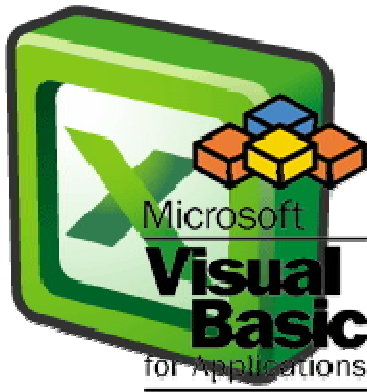


آموزش

# برنامه نویسی در اکسل

A screenshot of the Microsoft Visual Basic for Applications editor. The main window displays the code for 'Module3' in 'Book2.xlsm'. The code includes a function 'multiply', a sub 'macro1', a sub 'SetValue', a sub 'test', and a sub 'Macro11'. The 'Watches' window at the bottom is empty.

```
Function multiply(a, b)
    multiply = a * b
End Function
Sub macro1()
    r = multiply(4, 7)
    MsgBox ("r = " & r)
End Sub
Sub SetValue(ByRef i As Integer)
    i = i * 5
End Sub
Sub test()
    Dim x As Integer
    x = 5
    Call SetValue(x)
    MsgBox ("x = " & x)
End Sub
Sub Macro11()
    Dim w1 As Workbook, s1 As Worksheet
    Set w1 = Workbooks("book1")
    Set s1 = w1.Sheets("sheet1")
End Sub
```



گروه فردانش

تالیف : وحید فرزام

نگارش اول

Vahid.Farzam@gmail.com

www.Fardanesh.ir

5	معرفی و ایجاد ماکرو
5	روبان Developer
5	ابزار Record Macro
6	روشهای اجرای ماکرو
6	چند نکته
8	آشنایی با Visual Basic for Application (VBA)
8	ماژول (Module)
9	رویه (Procedure)
9	زیرروالها (Subroutines)
9	قواعد پایه برنامه نویسی VBA
9	عملگرها
12	متغیرها و انواع دادهها
12	اعلان متغیر
12	میدان دید و طول عمر متغیر
13	انواع دادهها در VBA
14	بررسی نوع داده یک متغیر
15	کار با آرایه ها
16	تعریف نوع داده جدید توسط کاربر
17	ثابت ها
18	ماژولها، توابع و زیرروالها
18	رویه یا Procedure
18	زیرروالها (Subroutines)
20	توابع (Functions)
22	تعریف آرگومان اختیاری برای توابع
22	نحوه انتقال آرگومان
23	مثال هایی از توابع
25	دستورهای کنترلی در VBA
25	دستور IF



۲۶.....	Select Case دستور
۲۷.....	For-Next حلقه
۲۸.....	For Each حلقه
۲۹.....	Do While   Until حلقه
۲۹.....	With—End With دستور
30.....	VBA توابع و دستورها
۳۰.....	توابع ویرایش متن (کلاس Strings)
۳۱.....	برخی از توابع کلاس Math
۳۱.....	توابع تاریخ و زمان (کلاس DateTime)
۳۱.....	معرفی چند دستور مهم
۳۱.....	کادر پیام (MsgBox)
۳۳.....	پنجره ورودی (InputBox)
35.....	مدیریت خطاها در برنامه نویسی اکسل
۳۵.....	انواع خطاها
۳۵.....	حالت های یک برنامه VBA
۳۷.....	دستور GoTo
۳۷.....	مدیریت خطاهای زمان اجرا
۳۸.....	دستور Resume
۳۹.....	مدیریت خطا توسط GoTo
۴۰.....	مثال های مدیریت خطا
42.....	مدل شیء Excel
۴۳.....	خصوصیات و متدها
۴۳.....	استفاده از Object Browser
۴۴.....	شیء Application
۴۵.....	شیء Workbook
۴۷.....	شیء Range
۴۸.....	شیء Cells
۴۹.....	مثالهای دیگر
51.....	رویدادها

۵۱	.....	Worksheet	رویدادهای
۵۵	.....	WorkBook	رویدادهای
57	.....		آشنایی با فرم ها
۵۷	.....	Data Form	نمایش
۵۸	.....		درج فرم
۵۸	.....		نمایش فرم
۵۹	.....		درج کنترل ها
۵۹	.....		انواع کنترل ها
۶۲	.....		تغییر ویژگی های یک کنترل
۶۳	.....	مثال 1	
۶۳	.....	مثال 2	
۶۵	.....	مثال 3	
67	.....	VBA	تکنیک های
۶۷	.....		کپی کردن یک محدوده
۶۸	.....		انتقال یک محدوده
۶۸	.....		کپی کردن محدوده متغیر
۶۸	.....		انتخاب محدوده اطلاعات
۶۹	.....		دریافت ورودی برای خانه فعال
۶۹	.....		شمارش خانه های ناحیه انتخاب شده
۷۰	.....		حذف سطر های غیر خالی
۷۰	.....		تعیین نوع اطلاعات یک خانه
۷۱	.....		ذخیره تمام فایل ها
۷۱	.....		شمارش خانه های بین دو مقدار
72	.....	Office	ارتباط با دیگر برنامه های
۷۲	.....	MS Word	ارتباط با
77	.....	Ribbon	مدیریت ها
۷۷	.....		فعال کردن یک روبان
۷۸	.....		ویرایش روبان ها
81	.....	Add-Ins	ایجاد



- ۸۱.....VBA محافظت از کدهای
- ۸۲.....تفاوتهای Add-Ins و یک فایل معمولی اکسل:
- ۸۲.....Add-In نصب یک
- ۸۳.....Add-In ایجاد یک
- ۸۴.....Add-In ویرایش
- ۸۵.....استفاده از Add-In نصب شده
- ۸۵.....Add-In رویدادهای



## معرفی و ایجاد ماکرو (فصل 1)

ماکرو به مجموعه دستوراتی گفته می‌شود که برای کاربر این امکان را فراهم می‌سازد تا اعمالی که زیاد اجرا می‌شوند را به صورت خودکار اجرا کند و یا فرمانهای سفارشی را تعریف و اجرا کند.

### 1-1 روبان Developer

اولین قدم در کار با ماکروها فعال کردن روبان Developer است. روبان Developer ابزارهای مربوط به امکانات پیشرفته اکسل (از جمله ماکروها) را در اختیار کاربر قرار می‌دهد. برای فعال کردن روبان Developer از منوی File گزینه Options را کلیک کنید. سپس به برگه Customize Ribbon مراجعه کنید و از لیست روبانهای ارائه شده در سمت راست پنجره، روبان Developer را انتخاب کنید تا نمایش یابد.

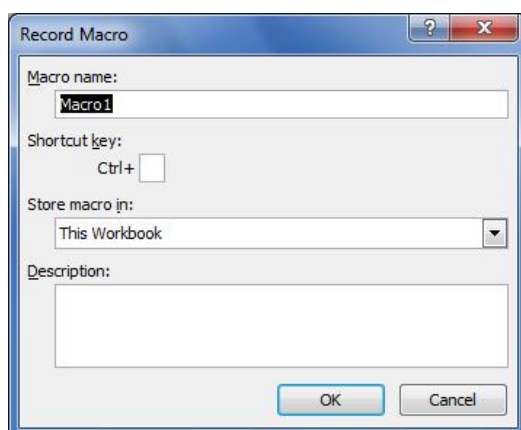


شکل ۱-۱

### 1-2 ابزار Record Macro

ساده ترین روش برای ایجاد ماکرو استفاده از ابزار Record Macro می‌باشد. این ابزار عملیات انجام شده در کاربرگ را به کدهای VBA تبدیل می‌کند و به این ترتیب کاربر بدون آگاهی زیاد از کدهای VBA می‌تواند ماکروهای ساده ایجاد کند و با بررسی و تغییر کدهای ایجاد شده بعد از کسب آگاهی نسبی در رابطه با قواعد کد نویسی به زبان VBA، کدها را ویرایش کرده و ماکروهای مورد نظر خود را ایجاد کند.

برای ثبت ماکرو ابتدا از روبان Developer دستور Record Macro را کلیک کنید تا پنجره Record Macro نمایش یابد.



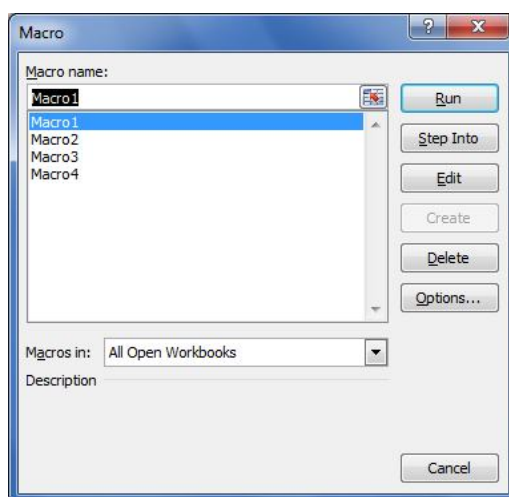
شکل ۲-۱

در پنجره باز شده ابتدا نام ماکرو را در کادر بالا وارد کنید و سپس Shortcut key یا کلید ترکیبی دلخواه را برای اجرای سریع ماکرو تعریف کنید و دکمه OK را کلیک کنید.

سپس اعمال دلخواه را در کاربرگ انجام دهید و در پایان روی دکمه Stop Recording در روبان Developer کلیک کنید تا ضبط ماکرو پایان یابد.

## 1-3 روشهای اجرای ماکرو

ابتدایی ترین روش برای اجرای یک ماکرو این است که روی دکمه Macros در گروه Code از روبان Developer کلیک کنید. سپس در پنجره باز شده ماکرو مورد نظر را در لیست ماکروها انتخاب کرده و برای اجرای آن روی دکمه Run کلیک کنید. (شکل ۳-۱)



شکل ۳-۱

همانطور که در بخش ضبط ماکرو گفته شد، شما می‌توانید برای یک ماکرو کلید میانبر (Shortcut Key) تعریف کنید و بعد از آن هر زمان که می‌خواهید ماکرو را اجرا کنید، کلید میانبر تعریف شده را از صفحه کلید فشار دهید.

روش دیگر اجرای ماکرو با کلیک روی یک دکمه می‌باشد. برای انجام این کار ابتدا یک Shape را که می‌خواهید وظیفه Button یا دکمه را داشته باشد، روی کاربرگ رسم کنید. سپس روی دکمه رسم شده راست کلیک کرده و گزینه Assign Macro را انتخاب کنید. در پنجره باز شده ماکرو مورد نظر را انتخاب کرده و دکمه OK را کلیک کنید. بعد از این عملیات با هر بار کلیک روی شکل، ماکرو انتخاب شده اجرا خواهد شد.

به عنوان روش آخر می‌توانید یک دکمه بر روی QAT اضافه کنید که با کلیک روی آن یک ماکروی خاص اجرا شود.

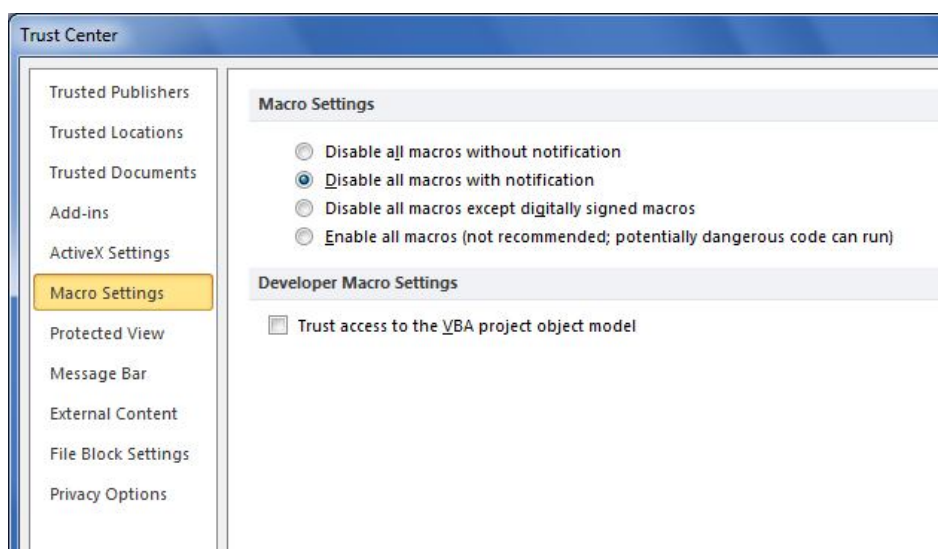
## 1-4 چند نکته

۱. برای ذخیره فایل های حاوی ماکرو باید در نوار Save as Type در پنجره Save, پسوند xism را انتخاب کنید تا ماکرو های ایجاد شده نیز همراه فایل اکسل ذخیره شوند.

۲. برای دسترسی به تنظیمات امنیتی ماکروها مسیر زیر را دنبال کنید:

Excel Options → Trust Center → Trust Center Setting → Macro Setting

در پنجره باز شده (شکل ۴-۱) چهار سطح امنیتی برای فایل های دارای ماکرو قابل انتخاب می باشد.



شکل ۱-۴

۳. برای مشاهده کدهای VBA مربوط به ماکرو ذخیره شده ابتدا روی دکمه Macros در روبان Developer کلیک کنید. سپس ماکرو مورد نظر را در لیست ماکروها انتخاب کرده و برای مشاهده کد مربوط به آن روی دکمه Edit کلیک کنید.
۴. یکی از راههای یادگیری کدهای VBA مشاهده همزمان پنجره Excel و پنجره VBA در هنگام ضبط ماکرو می‌باشد. به این صورت که قبل از رکورد ماکرو دو پنجره اکسل و VBE را همزمان باز کرده و ماکرو را ضبط کنید. مشاهده خواهید کرد که همزمان با انجام اعمال در محیط اکسل، کدهای مربوط به آن در محیط VBE تولید می‌شوند.





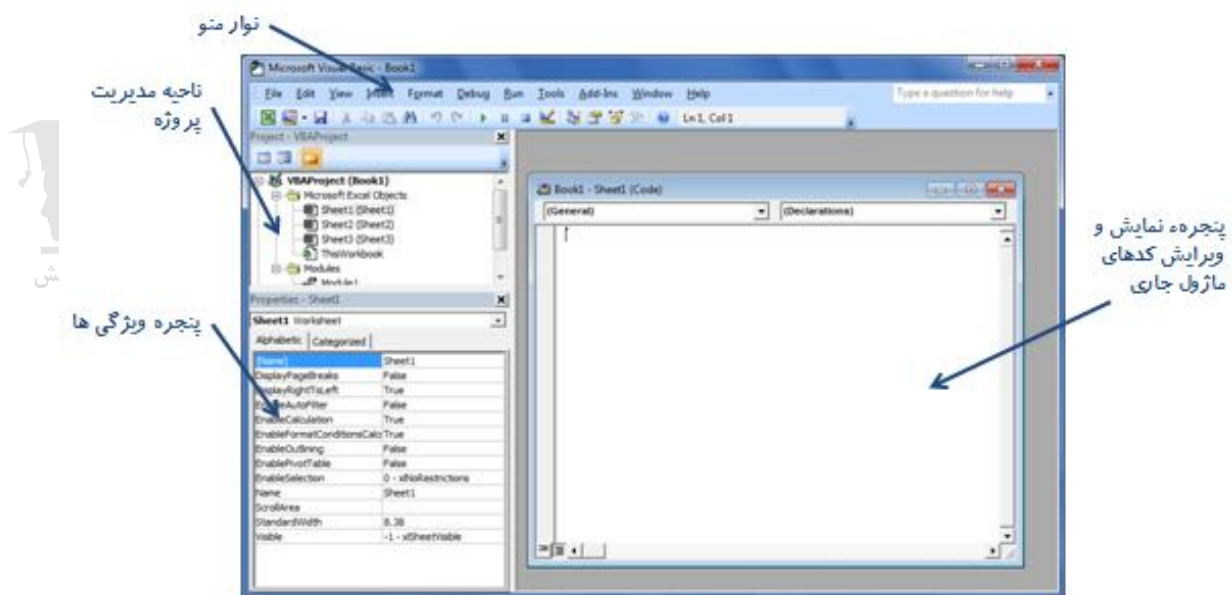
## Visual Basic for (فصل 2) آشنایی با Application (VBA)

برای مشاهده محیط برنامه نویسی VBA در اکسل، روی دکمه Visual Basic در گروه Code از روبان Developer (شکل 1-2) کلیک کنید یا اینکه کلیدهای Alt+F11 را از فشار دهید.



شکل 1-2

در تصویر زیر (شکل 2-2)، اجزاء معمول محیط VBE را مشاهده می‌کنید.



شکل 2-2

### 1-2 ماژول (Module)

ماژول یا Module به ظرف یا فضایی گفته می‌شود که کدهای VBA در قالب یک یا چند رویه (Procedure)، درون آن نگهداری می‌شوند. (کادر سمت راست در شکل 2-2 یک ماژول را نشان می‌دهد که البته هنوز کدا داخل آن نوشته نشده است)

پروژه‌های VBA حداقل از یک ماژول برای ذخیره‌سازی رویه‌ها استفاده می‌کنند. برای ایجاد یک ماژول، در محیط VBE از منوی Insert گزینه Module را انتخاب کنید.

برای مشاهده ماژول‌های فایل جاری، به ناحیه مدیریت پروژه (Project Explorer) در محیط VBE مراجعه کنید شکل 2-2 و سپس روی

عبارت Module کلیک کنید تا لیست ماژول ها نمایش یابد.

**نکته:** وقتی شما از ابزار Record Macro برای ایجاد یک ماکرو استفاده می‌کنید، به طور خودکار یک ماژول در پروژه شما ایجاد شده و کدهای ایجاد شده در قالب یک رویه (در ادامه با موضوع رویه آشنا خواهید شد) درون آن درج می‌شوند.

## 2-2 رویه (Procedure)

رویه یا Procedure مجموعه ای از کدهای مرتبط با هم است که درون یک ماژول در محیط VBE نوشته شده و عمل خاصی را انجام می‌دهند.

دو نوع رویه (Procedure) در VBA وجود دارد:

1. زیرروال‌ها (Subroutines)

2. توابع (Functions)

برای ایجاد یک رویه در ماژول جاری، می‌توانید در محیط VBE از منوی Insert گزینه Procedure را انتخاب کنید.

## 2-3 زیرروال‌ها (Subroutines)

زیرروال یا Subroutine مجموعه ای از کدهای VBA است که درون یک ماژول نوشته شده و در زمان اجرا توسط کاربر یا فراخوانی توسط زیرروال دیگر، عمل خاصی را انجام می‌دهد.

زیرروال‌ها همیشه با کلمه کلیدی Sub همراه با نام ماکرو یا زیرروال شروع شده و با عبارت End Sub پایان می‌یابند.

**نکته 1:** درج کلمه کلیدی Private قبل از نام یک زیرروال باعث می‌شود که فراخوانی زیرروال مورد نظر فقط در ماژول جاری امکان‌پذیر باشد و زیرروالی که با کلمه کلیدی Private اعلان می‌شود را دیگر نمی‌توانید در لیست ماکروهای تعریف شده مشاهده کرده و مورد استفاده قرار دهید.

**نکته 2:** برای اجرای آزمایشی زیرروال‌های بدون آرگومان، می‌توانید بعد از کلیک درون زیرروال، دکمه F5 را فشار دهید و یا از منوی Run دستور Run Sub را انتخاب کنید.

## 2-4 قواعد پایه برنامه نویسی VBA

چند نکته در برنامه نویسی VBA:

- از سطرهای خالی و تورفتگی برای خوانا بودن بیشتر کد استفاده کنید.
- بعد از فشار دادن Enter در صورت عدم وجود خطا، کد نوشته شده دوباره فرمت بندی می‌شود (شناسایی واژه های کلیدی)
- اگر می‌خواهید یک عبارت چند سطری بنویسید باید در انتهای هر سطر یک فاصله و سپس علامت Underline ( \_ ) را وارد کنید.
- برای اضافه کردن سطر توضیح کافیسیت که اول سطر یک علامت آپستروف وارد کنید.
- برای نوشتن یک دستور چند خطی باید در انتهای هر سطر (به غیر از سطر آخر) یک فاصله همراه با Underline وارد کنید.

## 2-5 عملگرها

عملگرها و توابع از جمله ابزارهای VBA برای انجام محاسبات می‌باشند.

عملگرهای VBA دارای تقدم هایی نسبت به هم می باشند که برای ایجاد تغییر در این تقدم ها می توانید از پرانتز استفاده کنید.

### 2-5-1 عملگرهای حسابی

- عملگر ^ : توان
- عملگر - : منفی
- عملگر \* : ضرب
- عملگر / : تقسیم
- عملگر \ : خارج قسمت تقسیم صحیح
- عملگر Mod : باقیمانده تقسیم صحیح
- عملگر - : تفریق
- عملگر + : جمع
- عملگر & : الحاق دو رشته متنی به هم

### 2-5-2 عملگرهای مقایسه‌ای

- عملگر =
- عملگر >
- عملگر <
- عملگر >=
- عملگر <=
- عملگر <>
- عملگر Like

این عملگر دو رشته متنی را بر طبق شرایط تعریف شده با یکدیگر مقایسه می کند.

در تعریف نوع مقایسه می توانید از کارکترهای جایگزین زیر استفاده کنید :

مفهوم	کارکتر
یک کارکتر	?
چند کارکتر	*
یک رقم از 0 تا 9	#
یک کارکتر از charlist	[charlist]
کارکتری که در charlist نباشد	[!charlist]



```
Sub like_operator()  
Dim MyCheck  
MyCheck = "aBBBa" Like "a*a"           ' Returns True.  
MyCheck = "F" Like "[A-Z]"             ' Returns True.  
MyCheck = "F" Like "[!A-Z]"           ' Returns False.  
MyCheck = "a2a" Like "a#a"            ' Returns True.  
MyCheck = "aM5b" Like "a[L-P]#[!c-e]" ' Returns True.  
MyCheck = "BAT123khg" Like "B?T*"     ' Returns True.  
MyCheck = "CAT123khg" Like "B?T*"     ' Returns False.  
End Sub
```

## 3-5-2 عملگرهای منطقی

- عملگر And
- عملگر Or
- عملگر Not
- عملگر Xor
- عملگر Eqv
- عملگر Imp



## متغیرها و انواع داده‌ها (فصل 3)

متغیر محلی موقت برای ذخیره سازی اطلاعات می باشد و معمولا از متغیرها برای نگهداری اطلاعات در زمان اجرای یک کد استفاده می شود.

### 3-1 اعلان متغیر

برای اعلان متغیرها از دستور زیر استفاده می شود :

[نوع متغیر as] نام متغیر Dim

**نکته 1:** اگر نوع متغیری را مشخص نکنید به طور پیش فرض نوع Variant برای آن در نظر گرفته می شود.

**نکته 2:** در نامگذاری متغیرها موارد زیر را رعایت کنید :

- نام متغیر باید با یکی از حروف آغاز شود
- نام متغیر تنها می تواند شامل حروف, اعداد و کارکتر Underline باشد
- از کلمات ذخیره شده نمی توانید به عنوان نام متغیر استفاده کنید.

**نکته 3:**

اگر در بخش اعلان ماژول عبارت Option Explicit وارد شود, قبل از استفاده از هر متغیر باید آن را تعریف کرد و عدم تعریف یا اعلان متغیر باعث بروز پیغام خطا خواهد شد. در غیر این صورت شما می توانید مستقیما از یک متغیر استفاده کنید, بدون اینکه آن را تعریف کرده باشید.

گروه‌فردانش

در مثال زیر بدون تغییر متغیر m از آن استفاده شده است.

```
Sub test5 ()
m = 5
MsgBox m
End Sub
```

اعلان متغیر رشته‌ای با طول متغیر

Dim temp1 as String

اعلان متغیر رشته‌ای با طول ثابت

Dim temp1 as String \* 20

نکته : اگر طول رشته اختصاص داده شده به متغیر کمتر از طول تعریف شده برای متغیر باشد, کارکترهای باقیمانده با فاصله خالی پر می شوند

### 3-2 میدان دید و طول عمر متغیر

متغیرهای تعریف شده در یک رویه, تنها درون همان رویه قابل دسترس خواهند بود و مقدار خود را تنها تا زمان پایان اجرای رویه, حفظ می کنند.

در تصویر زیر حتی اگر قبل از اجرای Test6، ماکروی Test5 را اجرا کنید باز هم مقدار نشان داده شده برای m در Test6 برابر null یا تهی خواهد بود.

```
Sub test5 ()
m = 5
MsgBox m
End Sub
Sub test6 ()
MsgBox m
End Sub
```

برای حفظ مقدار یک متغیر رویه در طول زمان اجرا، باید متغیر را به جای عبارت Dim با عبارت Static اعلان کنید. در مثال زیر، با هر بار اجرای رویه، 5 واحد به مقدار متغیر a اضافه می‌شود:

```
Sub StaticVar ()
Static a As Integer
MsgBox a
a = a + 5
End Sub
```

با قرار دادن عبارت Static قبل از عنوان یک رویه، تمام متغیرهای رویه به صورت Static در نظر گرفته می‌شوند.

متغیرهای تعریف شده در سطح اعلان ماژول، در تمام رویه‌های ماژول قابل دسترس خواهند بود و مقدار خود را در زمان اجرای برنامه حفظ می‌کنند. متغیرهای سطح ماژول تا آخر برنامه باقی مانده و مقدار خود را حفظ می‌کنند.

در مثال زیر با اجرای ماکروی Test6 مقدار 5 نمایش می‌یابد.

(General)	(Declarations)
Dim m As Integer	
Sub test5 ()	
m = 5	
MsgBox m	
End Sub	
Sub test6 ()	
MsgBox m	
End Sub	

متغیرهایی که با دستور Global در سطح اعلان ماژول تعریف می‌شوند، در همه جای برنامه قابل دسترس خواهند بود و مقدار خود را در زمان اجرای برنامه حفظ می‌کنند.

### 3-3 انواع داده‌ها در VBA

کارکتر معرف	شرح	نوع متغیر
-------------	-----	-----------

	عدد صحیح یک بایتی	Byte
%	عدد صحیح دو بایتی	Integer
&	عدد صحیح چهار بایتی	Long
!	عدد اعشاری چهار بایتی	Single
#	عدد اعشاری هشت بایتی	Double
@	عدد هشت بایتی با اعشار ثابت	Currency
\$	رشته متنی	String
	تاریخ و ساعت	Date
ندارد	نوع داده متغیر	Variant

نکته: متغیر از نوع Variant می‌تواند هر نوع داده‌ای را در خود جای دهد و آزادانه در طول اجرای برنامه نوع خود را تغییر دهد.

### 3-4 بررسی نوع داده یک متغیر

برای تعیین نوع داده متغیر از تابع VarType استفاده می‌شود. مقدار برگشتی از این تابع، بسته به نوع متغیر ورودی، یکی از مقادیر زیر خواهد بود:

نوع متغیر ورودی	مقدار برگشتی
Empty	0
Null	1
Integer	2
Long	3
Single	4
Double	5
Currency	6
Date/Time	7
String	8
Object	9
مقدار خطا	10

مثال:

```
Sub testVarType ()
Dim a
a = "ali"
MsgBox VarType(a) ' return 8

a = 23
MsgBox VarType(a) ' return 2

Set a = Sheets(2)
MsgBox VarType(a) ' return 9

End Sub
```

نکته: برای بررسی نوع یک متغیر می‌توانید از توابع زیر نیز استفاده کنید:

IsNumeric •

مثال: جواب کد زیر مقدار False خواهد بود.

```
Sub TestNumeric()
temp = "ali"
MsgBox IsNumeric(temp)
End Sub
```

- IsDate
- IsEmpty

تنها متغیر هایی از نوع Varient که هنوز مقداری به آنها اختصاص نیافته است دارای مقدار Empty می باشند و این امر توسط تابع IsEmpty کنترل می شود.

- IsNull

تنها متغیر هایی از نوع Varient میتوانند مقدار Null داشته باشند. متغیرها حاوی مقدار Null نخواهند بود مگر اینکه برای این کار کدی نوشته شود.

### 3-5 کار با آرایه ها

برای اعلان یک متغیر به صورت یک آرایه، بعد از نام متغیر باید طول آرایه تعریف شود.

مثال: عبارت مقابل یک آرایه 11 عنصری تعریف می کند که اندیس عناصر آن از 0 تا 10 می باشد.

```
Dim fname(10) as String
```

نکته 1: اگر عبارت Option Base 1 را در سطح اعلان ماژول وارد کنید، برای عنصر اول آرایه با اندیس 1 در نظر گرفته می شود.

نکته 2:

روشی دیگر برای اعلان آرایه:

```
Dim fname(1 to 10) as String
```

روش تعریف آرایه چند بعدی:

```
Dim fname(10,10) as String
```

تعریف آرایه دینامیکی یا آرایه بدون بعد:

```
Dim temp() as String
```

برای ایجاد تغییر در تعداد عناصر یک آرایه دینامیکی، از دستور ReDim استفاده می شود.

دستور ReDim تمام مقادیر قبلی موجود در آرایه را حذف می کند.



برای ایجاد تغییر در عناصر آرایه همراه با حفظ مقادیر قبلی، باید از کلمه کلیدی Preserve استفاده شود.

در رویه زیر، ابتدا یک آرایه دینامیکی با سه عنصر تعریف می‌شود و سپس با دستور ReDim Preserve، طول آن به شش عنصر افزایش پیدا می‌کند ولی مقادیر اولیه حفظ می‌شوند:

```
Sub ReDim_Array()
    Dim a()
    ReDim a(1 To 3)
    For i = 1 To 3
        a(i) = i ^ 2
    Next i
    ReDim Preserve a(1 To 6)
    For i = 1 To 6
        MsgBox a(i)
    Next i
End Sub
```

برای تعیین مرز بالا و پایین یک آرایه، به ترتیب از توابع UBound و LBound استفاده می‌شود. مثال:

```
Option Explicit
Option Base 1
Sub array1()
    Dim temp1(12, 8, 15) As Integer
    Dim temp2(10)
    MsgBox UBound(temp2)      ' 10
    MsgBox LBound(temp2)     ' 1
    MsgBox UBound(temp1, 1)  ' 12
    MsgBox UBound(temp1, 2)  ' 8
    MsgBox UBound(temp1, 3)  ' 15
End Sub
```



### 3-6 تعریف نوع داده جدید توسط کاربر

شما می‌توانید با استفاده از کلمه کلیدی Type در بخش اعلان ماژول، متغیر جدیدی با نوع داده‌های درونی دلخواه تعریف کنید.

```
Option Explicit
Option Base 1
Type Employee
    FName As String
    LName As String
    Salary As Currency
End Type
Sub NewVar()
    Dim temp As Employee
    temp.FName = "Ali"
    temp.LName = "eSTEGHAMAT"
    temp.Salary = 2000000
End Sub
```

### 3-7 ثابت ها

ثابت‌ها متغیرهایی هستند که تغییر نمی کنند. روش اعلان ثابت مانند اعلان متغیر است با این تفاوت که مقدار ثابت تعریف شده را نمی توان در طول اجرای کد تغییر داد.

Const Rate=.12

در مدل شیء اکسل ثابت های از پیش تعریف شده ای وجود دارد که با حروف xl آغاز می شوند. همچنین ثابت های تعریف شده در کتابخانه VBA با حروف کوچک vb شروع می شوند.



## ماژول‌ها، توابع و زیرروال‌ها (فصل 4)

Module یا ماژول به طرف یا فضایی گفته می‌شود که کدهای VBA، در قالب یک یا چند رویه (Procedure)، درون آن نگهداری می‌شوند. پروژه‌های VBA حداقل از یک ماژول برای ذخیره‌سازی رویه‌ها استفاده می‌کنند. برای ایجاد یک ماژول، در محیط VBE از منوی Insert گزینه Module را انتخاب کنید.

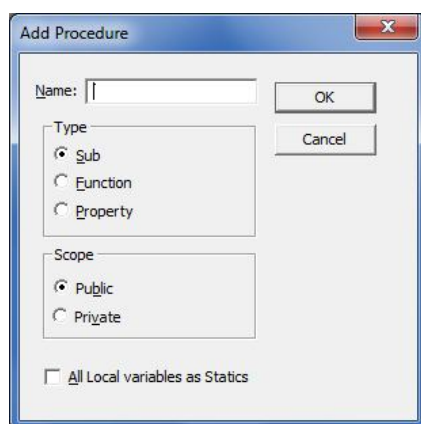
### 4-1 رویه یا Procedure

رویه یا Procedure مجموعه‌ای از کدهای مرتبط با هم است که درون یک ماژول در محیط VBE نوشته شده و عمل خاصی را انجام می‌دهند.

دو نوع رویه (Procedure) در VBA وجود دارد:

1. زیرروال‌ها (Subroutines)
2. توابع (Functions)

یک روش ساده برای ایجاد یک رویه در ماژول جاری این است که در محیط VBE از منوی Insert گزینه Procedure را انتخاب کنید. سپس در پنجره باز شده (شکل ۴-۱) نام، نوع و ویژگی‌های رویه را انتخاب کرده و دکمه OK را کلیک کنید.



شکل ۴-۱

همچنین می‌توانید مستقیماً در محیط ماژول با نوشتن دستورات مورد نیاز اقدام به ایجاد یک رویه کنید.

### 4-2 زیرروال‌ها (Subroutines)

زیرروال یا Subroutine مجموعه‌ای از کدهای VBA است که درون یک ماژول نوشته شده و در زمان اجرا توسط کاربر یا فراخوانی توسط زیرروال دیگر، عمل خاصی را انجام می‌دهد.

زیرروال‌ها همیشه با کلمه کلیدی Sub همراه با نام زیرروال شروع شده و با عبارت End Sub پایان می‌یابند.

شما می‌توانید با استفاده از کلمات کلیدی Public قبل از نام یک رویه، امکان فراخوانی آن رویه را در تمام ماژول‌ها فراهم کنید. همچنین استفاده

از کلمات کلیدی Private قبل از نام یک رویه، باعث می‌شود که فراخوانی رویه مورد نظر، فقط در ماژول جاری امکان پذیر باشد

**نکته:** زیرروال‌های عمومی (Public) را می‌توانید در لیست ماکروهای تعریف شده مشاهده کرده و مورد استفاده قرار دهید.

در تصویر زیر (شکل 2-4)، دو ماکرو (زیرروال) به نامهای Macro1 و Macro3 را مشاهده می‌کنید که درون یک ماژول به نام Module2 درج شده‌اند.

```

Book1 - Module2 (Code)
(General) Macro3
Sub Macro1 ()
    ActiveCell.FormulaR1C1 = "ali"
End Sub
Sub Macro3 ()
    Columns("D:D").Select
    Selection.Delete Shift:=xlToLeft
    Range("F8:H12").Select
    With Selection.Interior
        .Pattern = xlSolid
        .PatternColorIndex = xlAutomatic
        .Color = 65535
    End With
    Range("C17").Select
    ActiveCell.FormulaR1C1 = "sdfsdf"
    Range("C18").Select
End Sub
    
```

شکل ۲-۴

#### 4-2-1 فراخوانی یا اجرای زیرروال

برای فراخوانی یک رویه می‌توانید از کلمه کلیدی Call همراه با نام رویه استفاده کنید.

در شکل ۳-۴ یک زیرروال به نام helloWorld از درون زیر روال sayHello فراخوانی شده است.

```

Sub helloWorld()
    MsgBox "Hello World !"
End Sub
Sub sayHello()
    Call helloWorld
End Sub
    
```

شکل ۳-۴

#### 4-2-2 تعریف آرگومان برای زیرروال

رویه‌ها به طور کلی می‌توانند در جهت انجام عملیات خود مقادیری را به صورت متغیر دریافت کرده و آنها را در محاسبات خود استفاده کنند. برای انتقال مقادیر (همان آرگومان) به یک رویه باید آنها را درون پرانتز اعلان رویه تعریف کنید.

در شکل ۴-۴ زیرروال sayHello با انتقال آرگومان به زیر روال sayMessage، آن را فراخوانی کرده است.

```
Sub SayMessage (message)
  MsgBox message
End Sub
Sub sayHello ()
  Call SayMessage ("hello world !")
End Sub
```

شکل ۴-۴

**نکته 1:** برای اجرای آزمایشی زیرروالهای بدون آرگومان، می‌توانید بعد از کلیک درون زیرروال، دکمه F5 را فشار دهید و یا از منوی Run دستور Run Sub را انتخاب کنید.

**نکته 2:** اگر نوع آرگومان تعریف شده برای یک رویه را مشخص نکنید، نوع پیش فرض Variant برای آن در نظر گرفته خواهد شد.

## 4-3 توابع (Functions)

توابع به لحاظ عملکرد شبیه زیرروالها هستند و تنها تفاوت آنها با زیرروال این است که توابع مقداری را به عنوان نتیجه محاسبه به اجرا کننده خود بر می‌گردانند.

رویه تابع با کلمه کلیدی Function به همراه نام تابع شروع شده و با عبارت کلیدی End Function خاتمه می‌یابد و مانند زیرروال، آرگومانهای آن درون پرانتز معرفی می‌شوند.



مثالهایی از توابع بدون آرگومان :

```
Function sayHello()
  MsgBox "Hello World !"
End Function
```

```
Public Function user()
  ' Return the name of current user
  user = Application.UserName
End Function
```

مثالی از یک تابع با یک آرگومان :

```
Public Function sayHello(nm)
  sayHello = "Hello " & nm
End Function
```

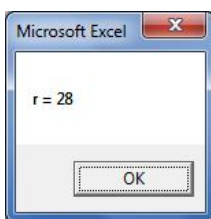
مقدار برگشتی یک تابع با استفاده از نام تابع مشخص می‌شود.

در شکل 5-4 تابع multiply دو آرگومان به نامهای a و b را دریافت کرده و حاصل ضرب آنها را برمی گرداند. سپس این تابع از درون زیرروال macro1 با ارسال آرگومانهای مورد نیاز فراخوانی شده و نتیجه محاسبه درون متغیر r ذخیره شده و اعلان می شود.

```
Function multiply(a, b)
    multiply = a * b
End Function
Sub macro1()
    r = multiply(4, 7)
    MsgBox ("r = " & r)
End Sub
```

شکل 5-4

نتیجه محاسبه :

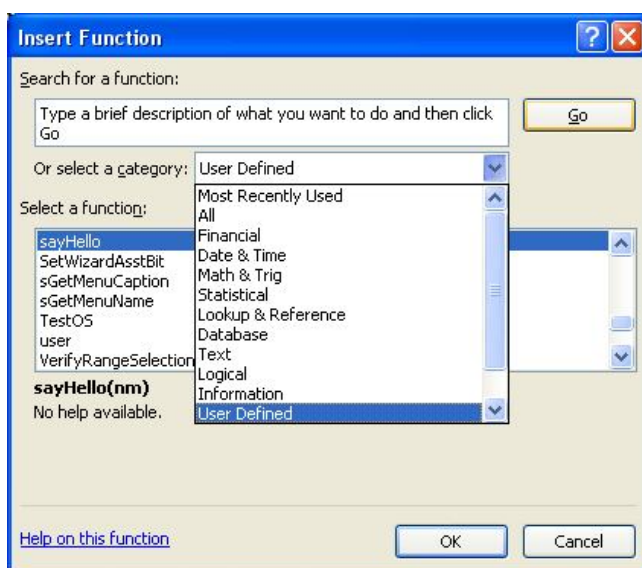


#### 1-3-4 استفاده از تابع تعریف شده در فرمول کاربرگ

رویه های Function را به دو روش می توان اجرا یا فراخوانی کرد :

1. فراخوانی تابع از یک رویه دیگر VBA که در مثال قبل توضیح داده شد.
2. استفاده از تابع در یک فرمول کاربرگ

توابع عمومی تعریف شده را می توانید در گروه User Defined در پنجره Insert Function مشاهده و استفاده کنید.



به عبارت دیگر شما با کسب مهارت لازم در نوشتن کدهای VBA می توانید مجموعه ای از توابع شخصی با ویژگی های دلخواه تعریف کرده و از

آنها در انجام محاسبات اکسل خود بهره ببرید.

### 4-3-2 نوشتن Help یا متن راهنما برای تابع سفارشی

همانند توابع پیش ساخته اکسل شما می‌توانید برای توابعی که خود ساخته اید، متن راهنما بنویسید. مراحل نوشتن Help یا متن راهنما برای تابع سفارشی تعریف شده به شرح زیر می‌باشد:

۱. در روبان Developer روی دکمه Macro کلیک کنید.
۲. نام تابع تعریف شده را در کادر Macro Name وارد کنید.
۳. روی دکمه Options کلیک کنید.
۴. متن راهنما را در کادر Description وارد کرده و دکمه OK را کلیک کنید.
۵. پنجره Macros را ببندید.

بعد از طی این مراحل با انتخاب تابع در پنجره Insert Function متن راهنما در زیر آن نمایش داده می‌شود.

### 4-4 تعریف آرگومان اختیاری برای توابع

برای تعریف آرگومان یک تابع به صورت یک آرگومان اختیاری، باید قبل از نام آرگومان از کلمه کلیدی Optional استفاده کنید. در مثال زیر آرگومان myText اجباری و آرگومان Num به صورت اختیاری تعریف شده است:

```
Function Function1(myText As String, Optional Num As Integer)
End Function
```



گروه‌فردانش

نکته: آرگومانهای اختیاری باید بعد از آرگومانهای اجباری معرفی شوند.

مثال:

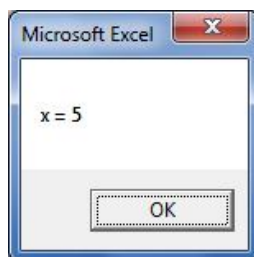
```
Function multiply1(a, Optional b)
If IsMissing(b) Then
multiply1 = a
Exit Function
Else
multiply1 = a * b
End If
End Function
```

### 4-5 نحوه انتقال آرگومان

شما می‌توانید با استفاده از کلمات کلیدی ByVal و ByRef، چگونگی انتقال آرگومان‌ها را به رویه مشخص کنید.

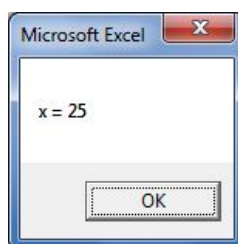
#### 4-5-1 انتقال توسط مقدار

```
Sub SetValue(ByVal i As Integer)
    i = i * 5
End Sub
Sub test()
    Dim x As Integer
    x = 5
    Call SetValue(x)
    MsgBox ("x = " & x)
End Sub
```



#### 4-5-2 انتقال توسط مرجع

```
Sub SetValue(ByRef i As Integer)
    i = i * 5
End Sub
Sub test()
    Dim x As Integer
    x = 5
    Call SetValue(x)
    MsgBox ("x = " & x)
End Sub
```



#### 4-6 مثال هایی از توابع

- مثال : این تابع یک آرگومان عددی دریافت می کند. اگر عدد وارد شده مثبت باشد, عبارت Positive , اگر منفی باشد عبارت Negative و در صورت صفر بودن, عبارت Zero را برمی گرداند.



```

Function numsign(num)
  Select Case num
    Case Is < 0
      numsign = "Negative"
    Case 0
      numsign = "Zero"
    Case Is > 0
      numsign = "Positive"
  End Select
End Function

```

- مثال : این تابع درآمد را دریافت کرده و بر اساس مقدار درآمد، مالیات را (با چهار نرخ) محاسبه می‌کند.

```

Function tax(income)
  Rate1 = 0.1
  Rate2 = 0.15
  Rate3 = 0.2
  Rate4 = 0.25
  Select Case income
    Case 0 To 5000000
      tax = income * Rate1
    Case 5000001 To 10000000
      tax = income * Rate2
    Case 10000001 To 15000000
      tax = income * Rate3
    Case Is > 15000000
      tax = income * Rate4
  End Select
End Function

```



- مثال : این تابع یک دامنه و یک عدد (n) را به عنوان آرگومان دریافت می‌کند و میانگین n عدد بزرگتر از دامنه داده شده را حساب می‌کند.

```

Function TopAvg(Data, n)
  Sum = 0
  For i = 1 To n
    Sum = Sum + WorksheetFunction.Large(Data, i)
  Next i
  TopAvg = Sum / n
End Function

```

## دستورهای کنترلی در VBA (فصل 5)

دستورهای کنترلی و تصمیم گیری زمانی استفاده می شوند که می خواهید حسب شرایط حاکم گزینه های مختلف تصمیم گیری داشته باشید تا برنامه شما انعطاف بیشتر و عملکرد بهتری داشته باشد.

### 5-1 دستور IF

فرم کلی این دستور به شکل زیر است :

**If condition Then**

[statements]

[Else

[elsestatements]]

**End If**

اگر عبارت شرطی بعد از IF درست باشد، جمله های بعد از Then و در غیر این صورت جمله های بعد از Else اجرا می شوند.

مثال : کد زیر مقدار درون سلولهای ناحیه انتخاب شده را بررسی می کند. اگر کمتر از 60 باشد با زمینه قرمز و در غیر این صورت با زمینه آبی نمایش داده می شود.

```
Sub Macro7()
    For Each cell In Selection
        If cell.Value < 60 Then
            cell.Interior.Color = vbRed
        Else
            cell.Interior.Color = vbBlue
        End If
    Next cell
End Sub
```



مثالهایی در مدل های مختلف کاربرد دستور IF

```
Public Sub vcompare()
    a = 4
    b = 5
    If a < b Then MsgBox "a is smaller"
End Sub
```

```
Public Sub vcompare()
    a = 4
    b = 5
    If a < b Then
        MsgBox "a is smaller"
        MsgBox "b is larger"
    End If
End Sub
```

```
Public Sub vcompare()
    a = 4
    b = 5
    If a < b Then MsgBox "a is smaller": MsgBox "b is larger"
End Sub
```

```
Public Sub vcompare()
    a = 4
    b = 5
    If a < b Then
        MsgBox "a is smaller"
    ElseIf a = b Then
        MsgBox "a = b"
    Else
        MsgBox "a is larger"
    End If
End Sub
```

```
Public Sub vcompare()
    a = 4
    b = 5
    c = 8
    If a < b And a < c Then
        MsgBox "a is smallest"
    End If
End Sub
```



## 5-2 دستور Select Case

این دستور زمانی استفاده می‌شود که برای چند حالت مربوط به یک عبارت تصمیم‌گیری کنید. فرم کلی این دستور به شکل زیر می‌باشد:

**Select Case** *testexpression*

[**Case** *expressionlist-n*

[*statements-n*]]

...

[**Case Else**

[*elstatements*]]

**End Select**

مثال: کد زیر مقدار درون سلولهای ناحیه انتخاب شده را بررسی می‌کند. اگر کمتر از 40 باشد به رنگ قرمز، بین 40 و 60 به رنگ زرد، بین 60 و 80 به رنگ سبز، بین 80 و 100 به رنگ آبی و در غیر این صورت به رنگ مشکی نمایش داده می‌شود.

```
Sub Macro7()
    For Each cell In Selection
        Select Case cell
            Case 0 To 40
                cell.Font.Color = vbRed
            Case 41 To 60
                cell.Font.Color = vbYellow
            Case 61 To 80
                cell.Font.Color = vbGreen
            Case 81 To 100
                cell.Font.Color = vbBlue
            Case Else
                cell.Font.Color = vbBlack
        End Select
    Next cell
End Sub
```

مثال دیگر :

```
Private Sub SelectCase()
    Dim Number As Integer
    Number = 18
    Select Case Number
        Case 1 To 5
            MsgBox "Between 1 and 5"
        Case 6, 7, 8
            MsgBox "Between 6 and 8"
        Case 9 To 10
            MsgBox "9 Or 10"
        Case Is <= 20
            MsgBox "Between 11 and 20"
        Case Else
            MsgBox "Not between 1 and 10"
    End Select
End Sub
```



## 3-5 حلقه For-Next

دستورهای نوشته شده درون حلقه For-Next به تعداد دفعات مشخصی (که یک شمارنده آن را کنترل می‌کند) اجرا می‌شوند. فرم کلی حلقه For-Next به شکل زیر است :

```
For counter = start To end [Step step]
[statements]
Exit For
[statements]
Next [counter]
```

مثال 1 : کد زیر با استفاده از حلقه For-Next اعداد 1 تا 1000 را درون خانه های A1 تا A1000 وارد می‌کند :

```
Private Sub for_next1()
    For i = 1 To 1000
        Cells(i, 1).Value = i
    Next i
End Sub
```

مثال 2:

```
Private Sub for_next2()
    For i = 1 To 100 Step 2
        Cells(i, 1).Value = i
    Next i
End Sub
```

مثال 3:

```
Private Sub for_next3()
    For i = 1 To 10
        For j = 1 To 10
            Cells(i, j).Value = "R" & i & "C" & j
        Next j
    Next i
End Sub
```



**نکته:** اگر در زمان اجرای حلقه به دلیلی بخواهید از حلقه خارج شوید، از عبارت Exit For استفاده کنید.

## 4-5 حلقه For Each

حلقه For Each برای پیمایش و پردازش اعضاء مجموعه‌ها یا آرایه‌ها استفاده می‌شود. (مانند مجموعه صفحه‌ها یا مجموعه نمودارها)

این حلقه به شما کمک می‌کند که اعضاء یک مجموعه را به صورت تک تک مرور کنید.

مثال 1: این مثال با استفاده از دستور MsgBox، لیست نام تمام صفحه‌ها را اعلان می‌کند.

```
Private Sub for_each1()
    For Each Worksheet In Worksheets
        MsgBox Worksheet.Name
    Next Worksheet
End Sub
```

مثال 2: مثال زیر متن خانه‌هایی از محدوده a1:e10 که عدد درونشان از 100 بزرگتر است را به رنگ قرمز رنگ آمیزی می‌کند.

```
Private Sub for_each2()
    For Each cell In Range("a1:e10")
        If cell.Value > 100 Then
            cell.Font.Color = vbRed
        End If
    Next cell
End Sub
```

## 5-5 حلقه Do While | Until

دستورهای درون حلقه Do While | Until تا زمان تحقق یا عدم تحقق یک شرط خاص اجرا می شوند.

فرم کلی این حلقه به دو صورت زیر می باشد :

```
Do [{While | Until} condition]
[statements]
[Exit Do]
[statements]
```

Loop

```
Do
[statements]
[Exit Do]
[statements]
```

```
Loop [{While | Until} condition]
```

نکته : برای خروج زود هنگام از این حلقه از دستور Exit Do استفاده می شود.

## 5-6 دستور With—End With

این دستور به کاربر کمک می کند تا بدون تکرار مکرر سلسله مراتب یک شیء، به ویژگی های آن دسترسی داشته باشد. فرم کلی این دستور به شکل زیر می باشد :

```
With object
```

```
[statements]
```

```
End With
```

در مثال زیر با استفاده از ساختار With به ویژگیهای شیء انتخاب شده (Selection) مراجعه شده است :

```
Sub Macro6()
With Application.Selection
.ClearContents
.Value = "Iran"
.Borders(xlEdgeBottom).LineStyle = xlContinuous
End With
End Sub
```



## (فصل 6) توابع و دستوره‌های VBA

### 6-1 توابع ویرایش متن (کلاس Strings)

#### 6-1-1 الحاق رشته های متنی

برای الحاق دو رشته متنی از عملگر & استفاده می‌شود.

مثال 1:

```
Sub Attach_str1()
    Dim n As Integer
    n = Worksheets.Count
    MsgBox "There are " & n & " sheets within the workbook"
End Sub
```

مثال 2:

```
Sub Attach_str2()
    For i = 1 To 5
        MsgBox "The value of i is " & i
    Next i
End Sub
```



#### 6-1-2 جدا کردن بخشی از یک رشته متنی :

تابع MID: این تابع برای جدا کردن بخشی از یک رشته متنی استفاده می‌شود.

تابع LEFT: این تابع برای جدا کردن بخشی از ابتدای یک رشته متنی استفاده می‌شود.

تابع RIGHT: این تابع برای جدا کردن بخشی از انتهای یک رشته متنی استفاده می‌شود.

#### 6-1-3 تغییر شکل حروف در رشته های متنی :

تابع Ucase تمام کارکتر های الفبایی را به صورت حروف بزرگ نمایش می‌دهد.

تابع Lcase تمام کارکتر های الفبایی را به صورت حروف کوچک نمایش می‌دهد.

برای جستجوی یک رشته متنی در متن دیگر از تابع Instr استفاده می شود.

InStr([start, ]string1, string2[, compare])

## 6-2 برخی از توابع کلاس Math

Abs(): قدر مطلق یک مقدار عددی را برمی گرداند.

Sqr(): ریشه دوم یک عدد را بر می گرداند.

Rnd(): در هر بار اجرا یک مقدار تصادفی اعشاری بین صفر و یک تولید می کند.

Round(): مقدار عددی را به مقدار دلخواه گرد می کند.

## 6-3 توابع تاریخ و زمان (کلاس DateTime)

Now: تاریخ و ساعت فعلی را برمی گرداند.

Date: تاریخ فعلی را با توجه به تنظیمات ویندوز بر می گرداند.

DateDiff: فاصله بین دو تاریخ مشخص را به فرمت دلخواه بر می گرداند.

DatePart: قسمت خاصی از یک تاریخ را بر می گرداند.

گروه فاردانesh

## 6-4 معرفی چند دستور مهم

### 6-4-1 دستور Send Keys

دستور SendKeys از متدهای شیء Application است و به شما این امکان را می دهد که فشرده شدن کلیدهای صفحه کلید را توسط کدهای VBA شبیه سازی کنید.

### 6-4-2 دستور Shell

این دستور برای اجرای یک برنامه اجرایی استفاده می شود و فرم کلی آن به صورت زیر است :

Shell(pathname[,windowstyle])

این دستور مقدار برگشتی دارد، به این صورت که اگر برنامه اجرا شود یک مقدار Double که معرف ID برنامه است و در غیر این صورت مقدار صفر را برمی گرداند.

## 6-5 کادر پیام (MsgBox)

دستور MsgBox معمولاً برای ارسال نتایج به کاربر استفاده می شود و همچنین با استفاده از این دستور می توانید از کاربر سوال کرده و ...



ساده ترین شکل استفاده از دستور MsgBox به صورت زیر است :

MsgBox "Hello World !"



فرم کامل دستور MsgBox به صورت زیر می باشد :

R=MsgBox(prompt[, buttons] [, title])

- Prompt : متن پیغام نمایش گذاشته شده
- Buttons : ترکیب دکمه های موجود در کادر پیغام
- Title : عنوان پنجره کادر پیغام
- R : متغیری با نام دلخواه که مقدار برگشتی از این دستور را (بسته به اینکه کاربر کدام دکمه را کلیک می کند) در خود نگهداری می کند.

#### جدول تعیین ترکیب دکمه ها

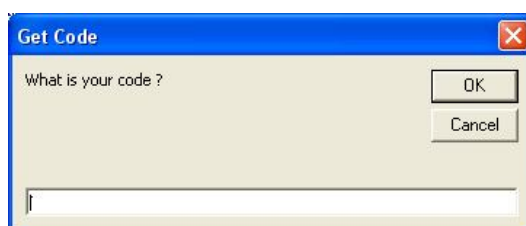
شرح	مقدار	ثابت
Display <b>OK</b> button only.	0	vbOKOnly
Display <b>OK</b> and <b>Cancel</b> buttons.	1	vbOKCancel
Display <b>Abort</b> , <b>Retry</b> , and <b>Ignore</b> buttons.	2	vbAbortRetryIgnore
Display <b>Yes</b> , <b>No</b> , and <b>Cancel</b> buttons.	3	vbYesNoCancel
Display <b>Yes</b> and <b>No</b> buttons.	4	vbYesNo
Display <b>Retry</b> and <b>Cancel</b> buttons.	5	vbRetryCancel
Display <b>Critical Message</b> icon.	16	vbCritical
Display <b>Warning Query</b> icon.	32	vbQuestion
Display <b>Warning Message</b> icon.	48	vbExclamation
Display <b>Information Message</b> icon.	64	vbInformation
First button is default.	0	vbDefaultButton1
Second button is default.	256	vbDefaultButton2
Third button is default.	512	vbDefaultButton3
Fourth button is default.	768	vbDefaultButton4
Application modal; the user must respond to the message box before continuing work in the current application.	0	vbApplicationModal
System modal; all applications are suspended until the user responds to the message box.	4096	vbSystemModal
Adds Help button to the message box.	16384	vbMsgBoxHelpButton
Specifies the message box window as the foreground window.	65536	VbMsgBoxSetForeground

Text is right aligned.	524288	vbMsgBoxRight
Specifies text should appear as right-to-left reading on Hebrew and Arabic systems.	1048576	vbMsgBoxRtlReading

### جدول مقادیر برگشتی از MsgBox

شرح	مقدار	ثابت
OK	1	vbOK
Cancel	2	vbCancel
Abort	3	vbAbort
Retry	4	vbRetry
Ignore	5	vbIgnore
Yes	6	vbYes
No	7	vbNo

## 6-6 پنجره ورودی (InputBox)



پنجره ورودی یک پنجره پیغام است که یک کادر متنی دارد و کاربر می‌تواند در پاسخ به این پنجره، مقداری را وارد کند.

روی پنجره ورودی فقط دکمه‌های OK و Cancel وجود دارند.

فرمت تابع InputBox به صورت زیر است :

```
strAnswer= InputBox(prompt[, title] [, default] [, xpos] [, ypos] [, helpfile, context])
```

آرگومانهای تابع InputBox عبارتند از :

تابع InputBox یک مقدار از نوع Variant را برمی‌گرداند.

اگر کاربر مقداری وارد نکند و یا دکمه Cancel را کلیک کند، تابع InputBox رشته‌ای به طول صفر (معادل "" یا Empty)

را برمی‌گرداند.

```
Sub InputBox1()  
    s1 = InputBox("What is your code ?", "Get Code")  
    If s1 <> Empty Then  
        MsgBox "your code is " & s1  
    Else  
        MsgBox "You didn't enter your code"  
    End If  
End Sub
```



## مدیریت خطاها در برنامه نویسی اکسل (فصل 7)

بروز خطا در یک برنامه می تواند باعث وقفه یا بوجود آمدن نتایج غیر قابل پیش بینی شود. یکی از مهمترین مهارتهای برنامه نویسان ماهر توانایی آنها در یافتن و رفع خطاهای برنامه می باشد.

### 7-1 انواع خطاها

در این بخش انواع خطاهایی که در زمان ایجاد و اجرای یک برنامه ممکن است با آن آشنا شوید به شما معرفی خواهند شد.

#### خطاهای ترجمه (Compile Errors)

خطاهای ترجمه در اثر نوشتن کد های بدون ساختار ایجاد می شوند. به طور مثال اگر از یک ویژگی تعریف نشده برای یک شیء استفاده کنید یا حلقه For بدون Next داشته باشید با این خطا مواجه خواهید شد. به زبان ساده تر این خطاها در اثر رعایت نکردن قوانین برنامه نویسی به وجود می آیند.

#### خطاهای زمان اجرا (Run Time Errors)

این خطاها ارتباطی به قوانین زبان VBA ندارند و تنها در زمان اجرا خود را نشان می دهند. به طور مثال اگر در کد خود بخواهید فایلی را باز کنید که وجود ندارد یا در محاسبات خود تقسیم بر صفر داشته باشید، با این خطا مواجه خواهید شد.



خطاهای منطقی در واقع خطا نیستند بلکه به حالتی می گویند که نتیجه اجرای برنامه با چیزی که شما انتظار داشته اید متفاوت بوده است و یافتن علت این نوع خطاها سخت ترین کار در مسیر برنامه نویسی می باشد.

### 7-2 حالت های یک برنامه VBA

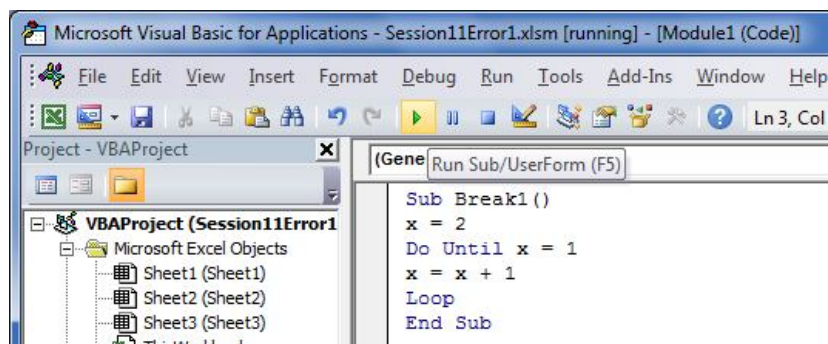
در زمان کار بر روی یک برنامه می توانید در یکی از سه وضعیت زیر قرار داشته باشید :

#### زمان طراحی

زمان طراحی یا Design Time زمانی است که بر روی یک کد کار می کنید یا فرمی را طراحی می کنید.

#### زمان اجرا

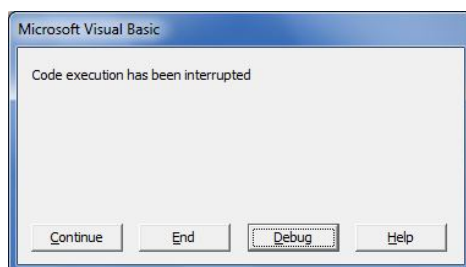
در زمان اجرای برنامه یا Runtime ممکن است که حتی کد را ببینید ولی نمی توانید بر روی آن تغییری انجام دهید. توجه کنید که در زمان اجرای برنامه در نوار عنوان پنجره VBE عبارت running دیده می شود. (شکل ۷-۱)



شکل ۱-۷

زمان وقفه

اگر در زمان اجرای برنامه کلیدهای **Ctrl+Break** را فشار دهید، اجرای کد متوقف شده و با پنجره زیر مواجه خواهید شد.



شکل ۲-۷

با کلیک روی کلید **Debug** به پنجره کد نویسی می روید و قسمتی از کد که در آن برنامه متوقف شده به رنگ زرد نمایش می یابد.

به کد زیر توجه فرمایید.

```

Sub Break1()
x = 2
Do Until x = 1
x = x + 1
Loop
End Sub

```

با توجه به اینکه  $x$  در زمان شروع حلقه مقدار 2 را دارد و درون حلقه به آن اضافه می شود،  $x$  هیچ وقت برابر 1 نخواهد شد. لذا اگر برنامه اجرا شود در یک حلقه بی پایان خواهیم افتاد. برنامه را اجرا می کنیم و سپس کلیدهای **Ctrl+Break** را فشار می دهیم. در پنجره نمایش داده شده دکمه **Debug** را کلیک کرده و به کد بر می گردیم.

توجه شما را به دو نکته در پنجره کد (شکل ۳-۷) جلب می کنم:

- سطری که کد در محل آن متوقف شده به رنگ زرد نمایش داده می شود.
- اگر در این حالت نشانه گر را بر روی یکی از متغیرها ببرید، مقدار فعلی آن را به صورت tooltip مشاهده خواهید کرد.

```
Sub Break1()
  x = 2

  Do Until x = 1
    x = 19482462
    x = x + 1
  Loop

End Sub
```

شکل ۳-۷

در این حالت می توانید با فشار دادن کلید F5 اجرای کد را ادامه دهید یا با انتخاب دستور Reset از منوی Run اجرا را متوقف کنید.

## 3-7 دستور GoTo

این دستور مستقیماً ارتباطی به خطاهای VBA ندارد و جزء دستورهای برنامه نویسی می باشد ولی به علت استفاده زیاد آن در مدیریت خطاها در این بخش آن را ذکر می کنیم. دستور GoTo برای رفتن به بخش خاصی از کد که قبلاً نام گذاری شده استفاده می شود.

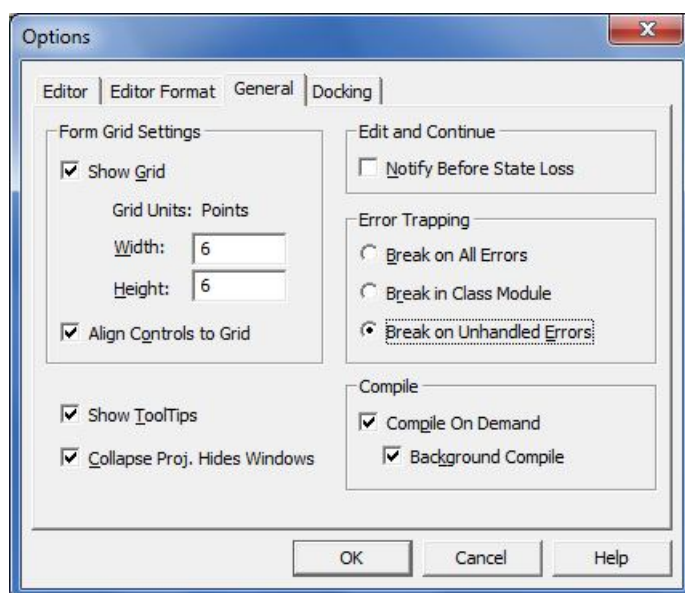
در کد زیر وقتی ط به مقدار 100 می رسد یک پرش اتفاق افتاده و اجرا به خطی می رود که با نام exitLine نام گذاری شده است و دستورات بعد از آن اجرا می شود.

```
Sub GoTo1()
  x = 2
  Do Until x = 1
    x = x + 1
    If x = 100 Then GoTo exitLine
  Loop
exitLine:
  MsgBox "x=100. Exit Time !!!"
Exit Sub
End Sub
```

## 4-7 مدیریت خطاهای زمان اجرا

ادامه موضوع ما در این فصل مدیریت خطاهای زمان اجرا می باشد. در زمان وقوع این نوع خطا برنامه به حالت وقفه رفته و یک پنجره پیغام نمایش می یابد.

قبل از هر چیز در محیط VBE از منوی Tools گزینه Options را انتخاب کنید. سپس در برگه General از پنجره باز شده در بخش Error Trapping گزینه سوم را انتخاب کنید. (شکل ۴-۷)



شکل ۷-۴

خطاهای زمان اجرا معمولاً به دو صورت مدیریت می شوند.

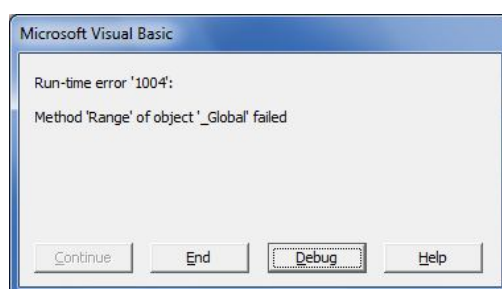
- خطا نادیده گرفته شده و اجازه دهیم که اجرای کد ادامه یابد.
- در صورت بروز خطا پرش به ناحیه خاصی از کد اتفاق بیفتد.

## 7-5 دستور Resume

کد زیر درون رویداد SheetActivate از Workbook نوشته شده و کاری که می کند این است که در صورت فعال کردن یک صفحه، خانه A1 از صفحه فعال را انتخاب می کند.

```
Private Sub Workbook_SheetActivate(ByVal Sh As Object)
Range("A1").Select
End Sub
```

حال فرض کنید که صفحه فعال شده از نوع صفحه نمودار باشد. در این حالت خانه A1 دیگر وجود ندارد و با پیغام خطای زیر مواجه خواهید شد.



ساده ترین روش برای حل این مشکل استفاده از دستور Resume به صورت زیر می باشد.

```
Private Sub Workbook_SheetActivate(ByVal Sh As Object)
On Error Resume Next
Range("a1").Select
End Sub
```

در کد بالا، دستور On Error Resume Next باعث می شود که در صورت بروز خطا، خطا نادیده گرفته شده و ادامه برنامه از سطر بعد از خطا اجرا شود.

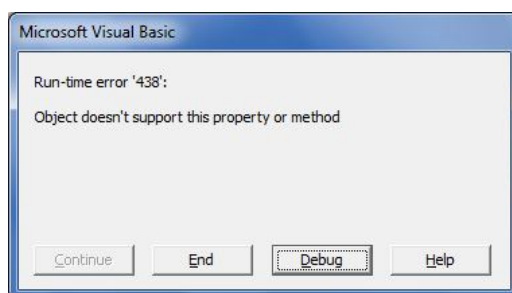
## 7-6 مدیریت خطا توسط GoTo

با استفاده از این دستور و ترکیب آن با دستور OnError می توانید در صورت وقوع خطا اجرا را به محل دلخواه از کد منتقل کنید. به مثال زیر توجه فرمائید :

مثال: کد زیر درون ناحیه انتخاب شده عدد 5 را می نویسد.

```
Sub ErrorDemo1()
Selection.Value = 5
End Sub
```

حال فرض کنید که به جای انتخاب یک ناحیه از صفحه، یک Shape یا نمودار روی صفحه را انتخاب کرده و کد بالا را اجرا کنید. در این صورت با پیغام زیر مواجه خواهید شد.



در کد زیر توسط دستور GoTo مشکل حل شده و به جای پیغام بالا و بروز خطا، توسط یک MsgBox بروز مشکل به کاربر اطلاع داده شده و اجرا متوقف می شود.

```
Sub ErrorDemo2()
On Error GoTo Handler
Selection.Value = 5
Exit Sub
Handler:
MsgBox "Cannot assign a value to the selection".
End Sub
```

نکته : کد زیر با استفاده از شماره خطا و متن خطا پیغام مفید تری را به کاربر نشان می دهد.

```
Sub ErrorDemo3()
On Error GoTo Handler
Selection.Value = 5
Exit Sub
Handler:
Msg = "Error " & Err.Number & ": " & Error(Err.Number)
Msg = Msg & vbNewLine
```



```
Msg = Msg & "Cannot assign a value to the selection".
MsgBox Msg
End Sub
```

## 7-7 مثال های مدیریت خطا

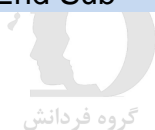
### مثال 1:

کد زیر را در حالتی اجرا کنید که صفحه ای به نام Sheet2 ندارید. طبیعتاً با خطا مواجه خواهید شد

```
Sub ErrorDemo4()
Worksheets("Sheet2").Name = "Sheet10"
End Sub
```

خطای به وجود آمده را می توانید با دستورات زیر کنترل کنید.

```
Sub ErrorDemo4()
On Error Resume Next
Worksheets("Sheet2").Name = "Sheet10"
If Err.Number <> 0 Then
MsgBox "Error " & Err.Number & ": " & Error(Err.Number)
Exit Sub
End If
End Sub
```



گروه فردانش

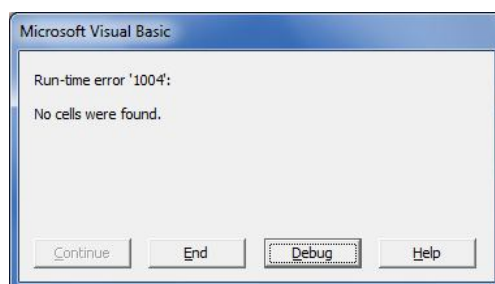
توضیح: اگر Err.Number مقدار صفر را برگرداند به این معنی است که خطایی رخ نداده است.

### مثال 2:

ماکروی زیر خانه های دارای فرمول را با استفاده از متد SpecialCells (شبه پنجره GoTo Special) انتخاب می کند.

```
Sub SelectFormulas1()
Selection.SpecialCells(xlFormulas).Select
' ...[more code goes here]
End Sub
```

حال اگر در صفحه خانه دارای فرمول وجود نداشته باشد با خطای زیر مواجه می شوید :



خطای بالا را می توان با ماکروی زیر مدیریت کرد:

```
Sub SelectFormulas2()
```

```
On Error Resume Next
Selection.SpecialCells(xlFormulas).Select
On Error GoTo 0
' ...[more code goes here]
End Sub
```

توضیح : دستور ON Error GoTo 0 روال های تعریف شده برای پیگیری خطا را لغو می کند و اجازه می دهد که تمام خطاها مانند روال معمول خود دیده شوند.

البته ماکروی زیر نیز روش دیگری برای مدیریت خطای بالا است که عدم وجود خانه های حاوی فرمول را اعلان می کند.

```
Sub SelectFormulas3()
On Error Resume Next
Selection.SpecialCells(xlFormulas).Select
If Err.Number = 1004 Then MsgBox "No formula cells were found."
On Error GoTo 0
' ...[more code goes here]
End Sub
```

### مثال 3 :

```
Sub CheckForFile()
Dim FileName As String
Dim x As Workbook
FileName = "BUDGET.XLSX"
On Error Resume Next
Set x = Workbooks(FileName)
If Err = 0 Then
MsgBox FileName & " is open."
Else
MsgBox FileName & " is not open."
End If
On Error GoTo 0
End Sub
```

## مدل شیء Excel (فصل 8)

در فهرست زیر با برخی از اشیاء اصلی در مدل شیء اکسل آشنا می‌شوید :

- **Application** : شیء مادر در مدل شیء اکسل است و به برنامه اکسل اشاره می‌کند.
- **Windows** : شیء گروهی است و به مجموعه پنجره های باز در محیط اکسل اشاره می‌کند.
- **Charts** : شیء گروهی است و به مجموعه نمودارها در محیط اکسل اشاره می‌کند.
- **Names** : شیء گروهی است و به مجموعه محدوده های نام گذاری شده در محیط اکسل اشاره می‌کند.
- **Worksheets** : شیء گروهی است و به مجموعه صفحه های موجود در فایل جاری اشاره می‌کند.
- **Range** : یکی از مهمترین اشیاء اکسل است و برای اشاره به یک خانه یا دامنه از آن استفاده می‌شود.
- **Columns** : شیء گروهی است و به مجموعه پنجره های باز در محیط اکسل اشاره می‌کند.
- **Rows** : شیء گروهی است و به مجموعه پنجره های باز در محیط اکسل اشاره می‌کند.
- **Worksheetfunction** : مجموعه توابع تعریف شده در محیط اکسل به صورت ویژگی های این شیء در دسترس می‌باشند.

### ساختار سلسله مراتبی اشیاء

در مدل شیء اکسل اشیاء به صورت سلسله مراتبی مرتب می‌شوند، به این صورت که هر شیء در بالا یا پایین شیء دیگر قرار دارد. در بالای مدل شیء اکسل شیء **Application** قرار دارد که همان برنامه اکسل است و اشیاء دیگر از آن منشعب می‌شوند. در ساختار سلسله مراتبی اشیاء با نقطه از هم جدا می‌شوند.

کد زیر خانه A1 از صفحه Sheet2 از فایل Book1 را اشاره می‌کند.

```
Workbooks("Book1").Worksheets("Sheet2").Range("A1")
```

### مجموعه ها

برخی از اشیاء در مدل شیء اکسل ماهیتی واحد دارند مانند شیء **Application** و برخی اشیاء ماهیت مجموعه ای دارند.

برای دسترسی به یک شیء در یک مجموعه از اشیاء می‌توانید موقعیت عددی شیء در مجموعه را مشخص کنید یا اینکه از طریق نام شیء به آن دسترسی داشته باشید. به طور مثال اگر صفحه اول شما Sheet1 باشد، دو سطر زیر هر دو به خانه A1 از Sheet1 اشاره می‌کنند.

```
Worksheets(1).Range("A1")
Worksheets("sheet1").Range("A1")
```

### اشیاء فعال

کد زیر به خانه A1 از Sheet2 اشاره می‌کند.

```
Worksheets("Sheet2").Range("A1")
```

اگر به جای کد بالا عبارت زیر را بنویسید، خانه A2 از صفحه فعال شما مورد اشاره قرار می گیرد

## Range("A1")

### ویژگی های اشیاء

هر شیء دارای یک سری از Properties یا خصوصیات است. بعضی از خصوصیتها فقط قابل خواندن هستند به این معنا که فقط می توانید مقدار آنها را مشاهده کنید و نمی توانید آنها را تغییر دهید. به طور مثال برای شیء Range خصوصیات Row و Column فقط خواندنی هستند در حالیکه خصوصیت Formula قابل تغییر می باشد. برای اشاره به ویژگی یک شیء، بعد از نام شیء یک نقطه و بعد از آن نام ویژگی را وارد کنید.

مثال :

## Range("a1").Formula = "=SUM(D1:D10)"

### متد های اشیاء

اشیاء علاوه بر ویژگی، دارای Method نیز می باشند. متد یا Method به اعمالی گفته می شود که می توان بر روی یک شیء انجام داد. در کد زیر با استفاده از متد ClearContents محتوای محدوده A1:A10 پاک شده است.

## Range("A1:A10").ClearContents

گروه فردانش

## 8-1 خصوصیات و متدها

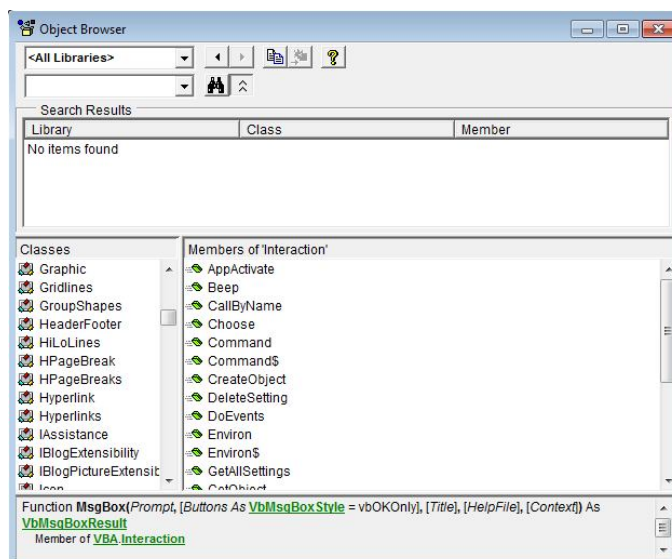
خصوصیات یک شیء به دو دسته تقسیم می شوند :

1. خصوصیات فقط خواندنی (read only) مانند ویژگی Count برای Sheets که تعداد صفحه ها را در خود نگه می دارد.
2. خصوصیات خواندنی/نوشتنی که امکان تغییر دادن و خواندن آنها در هر زمان فراهم است.

خصوصیات یک شیء معمولا با استفاده از کد و در زمان اجرا تغییر می کنند. البته امکان دسترسی به بعضی از خصوصیات اشیاء در زمان طراحی نیز وجود دارد

## 8-2 استفاده از Object Browser

Object Browser یک ابزار مفید برای مرور ویژگی ها و متد های یک شیء است.



## 8-3 شیء Application

### ویژگی های Active

نام ویژگی	توضیح
ActiveCell	خانه فعال
ActiveChart	نمودار فعال
ActiveSheet	صفحه فعال
ActiveWindow	پنجره فعال
ActiveWorkBook	فایل Workbook فعال
ThisWorkBook	فایلی که کد های VBA از آن اجرا شده است
Selection	شیء انتخاب شده (می تواند یک ناحیه از سلول ها، نمودار یا هر چیز دیگری باشد)

### ویژگی Selection

این ویژگی شیء انتخاب شده را برمی گرداند. (می تواند یک ناحیه از سلول ها، نمودار یا هر چیز دیگری باشد)

### ویژگی DisplayAlerts

این ویژگی برای فعال یا غیر فعال کردن پنجره های هشدار استفاده می شود.

در کد زیر ابتدا هشدار ها غیر فعال شده و سپس صفحه فعال حذف شده است.

```
Sub objTest1()
Application.DisplayAlerts = False
ActiveSheet.Delete
Application.DisplayAlerts = True
End Sub
```

**متد Evaluate**

این متد برای محاسبات توابع و پردازش‌آدرس محدوده های صفحه استفاده می شود. به مثال های زیر توجه فرمائید.

```
MsgBox Evaluate("=ISBLANK(A1)")
```

```
Evaluate("A1").Value = 10  
[A1].Value = 10
```

```
Sub evaluateTest2()  
Dim sFunctionName As String, sCellReference As String  
sFunctionName = "ISBLANK"  
sCellReference = ActiveCell.Address  
MsgBox Evaluate(sFunctionName & "(" & sCellReference & "(" &  
End Sub
```

**متد OnTime**

این متد یک یک ماکروی خاص را در زمان خاص اجرا می کند.

مثال 1: کد زیر 15 ثانیه بعد از زمان اجرا ماکرویی به نام my\_Procedure را اجرا می کند.

```
Sub OnTimeTest1()  
Application.OnTime Now + TimeValue("00:00:15"), "my_Procedure"  
End Sub
```

مثال 2: کد زیر در ساعت 5 بعد از ظهر ماکرویی به نام my\_Procedure را اجرا می کند.

```
Sub OnTimeTest2()  
Application.OnTime TimeValue("17:00:00"), "my_Procedure"  
End Sub
```

**8-4 شیء Workbook****متد Add**

این متد یک Workbook جدید ایجاد می کند.

مثال 1:

کد زیر دو فایل جدید ایجاد کرده و بعد از ذخیره آنها به ترتیب، فایل اول را فعال می کند.

```
Sub Addwb1()
Workbooks.Add
ActiveWorkbook.SaveAs Filename:="C:\Data1.xlsx"
Workbooks.Add
ActiveWorkbook.SaveAs Filename:="C:\Data2.xlsx"
Workbooks("Data1.xlsx").Activate
End Sub
```

مثال 2:

```
Sub Addwb2()
Dim wkb1 As Workbook
Dim wkb2 As Workbook
Set wkb1 = Workbooks.Add
Set wkb2 = Workbooks.Add
wkb1.Activate
End Sub
```

### متد Open

این متد یک فایل اکسل را باز می کند.

```
Sub Openwb1()
Set wkb1 = Workbooks.Open(Filename:="C:\Data1.xlsx")
End Sub
```

### متد SaveAs

این متد برای ذخیره فایل با نام دلخواه استفاده می شود.

### متد Close

این متد برای بستن فایل استفاده می شود.

مثال: ماکروی زیر فایل Data1 در درایو C را باز کرده و در خانه A1 از صفحه فعال آن تاریخ روز را ثبت کرده و بعد از ذخیره فایل را میبندد.

```
Sub CloseWorkbook()
Dim wkb1 As Workbook
Set wkb1 = Workbooks.Open(Filename:="C:\Data1.xlsx")
Range("A1").Value = Format(Date, "ddd mmm dd, yyyy")
Range("A1").EntireColumn.AutoFit
wkb1.Save
wkb1.Close
Set wkb1 = Nothing
End Sub
```

**متد Activate**

این متد دفتر کار انتخاب شده را فعال کرده و به طور خودکار به صفحه فعال در آن دفتر کار می رود.

**8-5 شیء Range**

احتمالا پر کاربردترین شیء VBA است و می تواند به یک خانه و یک یا چند دامنه اشاره کند. مثال :

```
Range("B2")
Range("A1:D10")
Range("A1:A10,C1:C10,E1:E10")
```

همچنین می توانید به جای آدرس یک دامنه از نامی که برای دامنه تعریف کرده اید استفاده کنید.

```
Range("SalesData")
```

برای اشاره به یک دامنه در صفحه دیگر به ترتیب زیر عمل می کنیم.

```
Worksheets("Sheet1").Range("C10")
```

کد زیر درون خانه پایین خانه فعال مقدار 5 را درج می کند.

```
ActiveCell.Range("A2") = 5
```

**8-5-1 متدهای Select و Activate**

متد Select برای Range باعث انتخاب دامنه می شود در حالی که متد Activate یک خانه را به عنوان خانه فعال انتخاب می کند.

توجه کنید که از بین خانه های انتخاب شده خانه ای فعال است که با عمل درج متن درون آن انجام شود. در کد زیر ناحیه ای انتخاب شده و سلول C8 به عنوان سلول فعال مشخص شده است.

```
Sub test1()
Range("A5:F11").Select
Range("C8").Activate
End Sub
```



	A	B	C	D	E	F	G
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							

### 2-5-8 ویژگی Resize

ویژگی Resize برای تغییر اندازه محدوده Range استفاده می شود. دامنه جدید از گوشه بالا و چپ ناحیه قبل شروع می شود.

```
Range("a1:d10").Resize(2, 3) === Range("a1:c2")
```

### 6-8 شیء Cells

ویژگی Cells برای اشیاء Application, Worksheet و Range استفاده می شود و به خانه های درون صفحه یا دامنه اشاره می کند.

```
ActiveSheet.Cells
Application.Cells
Cells
Range("A1:D10").Cells
```



کد زیر درون خانه C2 از صفحه Sheet1 مقدار 4 را درج می کند.

```
Worksheets("Sheet1").Cells(2,3)=4
```

کد زیر درون خانه پایین خانه فعال مقدار 5 را درج می کند.

```
ActiveCell.Cells(2, 1) = 5
```

در کد زیر با استفاده از ویژگی Offset در خانه بالای خانه فعال مقدار 12 درج می شود.

```
ActiveCell.Offset(-1,0).Value = 12
```

کد زیر درون خانه فعال و دو خانه پایین آن اعداد 1,2,3 را درج می کند.

```
Sub Macro1()
ActiveCell = 1
ActiveCell.Offset(1, 0) = 2
ActiveCell.Offset(2, 0) = 3
End Sub
```

مثال :

```
Sub FillCells()
    Dim lRow As Integer, lColumn As Integer
    For lRow = 1 To 10
        For lColumn = 1 To 5
            Cells(lRow, lColumn).Value = lRow * lColumn
        Next lColumn
    Next lRow
End Sub
```

نتیجه به صورت زیر خواهد بود.

	A	B	C	D	E	F
1	1	2	3	4	5	
2	2	4	6	8	10	
3	3	6	9	12	15	
4	4	8	12	16	20	
5	5	10	15	20	25	
6	6	12	18	24	30	
7	7	14	21	28	35	
8	8	16	24	32	40	
9	9	18	27	36	45	
10	10	20	30	40	50	
11						

## 8-7 مثالهای دیگر

ماکروی زیر در هر اجرا، یک صفحه به نام Result1,Result2,... ایجاد می کند :

```
Sub NewSheetResult()
    Static Num As Integer
    Dim newSheet As Worksheet
    Set newSheet = Worksheets.Add
    Num = Num + 1
    newSheet.Name = "Result" & Num
End Sub
```

ماکروی زیر تمام صفحه های فایل جاری را بر اساس حروف الفبا مرتب می کند.

```
Sub SortSheets()
    For i = 1 To Sheets.Count
        For j = 1 To Sheets.Count - 1
            If Sheets(j).Name > Sheets(i).Name Then
                Sheets(j).Move after:=Sheets(j + 1)
            End If
        Next j
    Next i
End Sub
```

ماکرو زیر پس از اجرا آدرس و فرمول همه خانه های حاوی فرمول از صفحه فعال را در یک صفحه جدید به نام Resault نمایش می دهد. کد زیر را وارد کرده و نتیجه را مشاهده نمایید.

```
Sub Macro10()
    s = ActiveSheet.Name
    Sheets.Add
    ActiveSheet.Name = "Resault"
    Range("a1").Value = "Address"
    Range("b1").Value = "Formula"
    Worksheets(s).Activate
    i = 2
    For Each cell In ActiveSheet.UsedRange
        If cell.HasFormula Then
            Worksheets("Resault").Range("a" & i).Value = cell.Address _
                (RowAbsolute:=False, ColumnAbsolute:=False)
            Worksheets("Resault").Range("b" & i).Value = " " & cell.Formula
            i = i + 1
        End If
    Next cell
    Worksheets("Resault").Activate
End Sub
```



## رویدادها (فصل 9)

رویه های نوشته شده در VBA زمانی که کاربر بخواهد اجرا می شوند. اکسل این قابلیت را دارد تا در زمان وقوع یک رویداد (مانند اضافه شدن یک صفحه) رویه ای را اجرا کند. به رویه های اینچنینی که در زمان وقوع یک رویداد به طور خودکار فراخوانی می شوند، رویه های رویدادی یا Event Procedure گفته می شود.

در اکسل برای اشیائی مانند Worksheet رویه هایی تعریف شده است که در ادامه با آنها آشنا خواهید شد.

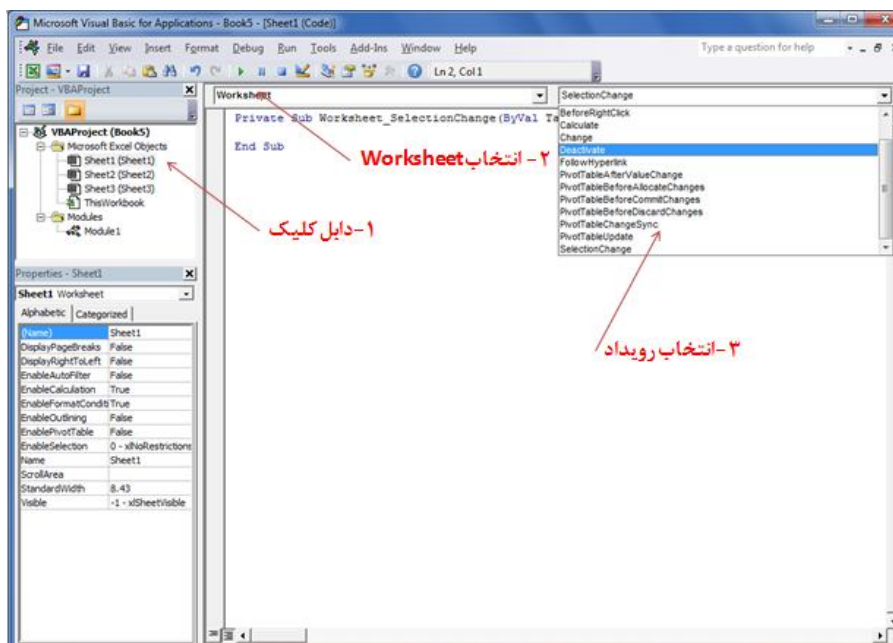
### 9-1 رویدادهای Worksheet

برای شیء Worksheet رویدادهای زیر تعریف شده است :

- Private Sub Worksheet\_Activate()
- Private Sub Worksheet\_BeforeDoubleClick(ByVal Target As Range, Cancel As Boolean)
- Private Sub Worksheet\_BeforeRightClick(ByVal Target As Range, Cancel As Boolean)
- Private Sub Worksheet\_Calculate()
- Private Sub Worksheet\_Change(ByVal Target As Range)
- Private Sub Worksheet\_Deactivate()
- Private Sub Worksheet\_FollowHyperlink(ByVal Target As Hyperlink)
- Private Sub Worksheet\_PivotTableUpdate(ByVal Target As PivotTable)
- Private Sub Worksheet\_SelectionChange(ByVal Target As Range)



برای دسترسی به رویدادهای یک Worksheet ابتدا در محیط VBE و در پنجره Project Explorer روی نام Worksheet مورد نظر دابل کلیک کنید تا ماژول Worksheet باز شود.



شکل ۱-۹

سپس مطابق شکل ۱-۹ رویداد مورد نظر را از لیست رویدادهای Worksheet انتخاب کنید.

### 1-1-9 رویداد BeforeRightClick

این رویداد قبل از راست کلیک بر روی صفحه اکسل فراخوانی می شود.

**مثال:** کد زیر راست کلیک بر روی صفحه را غیر فعال کرده و توسط MsgBox به کاربر این موضوع را اطلاع می دهد.



```
Private Sub Worksheet_BeforeRightClick(ByVal Target As Range, Cancel As Boolean)
Cancel = True
MsgBox "The shortcut menu is not available".
End Sub
```

توضیح: آرگومان Cancel برای لغو کردن رویداد استفاده می شود و آرگومان Target محدوده ای که روی آن راست کلیک شده است را بر می گرداند.

**مثال:** در مثال زیر اگر محتوای خانه ای که روی آن راست کلیک می شود عدد باشد، برگه Numbers از پنجره Format Cell نمایش داده می شود.

```
Private Sub Worksheet_BeforeRightClick (ByVal Target As Range, Cancel As Boolean)
If IsNumeric(Target) And Not IsEmpty(Target) Then
Application.CommandBars.ExecuteMso ("NumberFormatsDialog")
Cancel = True
End If
End Sub
```

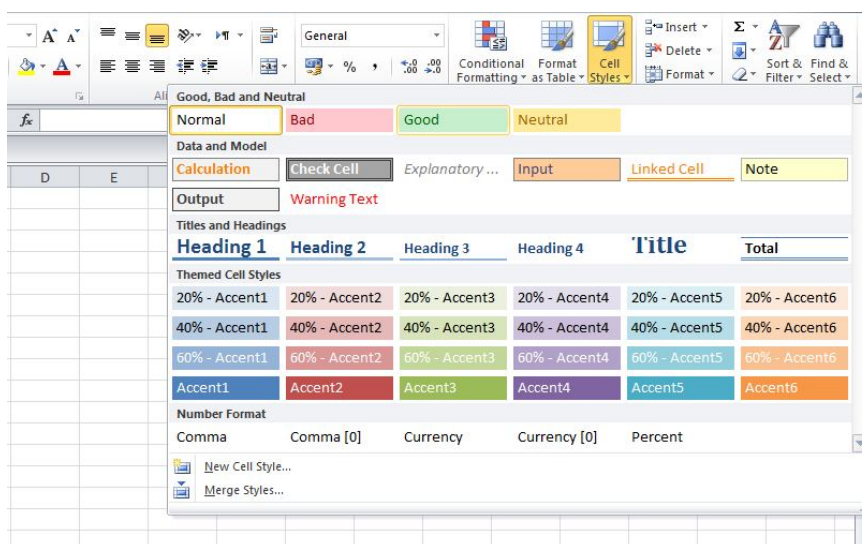
## 2-1-9 رویداد BeforeDoubleClick

این رویداد قبل از دابل کلیک بر روی صفحه اکسل فراخوانی می شود.

مثال : در کد زیر با دابل کلیک روی یک خانه، سبک اعمال شده از سبک Good به Normal و بالعکس تغییر می کند.

```
Private Sub Worksheet_BeforeDoubleClick (ByVal Target As Range, Cancel As Boolean)
If Target.Style = "Good" Then
    Target.Style = "Normal"
Else
    Target.Style = "Good"
End If
Cancel = True
End Sub
```

یاد آوری : سبک های تعریف شده در اکسل در ابزار Cell Style از روبان Home قابل دسترس می باشند. (شکل ۲-۹)



شکل ۲-۹

## 3-1-9 رویداد SelectionChange

این رویداد زمانی فراخوانی می شود که انتخاب محدوده جدیدی در صفحه اتفاق بیفتد.

مثال : کد زیر بعد از انتخاب یک محدوده در صفحه، زمینه سطر و ستون شامل خانه فعال را به رنگ قرمز آمیزی می کند.

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
Cells.Interior.ColorIndex = xlNone
With ActiveCell
.EntireRow.Interior.Color = vbRed
.EntireColumn.Interior.Color = vbRed
End With
End Sub
```

نتیجه اجرای کد :

	A	B	C	D	E	F	G	H	I	J	K
1											
2											
3											
4											
5											
6											
7											
8											
9											
10											
11											
12											
13											
14											

### 4-1-9 رویداد Change

این رویداد زمانی فراخوانی می شود که محتوای محدوده ای در صفحه توسط کاربر یا کدهای VBA تغییر کند.

مثال: کد زیر آدرس دامنه ای که محتوای آن تغییر کرده را توسط دستور MsgBox اعلان می کند.

```
Private Sub Worksheet_Change(ByVal Target As Range)
MsgBox "Range " & Target.Address & " was changed."
End Sub
```



مثال: در کد زیر اگر محتوای خانه ای که تغییر یافته یک فرمول باشد، به صورت ضخیم (Bold) نمایش داده می شود.

```
Private Sub Worksheet_Change(ByVal Target As Range)
Dim cell As Range
For Each cell In Target
cell.Font.Bold = cell.HasFormula
Next cell
End Sub
```

نکته : کد بالا یک اشکال خیلی مهم دارد. اگر یک سطر یا ستون به صفحه اضافه کنید تمام سطر یا ستون اضافه شده را به عنوان ناحیه تغییر یافته در نظر می گیرد و این باعث می شود که اجرای حلقه For Each زمان زیادی بگیرد.

در کد زیر که بهبود یافته کد بالا می باشد توسط تابع Intersect ناحیه مشترک بین ناحیه تغییر یافته و UsedRange به عنوان ناحیه مورد بررسی ذخیره شده و حلقه For Each بر روی آن اجرا می شود.

```
Private Sub Worksheet_Change(ByVal Target As Range)
Dim cell As Range
Set Target = Intersect(Target, Target.Parent.UsedRange)
If Not Target Is Nothing Then
For Each cell In Target
cell.Font.Bold = cell.HasFormula
Next cell
End If
End Sub
```

یک توضیح دیگر اینکه Target.Parent به صفحه ای اشاره می کند که تغییر در آن اتفاق افتاده است.

## 9-2 رویدادهای Workbook

تعدادی از رویدادهای تعریف شده برای شیء Workbook عبارتند از :

- Private Sub Workbook\_Activate()
- Private Sub Workbook\_AddinInstall()
- Private Sub Workbook\_AddinUninstall()
- Private Sub Workbook\_BeforeClose(Cancel As Boolean)
- Private Sub Workbook\_BeforePrint(Cancel As Boolean)
- Private Sub Workbook\_BeforeSave(ByVal SaveAsUI As Boolean, Cancel As Boolean)
- Private Sub Workbook\_Deactivate()
- Private Sub Workbook\_NewSheet(ByVal Sh As Object)
- Private Sub Workbook\_Open()
- Private Sub Workbook\_SheetActivate(ByVal Sh As Object)
- Private Sub Workbook\_SheetBeforeDoubleClick(ByVal Sh As Object, ByVal Target As Range, Cancel As Boolean)
- Private Sub Workbook\_SheetBeforeRightClick(ByVal Sh As Object, ByVal Target As Range, Cancel As Boolean)
- Private Sub Workbook\_SheetCalculate(ByVal Sh As Object)
- Private Sub Workbook\_SheetChange(ByVal Sh As Object, ByVal Target As Range)
- Private Sub Workbook\_SheetDeactivate(ByVal Sh As Object)
- Private Sub Workbook\_SheetFollowHyperlink(ByVal Sh As Object, ByVal Target As Hyperlink)
- Private Sub Workbook\_SheetSelectionChange(ByVal Sh As Object, ByVal Target As Range)
- Private Sub Workbook\_WindowActivate(ByVal Wn As Window)
- Private Sub Workbook\_WindowDeactivate(ByVal Wn As Window)
- Private Sub Workbook\_WindowResize(ByVal Wn As Window)



### 9-2-1 رویداد Open

این رویداد در زمان باز شدن فایل فراخوانی می شود.

**مثال:** در کد زیر اگر فایل در روز پنجشنبه باز شود، توسط دستور MsgBox تهیه نسخه پشتیبان یاد آوری می شود.

```
Private Sub Workbook_Open()
If Weekday(Now) = vbThursday Then
MsgBox "Today is thursday. Make sure that you do your weekly backup!"
End If
End Sub
```

### 9-2-2 رویداد Activate

این رویداد زمانی اجرا می شود که پنجره فایل باز شده، فعال شود.



**مثال:** کد زیر در زمان فعال شدن فایل، پنجره آن را به اندازه maximize تبدیل می کند.

```
Private Sub Workbook_Activate()
ActiveWindow.WindowState = xlMaximized
End Sub
```

### 3-2-9 رویداد SheetActivate

این رویداد در زمان فعال شدن یک صفحه از فایل فراخوانی می شود.

**مثال:** کد زیر در زمان فعال شدن یک صفحه، اگر از نوع Worksheet باشد خانه A1 آن را انتخاب خواهد کرد.

```
Private Sub Workbook_SheetActivate(ByVal Sh As Object)
If TypeName(Sh) = "Worksheet" Then Range("A1").Select
End Sub
```

### 4-2-9 رویداد NewSheet

این رویداد در زمان اضافه شدن یک صفحه جدید به فایل فراخوانی می شود.

**مثال:** کد زیر در زمان اضافه شدن یک صفحه، اندازه پهنای سطرها و ارتفاع ستون ها را تغییر می دهد.

```
Private Sub Workbook_NewSheet(ByVal Sh As Object)
If TypeName(Sh) = "Worksheet" Then
Sh.Cells.ColumnWidth = 20
Sh.Cells.RowHeight = 20
End If
End Sub
```

### 5-2-9 رویداد BeforeSave

این رویداد قبل از ذخیره فایل فراخوانی می شود.

**مثال:** کد زیر در زمان ذخیره فایل، اگر فایل قبلاً ذخیره نشده باشد، یاد آوری می کند که فایل در درایو J ذخیره شود.

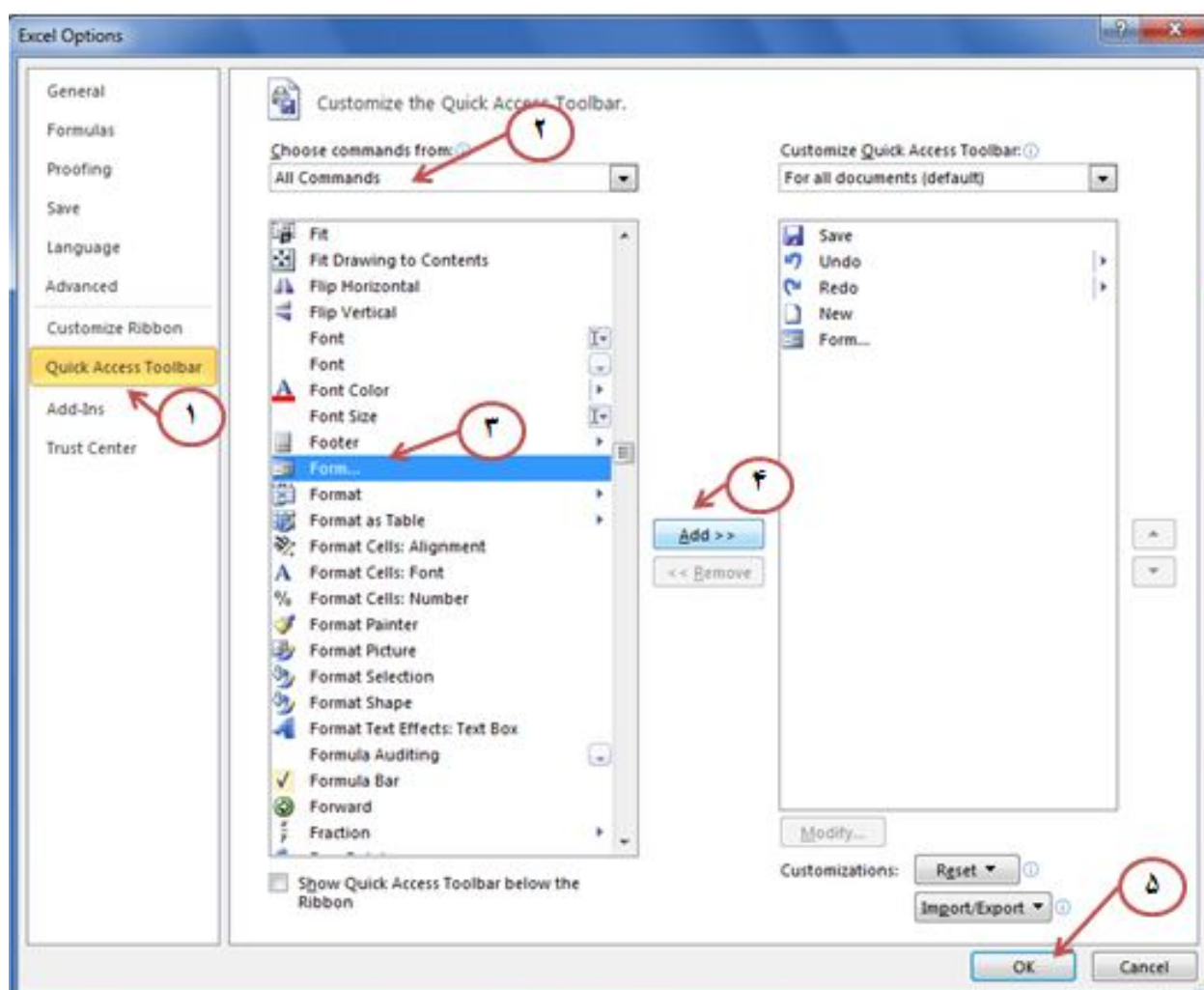
```
Private Sub Workbook_BeforeSave (ByVal SaveAsUI As Boolean, Cancel As Boolean)
If SaveAsUI Then
MsgBox "Make sure you save this file on drive J."
End If
End Sub
```

## (فصل 10) آشنایی با فرم ها

استفاده از فرم ها در اکسل به شما این امکان را می دهد تا اطلاعاتی را به کاربر نمایش دهید یا داده هایی را از طریق کنترل های فرم از کاربر دریافت کنید. ساده ترین جلوه های کاربرد یک فرم دستورهایی Inputbox و MsgBox می باشند.

### 10-1 نمایش Data Form

اکسل برای ویرایش اطلاعات موجود در یک لیست داده از Data Form استفاده می کند. برای نمایش Data Form ابتدا با طی کردن مسیر زیر دکمه مربوط به آن را در نوار QAT قرار دهید.



سپس درون لیست داده کلیک کنید و با کلیک روی دکمه Data Form، فرم ویرایش اطلاعات لیست باز خواهد شد. (شکل ۱۰-۱)

	A	B	C	D	E	F	G	H	I	J	K	L
1												
2												
3		کد پرسنلی	نام	شماره شناسنامه	نام خانوادگی	محل صدور	تاریخ تولد	تاهل	شروع کار			
4		745706	علام عباس	برزگر	7252	تهران	1362/02/2	متاهل	1386			
5		872795	محمد	نجفی	5970	سرخس	1361/11/0	مجرد	1379			
6		681590	محمود	قادر مزى	9388	همدان						
7		688493	روح الله	سیداورادی	9503	آستارا						
8		617831	الیکا	حاتمی	441	رودسر						
9		259383	غلامعباس	ابنیاچ	9551	خمین						
10		108298	سمیرا	دهقانیور	7653	بیرجند						
11		645346	سید حسین	گودرزی	8234	خلخال						
12		624855	غلامحسین	سیدخان	3100	خمین						
13		465122	نادر	علیزاده	8950	مهریز						
14		137530	الهام	هوشنگیان	8235	سمنان						
15		250334	سید مراد	کاشانی	4024	خرم‌آباد						
16		598368	یونس	غریب	5206	کرم‌تاشاد						
17		507967	علی محمدی	سیف اله	9080	اهواز						
18		797740	کاظم	مصطفایی	9684	کرم‌تاشاد						
19		252604	امیرپاشا	صحراگرد	6156	سبزوار						
20		861154	هوشنگ	جهانی	2209	مهریز						
21		857902	ناصر	احسانی	9804	دامغان						
22		750415	کامبیز	یونسی	6281	بروجرد	1366/07/0	متاهل	1384			
23		276247	فرزاد	تاجی	5950	کرمان	1351/03/2	مجرد	1370			

شکل ۱-۱۰

از این فرم می توانید برای ویرایش اطلاعات، درج سطر های جدید و فیلتر کردن اطلاعات استفاده کنید.

## 10-2 درج فرم

برای درج کردن یک فرم به پروژه خود، از منوی Insert گزینه UserForm را انتخاب کنید.

فرم باز شده با نام ... , UserForm1 نام گذاری خواهد شد و البته شما می توانید برای گویا تر کردن پروژه، به فرم های خود نام های معنی دار بدهید.

## 10-3 نمایش فرم

برای بارگزاری یک فرم در حافظه بدون نمایش آن از دستور زیر استفاده می شود.

Load UserForm1

همچنین می توانید با دستور Unload فرم را از حافظه خارج کنید.

برای نمایش یک فرم از دستور زیر استفاده می شود.

UserForm1.Show

راه دیگر برای نمایش یک یا چند فرم در یک ماژول:

```
Sub ShowForm()
    FormName = "UserForm1"
    VBA.UserForms.Add (FormName) . Show
End Sub
```

برای نمایش و پنهان کردن فرم از طریق کد، به ترتیب از متدهای Show و Hide استفاده می‌شود.

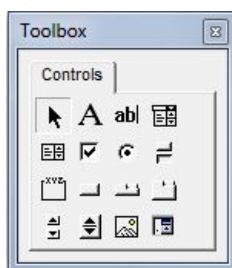
(متد Hide فرم را مخفی می‌کند ولی از حافظه خالی نمی‌کند)

نکته : دستور زیر فرم را در وضعیت غیر مودال باز می‌کند یعنی در حالتی که فرم باز است می‌توانید ناحیه ای غیر از فرم را کلیک کنید.

```
Sub macro2 ()  
frmPersonal.Show vbModeless  
End Sub
```

## 10-4 درج کنترل‌ها

برای ارتباط با کاربر می‌توانید روی فرم کنترل‌های ActiveX را درج کنید. برای اضافه کردن کنترل به فرم، از نوار ابزار Toolbox استفاده می‌شود. (شکل ۱۰-۲)

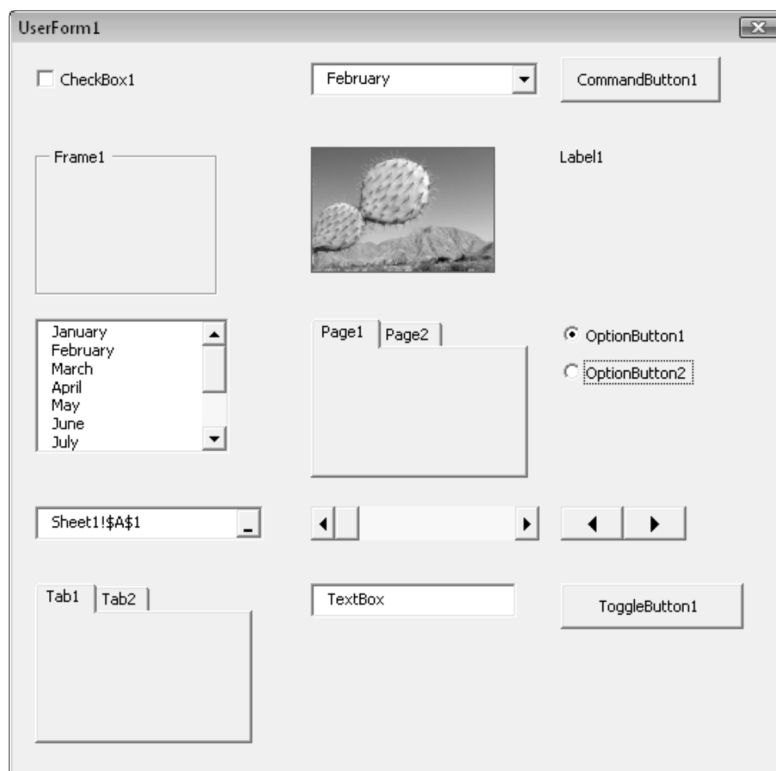


شکل ۱۰-۲

در صورتیکه جعبه Toolbox دیده نمی‌شود در محیط VBE از منوی View گزینه Toolbox را انتخاب کنید.

## 10-5 انواع کنترل‌ها

جعبه Toolbox حاوی انواع مختلفی از کنترل‌ها است که هر یک مورد استفاده خود را دارند و در ادامه با آنها آشنا خواهید شد :



### CheckBox

این کنترل برای انتخاب های دو گزینه ای استفاده می شود. وقتی CheckBox را تیک می زنید، مقدار آن برابر True و در غیر این صورت False خواهد بود.

### ComboBox

این کنترل لیستی از مقادیر را به صورت یک لیست باز شو (Drop Down List) به کاربر نمایش می دهد و در هر لحظه تنها یکی از مقادیر قابل انتخاب است.

### CommandButton

پر استفاده ترین کنترل است و دکمه ایست که با کلیک روی آن دستوراتی که در رویداد کلیک نوشته شده اجرا می شوند.

### Frame

این کنترل برای دسته بندی کنترل ها استفاده می شود و مهمترین استفاده آن زمانیست که شما بخواهید چند گروه از OptionButton ها را بر روی فرم قرار دهید. در این حالت OptionButton های مربوط به هم را درون یک Frame قرار می دهید.

### Image

از این کنترل برای نمایش تصویر بر روی فرم استفاده می شود.

### ListBox

این کنترل لیستی از مقادیر را به کاربر نمایش می دهد و کاربر یک یا چند آئتم را از بین مقادیر انتخاب کند.

## MultiPage

این کنترل به کاربر اجازه می دهد که بر روی فرم برگه های متعدد (Tab) ایجاد کند و روی هر برگه کنترل های دلخواه را درج کند.

## OptionButton

کنترل OptionButton به کاربر این امکان را می دهد که یک گزینه را از بین چند گزینه انتخاب کند. وقتی کاربر گزینه ای را انتخاب می کند، Value آن برابر True و Value دیگر گزینه ها برابر false خواهد بود.

## RefEdit

توسط این کنترل کاربر قادر خواهد بود که یک ناحیه را از روی Worksheet انتخاب کند. در این حالت آدرس ناحیه انتخاب شده توسط ویژگی Text از کنترل قابل دسترس خواهد بود.

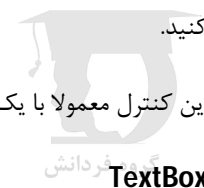
## ScrollBar

کنترل ScrollBar بسیار شبیه به کنترل SpinButton می باشد با این تفاوت که برای تغییر مقدار عددی می توانید لغزنده کنترل را جا به جا کنید.

## SpinButton

توسط کنترل SpinButton قادر خواهید بود با کلیک روی پیکان های دو طرف ، یک مقدار عددی بین ویژگی های Min و Max انتخاب کنید.

این کنترل معمولاً با یک TextBox استفاده می شود که مقدار value آن را نمایش دهد.



## TextBox

این کنترل برای درج متن توسط کاربر استفاده می شود.

## Label

کنترل Label برای نمایش یک متن بر روی فرم استفاده می شود. برای نمایش یک متن درون این کنترل کافایت که متن را به ویژگی Caption نسبت دهید.

## TabStrip

این کنترل به شما این امکان را می دهد که نواری از دکمه ها را بر روی فرم قرار دهید.

```
Private Sub TabStrip2_Change ()  
    MsgBox TabStrip2.SelectedItem.Caption  
    MsgBox TabStrip2.SelectedItem.Index  
End Sub
```

```

Private Sub TabStrip1_Change ()
Select Case TabStrip1.SelectedItem.Index
Case 0
    TabStrip1.ForeColor = QBColor(1)
Case 1
    TabStrip1.ForeColor = QBColor(5)
Case 2
    TabStrip1.ForeColor = QBColor(11)
Case 3
    TabStrip1.ForeColor = QBColor(15)
End Select
End Sub

```

## 10- 6 تغییر ویژگی های یک کنترل

ویژگی های یک کنترل را به دو روش می توان تغییر داد. در زمان طراحی و در زمان اجرا

### تغییر در زمان طراحی

برای تغییر ویژگی های یک کنترل در زمان طراحی کافیست که بعد از انتخاب کنترل ، به پنجره Properties مراجعه کنید و ویژگی مورد نظر را تغییر دهید.

از متداول ترین ویژگی هایی که برای تمام کنترل ها وجود دارد و معمولا در زمان طراحی تغییر می کنند عبارتند از :

ویژگی	شرح
Name	نام کنترل
Caption	عنوان کنترل
Top	فاصله کنترل از لبه بالای فرم
Left	فاصله کنترل از لبه چپ فرم
Width	پهنای کنترل
Height	ارتفاع کنترل

### تغییر در زمان اجرا

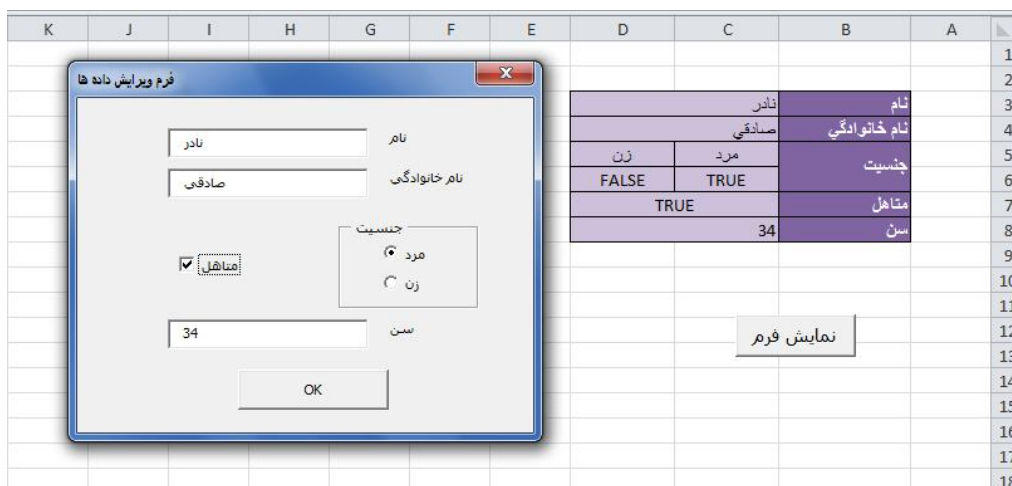
برای تغییر ویژگی یک کنترل در زمان اجرا کافیست که نام کنترل، نقطه و نام ویژگی را نوشته و سپس به آن مقدار دهی کنید.

مثال : دستور زیر OptionButton را به حالت انتخاب شده تبدیل می کند.

```
OptionButton1.Value = True
```

با انتخاب یک فرم یا یک کنترل درج شده بر روی فرم، می‌توانید ویژگی‌های آن را در پنجره Properties مشاهده کنید.

## 10-7 مثال 1



```
Private Sub cmdShowForm_Click()
    frmPersonal.Show
End Sub
```

```
Private Sub cmdOK_Click()
    Unload Me
End Sub
```



Control Source مقدار ویژگی	نام کنترل	نوع کنترل
Sheet1!c3	txtFirstName	TextBox
Sheet1!c4	txtLastName	TextBox
Sheet1!c6	optMale	Option Button
Sheet1!d6	optFemale	Option Button
Sheet1!c7	chkMarried	CheckBox
Sheet1!c8	txtAge	TextBox

## 10-8 مثال 2

در این مثال (شکل ۱۰-۳) یک فرم برای اضافه کردن اطلاعات جدید به انتهای لیست طراحی شده است.



	A	B	C	D	E	F	G	H	I
1									
2									
3		ردیف	نام خانوادگی	نام	تاهل				
4		1	بهباد	سعادت	Yes				
5		2	محمد	مصدق	No				
6		3	سید رهام	مهدی	Yes				
7		4	کریم	ربیعی	Yes				
8									
9									
10									
11									
12									
13									
14									
15									
16									
17									
18									
19									
20									
21									

شکل ۳-۱۰

برای دکمه Exit تنها کد زیر درج شده است :

```
Private Sub cmdExit_Click()
Unload UserForm1
End Sub
```

کد دکمه Add به صورت زیر می باشد :

```
Private Sub cmdAdd_Click()
Dim NextRow As Long
Sheets("Sheet1").Activate
NextRow = Application.WorksheetFunction.CountA(Range("B:B")) + 3
If txtFname.Text = "" Or txtLname.Text = "" Then
MsgBox "You must enter a name."
txtFname.SetFocus
Exit Sub
End If
Cells(NextRow, 1) = Cells(NextRow - 1, 1) + 1
Cells(NextRow, 2) = txtFname.Text
Cells(NextRow, 3) = txtLname.Text
If optYes Then Cells(NextRow, 4) = "Yes"
If optNo Then Cells(NextRow, 4) = "No"
txtFname.Text = ""
txtLname.Text = ""
txtFname.SetFocus
End Sub
```

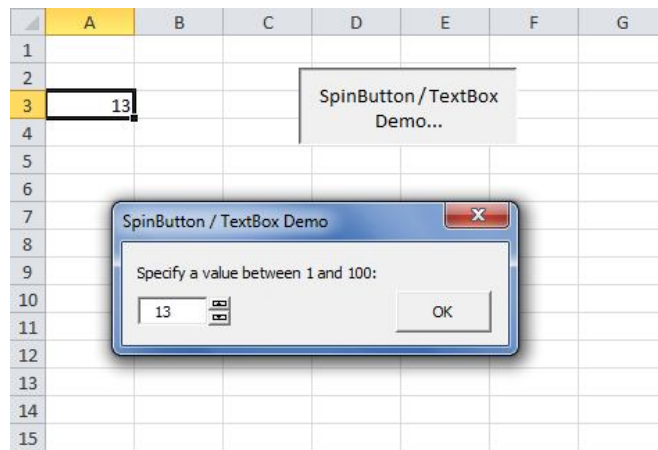
توضیح کد :

1. صفحه Sheet1 فعال می شود.
2. متغیر NextRow تعداد خانه های غیر خالی را شمرده و با توجه به اینکه جدول از سطر سوم شروع می شود، مقدار حاصل را با 3

3. جمع کرده و در خود ذخیره می کند. به این ترتیب این متغیر شماره سطر جدید را در خود نگه می دارد.
3. توسط دستور IF کادر های نام و نام خانوادگی کنترل می شود. اگر هرکدام از آنها خالی باشند، با یک پیام ادامه مسیر متوقف شده و کاربر مجبور به نوشتن نام و نام خانوادگی می شود.
4. خانه جدید ردیف مقدار قبلی را یک واحد اضافه کرده و در خود نگه می دارد.
5. متن درون کادر نام در زیر ستون نام درج می شود.
6. متن درون کادر نام خانوادگی در زیر ستون نام خانوادگی درج می شود.
7. کنترل های **Option Button** چک شده و بسته به انتخاب هر کدام، عبارت متنی معادل در ستون آخر درج می شود.
8. کادر های نام و نام خانوادگی برای درج اطلاعات جدید خالی می شوند.
9. فوکوس به درون کادر نام بر می گردد.

### 10-9 مثال 3

در این مثال از یک **SpinButton** که مقدار **Min** و **Max** آن به ترتیب 0 و 100 است، برای درج مقدار در **ActiveCell** استفاده شده و همزمان با تغییر **SpinButton**، مقدار آن توسط **TextBox** نمایش می یابد.



کد درون رویداد **Change** مربوط به **SpinButton**

```
Private Sub SpinButton1_Change ()
    TextBox1.Text = SpinButton1.Value
End Sub
```

کد درون رویداد **Change** مربوط به **TextBox**

```
Private Sub TextBox1_Change ()
    Dim NewVal As Integer
    NewVal = Val(TextBox1.Text)
    If NewVal >= SpinButton1.Min And NewVal <= SpinButton1.Max Then
        SpinButton1.Value = NewVal
    End If
End Sub
```

توضیح: تابع **Val** متن درون کادر متنی را به معادل عددی تبدیل می کند.

کد درون رویداد **Click** مربوط به دکمه **OK**

```
Private Sub OKButton_Click()  
    If CStr(SpinButton1.Value) = TextBox1.Text Then  
        ActiveCell = SpinButton1.Value  
        Unload Me  
    Else  
        MsgBox "Invalid entry.", vbCritical  
        TextBox1.SetFocus  
        TextBox1.SelStart = 0  
        TextBox1.SelLength = Len(TextBox1.Text)  
    End If  
End Sub
```



## (فصل 11) تکنیک های VBA

### 11-1 کپی کردن یک محدوده

در صورتی که از Record Macro برای کپی کردن خانه A1 بر روی B1 استفاده کنید کد زیر را مشاهده خواهید کرد :

```
Sub Macro1()  
Range("A1").Select  
Selection.Copy  
Range("B1").Select  
ActiveSheet.Paste  
Application.CutCopyMode = False  
End Sub
```

ب اینک Record Macro ابزاری بسیار مفید است ولی همیشه بهترین راه را نشان نمی دهد. به نمونه دیگر که خلاصه تر نوشته شده است توجه فرمایید

```
Sub CopyRange()  
Range("A1").Copy Range("B1")  
End Sub
```

گروه فردانش

نمونه دیگر برای کپی از یک فایل بر روی فایل دیگر

```
Sub CopyRange3()  
Dim Rng1 As Range, Rng2 As Range  
Set Rng1 = Workbooks("File1.xlsx").Sheets("Sheet1").Range("A1")  
Set Rng2 = Workbooks("File2.xlsx").Sheets("Sheet2").Range("A1")  
Rng1.Copy Rng2  
End Sub
```

توجه کنید که مقصد کپی می تواند تنها آدرس اولین خانه را نشان دهد. به این مثال توجه فرمایید :

```
Sub CopyRange4()  
Range("A1:C800").Copy Range("D1")  
End Sub
```

## 11-2 انتقال یک محدوده

همانند کپی، برای انتقال یک محدوده نیز می توان کد ها را خلاصه تر و کارآمد تر نوشت. مثال :

```
Sub MoveRange1()
Range("A1:C6").Cut Range("H1")
End Sub
```

## 11-3 کپی کردن محدوده متغیر

در شکل ۱۱-۱ محدوده ای از اطلاعات را میبینید که تعداد سطر های آن به طور مداوم تغییر می کند.

	A	B	C	D	E	F	G	H
1								
2		کد پرسنلی	نام	نام خانوادگی	شماره شناسنامه	محل صدور		
3		745706	غلام عباس	برزگر	7252	تهران		
4		872795	محمد	نجفی	5970	سرخس		
5		681590	محمود	قادر مزی	9388	همدان		
6		688493	روح الله	سید اورادی	9503	استارا		
7		617831	الیکا	حاجتی	441	رودسر		
8		259383	غلامعباس	ابنیاچ	9551	خمین		
9		108298	سمیرا	دهقانپور	7653	بیرجند		
10		645346	سید حسین	گوردزی	8234	خلخال		
11		624855	غلامحسین	سیدخان	3100	خمین		
12		465122	نادر	علیزاده	8950	مهریز		
13		137530	الهام	هوشنگیان	8235	سمنان		
14		250334	سید مراد	کاشانی	4024	خرمآباد		
15		598368	یونس	غریب	5206	کرمانشاه		
16		507967	علی محمدی	سیف اله	9080	اهواز		
17		797740	کاظم	مصطفایی	9684	کرمانشاه		
18								
19								
20								
21								



شکل ۱۱-۱

برای کپی کردن این محدوده به Sheet2 از کد زیر می توانید استفاده کنید.

```
Sub CopyCurrentRegion2()
Range("C4").CurrentRegion.Copy Sheets("Sheet2").Range("A1")
End Sub
```

نکته : اگر لیست شما به صورت Table تبدیل شده باشد می توانید از کد زیر استفاده کنید :

```
Sub CopyTable()
Range("Table1[#All]").Copy Sheets("Sheet2").Range("A1")
End Sub
```

## 11-4 انتخاب محدوده اطلاعات

برای انتخاب محدوده اطلاعات نشان داده شده در شکل ۱۱-۱ ، بعد از انتخاب خانه ای در محدوده اطلاعات می توانید از کد زیر استفاده کنید :

```
Sub SelectCurrentRegion()
ActiveCell.CurrentRegion.Select
End Sub
```

## 11-5 دریافت ورودی برای خانه فعال

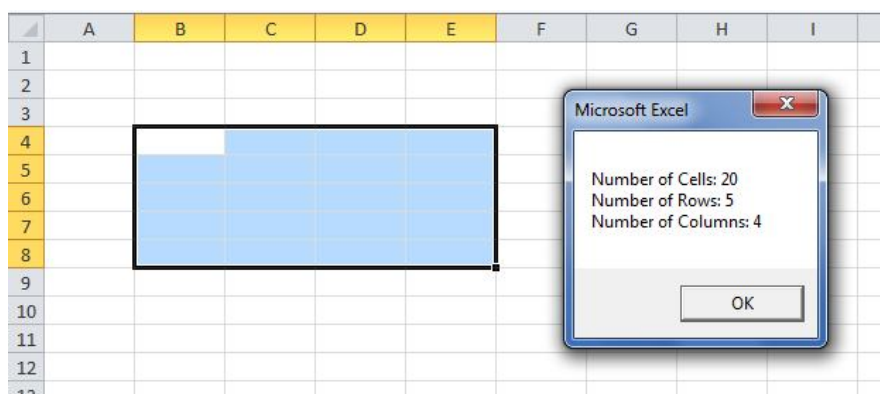
در کد زیر توسط تابع `InputBox` مقداری برای خانه فعال دریافت می شود و اگر این مقدار خارج از محدوده 1 تا 12 باشد به عنوان مقدار غیر معتبر تشخیص داده می شود.

```
Sub GetValue3()
Dim UserEntry As Variant
Dim Msg As String
Const MinVal As Integer = 1
Const MaxVal As Integer = 12
Msg = "Enter a value between " & MinVal & " and " & MaxVal
Do
    UserEntry = InputBox(Msg)
    If UserEntry = "" Then Exit Sub
    If IsNumeric(UserEntry) Then
        If UserEntry >= MinVal And UserEntry <= MaxVal Then Exit Do
    End If
    Msg = "Your previous entry was INVALID".
    Msg = Msg & vbNewLine
    Msg = Msg & "Enter a value between " & MinVal & " and " & MaxVal
Loop
ActiveCell.Value = UserEntry
End Sub
```

## 11-6 شمارش خانه های ناحیه انتخاب شده

ماکروی زیر تعداد خانه ها، سطر ها و ستون های ناحیه انتخاب شده را اعلان می کند :

```
Sub SlectionCount()
Dim Msg As String
Msg = "Number of Cells: " & Selection.Count
Msg = Msg & vbNewLine
Msg = Msg & "Number of Rows: " & Selection.Rows.Count
Msg = Msg & vbNewLine
Msg = Msg & "Number of Columns: " & Selection.Columns.Count
MsgBox Msg
End Sub
```



## 11-7 حذف سطر های غیر خالی

ماکروی زیر سطر های خالی را در محدوده استفاده شده (UsedRange) حذف می کند.

```

Sub DeleteEmptyRows()
Dim LastRow As Long
Dim r As Long
Dim Counter As Long
Application.ScreenUpdating = False
LastRow = ActiveSheet.UsedRange.Rows.Count + _
ActiveSheet.UsedRange.Rows(1).Row - 1
For r = LastRow To 1 Step -1
If Application.WorksheetFunction.CountA(Rows(r)) = 0 Then
Rows(r).Delete
Counter = Counter + 1
End If
Next r
Application.ScreenUpdating = True
MsgBox Counter & " empty rows were deleted."
End Sub

```

توضیح: ویژگی ScreenUpdating برای شیء Application در زمان اجرای ماکرو False شده است تا سرعت اجرای ماکرو بیشتر باشد.

## 11-8 تعیین نوع اطلاعات یک خانه

تابع زیر یک خانه (یا محدوده) را به عنوان ورودی دریافت کرده و نوع اطلاعات موجود در خانه را به صورت یک رشته متنی بر میگرداند.

```

Function CellType(Rng) As String
Dim TheCell As Range
Set TheCell = Rng.Range("A1")
Select Case True
Case IsEmpty(TheCell)
CellType = "Blank"
Case Application.IsText(TheCell)
CellType = "Text"

```

```

Case Application.IsLogical(TheCell)
    CellType = "Logical"
Case Application.IsErr(TheCell)
    CellType = "Error"
Case IsDate(TheCell)
    CellType = "Date"
Case InStr(1, TheCell.Text, ":") <> 0
    CellType = "Time"
Case IsNumeric(TheCell)
    CellType = "Number"
End Select
End Function
    
```

توضیح: دستور زیر باعث می شود که از ناحیه وارد شده به تابع تنها خانه اول در متغیر TheCell ذخیره شود.

```
Set TheCell = Rng.Range("A1")
```

## 11-9 ذخیره تمام فایل ها

ماکرو زیر تمام فایل های باز که قبلا ذخیره شده اند را ذخیره می کند.

```

Public Sub SaveAllWorkbooks()
Dim Book As Workbook
For Each Book In Workbooks
    If Book.Path <> "" Then Book.Save
Next Book
End Sub
    
```

توضیح: دستور If بررسی می کند که فایل قبلا ذخیره شده است یا خیر. به عبارت دیگر در صورتی که فایل قبلا ذخیره شده باشد ویژگی Path آن آدرس را بر می گرداند.

## 11-10 شمارش خانه های بین دو مقدار

تابع زیر تعداد مقادیر بین num1 و num2 را در محدوده InRange شمارش می کند.

```

Function CountBetween(InRange, num1, num2) As Long
With Application.WorksheetFunction
    If num1 <= num2 Then
        CountBetween = .Countifs(InRange, ">=" & num1, InRange, "<=" & num2)
    Else
        CountBetween = .Countifs(InRange, ">=" & num2, InRange, "<=" & num1)
    End If
End With
End Function
    
```



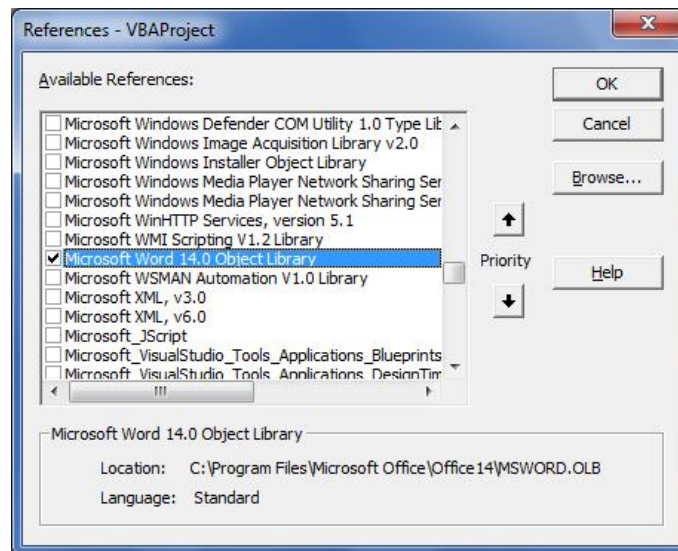
## Office (فصل 12) ارتباط با دیگر برنامه‌های Office

پشت صحنه تمام برنامه‌های MS Office از زبان به عنوان ابزار ایجاد ماکروها استفاده می‌شود و هر کدام از برنامه‌های Office مانند اکسل مدل شیء مخصوص به خود را دارا می‌باشند.

این مزیت مهمی برای Ms Office است به این معنی که کاربر Office می‌تواند به راحتی و با استفاده از مدل شیء هر برنامه از برنامه دیگر، اقدام به ویرایش در محیط برنامه دیگر نماید.

### 12-1 ارتباط با MS Word

برای استفاده از مدل شیء Word در اکسل ابتدا باید کتابخانه ورد را به اکسل اضافه کنید. به این صورت که در محیط VBE از منوی Tools گزینه Preferences را انتخاب کنید. سپس در پنجره باز شده گزینه Microsoft Word 14 Object Library را انتخاب کنید. (شکل 1-12)



شکل ۱-۱۲

#### مثال 1:

با اجرای کدهای ماکروی زیر یک فایل جدید ورد ایجاد می‌شود و بعد از درج یک متن نمونه در آن، با نام Test در درایو C ذخیره می‌شود.

```
Sub Word_Test ()
Dim wordApp As Word.Application
Dim wordDoc As Word.Document
Set wordApp = CreateObject("word.application")
Set wordDoc = wordApp.Documents.Add
wordDoc.Sections(1).Range.Text = "test for new word document"
wordDoc.SaveAs ("c:\test.docx")
wordDoc.Close
wordApp.Quit
Set wordApp = Nothing
Set wordDoc = Nothing
End Sub
```

به شکل ۲-۱۲ توجه فرمائید. می خواهیم فیش حقوقی هر شخص به نام خودش در یک فایل Word ذخیره شود.

I	H	G	F	E	D	C	B	A	
									1
									2
									3
									4
									5
									6
									7
									8
									9
									10
									11
									12
									13
									14
									15
									16
									17
									18
									19
									20
									21
									22

شکل ۲-۱۲

برای انجام این کار کد زیر در رویداد کلیک دکمه Save to word نوشته شده است.

```
Private Sub cmdSave_Click()
Dim wordApp As Word.Application
Dim wordDoc As Word.Document
Dim fileName As String
fileName = Range("c15").Value & Range("e15").Value
Sheets("sheet1").Range("B10:e22").Select
Selection.Copy
Set wordApp = CreateObject("Word.application")
Set wordDoc = wordApp.Documents.Add
wordApp.Selection.PasteExcelTable False, False, False
fileName = "C:\" & fileName
wordDoc.SaveAs (fileName)
wordDoc.Close
wordApp.Quit
Set wordDoc = Nothing
Set wordApp = Nothing
Application.CutCopyMode = False
End Sub
```

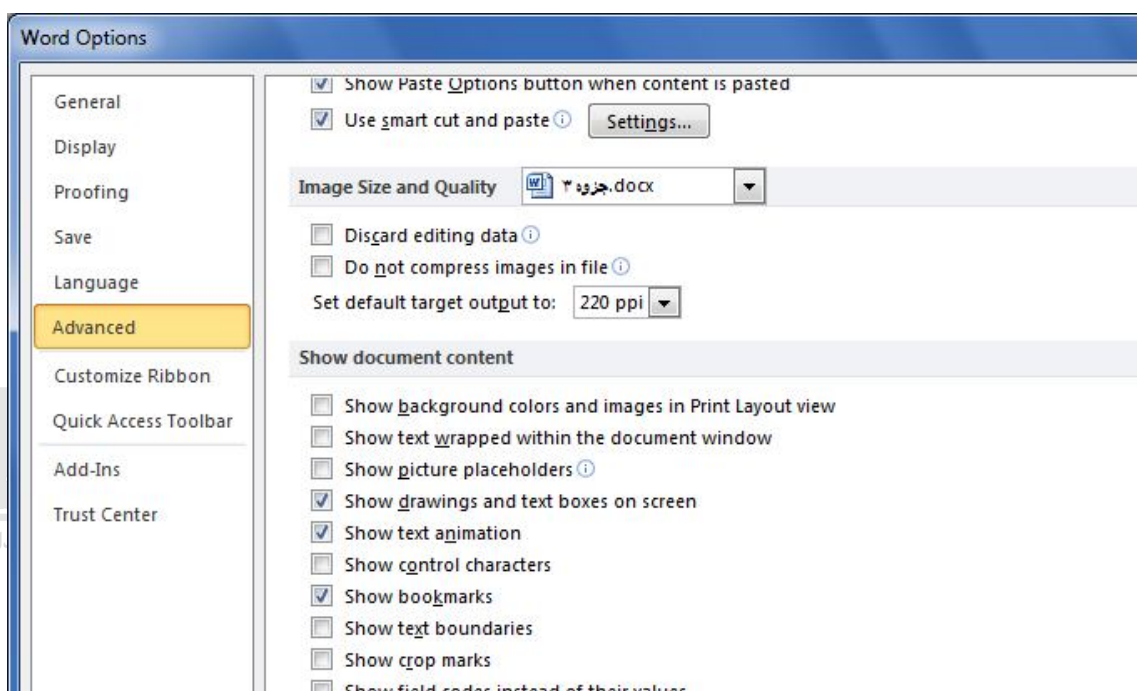
## 12-1-1 ارتباط با Bookmark های Word

مهمترین موضوع در مورد بوک مارک های Word این است که دو نوع بوک مارک در Word وجود دارد :

Placeholder bookmarks

Enclosing bookmarks

پیش از آشنایی با این موضوع برای روشن کردن وضعیت نمایش بوک مارک ها به برگه Advanced از پنجره Word Options رفته و قسمت Show document content، گزینه Show Bookmark را انتخاب کنید. (شکل ۳-۱۲)



شکل ۳-۱۲

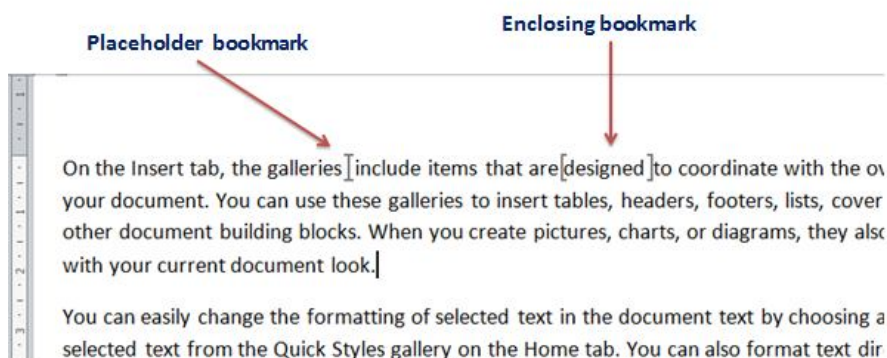
### Placeholder bookmarks

اگر در جایی از متن **کلیک** کنید و از روبان Insert گزینه Bookmark را انتخاب کنید، بوک مارک ایجاد شده از نوع Placeholder bookmark خواهد بود.

### Enclosing bookmarks

اگر بخشی از متن را انتخاب کنید و از روبان Insert گزینه Bookmark را انتخاب کنید، بوک مارک ایجاد شده از نوع Enclosing bookmark خواهد بود.

نکته : اگر وضعیت نمایش بوک مارک ها روشن باشد، دو نوع بوک مارک به صورت زیر دیده می شوند.



شکل ۱۲-۴

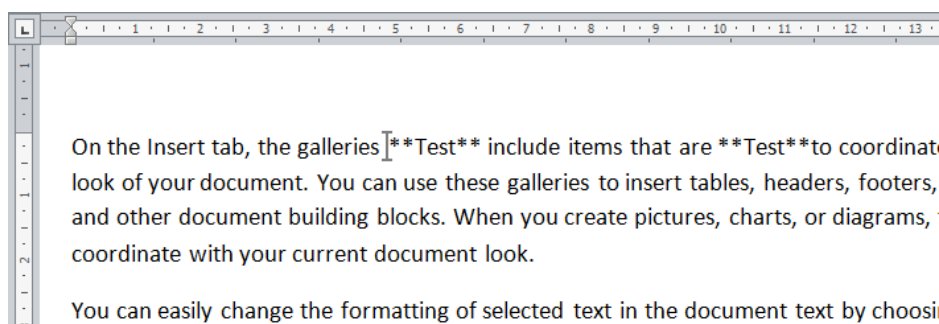
در شکل ۱۲-۴ نام بوک مارک ها به ترتیب B1 و B2 می باشد.

حال ماکروی زیر را بر روی سند بالا اجرا می کنیم.

```
Sub Macro1 ()
    '
    ' Macro1 Macro
    '
    '
    ActiveDocument.Bookmarks ("B1").Range.Text = "***Test**"
    ActiveDocument.Bookmarks ("B2").Range.Text = "***Test**"
End Sub
```



و نتیجه به صورت زیر خواهد بود :



ملاحظه می کنید که در بوک مارک اول متن به محل بوک مارک اضافه شده و در بوک مارک دوم متن جدید جایگزین متن پیشین شده ولی خود بوک مارک حذف شده و قابل استفاده مجدد نیست.

نتیجه : برای تعریف بوک مارک هایی که قابلیت جایگزینی متن داشته باشند، از Enclosing Bookmark استفاده می کنیم و برای تغییر متن بوک مارک از ماکروی زیر کمک می گیریم :

```
Sub Macro2 ()
'
' Macro2 Macro
'
'
Dim bmRange As Range

Set bmRange = ActiveDocument.Bookmarks("B2").Range

bmRange.Text = "***Test***"

ActiveDocument.Bookmarks.Add Name:="B2", Range:=bmRange

End Sub
```

برای تغییر مکرر بوک مارک های یک فایل Word از محیط اکسل یا هر جای دیگر می توان یک روبه با آرگومانهای ورودی به صورت زیر تعریف کرد:

```
Sub UpdateBookmark(BookmarkToUpdate As String, TextToUse As String)
Dim BMRRange As Range
Set BMRRange = ActiveDocument.Bookmarks(BookmarkToUpdate).Range
BMRRange.Text = TextToUse
ActiveDocument.Bookmarks.Add BookmarkToUpdate, BMRRange
End Sub
```

حال می توان این روبه را در هر ماکرو فراخوانی کرد:

```
Sub Macro3 ()
'
' Macro3 Macro
'
'
Call UpdateBookmark("B2", "^^Test^^")

End Sub
```



## مدیریت Ribbon ها (فصل 13)

در اکسل 2010 بیش از 1700 کنترل تعریف شده و برخی از آنها بر روی روبان های برنامه در دسترس هستند. برای کار با کنترل های تعریف شده از شیء ComandBars استفاده می شود.

متد های شیء ComandBars عبارتند از :

**ExecuteMso** : یک دستور را اجرا می کند.

**GetEnabledMso** : اگر کنترل فعال باشد مقدار True را برمی گرداند.

**GetImageMso** : آیکن کنترل را برمی گرداند.

**GetLabelMso** : متن کنترل را بر میگرداند.

**GetPressedMso** : اگر کنترل کلیک شود مقدار True را برمی گرداند.

تمام این متد ها تنها یک آرگومان دارند و آن نام کنترل می باشد.

مثال : کد زیر پنجره Paste Special را باز می کند

```
Sub controlTest1()
Application.CommandBars.ExecuteMso ("PasteSpecialDialog")
End Sub
```

مثال : کد زیر اگر نوار فرمول آشکار باشد مقدار True را اعلان می کند

```
Sub controlTest2()
MsgBox Application.CommandBars.GetPressedMso("ViewFormulaBar")
End Sub
```

مثال : کد زیر دستور Merge & Center انتخاب شده باشد مقدار True را اعلان می کند

```
Sub controlTest3()
MsgBox Application.CommandBars.GetEnabledMso("MergeCenter")
End Sub
```

### 13-1 فعال کردن یک روبان

برای فعال کردن یک روبان دستوری تعریف نشده ولی می توانید از متد SendKeys برای انجام این کار استفاده کنید :

مثال: کد زیر روبان Home را فعال می کند.

```
Sub controlTest4()
```

## Application.SendKeys "%h{F6}" End Sub

برای فعال کردن روبان های دیگر از آرگومان های زیر استفاده کنید :

- **Insert:** "%n{F6}"
- **Page Layout:** "%p{F6}"
- **Formulas:** "%m{F6}"
- **Data:** "%a{F6}"
- **Review:** "%r{F6}"
- **View:** "%w{F6}"
- **Developer:** "%l{F6}"
- **Add-Ins:** "%x{F6}"

## 2-13 ویرایش روبان ها

مفید ترین روش برای ویرایش دستورهایی موجود در روبان ها استفاده از نرم افزار Custom UI Editor می باشد.

### مثال 1:

یک فایل جدید ایجاد کنید و درون یک ماژول جدید ماکرو های زیر را بنویسید :

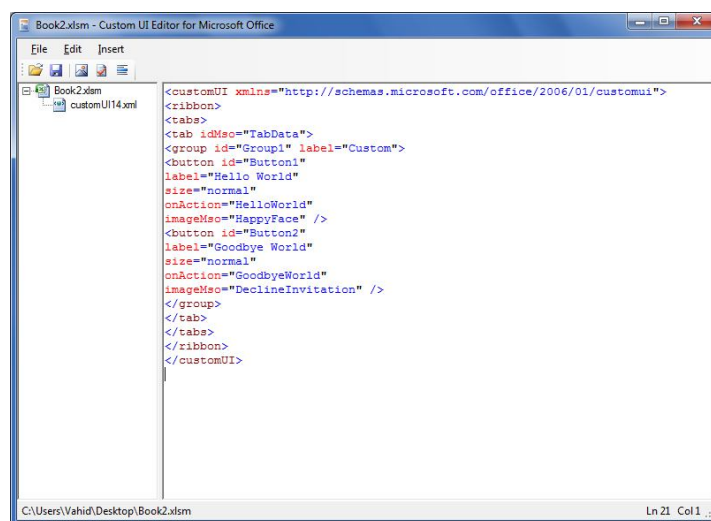
```
Sub HelloWorld(control As IRibbonControl)
MsgBox "Hello World!"
End Sub

Sub GoodbyeWorld(control As IRibbonControl)
ThisWorkbook.Close
End Sub
```



سپس فایل را با پسوند XLSM ذخیره کرده و ببندید.

در ادامه نرم افزار Custom UI Editor را باز کرده و از درون آن فایل اکسل را باز کنید. سپس برای اکسل 2010 از منوی Insert گزینه Office 2010 Custom UI Part را انتخاب کرده و کد زیر را درون آن بنویسید :



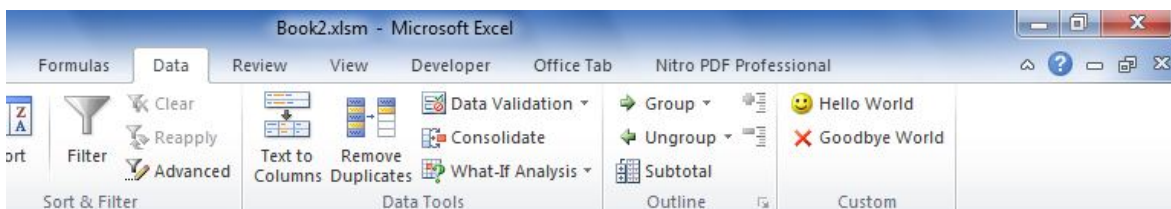
```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/
customui ">
<ribbon>
<tabs>
```

```

<tab idMso="TabData">
  <group id="Group1" label="Custom">
    <button id="Button1"
      label="Hello World"
      size="normal"
      onAction="HelloWorld"
      imageMso="HappyFace" />
    <button id="Button2"
      label="Goodbye World"
      size="normal"
      onAction="GoodbyeWorld"
      imageMso="DeclineInvitation" />
  </group>
</tab>
</tabs>
</ribbon>
</customUI>

```

بعد از ذخیره و بستن فایل، آن را با اکسل مجدداً باز کنید. مشاهده خواهید کرد که در انتهای روبان Data کنترل های زیر اضافه شده است.



مثال 2: فردانش

کد زیر را برای یک فایل اکسل در برنامه Custom UI Editor بنویسید و بعد از ذخیره و باز کردن فایل در محیط اکسل مشاهده خواهید کرد که فایل هیچ روبانی را نمایش نمی دهد.

```

<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon startFromScratch="true">
  </ribbon>
</customUI>

```

توضیح : تگ startFromScratch وقتی مقدار True بگیرد تمام روبان ها مخفی می شوند.

مثال 3 :

کد زیر را برای یک فایل اکسل در برنامه Custom UI Editor بنویسید و بعد از ذخیره و باز کردن فایل در محیط اکسل مشاهده خواهید کرد که روبان Home مخفی شده است.

```

<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
<ribbon>
<tabs>
<tab idMso="TabHome" visible="false" />
</tabs>
</ribbon>
</customUI> >

```



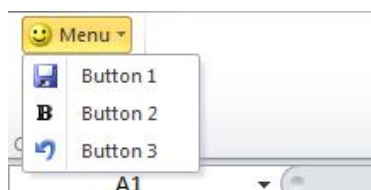
مثال 4: کد زیر را برای یک فایل اکسل در برنامه Custom UI Editor بنویسید و بعد از ذخیره و باز کردن فایل در محیط اکسل مشاهده خواهید کرد که دکمه Bold در روبان Home غیر فعال شده است، روبان Insert غیر فعال شده است و یک روبان جدید با یک دکمه بر روی آن به روبانها اضافه شده است.

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <commands>
    <command idMso="Bold" enabled="false" />
  </commands>
  <ribbon>
    <tabs>
      <tab idMso="TabInsert" visible="false" >
      </tab>
      <tab id="CustomTab" label="Custom Tab">
        <group id="CustomGroup" label="Custom Group">
          <button id="CustomButton" label="Custom Button"
            size="large" imageMso="HappyFace" onAction="OnButtonClick" />
        </group>
      </tab>
    </tabs >
  </ribbon>
</customUI>
```

مثال 5:

کد زیر را برای یک فایل اکسل در برنامه Custom UI Editor بنویسید و بعد از ذخیره و باز کردن فایل در محیط اکسل مشاهده خواهید کرد که یک روبان جدید با یک دکمه به صورت منو به سه گزینه بر روی آن به روبانها اضافه شده است.

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon>
    <tabs>
      <tab id="CustomTab" label="Custom Tab">
        <group id="CustomGroup" label="Custom Group">
          <menu id="menu" label="Menu" imageMso="HappyFace" >
            <button id="button1" label="Button 1" imageMso="FileSave" />
            <button id="button2" label="Button 2" imageMso="Bold" />
            <button id="button3" label="Button 3" imageMso="Undo" />
          </menu>
        </group>
      </tab>
    </tabs >
  </ribbon>
</customUI>
```



## فصل 14 ایجاد Add-Ins

استفاده از Add-Ins ها مهمترین روش برای توسعه قابلیت های برنامه Excel می باشد. مطمئنا با Solver به عنوان یک Add-In مهم که همراه نرم افزار اکسل بر روی سیستم کپی می شود آشنا هستید و از آن برای تحلیل های آماری استفاده کرده اید.

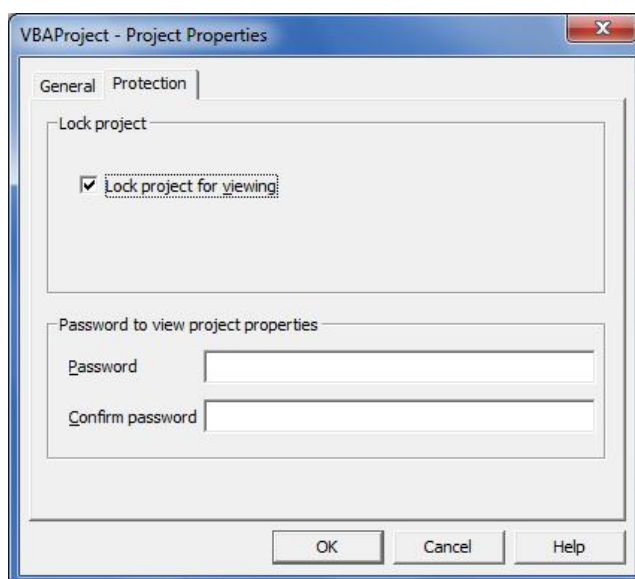
کاری که در این فصل می خواهیم انجام دهید تبدیل یک فایل با قابلیت های عمومی به یک Add-In می باشد تا بتوانیم آن را به راحتی در دسترس دیگران قرار دهیم.

وقتی یک فایل اکسل را به یک Add-Ins تبدیل می کنید، ماکرو ها و توابع عمومی موجود در آن در زمان باز بودن برنامه اکسل قابل استفاده خواهد بود.

### 14-1 محافظت از کدهای VBA

احتمالا اولین قدم شما در مسیر ایجاد یک Add-In (در واقع تبدیل فایل به Add-In) این است که از کدهای VBA فایل خود محافظت کنید تا کسی نتواند آنها را تغییر دهد.

برای انجام این کار در محیط VBE از منوی Tools گزینه VBA Project Properties را انتخاب کرده و در برگه Protection گزینه Lock Project for viewing را انتخاب کنید. (شکل ۱۴-۱)



شکل ۱۴-۱

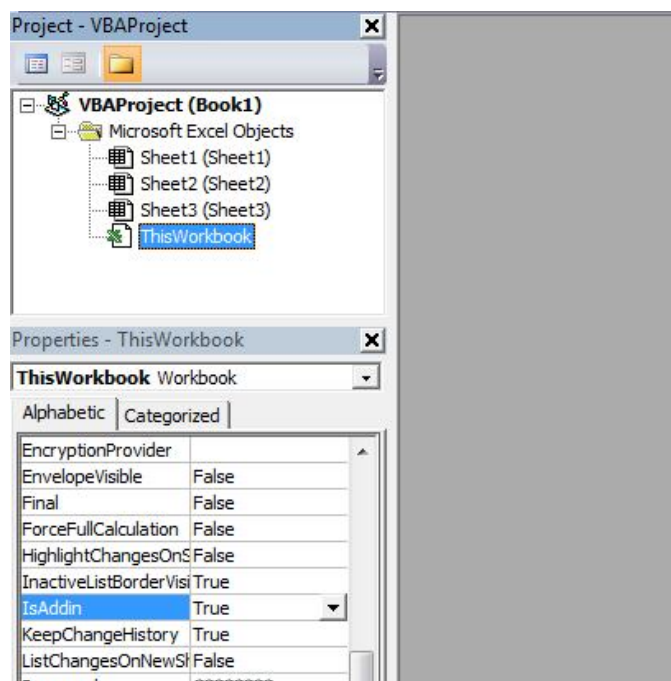
سپس رمز دلخواه را در کادر Password بنویسید و بعد از تأیید و بستن پنجره فایل را ذخیره کنید.

از این به بعد دیدن کدها در پنجره Project Explorer منوط به داشتن رمز می باشد.

## 2-14 تفاوت‌های Add-Ins و یک فایل معمولی اکسل :

مهمترین تفاوت‌های بین یک Add-In و یک فایل اکسل معمولی به شرح زیر می‌باشند :

- برای فایل Add-Ins، ویژگی Is AddIn از شیء Workbook برابر مقدار True می‌باشد.

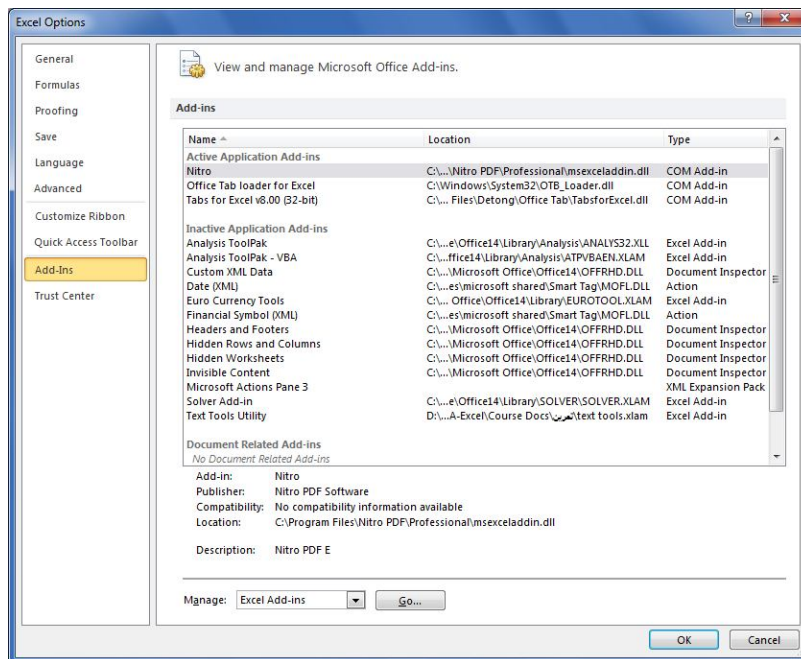


گروه فردانش

- پنجره Workbook مربوط به فایل Add-In نصب شده به طور پیش فرض مخفی می‌باشد.
- فایل Add-In نصب شده عضوی از شیء مجموعه Workbooks نمی‌باشد.
- پنجره ماکروهای اکسل، ماکروهای تعریف شده برای فایل Add-In نصب شده را در لیست ماکروهای خود نمایش نمی‌دهد.
- شما می‌توانید توابع تعریف شده در فایل Add-In نصب شده را در هر صفحه استفاده کنید بدون اینکه لازم باشد به نام فایل Add-In نصب شده در ابتدای نام تابع اشاره کنید.

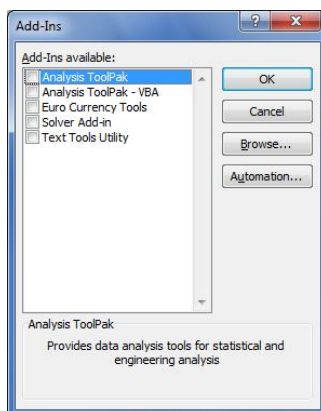
## 3-14 نصب یک Add-In

برای نصب یک فایل Add-In در پنجره Excel Options به برگه Add-Ins مراجعه کنید. (شکل ۱۴-۲)



شکل ۲-۱۴

سیس از لیست manage در پایین پنجره گزینه Excel Add-Ins را انتخاب کنید و با کلیک روی دکمه Go پنجره Add-Ins برای شما نمایش می یابد. (شکل ۳-۱۴)



شکل ۳-۱۴

حال با کلیک روی دکمه Browse و انتخاب فایل Add-In مورد نظر، نصب فایل انجام می شود.



## 4-14 ایجاد یک Add-In

مهمترین موضوع قبل از تبدیل یک فایل به Add-In این است که فایلی می تواند به Add-In تبدیل شود که ماهیتش قابلیت استفاده عمومی را داشته باشد و حداقل دارای یک WorkSheet باشد.

نکته دیگر اینکه قبل از تبدیل فایل و انتشار آن می توانید با رمز گذاری بر روی کدهای VBA مالکیت محتوا را در اختیار خود حفظ کنید.

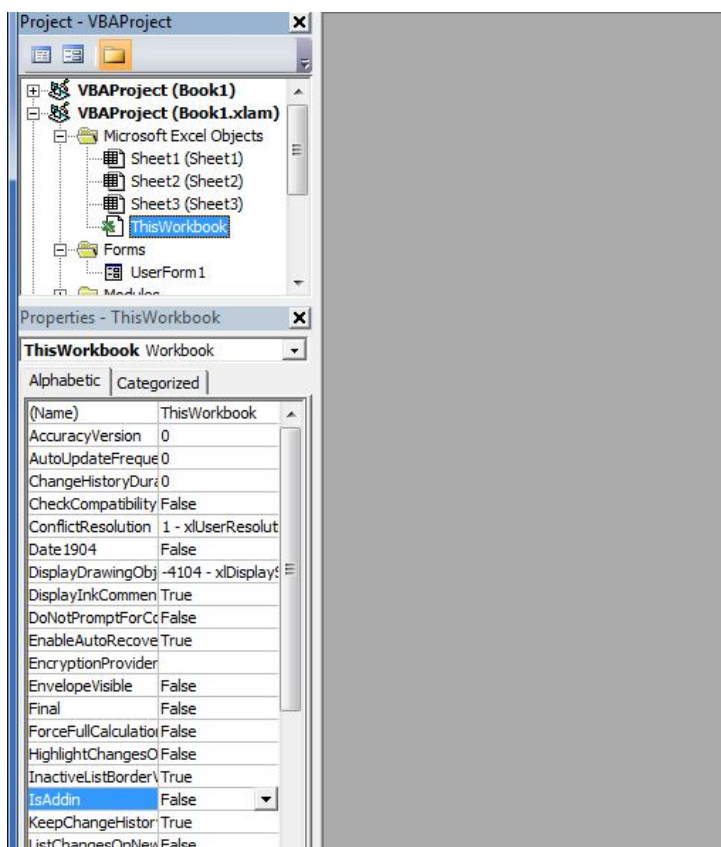
بعد از ذخیره یک کپی از فایل با پسوند XLSM و انجام حفاظت، مجدداً پنجره Save as را باز کنید و فایل را با پسوند XLAM ذخیره کنید.

حال برای نصب Add-In ذخیره شده کافیست که پنجره Add-In Manager را باز کنید و با کلیک روی دکمه Browse فایل را برای نصب

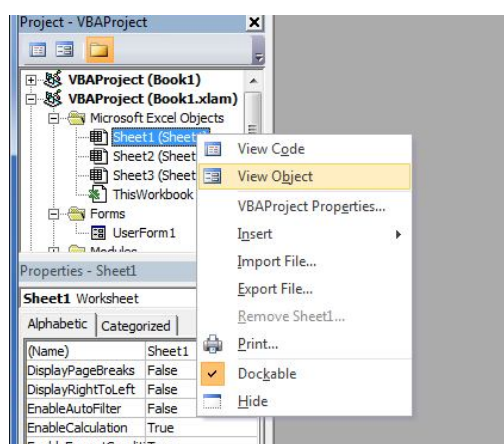
انتخاب کنید.

## 5-14 ویرایش Add-In

برای ویرایش Add-In نصب شده کافیست که کدهای آن را از حالت حفاظت خارج کنید و سپس به کدها دسترسی خواهید داشت. نکته: اگر می خواهید Worksheet های فایل Add-In را مشاهده کنید باید ابتدا ویژگی IsAddIn را برای شیء ThisWorkbook از طریق پنجره Properties به مقدار False تغییر دهید.



سپس روی Worksheet مورد نظر از فایل Add-In در پنجره Project Explorer راست کلیک کرده و گزینه View Object را انتخاب کنید.



## 14-6 استفاده از Add-In نصب شده

در مورد استفاده از یک Add-In نصب شده نکات زیر مورد توجه می باشد :

1. معمولا برای دسترسی به فرم ها و امکانات یک Add-In کنترل هایی به روبان ها اضافه می شوند و این مهمترین روش استفاده از امکانات یک Add-In می باشد. به طور مثال وقتی Solver را نصب می کنید ، یک کنترل به همین نام در انتهای روبان Data اضافه می شود.
2. اگر نام رویه های عمومی یک Add-In را می دانید، می توانید با باز کردن پنجره ماکرو ها و نوشتن نام رویه و کلیک روی دکمه Run، رویه را اجرا کنید.
3. توابع عمومی تعریف شده در فایل Add-In، در گروه User Defined از پنجره توابع قرار گرفته و به طور عمومی قابل استفاده می باشند.

## 14-7 رویدادهای Add-In

دو رویداد از شیء Workbook مستقسما به Add-Ins مرتبط هستند.

AddInInstall : در زمان نصب Add-In فراخوانی می شود.

AddInUnInstall : در زمان حذف Add-In فراخوانی می شود.

مثال : کد زیر در زمان نصب Add-In، نام فایل و نصب آن را به کاربر اعلان می کند.

```
Private Sub Workbook_AddInInstall()
MsgBox ThisWorkbook.Name & " add-in has been installed."
End Sub
```