

CS2034B / DH2144B

Data Analytics: Principles and Tools



Western
UNIVERSITY • CANADA

Week 5

Programming Part 1



Getting Started With Visual Basic for Applications (VBA)

Example: Daily Interest

Sometimes formulas can get large enough to be error prone.

We may also want to make it easier to reuse a formula that we frequently need.



Example: Daily Interest

- Suppose we have interest compounded daily at a given annual rate r_a .
- How much interest accumulates after a certain number of days d ?



Example: Daily Interest

After d days a principal amount P at annual rate r_a will earn interest equal to

$$\text{Interest} = ((1 + r_a)^{d/365} - 1)P$$



Example: Daily Interest

After d days a principal amount P at annual rate r_a will earn interest equal to

$$\text{Interest} = ((1 + r_a)^{d/365} - 1)P$$

We can write this as an Excel formula, e.g.

$$=((1+B1)^{(B2/365)}-1)*B3$$

SUM		✕ ✓ f _x		=((1+B1)^(B2/365)-1)*B3	
	A	B	C	D	E
1	Annual Rate	6.50%	5.50%	2.50%	5.00%
2	Number of Days	33	103	44	365
3	Principal	\$1,000.00	\$33,950.00	\$3.95	\$100.00
4					
5	Interest	5)-1)*B3	\$516.84	\$0.01	\$5.00



Example: Daily Interest

- If a formula such as this appears in many cells, then it is susceptible to hard-to-spot editing errors.
- We would like to give formulas such as these short names and re-use them by that name, e.g.

=DailyInterest(B1, B2, B3)

- We do this by programming new Excel functions in VBA.



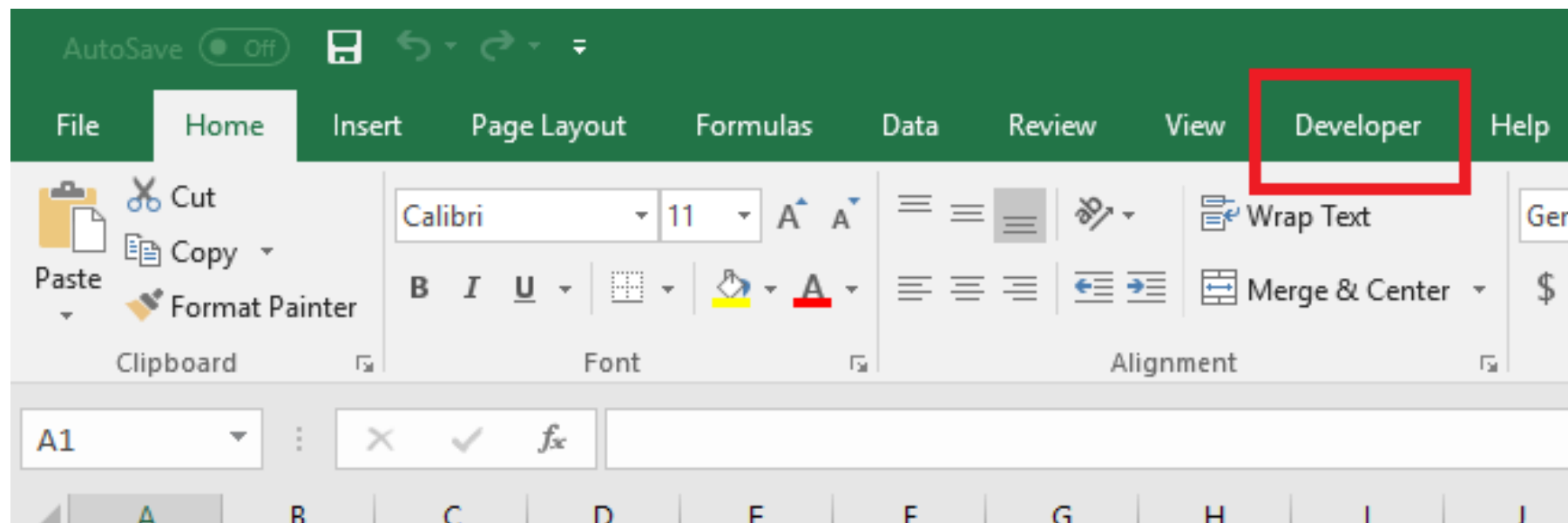
VBA

VBA is the programming language used within Excel to develop functions, subroutines, and macros. It stands for **Visual Basic for Applications**



Exposing the Developer Tab

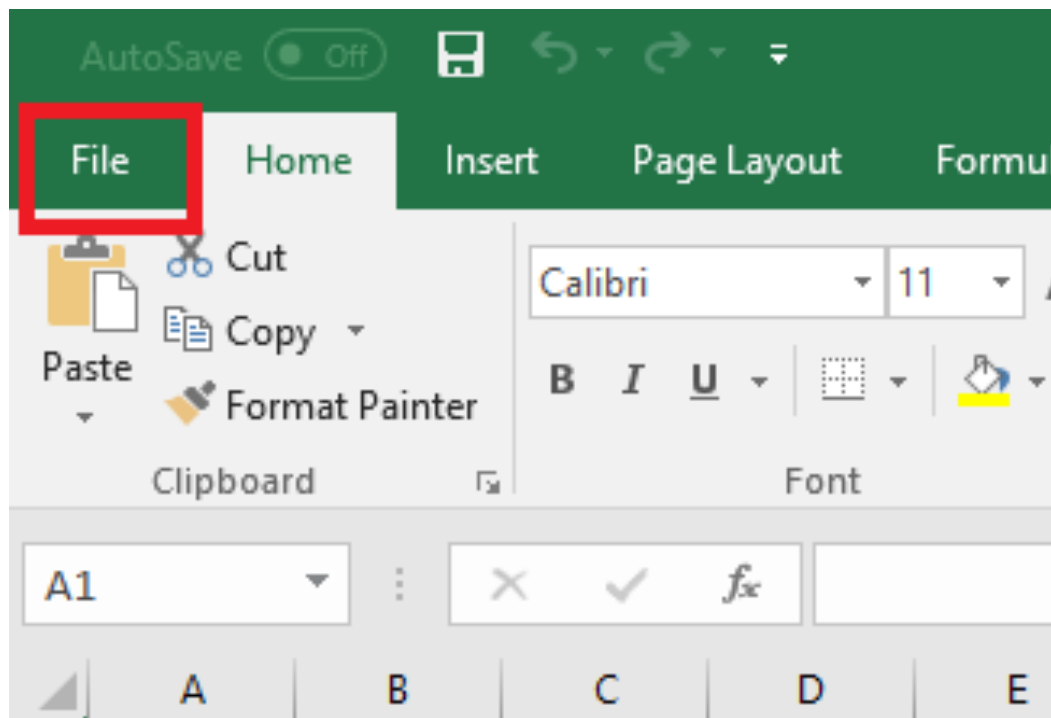
To program with VBA in Excel, we first need access to the developer tab:



Exposing the Developer Tab

In most cases, this tab is not displayed by default and we have to expose it as follows:

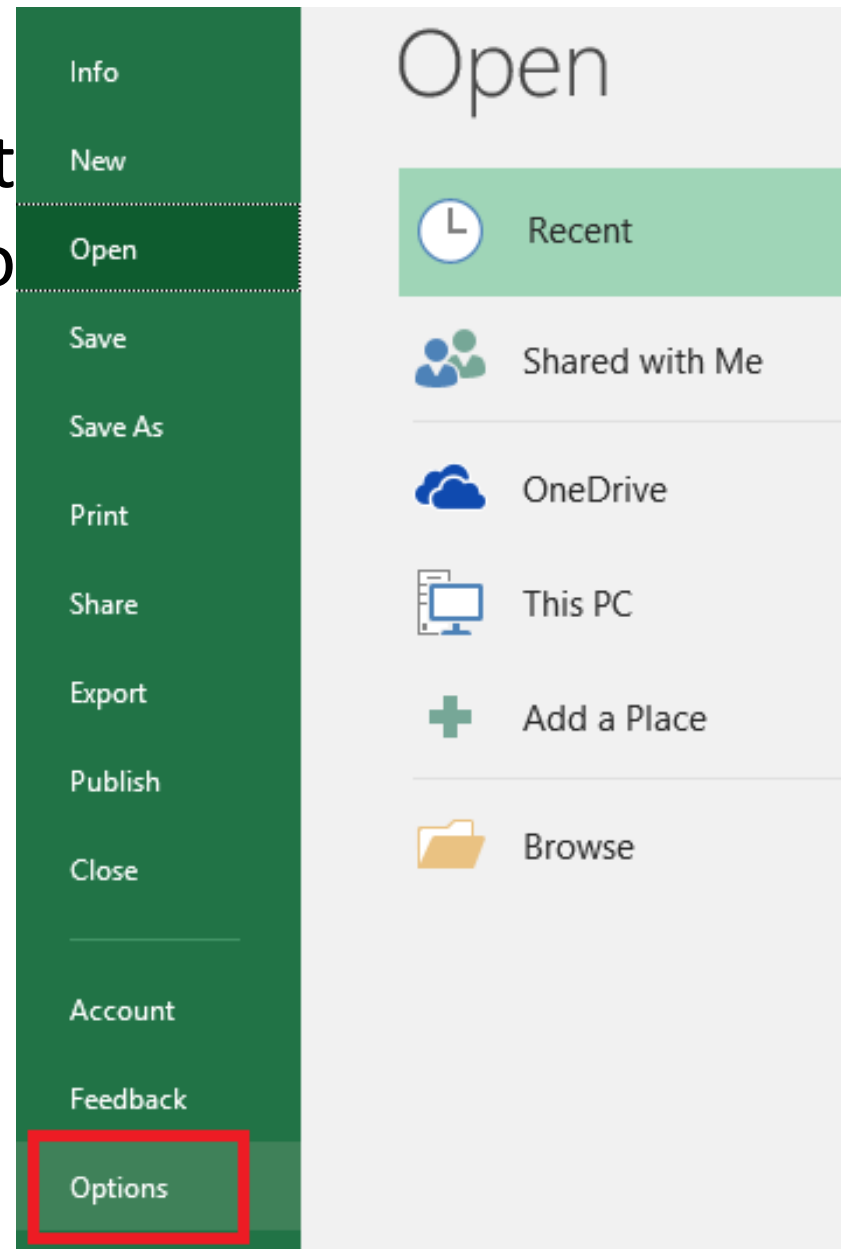
1. Click the file tab



Exposing the Developer Tab

In most cases, this tab is not visible, so we have to expose it as follows:

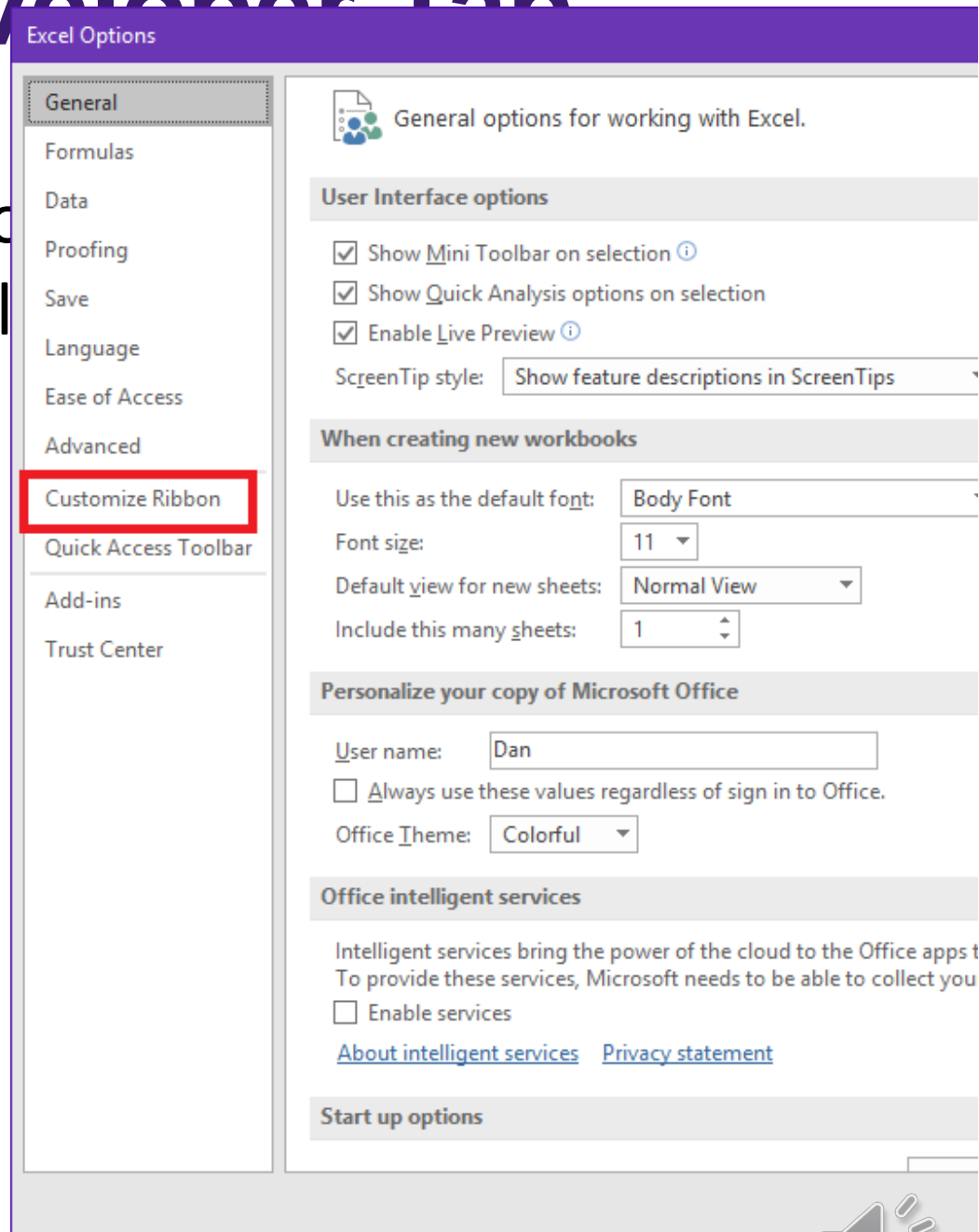
1. Click the file tab
2. Click options



Exposing the Developer Tab

In most cases, this tab is not visible, so we have to expose it as follows:

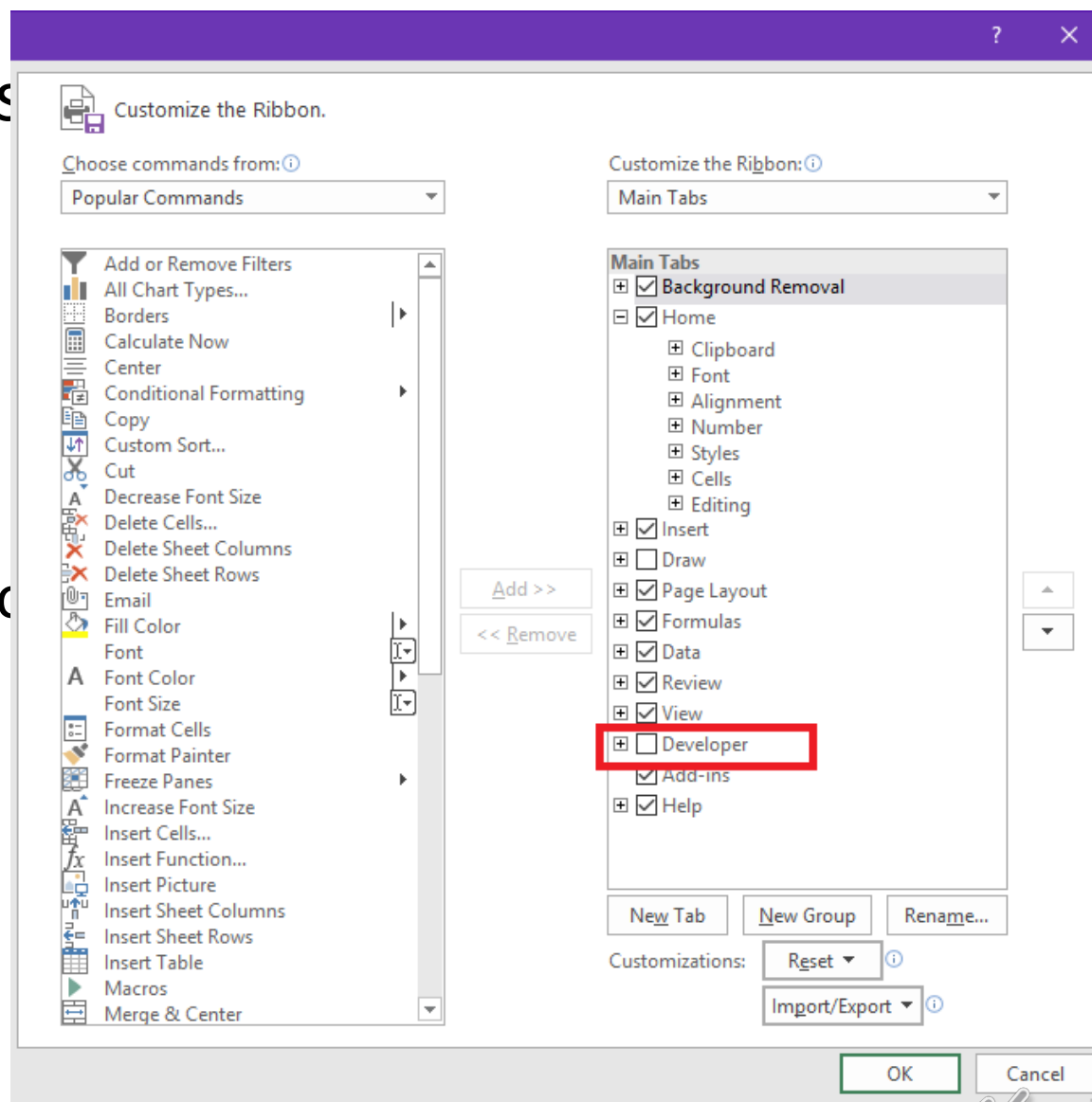
1. Click the File tab
2. Click Options
3. Select "Customize Ribbon"



Exposing the Developer Tab

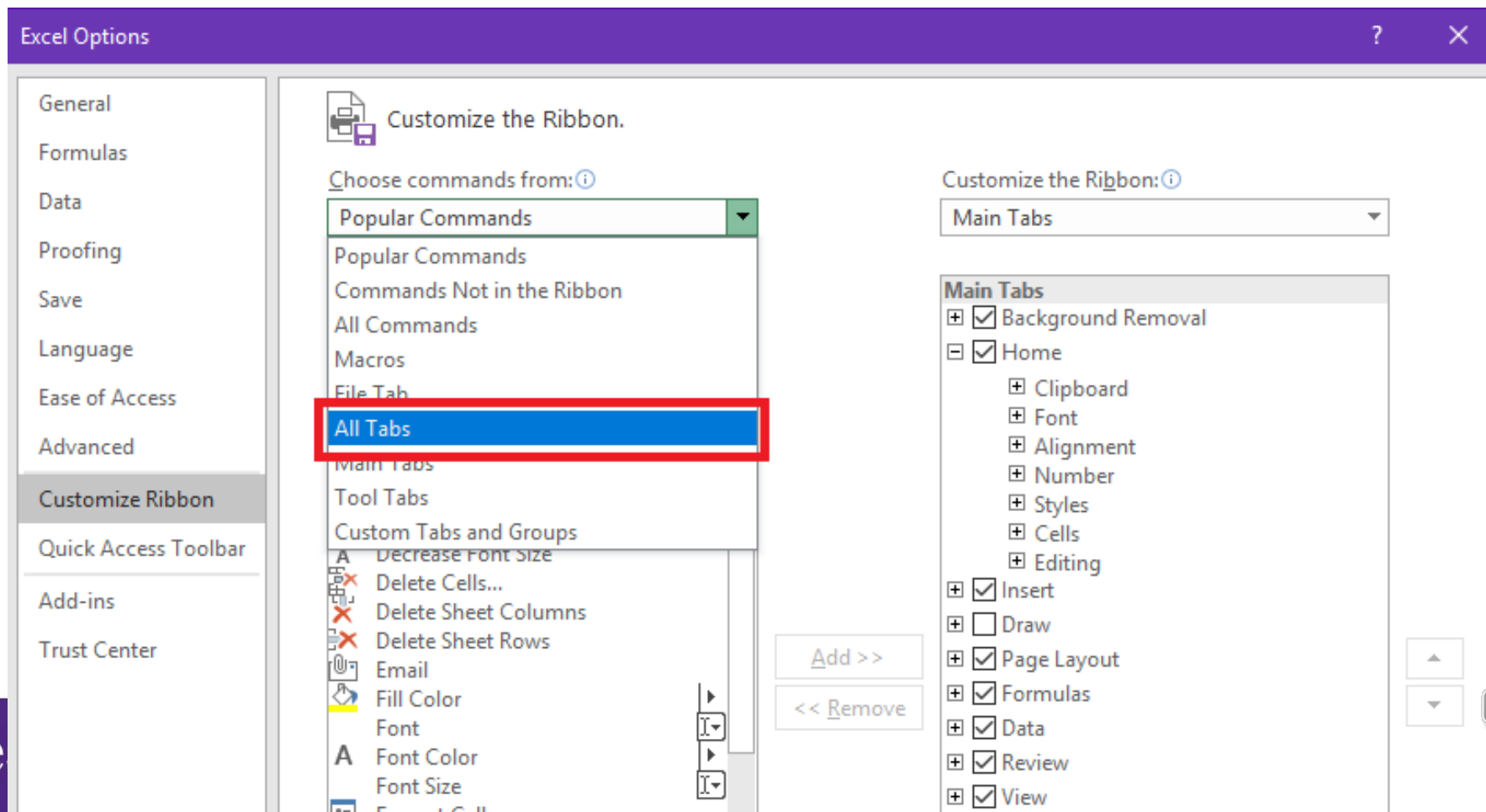
In most cases, this tab is hidden, so we have to expose it as follows:

1. Click the file tab
2. Click options
3. Select "Customize Ribbon"
4. Check the box next to Developer on the right & click OK



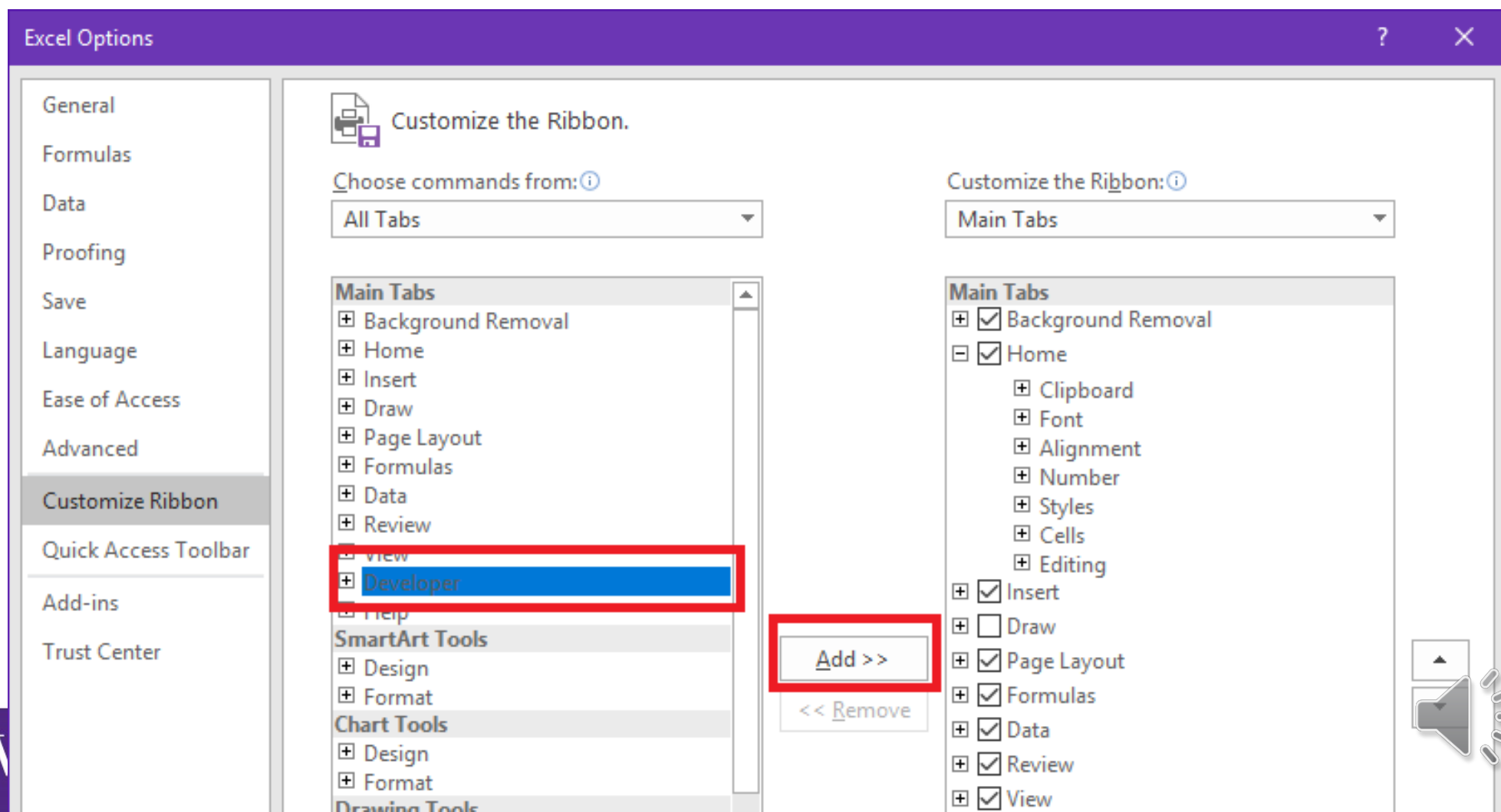
Exposing the Developer Tab

In some cases, the Developer tab option is not shown as an option to check. If this is the case, select “All Tabs” in the “Choose commands from:” drop down menu.

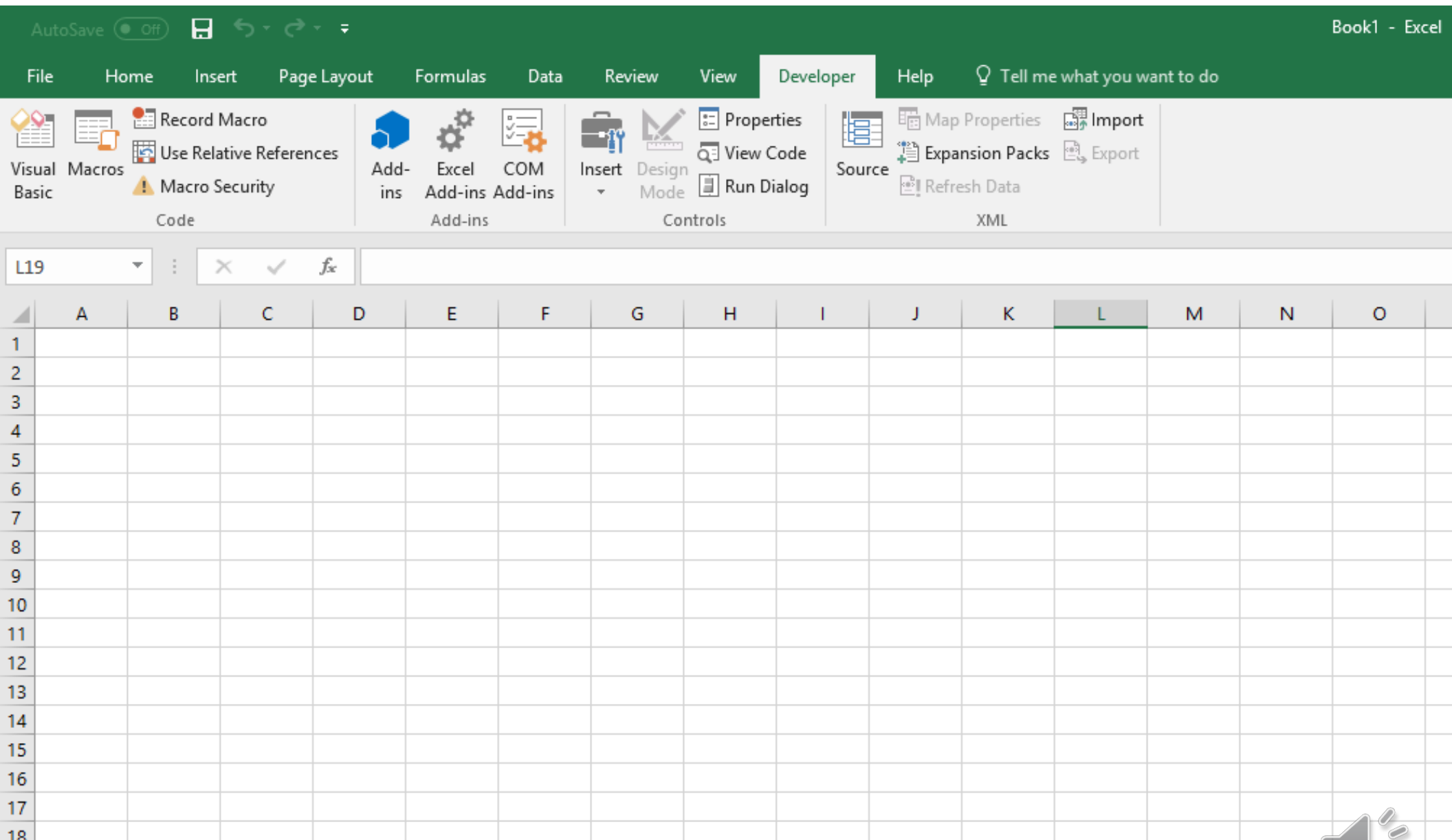


Exposing the Developer Tab

Select “Developer” and click the “Add >>” button. You should now be able to check the Developer check box on the right.

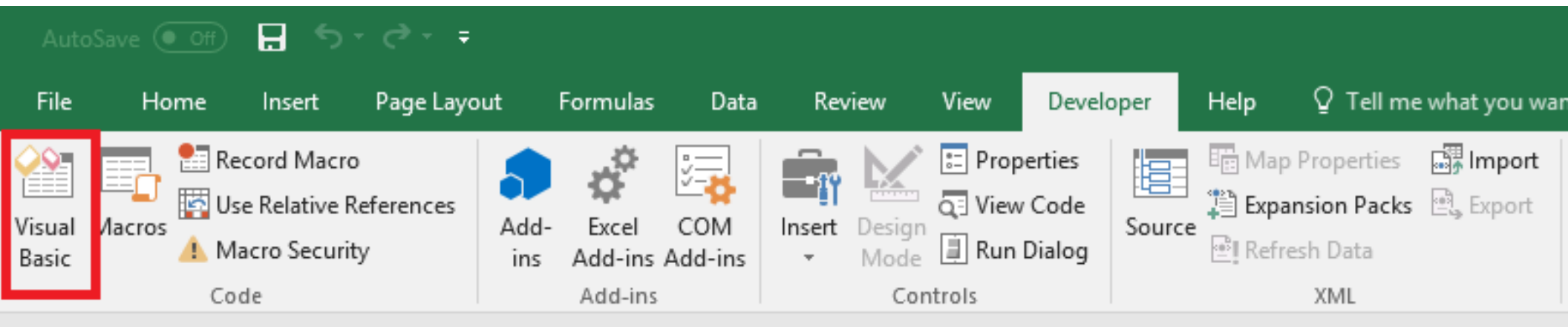


Exposing the Developer Tab

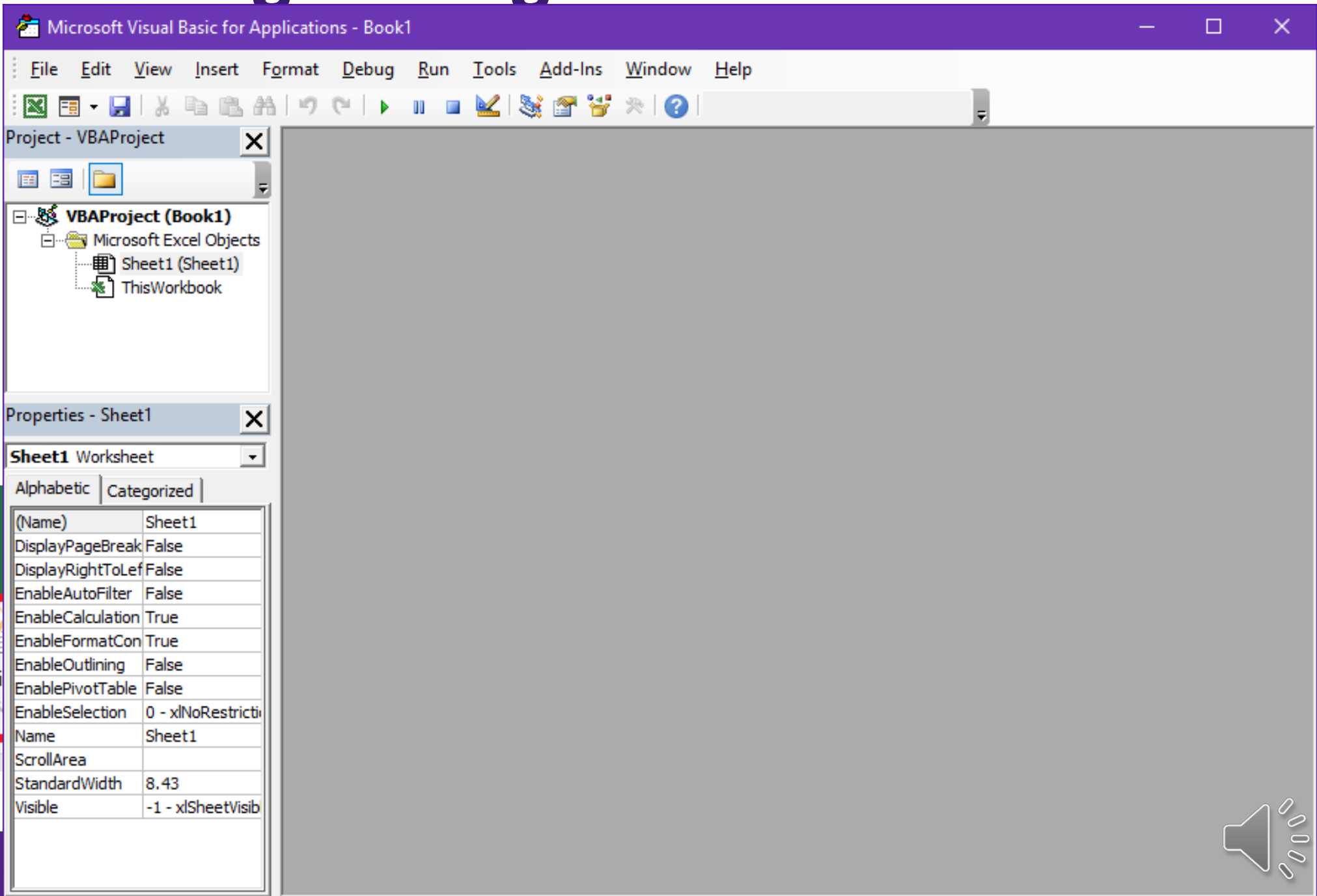


Adding a Program

- On the DEVELOPER tab, click Visual Basic.
- The Visual Basic for Applications (VBA) Integrated Development Environment (IDE) should appear.
- You can also get to the VBA IDE by typing Alt-F11.

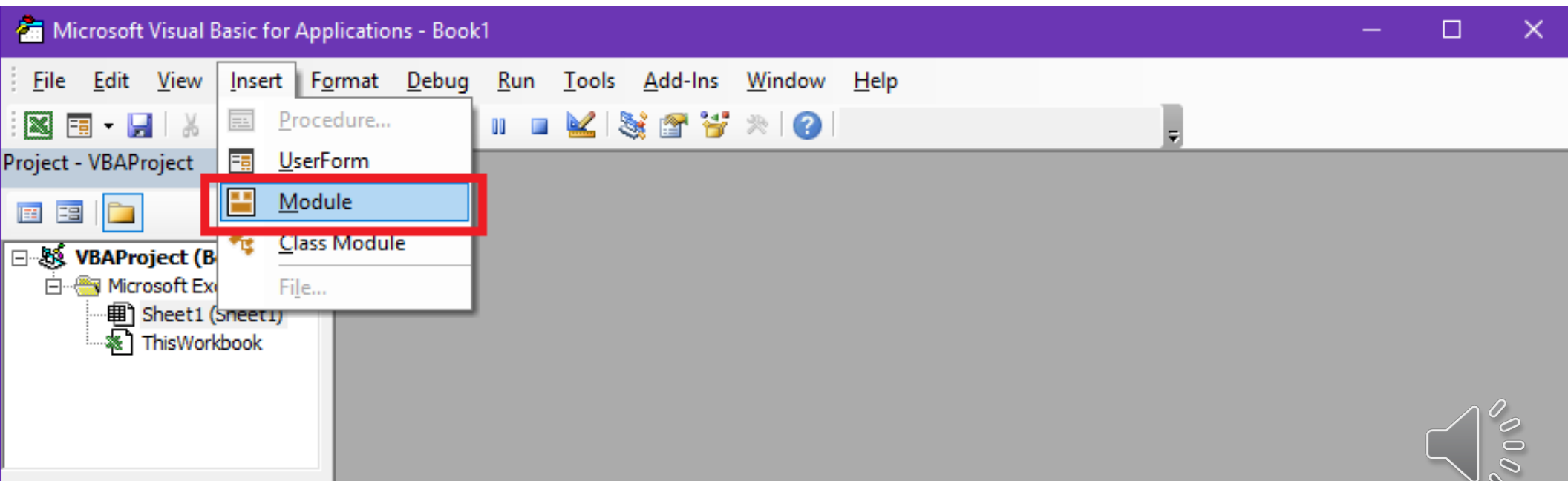


Adding a Program

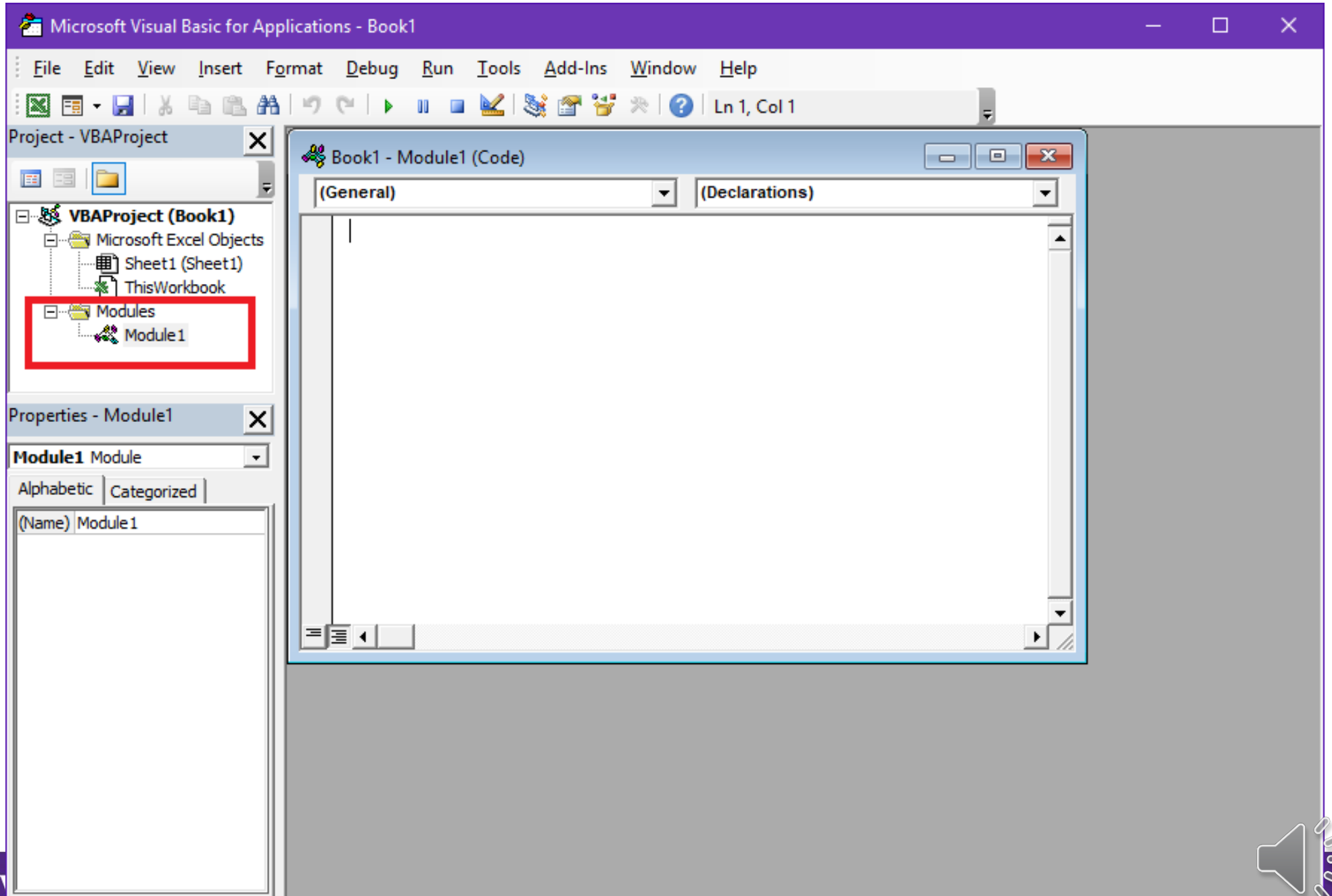


Adding a Program

- All of our programs must exist in a “module”, about which we will learn more later.
- Create a module by going to the **Insert** menu and selecting **Module**.
- This will create a module called Module1.

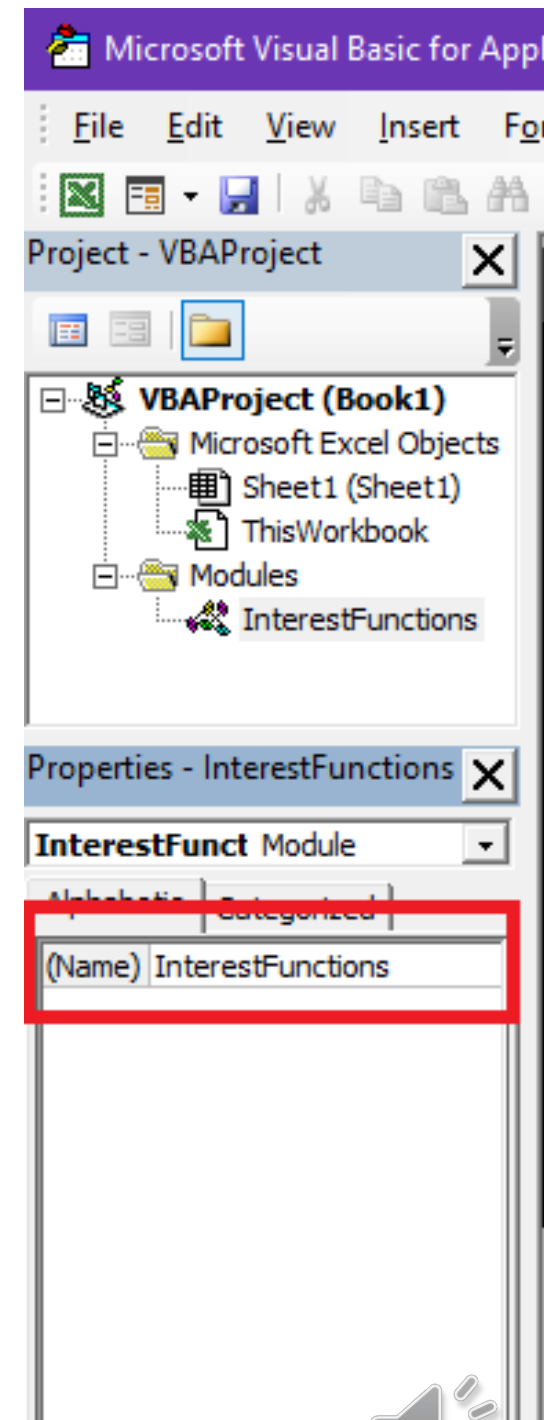


Adding a Program



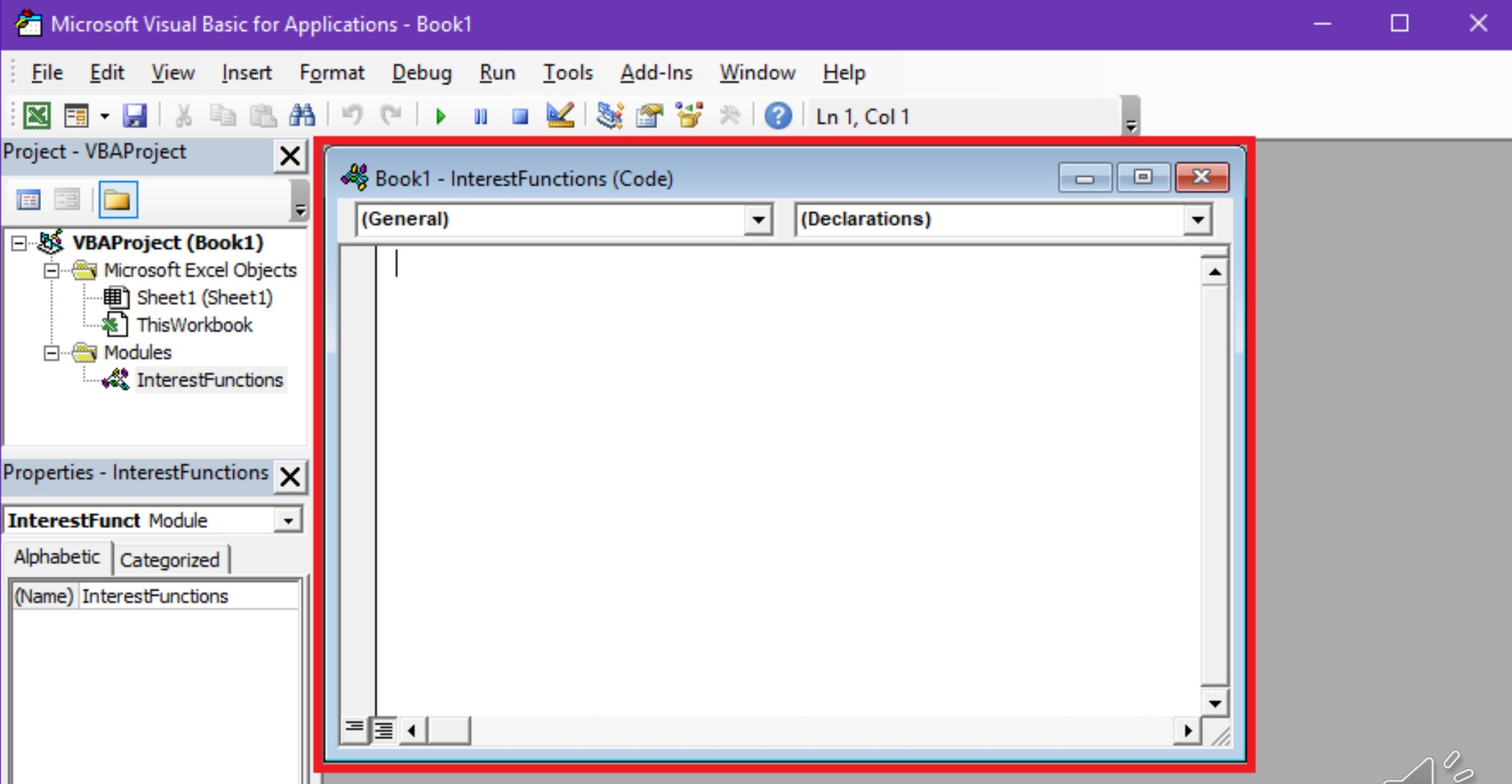
Adding a Program

- You can rename Module1 to something more meaningful by giving a new name in the Properties box.
- Then you can enter the code for the function.
- Here we add code to compute the **DailyInterest**.



Adding a Program

Enter the code for your functions here



Adding a Program

(General)

DailyInterest

```
Function DailyInterest(ra As Double, d As Double, P As Double) As Double  
    DailyInterest = ((1 + ra) ^ (d / 365) - 1) * P  
End Function
```

Adding a Program

- The worksheet should be saved as an “xlsm” file. This allows macros to be enabled when the worksheet is re-opened.
- In this worksheet, it is now possible to use the function **DailyInterest** as though it were built in.

=DailyInterest(B1, B2, B3)

Adding a Program

SUM				X ✓ f _x =D			
	A	B	C				
1	Annual Rate	6.50%	5.50%				
2	Number of Days	33	103				
3	Principal	\$1,000.00	\$33,950.00				
4							
5	Interest	=D	\$516.84				
6							
7							
8							
9							
10							

f_x DailyInterest

f_x DATE

f_x DATEVALUE

f_x DAVERAGE

f_x DAY

f_x DAYS

f_x DAYS360

f_x DB

f_x DCOUNT

f_x DCOUNTA

f_x DDB

f_x DEC2BIN

Adding a Program

SUM		✕ ✓ <i>fx</i>		=DailyInterest(B1,B2,B3)	
	A	B	C	D	E
1	Annual Rate	6.50%	5.50%	2.50%	5%
2	Number of Days	33	103	44	365
3	Principal	\$1,000.00	33,950.00	3.95	100.00
4					
5	Interest	=DailyInterest(B1,B2,B3)			\$5.00
6					



The Structure of a Function

```
Function DailyInterest(ra As Double, d As Double, P As Double) As Double
    DailyInterest = ((1 + ra) ^ (d / 365) - 1) * P
End Function
```

The Structure of a Function

Function Header

Describes the function and its parameters




```
Function DailyInterest(ra As Double, d As Double, P As Double) As Double
    DailyInterest = ((1 + ra) ^ (d / 365) - 1) * P
End Function
```

The Structure of a Function

Function keyword

Tells VBA we are making a
new function



```
Function DailyInterest(ra As Double, d As Double, P As Double) As Double  
    DailyInterest = ((1 + ra) ^ (d / 365) - 1) * P  
End Function
```

The Structure of a Function

Function name

Name that will allow us
to refer to this function



```
Function DailyInterest(ra As Double, d As Double, P As Double) As Double
    DailyInterest = ((1 + ra) ^ (d / 365) - 1) * P
End Function
```



The Structure of a Function

Parameters

Parameters the function
will take



```
Function DailyInterest(ra As Double, d As Double, P As Double) As Double  
    DailyInterest = ((1 + ra) ^ (d / 365) - 1) * P  
End Function
```



The Structure of a Function

Parameters

Parameters the function
will take

Parameter Parameter

Name

Type



```
Function DailyInterest(ra As Double, d As Double, P As Double) As Double  
    DailyInterest = ((1 + ra) ^ (d / 365) - 1) * P  
End Function
```



The Structure of a Function

Function Return Type
Type of data the function
returns



```
Function DailyInterest(ra As Double, d As Double, P As Double) As Double  
    DailyInterest = ((1 + ra) ^ (d / 365) - 1) * P  
End Function
```

The Structure of a Function

```
Function DailyInterest(ra As Double, d As Double, P As Double) As Double
    DailyInterest = ((1 + ra) ^ (d / 365) - 1) * P
End Function
```



Function Body

Describes the steps, operations and calculations the function will take

The Structure of a Function

```
Function DailyInterest(ra As Double, d As Double, P As Double) As Double
    DailyInterest = ((1 + ra) ^ (d / 365) - 1) * P
End Function
```

Assignment of Result to Return

The value on the right of the equals will be the result of this function

Result to Return
Expression that computes the result

The Structure of a Function

```
Function DailyInterest(ra As Double, d As Double, P As Double) As Double
    DailyInterest = ((1 + ra) ^ (d / 365) - 1) * P
End Function
```



End of Function

Tells VBA that this is the
end of our function.



User-Defined Functions

- User-defined functions may be used to perform complex calculations and return results for use in the worksheet.
- User-provided code may also be used to perform almost any action that you would do by hand on the worksheet. Code that **does not return a specific result** is called a “**subroutine**”, and is written with the keyword “sub” in VBA.
- To understand what can be done, we need to learn some programming.

Programming Part 1

Basic Data Types

Data Type	Memory	Range	Description	Examples
Double	8 Bytes	For negative values: -1.79769313486231E308 to -4.94065645841247E-324 For positive values: 4.94065645841247E-324 to 1.79769313486232E308	Positive and negative numbers that may have decimal points.	3.1456 -123.4567 5 9.0
Integer	2 Bytes	-32,768 to 32767	Positive and negative whole numbers without decimal points.	10 -1234 42
Long	4 Bytes	-2,147,483,648 to 2,147,483,647	Like an integer but longer (larger range).	10 -2140478234
Boolean	2 Bytes	True or False	Boolean values (true or false).	True False
String	1 Byte per char	Varies	Text and strings of characters.	"Hello World!" "123ABC!@#%"
Date	8 Bytes	1/1/100 to 12/31/9999	A calendar date.	12/4/2017 30/1/2018



Basic Operators

Arithmetic

+, -, *, /, (,), ^, Mod, \

Comparison

>, <, =, >=, <=, <>

String

&

Logic

And, Or, Not

Assignment

=



Basic Operators

Arithmetic

$+$, $-$, $*$, $/$, $($, $)$, $^$, Mod, \backslash

Comparison

$>$, $<$, $=$, $>=$, $<=$, $<>$

String

$\&$

Logic

And, Or, Not

Assignment

$=$

Same as we saw in Excel

Basic Operators

Arithmetic

+, -, *, /, (,), ^, **Mod**, \

Logic

And, Or, Not

Comparison

>, <, =, >=, <=, <>

Assignment

=

String

&

Finds the remainder after dividing two numbers

10 Mod 2

0

15 Mod 9

6

5 Mod 2

1

0 Mod 4

0

42 Mod 4

2

4 Mod 0

#VALUE

Basic Operators

Arithmetic

+, -, *, /, (,), ^, Mod, \

Logic

And, Or, Not

Comparison

>, <, =, >=, <=, <>

Assignment

=

Integer division

String

&

10 \ 2

5

15 \ 9

1

5 \ 2

2

0 \ 4

0

42 \ 4

10

4 \ 0

#VALUE

Basic Operators

Arithmetic

+, -, *, /, (,), ^, Mod, \

Comparison

>, <, =, >=, <=, <>

String

&

Logic

And, Or, Not

Assignment

=

Boolean logic operators

True And False

False

False Or False

False

True And True

True

Not True

False

True Or False

True

Not False

True

Basic Operators

Arithmetic

+, -, *, /, (,), ^, Mod, \

Comparison

>, <, =, >=, <=, <>

String

&

Logic

And, Or, Not

Assignment

=

Boolean logic operators

5 > 4 And 5 < 2 False

5=6 Or 5<>6 True

NOT (6>=6) False

Basic Operators

Arithmetic

+, -, *, /, (,), ^, Mod, \

Logic

And, Or, Not

Comparison

>, <, =, >=, <=, <>

Assignment

=
↑

Assignment operator is used to assign a value to be the result of a function or the value of a variable.

String

&



Basic Operators

Arithmetic

+, -, *, /, (,), ^, Mod, \

Logic

And, Or, Not

Comparison

>, <, =, >=, <=, <>

Assignment

=
↑

Assignment operator is used to assign a value to be the result of a function or the value of a variable.

String

&

DailyInterest = ((1 + ra) ^ (d / 365) - 1) * P

Value on right is assigned to function or variable on left.

Example Function

Create a function that calculates the area of a circle given a radius.

$$A = \pi r^2$$

$$\pi \cong 3.14159$$

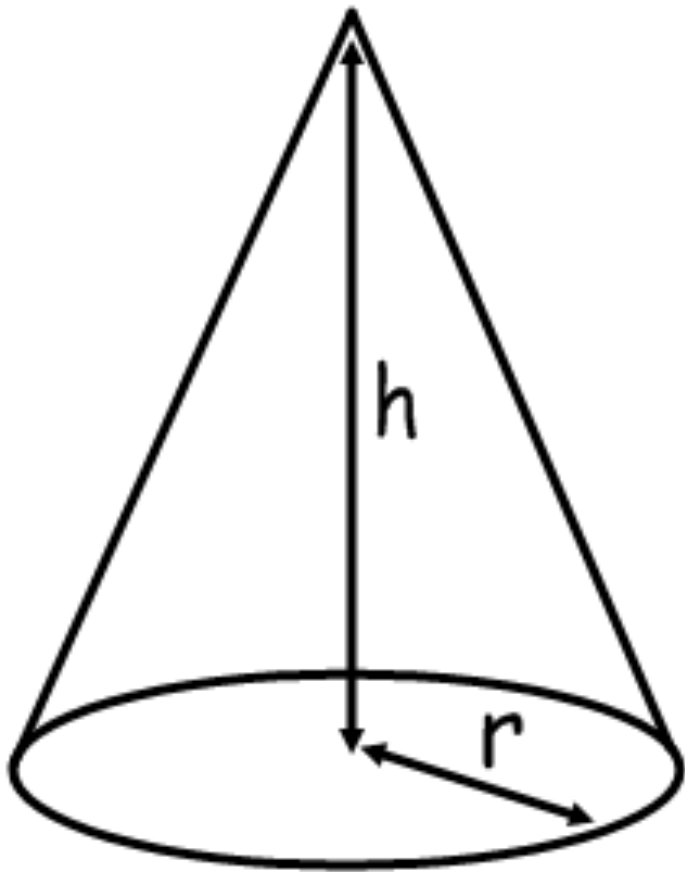
Example Function

Create a function that calculates the area of a circle given a radius.

```
Function CircleArea(radius As Double) As Double  
    CircleArea = 3.14159 * radius ^ 2  
End Function
```

Another Example

Create a function that calculates the volume of a cone given the height and radius.



$$V = \pi r^3 h$$

Another Example

Create a function that calculates the volume of a cone given the height and radius.

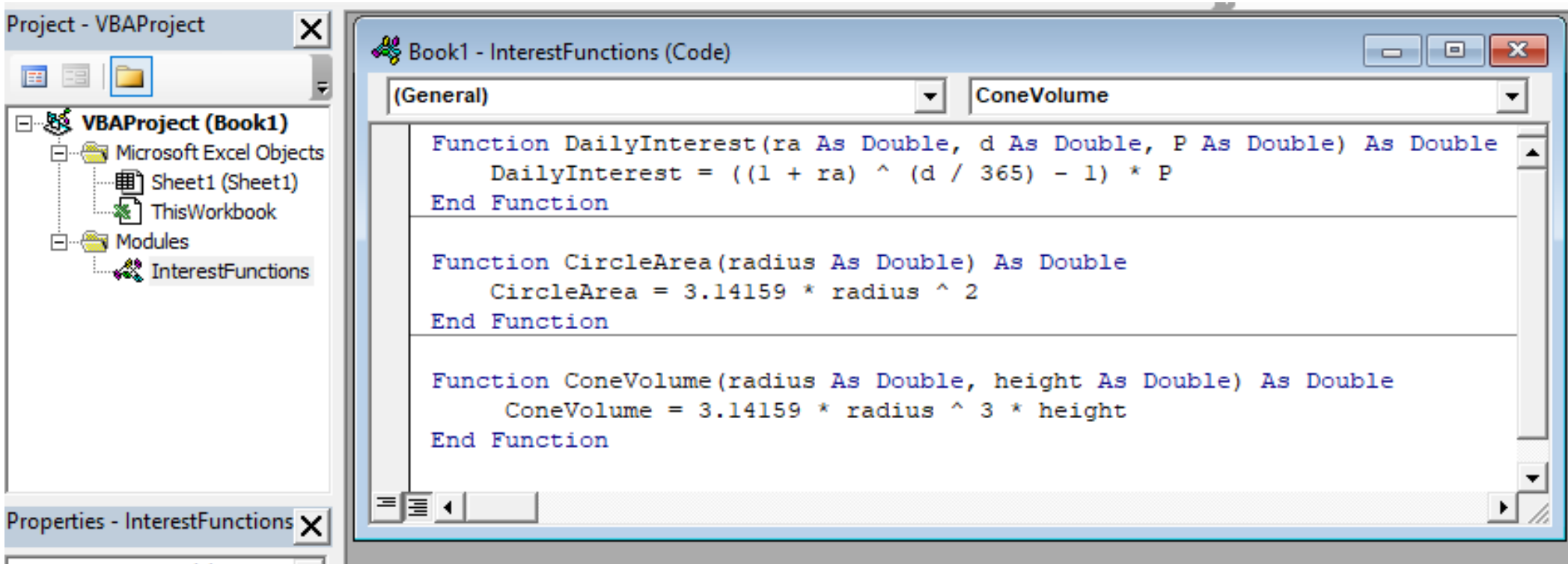
```
Function ConeVolume(radius As Double, height As Double) As Double
    ConeVolume = 3.14159 * radius ^ 3 * height
End Function
```



Naming Functions and Modules

We can have multiple functions in a module, so long as the names do not conflict.

Each function and module needs a unique name.



String Example

Create a function that adds the text “ing” to the end of strings.

```
Function AddIng(text As String) As String  
    AddIng = text & "ing"  
End Function
```

Boolean Example

Create a function that takes an Integer and returns True if the value is odd otherwise it returns False.

```
Function IsOdd(n As Integer) As Boolean
    IsOdd = (n Mod 2) <> 0
End Function
```

Variables

- A variable is a named storage location in a computer program
- Programs use variables to store values.
- These are like the cells names (e.g, A1, B2) but they are not part of a worksheet.
- Each variable holds a value, and the value can be changed by the execution of the program.



Variable Names

- Variable names cannot be reserved key words
 - Examples of reserved words are Sub, Integer, Function, Array, Else
- They are given names, like named cells in a worksheet.
- Name must start with a letter but following characters can have numbers or a _
- RegEx: `[A-Za-z][A-Za-z0-9_]*`

Variable names are not case sensitive and by default the interpreter adjusts the names of all variables with the same letters so that their case matches the case in the variable declaration



Variable Names

Which of the following are valid variable names?

age

Integer

1stName

First name

_name

Name_1

#name

Names2

myPercent%

Variable Type

- A variable is given a “type” to indicate what kind of values it will store.
- Same types we saw before, including:
 - Double
 - Integer
 - Long
 - Boolean
 - String
 - Date



Variable Declarations

To tell VBA that you wish to use a variable, you give the name and the type in a “Dim” declaration.
 (“Dim” is short for “dimension”.)

For now, we will put these declarations inside a function or subroutine.

```
Dim n As Integer  
Dim r As Integer, c As Integer  
Dim x As Double, y As Double  
Dim b as Boolean, s As String, i as Integer  
Dim myLargeNum as Long  
Dim myDate as Date
```



Variable Assignment

We use the assignment operator, =, to give variables a value.

```
Dim n As Integer  
n = 42
```

```
Dim d As Double  
d = 65 / 3
```

```
Dim s As String  
s = "Hello World!"
```

```
Dim b as Boolean  
b = 5 > 7 Or True
```



Variable Use

We can use the value of a variable in an expression by using its name.

```
Dim n As Integer  
Dim d As Double  
Dim b as Boolean
```

```
n = 42  
d = n / 3  
b = n > 20 Or n <= 14
```



Variable Use

We can use the value of a variable in an expression by using its name.

```
Dim n As Integer  
Dim d As Double  
Dim b as Boolean
```

```
n = 42  
d = n / 3  
b = n > 20 Or n <= 14
```

Variable	Value

Variable Use

We can use the value of a variable in an expression by using its name.

```
Dim n As Integer  
Dim d As Double  
Dim b as Boolean
```

```
n = 42  
d = n / 3  
b = n > 20 Or n <= 14
```

Variable	Value
n	

Variable Use

We can use the value of a variable in an expression by using its name.

```
Dim n As Integer  
Dim d As Double  
Dim b as Boolean
```

```
n = 42  
d = n / 3  
b = n > 20 Or n <= 14
```

Variable	Value
n	
d	

Variable Use

We can use the value of a variable in an expression by using its name.

```
Dim n As Integer  
Dim d As Double  
Dim b as Boolean
```

```
n = 42  
d = n / 3  
b = n > 20 Or n <= 14
```

Variable	Value
n	
d	
b	



Variable Use

We can use the value of a variable in an expression by using its name.

```
Dim n As Integer  
Dim d As Double  
Dim b as Boolean
```

```
n = 42  
d = n / 3  
b = n > 20 Or n <= 14
```

Variable	Value
n	42
d	
b	



Variable Use

We can use the value of a variable in an expression by using its name.

```
Dim n As Integer
Dim d As Double
Dim b as Boolean
```

Variable	Value
n	42
d	14.0
b	

```
n = 42
```

```
d = n / 3
```

Equal to 42 / 3

```
b = n > 20 Or n <= 14
```

Variable Use

We can use the value of a variable in an expression by using its name.

```
Dim n As Integer  
Dim d As Double  
Dim b as Boolean
```

Variable	Value
n	42
d	14.0
b	True

```
n = 42  
d = n / 3  
b = n > 20 Or n <= 14
```

Equal to 14 > 20 Or 14 <= 14

Variables

When using variables there are 2 steps you need to complete:

- **Declare** the variable – declare the symbolic variable name and the data type
- **Define** the variable – set a value to the variable

```
1 | Dim myVar As Integer 'Declaration  
2 | myVar = 10 'Definition
```

