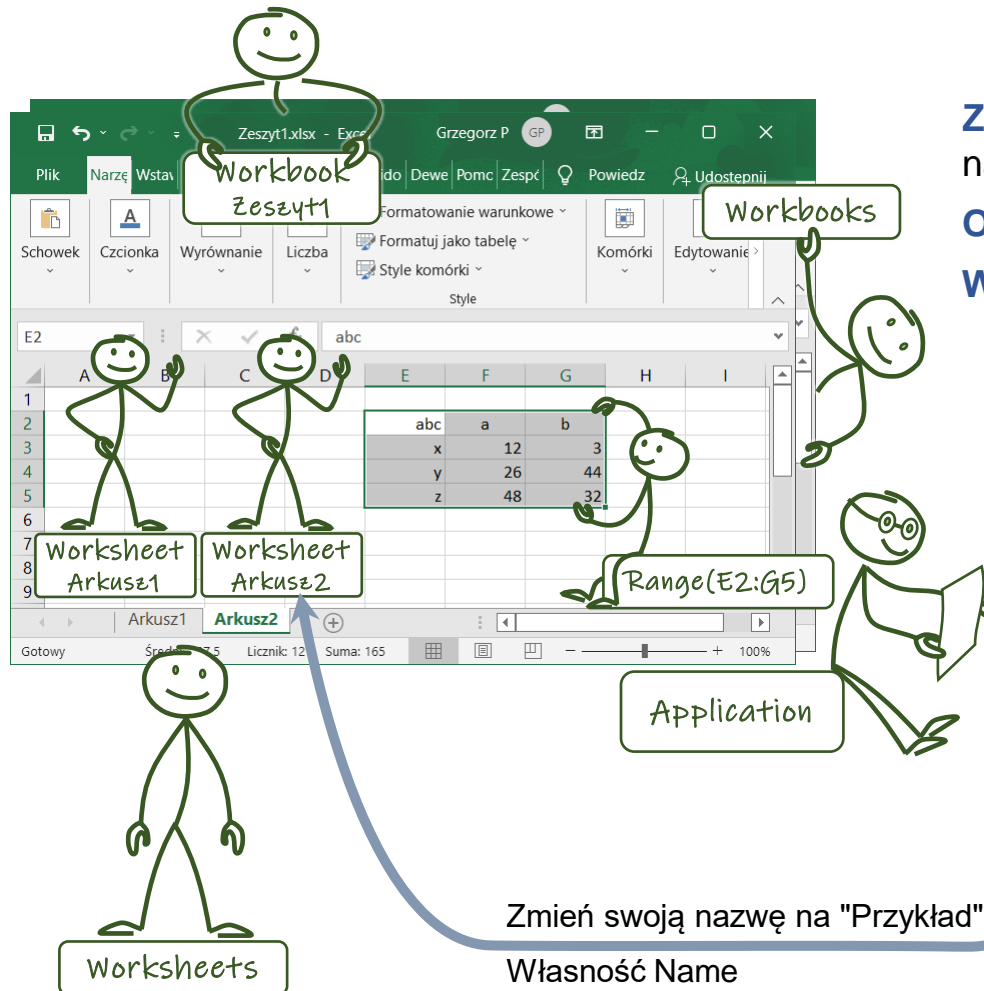
**Object Browser** – narzędzie dostępne w edytorze VBA, które pozwala na przeglądanie dostępnych klas, ich własności i metod.

The screenshot shows the Object Browser window with the following components and annotations:

- Poszukiwany element**: Points to the search input field containing "Range".
- Wyniki poszukiwania**: Points to the "Search Results" table listing classes like AllowEditRange, AllowEditRanges, Range, Ranges, and ShapeRange.
- Lista elementów (klasy, moduły, typy)**: Points to the "Classes" list on the left, which includes PublishObject, PublishObjects, Queries, QueryTable, QueryTables, QuickAnalysis, Range, Ranges, RecentFile, RecentFiles, RectangularGrac, ReflectionForma, Research, RoutingSlip, RTD, and Ruler2.
- Składniki elementu (własności, metody, itp.)**: Points to the "Members of 'Range'" list, which includes Addindent, Address, AddressLocal, AllowEdit, Application, Areas, Borders, Cells, Characters, Column, Columns, ColumnWidth, Comment, CommentThreaded, Count, and Countl arae.
- obiekty globalne**, **klasy**, **moduły**, **typy**: Labels with icons pointing to the "Classes" list.
- własności**, **metody**, **zdarzenia**, **składowe typów**: Labels with icons pointing to the "Members of 'Range'" list.
- Składnia**: Points to the context menu for the "Cells" member, which includes options like Copy, View Definition, Find Whole Word Only, Group Members, Show Hidden Members, References..., Properties..., Help, Dockable, and Hide.

The "Property Cells As Range" section at the bottom shows: read-only, Member of Excel.Range.

# Idea pracy z obiektami – przykład I



**Zadanie:** Zmienić nazwę arkusza "Arkusz2" na "Przykład".

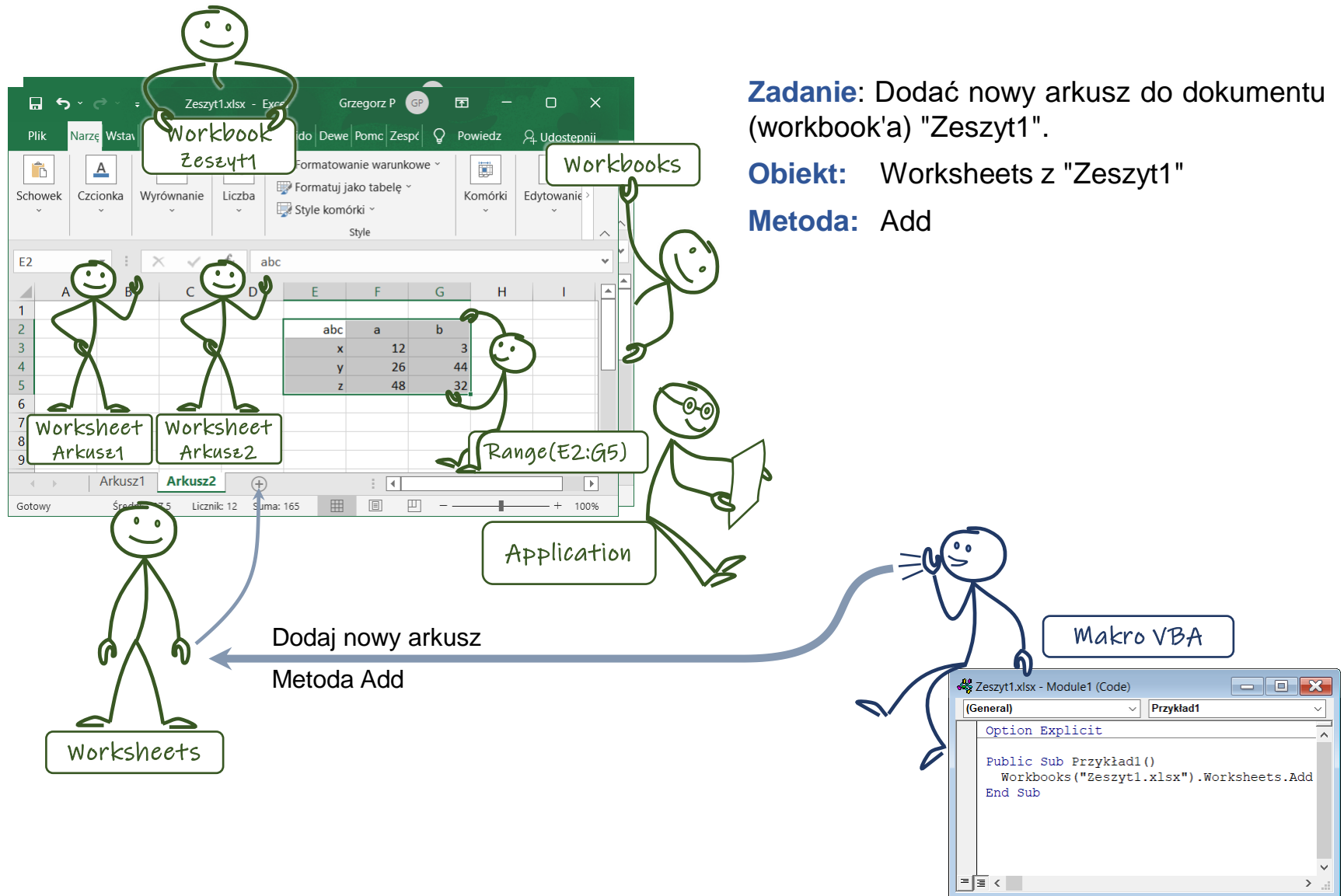
**Obiekt:** Worksheet Arkusz2.

**Własność:** Name.

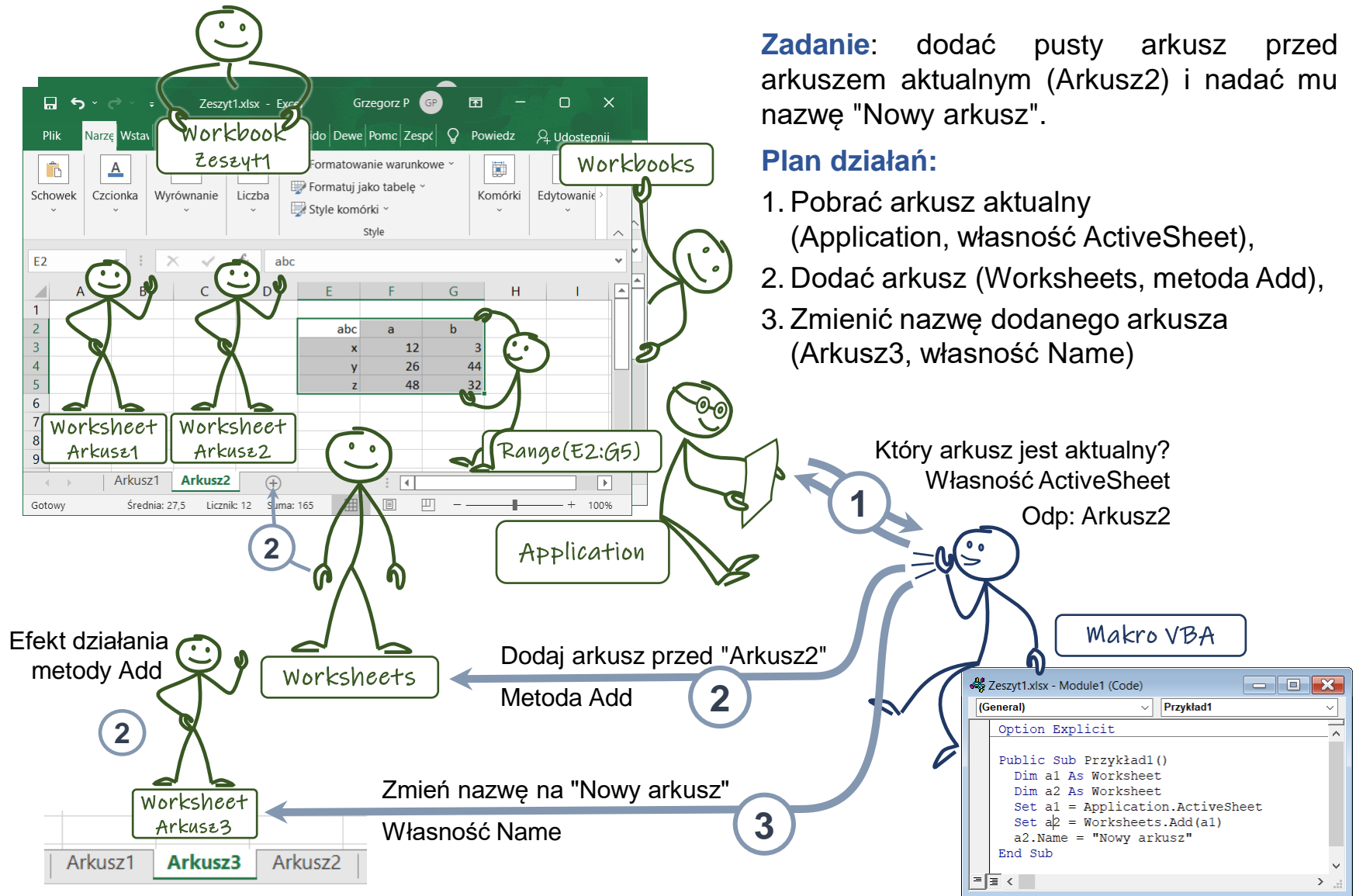
```
Option Explicit

Public Sub Przykład1()
    Arkusz2.Name = "Przykład"
End Sub
```

# Idea pracy z obiektami – przykład II



# Idea pracy z obiektami – przykład III



**Zadanie:** dodać pusty arkusz przed arkuszem aktualnym (Arkusz2) i nadać mu nazwę "Nowy arkusz".

## Plan działań:

1. Pobrać arkusz aktualny (Application, własność ActiveSheet),
2. Dodać arkusz (Worksheets, metoda Add),
3. Zmienić nazwę dodanego arkusza (Arkusz3, własność Name)

Który arkusz jest aktualny?  
Własność ActiveSheet  
Odp: Arkusz2

Dodaj arkusz przed "Arkusz2"  
Metoda Add

Zmień nazwę na "Nowy arkusz"  
Własność Name

Makro VBA

# Podstawowe zasady programowania w VBA

---

- ❑ Wielkość liter nie jest rozróżniana (dotyczy wszystkich elementów programu: nazw instrukcji, elementów predefiniowanych i definiowanych przez użytkownika),
- ❑ Nazwy definiowanych elementów muszą zaczynać się od litery, nie mogą zawierać znaków specjalnych (spacja , \* ; . = itp.),
- ❑ Pojedyncza instrukcja znajduje się w jednym wierszu programu,
- ❑ W celu podziału instrukcji na kilka wierszy należy zastosować symbol kontynuacji „ \_ ” (spacja i znak podkreślenia),
- ❑ W jednym wierszu można umieścić kilka instrukcji rozdzielając je znakiem „:” (zmniejsza czytelność kodu, należy stosować tylko w szczególnych przypadkach),
- ❑ Puste wiersze są pomijane, mogą być wykorzystane do podziału kodu na mniejsze bloki (istotne z punktu widzenia czytelności programu),
- ❑ Słowa kluczowe języka (nazwy instrukcji, elementy nagłówek, itp.) są zastrzeżone i nie mogą być wykorzystane jako nazwy elementów definiowanych w programie,
- ❑ Tekst rozpoczynający się znakiem apostrofu jest komentarzem, nie wpływa na sposób wykonania programu.

## Odwołanie do własności i metody obiektu

```
nazwa_obiektu.nazwa_własności  
nazwa_obiektu.nazwa_metody
```

## Przykłady

`Application.ActiveSheet` – odwołanie do aktywnego arkusza

`Application.ActiveSheet.Name` – odwołanie do nazwy aktywnego arkusza

`Application.Worksheets.Add` – wywołanie metody „Add” (nowy arkusz)

## Odwołanie do kolekcji

```
kolekcja(indeks)
```

indeks określa element, jego postać zależy od rodzaju kolekcji.

## Przykłady

`Application.Workbooks(1)` – odwołanie do pierwszego dokumentu

`Application.Workbooks("Z1.xlsx")` – odwołanie do dokumentu „Z1.xlsx”

`Application.Workbooks("Z2.xlsx").Worksheets("Arkusz1")` – odwołanie do arkusza „Arkusz1” w dokumencie „Z2.xlsx”

**Obiekty domyślne** mogą być pomijane przy odwołaniach do ich własności obiektowych.

## Obiekty domyślne w VBA

- ❑ Obiekt **Application** zawsze jest obiektem domyślnym i może być pominięty w większości odwołań, stąd odwołania postaci:

```
Application.Selection
```

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

mogą być skrócone do:

```
Selection
```

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

- ❑ Obiekty **Workbook** i **Worksheet** domyślnie odwołują się do aktywnego dokumentu oraz arkusza, stąd odwołanie postaci:

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

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

mogą być skrócone do:

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

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

**Typ danych** określa zakres wartości, które może przyjmować dany element programu (np. własność obiektu). Każdy element przechowujący dane ma określony typ.

## Predefiniowane typy danych VBA

- **Byte** – bajt, liczba całkowita 0 .. 255
- ➔ ▪ **Integer** – liczba całkowita -32 768 .. 32 767
- **Long** – długa liczba całkowita -2 147 483 648 .. 2 147 483 647
- ➔ ▪ **Single** – liczba rzeczywista (pojedyncza precyzja , 8 cyfr) 1.40e-45 .. 3.40e38
- **Double** – liczba rzeczywista (podwójna precyzja, 16 cyfr) 4.94e-324 .. 1.79e308
- **Currency** – waluta (4 miejsca po przecinku)  
-922 337 203 685 477.5808 .. 922 337 203 685 477.5807
- ➔ ▪ **Boolean** – logiczny, dozwolone dwie wartości: True i False
- **Date** – data/czas 1 stycznia 100 roku .. 31 grudnia 9999
- ➔ ▪ **String** – ciąg znaków, wartości podawane w cudzysłowach, maksymalnie 2 bln.
- **Object** – wskaźnik na dowolny obiekt
- ➔ ▪ **Variant** – dowolny typ, może przechowywać każdą z powyższych wartości

*Uwaga:* separatorem części ułamkowej w kodzie programu jest kropka

$1.756e02=1.756 \cdot 10^2=175.6$ ,     $2.5e-03=2.5 \cdot 10^{-3}=0.0025$



- 127 – liczba całkowita (**Byte**, **Integer** lub **Long**)
- 5786 – liczba całkowita (**Integer** lub **Long**)
- "127" – łańcuch znakowy (**String**)
- 12.576 – liczba rzeczywista (**Single** lub **Double**, ewentualnie **Currency**)
- True – wartość logiczna (**Boolean**)
- "False" – łańcuch znakowy (**String**)
- 253,8 – nieprawidłowa wartość (przecinek zamiast kropki)
- 25.39e-11 – liczba rzeczywista (**Single** lub **Double**, ewentualnie **Currency**)
- #20/5/2019 8:20:00 PM# – data i czas (**Date**)
  
- ActiveCell – własność Application, obiekt (**Object**, dokładnie Range)
- Selection – własność Application, obiekt (**Object**, dokładnie Range)
- Worksheets (1) – własność Application, obiekt (**Object**, dokładnie Worksheet)
- ActiveCell.Value – zależne od wartości w komórce, **Variant**

# Instrukcja przypisania, zmiana własności

---

## Instrukcja przypisania

```
nazwa_elementu = wartość
```

Instrukcja przypisania nadaje pewnemu elementowi (np. własności obiektu) określoną wartość. **Typ przypisywanej wartości musi być zgodny z typem elementu.**

## Przykład 1 – ustawienie wartości w aktywnej komórce arkusza

```
ActiveCell.Value = 125
```

```
ActiveCell.Value = "Suma"
```

## Przykład 2 – zmiana czcionki w komórce "C7"

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

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

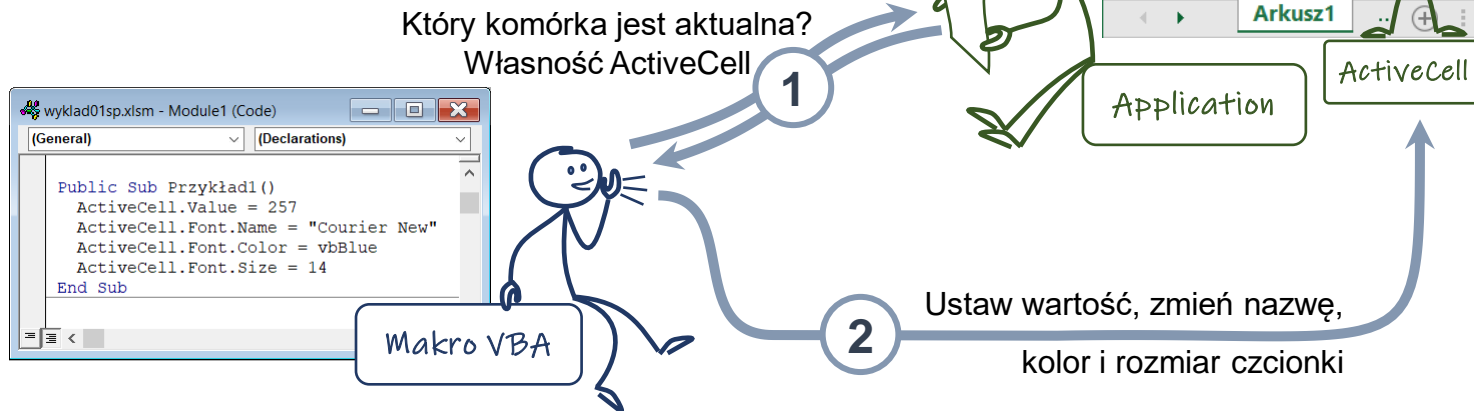
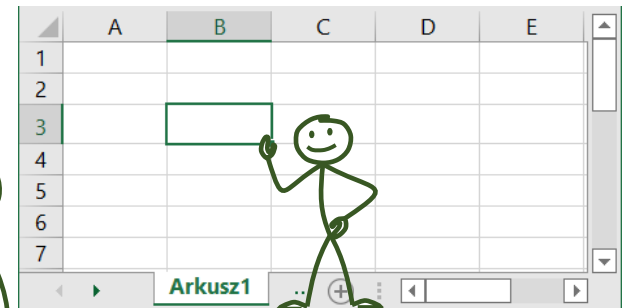
```
Range("C7").Font.Name = "Courier New"
```

*Uwaga 1.:* Font jest własnością obiektową, zawiera własności Color, Name, Size, itp.

*Uwaga 2.:* Kolor można określić używając funkcji RGB, podając jako parametry nasycenie składowej czerwonej, zielonej i niebieskiej lub posługując się zdefiniowanymi stałymi o nazwach zgodnych ze schematem vb<nazwa\_koloru>.

Modyfikacja aktywnej komórki: ustawienie wartości 257, zmiana czcionki na Courier New, niebieska, rozmiar 14pt.

```
Public Sub Przykład1()  
    ActiveCell.Value = 257  
    ActiveCell.Font.Name = "Courier New"  
    ActiveCell.Font.Color = vbBlue  
    ActiveCell.Font.Size = 14  
End Sub
```

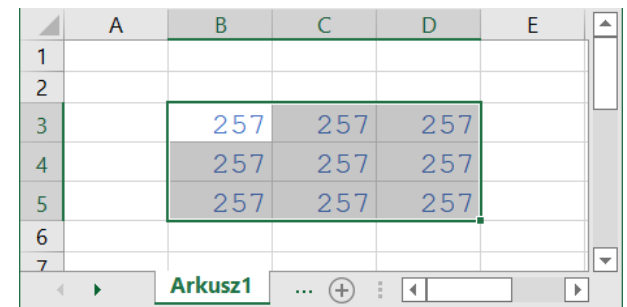


**Uwaga:** ActiveCell jest własnością obiektu Application, ponieważ jest to obiekt domyślny w odwołaniu jego nazwa może być pominięta.

Kody dostępne na stronie przedmiotu

Modyfikacja wszystkich komórek w aktualnie wybranym zakresie: ustawienie wartości 257, zmiana czcionki na Courier New, niebieska, rozmiar 14pt.

```
Public Sub Przykład2()  
    Selection.Value = 257  
    Selection.Font.Name = "Courier New"  
    Selection.Font.Color = vbBlue  
    Selection.Font.Size = 14  
End Sub
```



*Uwaga:* zmiana własności obiektu typu Range (zakres) powoduje zmianę odpowiednich własności wszystkich komórek w tym zakresie.

Przepisanie zawartości zakresu A1:A5 z Arkusz1 do zakresu C3:E7 na Arkusz2, aktywacja Arkusza2.

```
Public Sub Przykład3()  
    Worksheets("Arkusz2").Range("C3:E7").Value = _  
    Worksheets("Arkusz1").Range("A1:A5").Value  
    Worksheets("Arkusz2").Activate  
End Sub
```

*Uwaga:* jeżeli pojedyncza instrukcja jest zapisywana w kilku wierszach należy użyć symbolu kontynuacji wiersza: " \_" (spacja i znak podkreślenia).

Przepisanie zawartości aktualnego zakresu o dwie kolumny w prawo

```
Public Sub Przykład4()  
    Selection.Offset(0, 2).Value = Selection.Value  
End Sub
```

**With** <obiekt>

*odwołania do własności i metod*

**End With**

Wewnątrz instrukcji **With** wszystkie odwołania do własności i metod obiektu mogą być skrócone do postaci:

.nazwa\_własności

.nazwa\_metody

## Przykład

Procedury ustawiają wartość aktywnej komórki na 257, zmieniają kolor i rozmiar czcionki.

```
Public Sub Przykład1a()  
    ActiveCell.Value = 257  
    ActiveCell.Font.Color = vbBlue  
    ActiveCell.Font.Size = 14  
End Sub
```

```
Public Sub Przykład1b()  
    With ActiveCell  
        .Value = 257  
        .Font.Color = vbBlue  
        .Font.Size = 14  
    End With  
End Sub
```

Do wyświetlania informacji w formie okna komunikatu służy instrukcja:

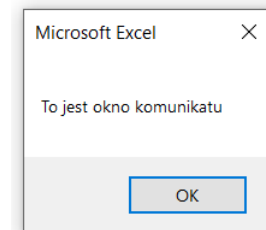
```
MsgBox <treść>
```

gdzie <treść> jest informacją, która powinna pojawić się w oknie.

## Przykłady

Komunikat tekstowy

```
Public Sub Przykład5()  
    MsgBox "To jest okno komunikatu"  
End Sub
```



Nazwa użytkownika, na którego został zarejestrowany program

```
Public Sub Przykład6()  
    MsgBox Application.UserName  
End Sub
```

*Uwaga:* przy odwołaniu do własności `UserName` nie można pominąć obiektu `Application` (`UserName` nie jest własnością obiektową).

*Kody dostępne na stronie przedmiotu*

## Operatory arytmetyczne

+	dodawanie	\	dzielenie całkowite
-	odejmowanie	mod	dzielenie modulo
*	mnożenie	^	potęgowanie
/	dzielenie	( )	grupowanie działań

Operacje wykonywane są od lewej do prawej z zachowaniem priorytetów (po pierwsze potęgowanie, następnie mnożenie i dzielenie, później dodawanie i odejmowanie). Nie można pomijać symboli działań, nawiasy okrągłe zmieniają kolejność operacji.

*Uwaga:*  $x + 5/3*y = x + \frac{5}{3}y$ ,  $(x + 5)/(3*y) = \frac{x+5}{3y}$

## Funkcje arytmetyczne (przykłady)

Abs (x) – wartość bezwzględna,

Log (x) – logarytm naturalny,

Round (x) – zaokrąglenie bankowe,

Sin (x), Cos (x), Tan (x) – sinus, cosinus, tangens,

Sqr (x) – pierwiastek kwadratowy.



Procedura pobiera wartość aktualnej komórki, powiększa ją o dwa, wynik zapisuje w komórce znajdującej się na prawo (ten sam wiersz, następną kolumną) i zaznacza ją.

## Odwołania

- Aktualnie wybrana komórka: `ActiveCell`
- Odwołanie do wartości aktualnej komórki: `ActiveCell.Value`
- Odwołanie do wartości komórki sąsiedniej: `ActiveCell.Offset(0, 1).Value`
- Zaznaczenie komórki sąsiedniej: `ActiveCell.Offset(0, 1).Activate`

```
Public Sub InkrementujWPrawo()
```

```
    With ActiveCell
```

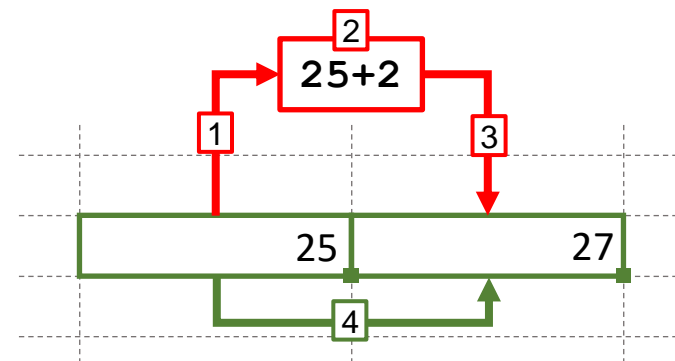
```
        3.Offset(0, 1).Value = 1.Value + 2
```

```
        4.Offset(0, 1).Activate
```

```
    End With
```

```
End Sub
```

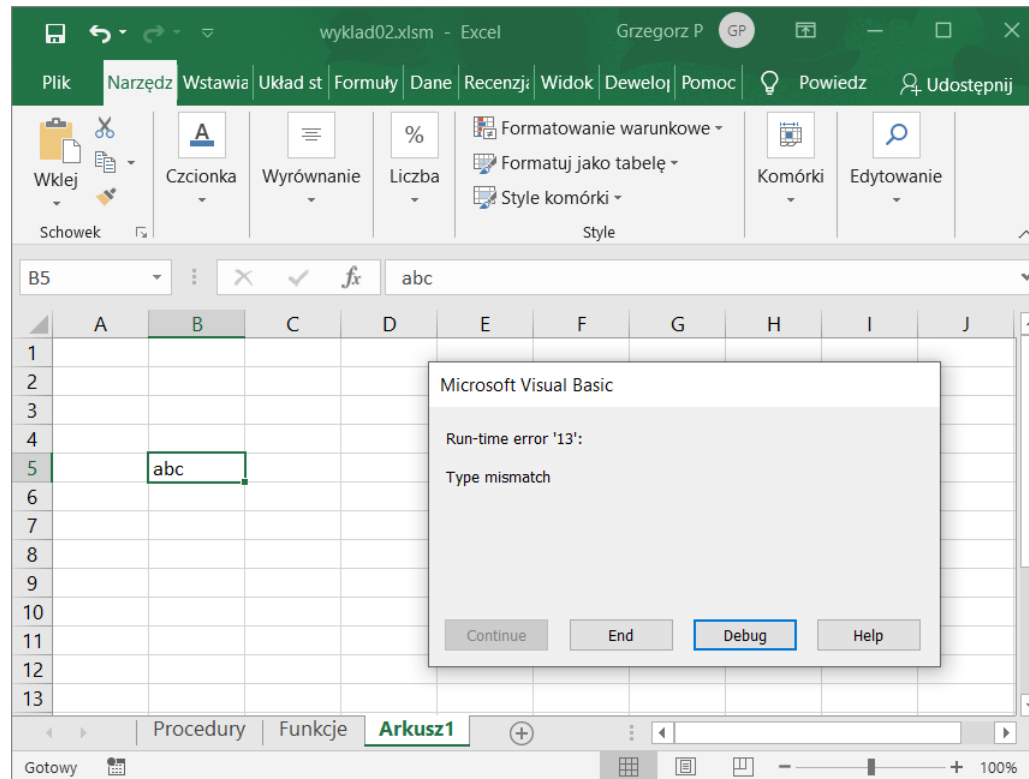
*Kod dostępny na stronie przedmiotu*



Operacje arytmetyczne są zdefiniowane tylko dla wartości liczbowych (liczba całkowita, rzeczywista, waluta lub data), więc instrukcja:

```
ActiveCell.Offset(0, 1).Value = ActiveCell.Value + 2
```

zgłasza błąd, jeżeli aktywna komórka zawiera tekst (wartość String).



**Zmienna** – element języka programowania używany do przechowywania danych. Podczas pracy programu wartości zmiennej mogą ulegać modyfikacji.

**Stała** – symbol reprezentujący pewną niezmienną wartość, która nie może być zmodyfikowana podczas pracy programu.

## Schemat deklaracji zmiennej

```
Dim <nazwa> As typ
```

## Schemat deklaracji stałej

```
Const <nazwa> [As typ] = wartość
```

<nazwa> musi być unikalną (nie mogą istnieć dwa elementy o takiej samej nazwie), powinna spełniać warunki opisane w punkcie 2 na s.1-23.

## Przykłady

```
Dim x As Integer
```

```
Dim komórka As Range
```

```
Const VAT As Single = 0.23
```

*Uwaga:* Deklaracja zmiennych nieobiektyowych nie jest wymagana. Niezadeklarowana zmienna otrzymuje typ **Variant**. Deklaracja zmiennych może być wymuszona przez umieszczenie dyrektywy **Option Explicit** na początku modułu.

## Ustawienie wartości zmiennej nieobiektovej

*<nazwa> = wyrażenie*

## Ustawienie wartości zmiennej obiektowej

**Set** *<nazwa> = wyrażenie*

*<nazwa>* jest nazwą zmiennej

*wyrażenie* jest dowolnym wyrażeniem VBA (w tym również wartością stałą, nazwą zmiennej, itp.) o typie zgodnym z typem zmiennej.

## Przykłady

```
Dim x As Integer
```

```
Dim y As Integer
```

```
Dim k As Range
```

```
x = 5
```

```
y = 29 + x ^ 2
```

```
Set k = Worksheets("Arkusz5").Range("C3:E5")
```

Procedura zamienia miejscami wartość komórki aktywnej i komórki znajdującej się na prawo od niej (ten sam wiersz kolejna kolumna).

## Algorytm

1. Odczytaj wartość komórki aktywnej do zmiennej
2. Zapisz w komórce aktywnej wartość komórki po prawej
3. Zapisz w komórce po prawej wartość zapisaną w zmiennej

```
Public Sub ZamieńWPrawo()
```

```
    Dim x As Integer
```

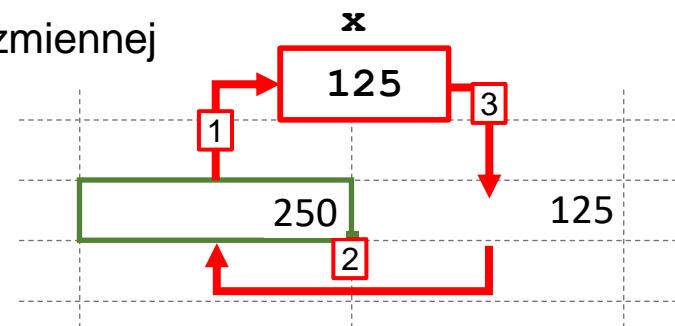
```
    x = ActiveCell.Value
```

```
    ActiveCell.Value = ActiveCell.Offset(0, 1).Value
```

```
    ActiveCell.Offset(0, 1).Value = x
```

```
End Sub
```

*Kod dostępny na stronie przedmiotu*



*Uwaga: Procedura działa prawidłowo tylko dla wartości typu **Integer**.*

*Do przemyślenia (1): Zamiana dowolnych wartości.*

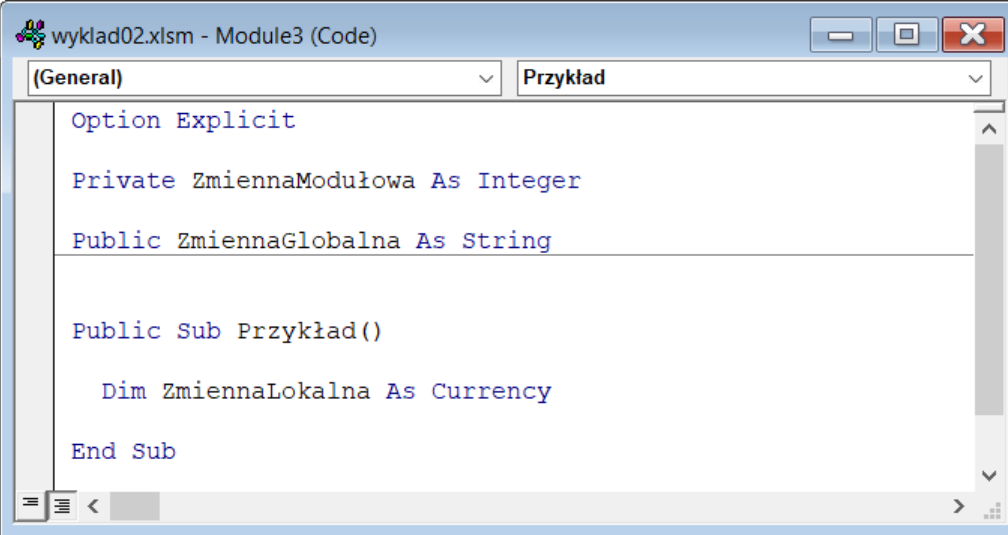
*Do przemyślenia (2): Zamiana całego zakresu.*

# Czas życia i zasięg zmiennych

**Zmienna lokalne** – zmienna zadeklarowana wewnątrz procedury VBA przy pomocy **Dim**. Istnieje tylko podczas wykonania procedury, jest usuwana po jej zakończeniu.

**Zmienna modułowa** – zmienna zadeklarowana na poziomie modułu (przed pierwszą procedurą) przy pomocy **Dim** lub **Private** (równoważne, ale zalecane **Private**). Jest dostępna we wszystkich procedurach modułu, istnieje przez cały czas wykonania programu.

**Zmienna globalna** – zadeklarowana na poziomie modułu przy pomocy **Public**. Jest dostępna we wszystkich modułach, istnieje przez cały czas wykonania programu.



```
Option Explicit

Private ZmiennaModułowa As Integer

Public ZmiennaGlobalna As String

Public Sub Przykład()

    Dim ZmiennaLokalna As Currency

End Sub
```