الإصدار 1.0

أساسيات البرمجة باستخدام

EXCEL VBA

نضال الشامي

أساسيات البرمجة باستخدام

Excel VBA

نضال الشامي

أساسيات البرمجة باستخدام

Excel VBA

المؤلف

نضال الشامي

الإصدار الأول

2020

الغلاف

نضال الشامي

صورة الغلاف

من الانترنت

محفوظة لصاحبها

جدول المحتويات

جدول المحتويات

جدول المحتويات

مقدمة

تُعد لغة البرمجة؛ فيجوال بيسيك للتطبيقات Visual Basic for Applications (VBA) من الأدوات القوية جداً التي يمكن استخدامها مع مجموعة برامج أوفيس لأتمتة المهام المتكررة واضافة ميزات جديدة مثل انشاء دوال جديدة يمكن استخدامها ضمن أوراق العمل في اكسيل أو انشاء برامج خاصة تعمل ضمن البرنامج المضيف (مثل الاكسيل) فمثلاً يمكنك انشاء برنامج لإدارة العملاء أو اصدار الفواتير أو ادخال البيانات وغيرها. وقد أعلنت شركة مايكروسوفت في أكثر من مناسبة بأنها سوف تستمر بدعم لغة VBA وأنها لا تنوي التخلي عنها ولا صحة للإشاعات التي انتشرت ذات مرة بأنها تنوي التخلي عن لغة VBA.

وعلى الرغم من القوة والمرونة التي توفرها لغة VBA إلا أنك سوف تتفاجأ من مدى سهولتها. حيث أنها تقدم لك العديد من الأدوات التي تسهل عليك كتابة برامجك ويعتبر محرر الأكواد الخاص بـ VBA أحد أفضل البيئات البرمجية وأكثرها سهولة ومرونة حيث يتمتع بنظام مساعدة Help فعال يساعدك في أثناء كتابة أكوادك من خلال ميزات مثل الاكمال التلقائي واكتشاف الأخطاء الفوري وغيرها من الميزات.

بل إنك تستطيع انشاء بعض برامج الـ VBA بدون كتابة سطر برمجي واحد وذلك عن طريق استخدام مسجل الماكرو! حيث أنك في هذه الحالة تقوم بتشغيل مسجل الماكرو ثم تقوم بأداء المهام التي تريد أن يقوم الماكرو بتكرارها وعند الانتهاء توقف التسجيل وتقوم بحفظ الماكرو لاستدعائه عند الحاجة. هذه الوسيلة فعالة ليس فقط لإنجاز أعمالك بسرعة ودقة، بل أيضاً للتعلم حيث أنك إذا أردت أن تعرف كيفية القيام بعملية معينة من خلال الـ VBA فإنه بإمكانك أن تُشغل مسجل الماكرو ثم تقوم بأداء تلك العملية ثم توقف المسجل وتعرض الكود الموافق للماكرو الذي تم تسجيله.

ومن الميزات العظيمة للغة VBA أن الكود الخاص بها يستخدم كلمات وجمل واضحة وقريبة من استخدامنا اليومي مما يسهل كتابة وقراءة الكود. خذ عندك الكود التالي كمثال:

```
Worksheets("sheet1").Activate
Range("A1").Value = "Hello World!!"
Range("A1").Interior.Color = vbGreen
```

يكفي أن يكون لديك معرفة بأساسيات الاكسيل لتدرك أن هذا الكود يقوم بتنشيط ورقة العمل Sheet1 ثم يكتب جملة "Hello World!!" بداخل الخلية A1 ويقوم بتلوين تلك الخلية باللون الأخضر، حتى ولو لم يكن لك خبرة مسبقة بأي من لغات البرمجة.

يغطي هذا الكتاب أساسيات برمجة الاكسيل بلغة VBA من الصفر، انطلاقاً من فرضية أن القارئ ليس له سابق خبرة بالبرمجة سواءً بلغة VBA أو غيرها. إذا لم يكن لك سابق خبرة بتسجيل واستخدام وحدات الماكرو أو كتابة أكواد VBA فإن هذا الكتاب سيأخذ بيدك خطوة خطوة على طريق اجادة التعامل مع هذه اللغة.

لقد اعتمدنا في كتابة هذا الكتاب على تقديم المعلومة بأبسط شكل وعلى تقديم أمثلة راعينا فيها قدر المستطاع أن تكون ذات فائدة عملية وفي نفس الوقت توضح المبدأ التي وُضعت لأجله. بالنسبة للأمثلة فقد رأينا أن عدم تعقيدها يساهم في وصول الفكرة بشكل أفضل للقارئ بدون تشتيت، لذلك فقد بسطنا الأمثلة قدر المستطاع بدون اخلال بفكرة المثال أو بالمبدأ الذي يوضحه المثال. ومن المفترض أنك إذا فهمت المثال المُعطى بشكل جيد فإنه سوف يكون من السهل عليك تطوير المثال ليصبح أكثر تعقيداً.

وعلى الرغم من أن هذا الكتاب مكتوب خصيصاً لبرمجة الاكسيل بلغة VBA إلا أن إلمامك بالأساسيات المذكورة في الكتاب سوف يسهل عليك من تعلم برمجة برامج أخرى من طقم برامج أوفيس (مثل وورد وأكسس) بلغة VBA. بل إن الأساسيات المذكورة هنا سوف تساعدك في تعلم أي لغة برمجة أخرى مثل بايثون أو جافا سكربت.

تم تقسيم الكتاب إلى اثني عشر فصلاً هي كالتالي:

- **الفصل الأول، مدخل إلى VBA** : وفيه تم التطرق إلى أساسيات لغة VBA وأهم العناصر المكونة لها مثل الوحدات النمطية والدوال والاجرائيات كما وتم شرح مفهوم البرمجة كائنية التوجه

Object Oriented Programming ونموذج الكائنات Object Model وخصائصه. يعد هذا الفصل أهم فصول الكتاب وهو يمهد ويضع الأساس للفصول اللاحقة.

- **الفصل الثاني، محرر الأكواد VBE:** وفيه ثم التعرض لمحرر الأكواد Visual Basic Editor وهي البيئة البرمجية الخاصة بلغة VBA التي يتم من خلالها ادخال وتعديل الأكواد. تم شرح عناصر ونوافذ محرر الأكواد باستفاضة كما تم في هذا الفصل شرح كيفية كتابة برنامجك الأول في VBA وكيفية حفظه واستدعاءه عند الحاجة بأكثر من طريقة.
- **الفصل الثالث، مسجل الماكرو:** وفيه تم شرح طريقة استخدام مسجل الماكرو Macro Recorder الذي يعد أبسط وأسرع الطرق لإنشاء وحدات الماكرو (البرامج المكتوبة بلغة VBA يطلق عليها ماكرو). هنا تم شرح كيفية تسجيل وحفظ الماكرو ثم كيفية فحص الكود الموافق للماكرو المُسجل والتعديل عليه.
- **الفصل الرابع، المتغيرات:** وفيه تم شرح مفهوم المتغيرات Variables وأنواعها وكيفية الإعلان عنها ونطاقها وكيفية تعيين قيم لتلك المتغيرات. كما وتم التطرق في هذا الفصل للمصفوفات Arrays بأنواعها المختلفة وكيفية الإعلان عنها والتعامل معها.
- **الفصل الخامس، التعامل مع كائنات النطاق:** وفيه تم التطرق لكائنات النطاق Range Objects وخصائصها وتم شرح أشهر الوظائف Methods التي يمكن تطبيقها على النطاقات.
- **الفصل السادس، جمل التحكم:** من خلال جمل التحكم Control statements، تستطيع التحكم بسير البرنامج بناءً على تحقق شرط أو مجموعة من الشروط. في هذا الفصل تم التطرق لجمل التحكم في VBA مثل IF.... Then و Case ..... Select وكيفية استخدامها مع العديد من الأمثلة. كما تم شرح العوامل المنطقية Logical Operators وكيفية استخدامها مع جمل التحكم.
- **الفصل السابع، جمل التكرار:** من خلال جمل التكرار تستطيع تكرار تنفيذ أوامر محددة حتى يتحقق شرط معين. هنا تم شرح العديد من جمل التكرار مثل For....Next و Do.....While وغيرها بتفصيل كبير وبعدة حالات وأمثلة.
- **الفصل الثامن، الدوال:** الدالة Function هي عبارة عن مجموعة من الأسطر البرمجية تم تنظيمها بشكل معين بحيث تقوم ببعض الحسابات أو الإجراءات ومن ثم ترجع قيمة واحدة. وفي هذا الفصل تم التطرق لما هي الدوال؟ وأنواع الدوال في VBA حيث أنها تنقسم لثلاثة أنواع: الدوال المدمجة في VBA ودوال ورقة العمل والدوال المخصصة. تم شرح الأنواع الثلاثة مع إعطاء أمثلة على كل نوع.

- **الفصل التاسع، التعامل مع الأخطاء:** الأخطاء من الأمور الشائعة جداً عند التعامل مع VBA ومع أي لغة برمجة أخرى. وفي هذا الفصل تم التطرق لأنواع الأخطاء التي يمكن أن تظهر أثناء تشغيل برامج VBA وكيفية تحديد مكان الخطأ ثم التعامل معه بأكثر من طريقة. كما وتم التطرق في هذا الفصل لمفهوم "إزالة الشوائب" Debugging وتم شرح العديد من التقنيات التي يمكن استخدامها لإزالة شوائب البرامج.
- **الفصل العاشر، الأحداث:** احدى طرق تنفيذ اجرائيات VBA هي تنفيذها تلقائياً عند حدوث حدث معين. والحدث Event، هو ببساطة عبارة عن شيء حدث للاكسيل. وفي هذا الفصل سوف نتعرف على الأحداث الأكثر شيوعاً في الاكسيل وكيفية الاستفادة منها في تنفيذ الاجرائيات بشكل تلقائي.
- **الفصل الحادي عشر، التفاعل مع المستخدم من خلال دالتي MsgBox و InputBox:** وفيه تم عرض حالات مختلفة لكل من دالة MsgBox التي تقوم بعرض البيانات ضمن مربع حوار مع إمكانية أخذ تغذية راجعة من المستخدم، ودالة InputBox التي تستخدم لإدخال البيانات من خلال مربع حوار. إضافة إلى ذلك فقد تم التطرق لدالة Application.InputBox والتي تُستخدم لادخال أشياء مثل المعادلات أو مراجع النطاقات.
- **الفصل الثاني عشر، النماذج:** تعد النماذج User Forms من الوسائل الجيدة لإدخال البيانات وعرضها. حيث أنك من خلالها تستطيع التحكم بكيفية ادخال البيانات من حيث ترتيب ادخال البيانات أو اختيار الوسيلة المناسبة لكل نوع من البيانات المطلوبة أو اجراء عملية التحقق من الادخال للبيانات المدخلة وغيرها من الأشياء. ويمكنك أيضاً التحكم بكيفية عرض البيانات بالشكل الذي يلائم طبيعة البيانات المعروضة مما يعطيك مرونة عالية جداً في التعامل مع البيانات. في هذا الفصل تم شرح كيفية انشاء النماذج وكيفية إضافة عناصر التحكم عليها مثل أزرار الاختيار والقوائم المنسدلة وغيرها، وكيفية برمجة عناصر النموذج للاستجابة للأحداث المختلفة مثل الاستجابة للضغط على الأزرار أو الاختيار من القائمة المنسدلة.

لمن موجه هذا الكتاب

هذا الكتاب موجه لمستخدمي الاكسيل الذين يرغبون باكتساب المهارات الأساسية للبرمجة بلغة VBA، مما سيسهم بالانتقال بمستوى مهارتهم في الاكسيل للأعلى. للمضي قدماً في هذا الكتاب يجب أن يتمتع القارئ بمستوى متوسط على الأقل في استخدام برنامج اكسيل، ذلك يشمل أساسيات برنامج اكسيل، التعامل مع الخلايا والنطاقات وأوراق العمل، استخدام أدوات الاكسيل المختلفة مثل الجداول

والمخططات، واجادة التعامل مع المعادلات والدوال. ويمكن لمن أحب الرجوع لكتابنا " اكسيل 2019 – الدليل السهل" لاكتساب مثل هذه المعرفة أو إلى أي كتاب آخر يشرح مبادئ إكسيل. كما ويمكنك متابعة دورتنا على اليوتيوب " مهارات اكسيل من الصفر للاحتراف" من خلال الرابط بالأسفل.

كيف تقرأ هذا الكتاب؟

أنصحك بقراءة فصول هذا الكتاب بنفس الترتيب، مع تطبيق الأمثلة أولاً بأول وبعد الانتهاء من الفصل وفهمه تماماً الانتقال للفصل الذي يليه حيث أن الفصول تعتمد على بعضها البعض اعتماداً كبيراً. في غير هذا الكتاب كنت أنصح القارئ بأن يقرأ جدول المحتويات ثم يذهب للمواضيع التي تهمه اختصاراً لوقته، إلا أن الموضوع هنا مختلف لذا فأنصحك بقراءة كامل الكتاب حيث أنه في كل فصل هناك معلومات وتقنيات جديدة يمكن تطبيقها على موضوع الفصل وعلى موضوعات أخرى مشروحة في فصول أخرى، فالفصول تُكمل بعضها البعض ولن تصلك الصورة كاملة إلا بعد قراءة جميع فصول الكتاب.

روابط قد تهمك

رابط دورة مهارات اكسيل على اليوتيوب

رابط دورة الاكسيل المتقدم على اليوتيوب

رابط كتاب " اكسيل 2019 – الدليل السهل"

رابط كتاب "إكسيل 2013 – المستوى المتقدم"

في النهاية أتمنى أن يضيف هذا الكتاب جديداً للمكتبة العربية وأن يكون ذا عون بالنسبة للقراء والمهتمين.

نضال الشامي

غزة، نوفمبر 2019

الفصل الأول

1. مدخل إلى Excel VBA

في هذا الفصل سوف نتعرف إلى أساسيات لغة البرمجة VBA للاكسيل وسوف نتطرق إلى العناصر الأساسية المكونة لها مما يمهد الطريق للفصول اللاحقة. المعلومات الواردة في هذا الفصل مهمة جداً وتعتمد عليها بشكل كلي الفصول القادمة لذا ننصح بقراءته بعناية قبل الانتقال لباقي الفصول.

إذن، ما هي VBA ؟

لغة VBA هي اختصار لـ Visual Basic for Application وهي لغة برمجة مقدمة من مايكروسوفت تستطيع من خلالها برمجة تطبيقات الأوفيس مثل الوورد والاكسيل بالإضافة لبعض البرامج الأخرى مثل أوتوكاد للقيام بمهام غير موجودة في تلك البرامج. بالنسبة للاكسيل، هناك العديد من المهام التي تستطيع أن تقوم بها VBA مثل:

- أتمتة العمليات المتكررة لاختصار الوقت والجهد.
- تطوير دوال جديدة غير موجودة بالاكسيل.
- انشاء نماذج لتسهيل ادخال وعرض البيانات.
- انشاء وظائف إضافية Add-ins للاكسيل.
- إضافة النصوص المتكررة مثل اسم الشركة أو عنوان الايميل وغيرها من البيانات حيث يمكنك برمجة ذلك وتعيين زر لإدراج تلك النصوص بسهولة وسرعة.
- .... وغيرها الكثير

أهم العناصر والمفاهيم في لغة VBA في إكسيل

الوحدات البرمجية VBA Module

الكود البرمجي للـ VBA يتم إدخاله بداخل وحدة برمجية VBA Module، سواء أكان الكود مدخلاً يدوياً أو من خلال مسجل وحدات الماكرو Macro Recorder. يمكن مشاهدة وتحرير الكود من خلال "محرر الأكواد VB Editor"

سيتم التطرق لمحرر الأكواد VBE بتفصيل كبير في الفصل القادم "استخدام محرر الأكواد VBE"

تحتوي الوحدة البرمجية VBA Module على إجرائيات فرعية Sub Procedures و/أو دوال Functions.

الإجرائيات Procedures

تنقسم الاجرائيات إلى نوعين:

1- الاجرائيات الفرعية Sub-Procedures
2- الدوال Function

الإجرائية الفرعية Sub Procedure : هي عبارة عن مجموعة من الأوامر البرمجية التي تهدف إلى الوصول إلى نتيجة محددة. يجب أن تبدأ الإجرائية بالكلمة المفتاحية Sub ثم اسم الإجرائية متبوعاً بقوسين فارغين ( ) يلي ذلك الكود البرمجي وتنتهي بجملة End Sub. فيما يلي مثال على إجرائية تقوم بعرض رسالة الترحيب "Hello VBA":

```
Sub Hello_VBA()
  MsgBox ("Hello VBA")
End Sub
```

كما تم ذكره سابقاً لاحظ أن الإجرائية بدأت بـ Sub وانتهت بـ End Sub وما بينهما الكود البرمجي الخاص بالإجرائية، في حالتنا الكود يقوم بعرض مربع حوار يعرض الجملة "Hello VBA".

الإجرائية التي لا تظهر النتائج بالشكل المتوقع يطلق عليها Substandard (إجرائية دون المستوى)

الدالة Function: هي أيضاً عبارة عن إجرائية ولكنها بخلاف الإجرائية الفرعية تقوم بإرجاع قيمة محددة. بإمكانك استدعاء الدالة من داخل إجرائية أخرى أو استخدامها من داخل ورقة العمل بنفس طريقة استخدام دوال الاكسيل المدمجة Built-in functions. تبدأ الدالة بالكلمة المفتاحية Function متبوعاً باسم الدالة ثم الكود لبرمجي للدالة وتنتهي بالجملة End Function. ومن الممكن أن تأخذ الدالة مدخلات (تسمى معاملات الدالة Function parameters/arguments) ولكنها في جميع الأحوال يجب أن ترجع قيمة محددة. الدالة التالية هي مثال على دالة تقوم بأخذ رقمين من المستخدم ومن ثم ترجع حاصل جمعهما:

```
Function Add2Numbers(num1, num2)
   Add2Numbers = num1 + num2
End Function
```

لاحظ هنا أن الدالة قد تم تصميمها كي تأخذ معاملين هما num1 و num2 ومن ثم ترجع نتيجة جمعهما من خلال المتغير Add2Numbers وهنا ملاحظة مهمة جداً وهي أن القيمة التي ترجعها الدالة يجب أن يحملها متغير له نفس اسم الدالة.

ملاحظة////

للمزيد حول المتغيرات، انظر الفقرة الخاصة بالمتغيرات في هذا الفصل، أو انظر الفصل الرابع

بعد ادخال الدالة ضمن إحدى الوحدات البرمجية يصبح بالإمكان استخدامها من داخل أوراق العمل كما يظهر لنا في الشكل 1-1. كما أنه بالإمكان استدعاء الدالة من داخل الإجرائيات الفرعية أو دوال أخرى.

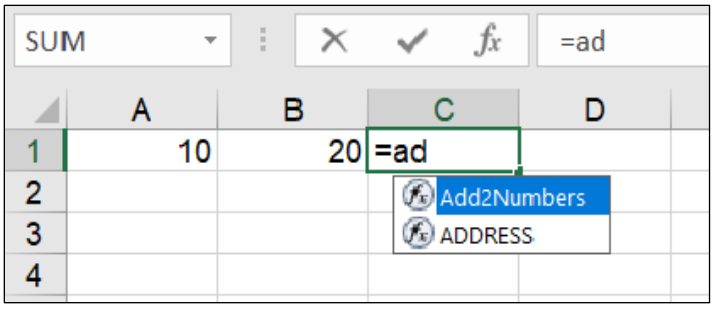


Figure 1-1

في الكود التالي تم استدعاء الدالة Add2Numbers من داخل الإجرائية الفرعية TestAdd

```
Sub TestAdd()
   arg1 = 10
   arg2 = 20
   result = Add2Numbers(arg1, arg2)
   MsgBox result
End Sub
```

البرمجة كائنية التوجه Object oriented Programming

فيجوال بيسيك للتطبيقات هي عبارة عن لغة برمجة كائنية التوجه Object Oriented Programming language. المبدأ الأساسي للغات البرمجة كائنية التوجه هي أن التطبيقات/البرمجيات (في حالتنا الإكسيل) تتكون من العديد من الكائنات Objects، كل كائن له مجموعة من الصفات/الخصائص Properties ومجموعة من الاستخدامات تسمى الدوال Methods. الاكسيل يحتوي على العديد من الكائنات مثل الخلايا Cells، أوراق العمل Worksheets، المخططات Charts، الأشكال Shapes، وغيرها الكثير.

بإمكانك أن تتعامل مع هذا المبدأ كما تتعامل مع أي كائن في الحياة الواقعية، لنأخذ السيارة كمثال؛ السيارة ككائن لها مجموعة من الخصائص Properties التي تصفها مثل اللون، الطول، العرض، والارتفاع. ولها مجموعة من الاستخدامات/الطرق Method مثل التحرك للأمام وللخلف كما أن لها مجموعة من الأحداث Events التي تحدد حالة السيارة الحالية مثل أن السيارة متوقفة أو متحركة هذه الأحداث يمكن استغلالها كقادح Trigger لتطبيق أوامر معينة عن بدء الحدث. الكود الذي يتم تطبيقه عند تحقق حدث معين يسمى Event Handler.

الاكسيل بالمثل يحتوي على العديد من الكائنات التي لها خصائص ودوال/طرق وأحداث تطبق عليها؛ فمثلاً المصنف Workbook هو عبارة عن كائن له مجموعة من الخصائص Features/Properties مثل اسم المصنف Name وحجمه Size ومجموعة من الدوال Methods مثل فتح المصنف Open وحفظ المصنف Save. وله مجموعة من الأحداث التي يمكن أن تُطبق عليه، مثل حدث فتح أو اغلاق المصنف.

نموذج الكائنات Object Model

يعد نموذج الكائنات لإكسيل هو القلب النابض لبرمجة VBA في برنامج الاكسيل، كل أمر برمجي يتم من خلال VBA يُرسل من خلال نموذج الكائنات. نموذج الكائنات هو عبارة عن قائمة طويلة بالكائنات التي يحتويها الاكسيل مثل أوراق العمل، النطاقات والخلايا.

كل شيء في الاكسيل هو عبارة عن كائن Object. يوفر لنا الاكسيل مجموعة كبيرة من الكائنات التي يمكن التحكم بها من خلال برمجة VBA. تشتمل الكائنات على المصنفات، أوراق العمل، الخلايا، المخططات وغيرها الكثير.

الكائنات تم ترتيبها بشكل هرمي Hierarchical arrangement بحيث أن بعض الكائنات تحتوي كائنات أخرى. في أعلى الهرم يقع برنامج الاكسيل نفسه. نعم الاكسيل بذات نفسه هو عبارة عن كائن ويدعى Application. كائن البرنامج Application يحتوي على كائنات مثل المصنفات Workbooks والوظائف الإضافية Add-Ins. يحتوي كائن المصنف Workbook على كائنات مثل أوراق العمل Worksheets والمخططات Charts. كائن ورقة العمل Worksheet يحتوي على كائنات مثل النطاقات Ranges، الخلايا Cells، والجداول المحورية PivotTables. المصطلح "نموذج الكائنات Object Model" يطلق على كيفية ترتيب الكائنات بداخل اكسيل.

التجميعات/المجموعات Collections

الكائنات من نفس النوع يطلق عليها "مجموعة Collection" فعلى سبيل المثال "مجموعة أوراق العمل Worksheets collection" تحتوي على جميع أوراق العمل بداخل المصنف ومجموعة الجداول المحورية PivotTables collection تحتوي على جميع الجداول المحورية بداخل المصنف. جدير بالملاحظة أن المجموعات Collections هي بحد ذاتها عبارة عن كائنات Objects.

تتم الإشارة إلى كائن ما عن طريق تحديد موقعه في التسلسل الهرمي للكائنات باستخدام النقطة كفاصلة. فمثلاً يمكن الإشارة إلى المصنف Workbook.xlsx من خلال:

```
Application.Workbooks ("Workbook1.xlsx")
```

وهنا تمت الإشارة إلى المصنف Workbook1.xlsx ضمن مجموعة المصنفات Workbook collection، حيث تقع مجموعة المصنفات ضمن "كائن البرنامج Application object". لتوسعة هذا الوصف ليشمل مستوىً آخر يمكن الإشارة لورقة العمل Sheet1 الواقعة ضمن المصنف Workbook1.xlsx من خلال:

```
Application.Workbooks("Workbook1.xlsx").Worksheets("Sheet1")
```

وللذهاب لمستوى آخر يمكن الإشارة إلى خلية محددة بداخل المصنف من خلال:

```
Application.Workbooks("Workbook1.xlsx").Worksheets("Sheet1").Range("A1")
```

عند حذف مرجع محدد يستخدم الاكسيل الكائنات النشطة Active objects. فمثلاً إذا كان المصنف Workbook1.xlsx هو المصنف النشط وأردنا أن نشير للخلية A1 الموجودة في ورقة العمل Sheet1 يمكن استخدام:

```
Worksheets("Sheet1").Range("A1")
```

وإذا كانت ورقة العمل Sheet1 هي ورقة العمل النشطة فيمكن الإشارة إلى الخلية A1 من خلال:

```
Range("A1")
```

خصائص الكائنات Objects Properties

كل كائن له مجموعة من الخصائص التي يمكن اعتبارها كإعدادات الكائن Object settings. فمثلاً كائن النطاق Range object له خصائص مثل العنوان Address والقيمة Value وبالمثل كائن ورقة العمل Worksheet object له خصائص مثل الاسم Name وعرض فواصل الصفحات DisplayPageBreaks.

من الممكن استخدام VBA لعرض قيمة خاصية محددة أو تعديلها.

يمكن الإشارة إلى خاصية كائن محدد من خلال ضم اسم الخاصية إلى اسم الكائن مفصولاً بينهما بنقطة. فيمكن الإشارة مثلاً إلى قيمة الخلية A1 بداخل ورقة العمل Sheet1 من خلال:

```
Workbooks("Workbook1.xlsx").Worksheets("Sheet1").Range("A1").Value
```

المتغيرات Variables

المتغير هو عبارة عن عنصر له اسم يستخدم لتخزين البيانات. من الممكن استخدام المتغيرات في الـ VBA لتخزين أشياء مثل النصوص، القيم أو اعدادات الخصائص للعناصر Object property settings. لإعطاء قيمة الخلية A1 في ورقة العمل Sheet1 للمتغير GoodsCost استخدم الصيغة التالية:

```
GoodsCost = Worksheets("Sheet1").Range("A1").Value
```

سيتم التطرق للمتغيرات بتفصيل كبير في الفصل الرابع

الاستخدامات/الدوال Methods

الدالة Method هي عبارة عن اجراء يقوم الاكسيل بتنفيذه على عنصر ما. على سبيل المثال؛ أحد الدوال الخاصة بالكائن Worksheet هي عبارة عن دالة الحذف Delete وهي تقوم بحذف ورقة العمل. بإمكانك أن تطبق دالة ما على أحد الكائنات من خلال ارفاق اسم الدالة للكائن المطلوب بحيث يفصل بينهما النقطة. كمثال على تطبيق دالة على أحد الكائنات انظر المثال التالي حيث يقوم السطر البرمجي التالي بحذف ورقة العمل المسماة Sheet1:

```
Worksheets("Sheet1").Delete
```

ما تم ذكره في الفقرات السابقة يمثل المكونات الأساسية للغة VBA، تأكد من فهمك الجيد لتلك المفاهيم قبل الانتقال للفصول اللاحقة.

الفصل الثاني

2. محرر الأوامر (VBE)

في هذا الفصل سوف نتعرف على محرر الأكواد VBE، والذي من خلاله يتم ادخال أو تعديل الكود البرمجي لوحدات الماكرو. كما سنقوم بكتابة أول برنامج VBA وسنتعرف على الطرق المختلفة لاستدعاء وتشغيل الماكرو.

ما هو محرر أوامر فيجوال بيسيك لإكسيل Visual Basic Editor (VBE) ؟

قد لا يعرف معظم مستخدمي الاكسيل أن هناك جانباً خفيا للإكسيل بخلاف أوراق العمل والمخططات والجداول المحورية وغيرها من الأدوات. هذا الجانب هو محرر أوامر فيجوال بيسيك VBE حيث أنه يكون متواجداً دوما في الخلفية ويتم تنفيذ أية أوامر موجودة بداخله إذا استدعت الحاجة.

هاتان البيئتان تعملان بشكل متزامن وتتبادلان البيانات مع بعضهما البعض وقتما استدعت الحاجة. محرر VBE هو عبارة عن بيئة تطوير برمجية Integrated Development Environment IDE حيث بإمكانك ادخال، تحرير، وتنفيذ الأكواد البرمجية الخاصة بلغة VBA من خلالها.

كيفية إظهار محرر الأكواد VBE

أسرع طريقة للوصول لمحرر الأكواد هي من خلال الضغط على زري Alt+F11. كما أنه بالإمكان الوصول للمحرر من خلال تبويب المطور Developer. لا يظهر تبويب المطور بالوضع الافتراضي وانما يمكن اظهاره من خلال تخصيص شريط الأدوات Ribbon من خلال التالي:

1- انقر بزر الماوس الأيمن فوق أي مكان على شريط الأدوات Ribbon ثم اختر تخصيص الشريط Customize the Ribbon
2- سوف يظهر لك مربع الحوار "خيارات اكسيل Excel Options"، في الجانب الأيمن من مربع الحوار صع علامة التحديد على المطور Developer كما في الشكل 2-1

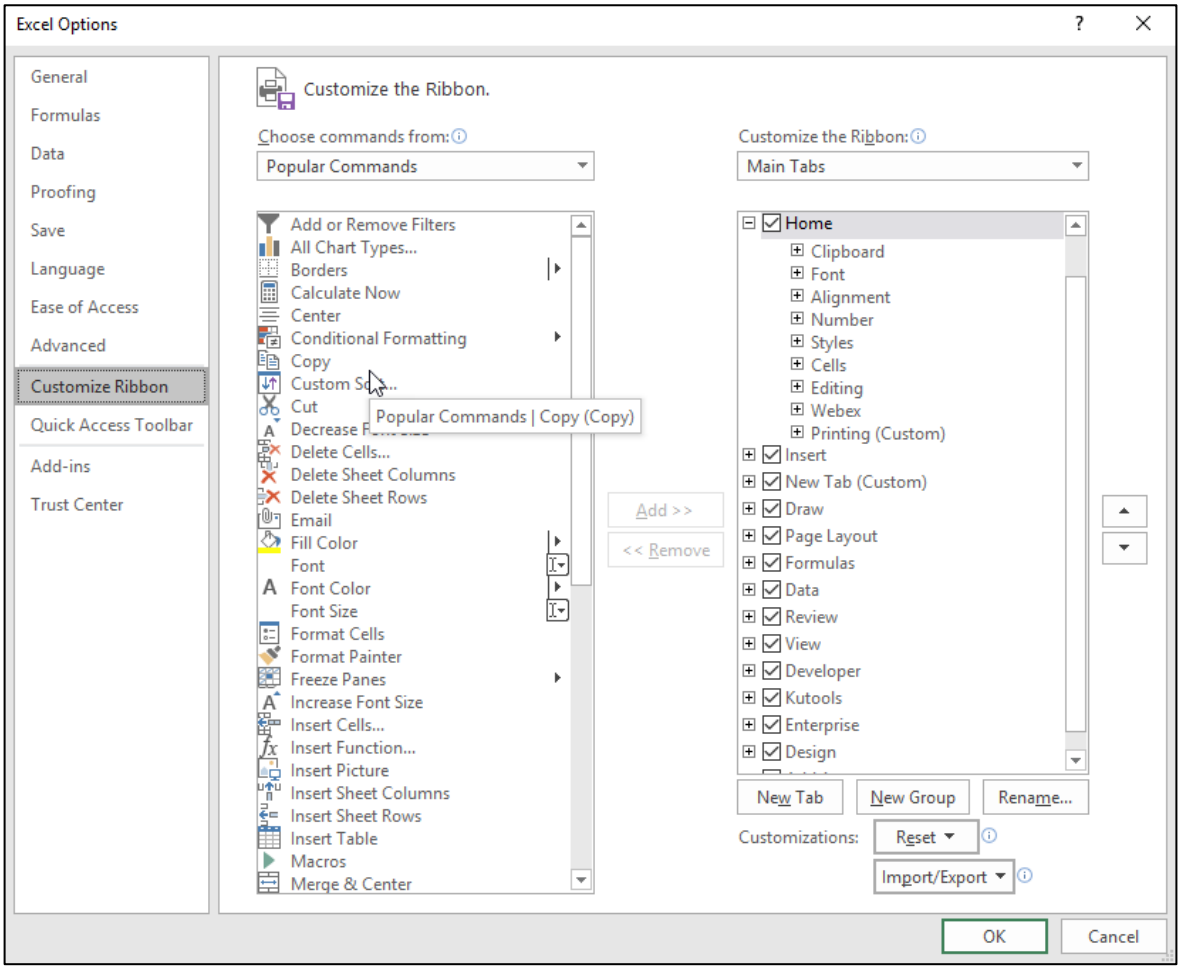


Figure2-1

لتشغيل محرر الأكواد من خلال شريط المطور، اذهب إلى تبويب Developer ثم اختر الأمر Visual Basic.

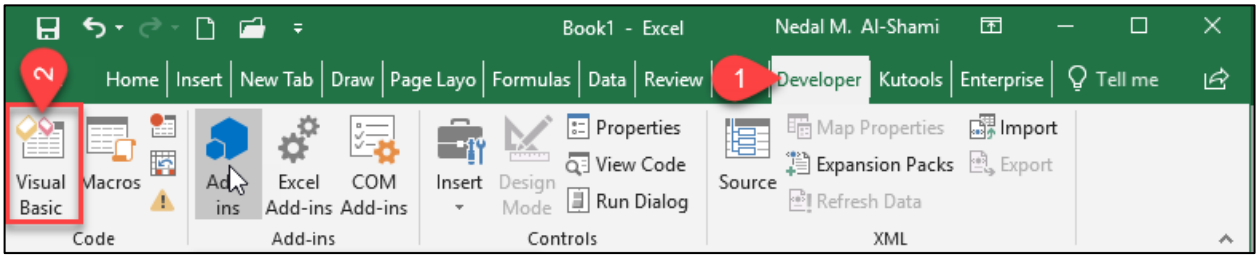


Figure2-2

عناصر ونوافذ محرر الأكواد VBE

يحتوي محرر الأكواد على العديد من النوافذ التي يمكن التحكم بإظهارها وإخفائها حسب الحاجة. في غالب الأوقات أنت قد تحتاج لأربعة نوافذ أساسية هي:

1- مستكشف المشاريع Project Explorer
2- نافذة الكود البرمجي Code Window
3- نافذة الخصائص Properties Window
4- نافذة الأوامر الفورية Immediate Window

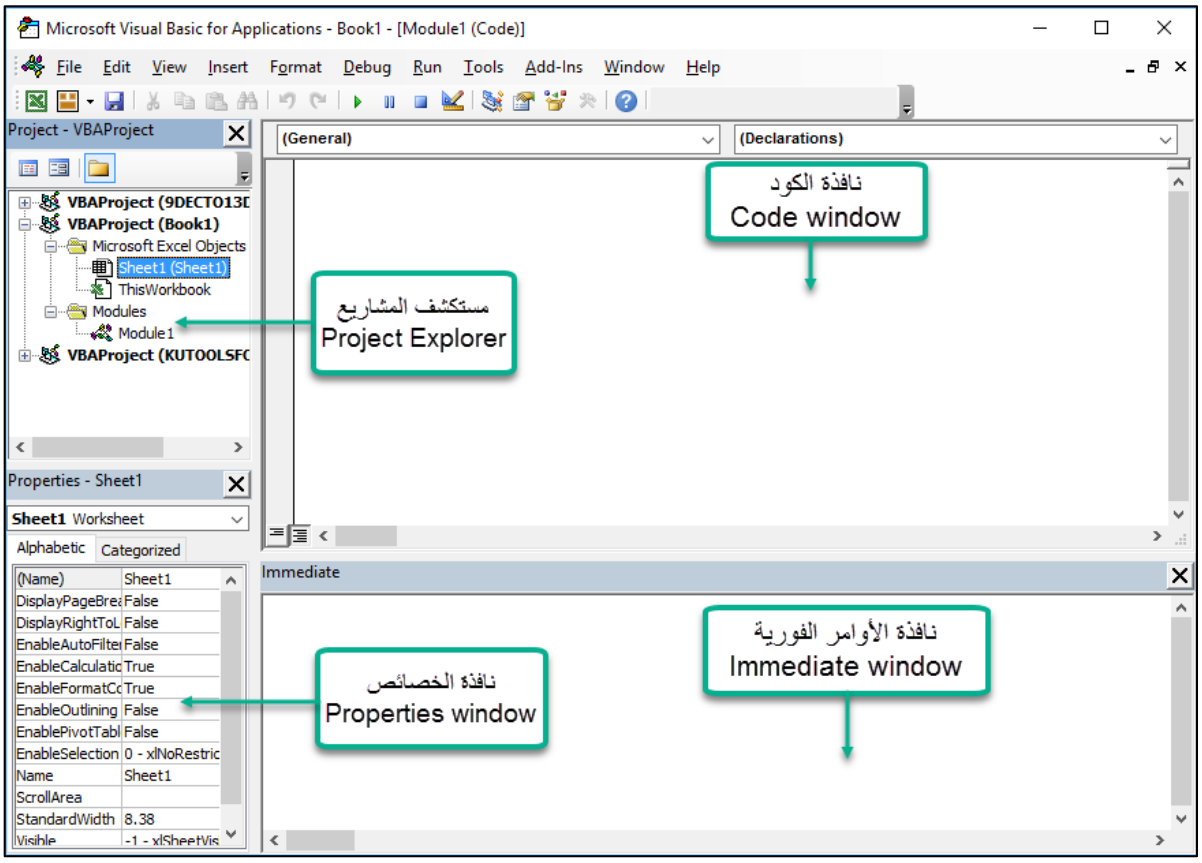


Figure2-3

مستكشف المشاريع Project Explorer

يظهر مستكشف المشاريع مخططاً هيكلياً شجرياً Tree diagram تظهر من خلاله المصنفات المفتوحة بما فيها من أوراق عمل (ذلك يشمل المصنفات المخفية Hidden workbook) كما تظهر من خلاله الوحدات البرمجية Modules الخاصة بكل مصنف مفتوح. ومن الممكن توسعة Expand أو طي العناصر القابلة للطي من خلاله.

عند النقر بالماوس نقراً مزدوجاً على أحد العناصر أسفل Microsoft Excel Objects أو Modules يظهر لنا كود الـ VBA الخاص بهذا العنصر في نافذة الكود Code Window.

إذا لم يكن مستكشف المشاريع ظاهراً فيمكن اظهاره من خلال الضغط على Ctrl + R أو من خلال الذهاب إلى View ➔ Project Explorer.

نافذة الكود Code Window

نافذة الكود هي المكان الذي تقوم بإدخال الكود فيه، كل عنصر يظهر في مستكشف المشاريع له نافذة كود مرتبطة به. لعرض الكود الخاص بأي عنصر انقر نقراً مزدوجاً فوق العنصر المطلوب فيتم اظهار الكود الخاص به في نافذة الكود (عادة تكون على يمين مستكشف المشاريع).

نافذة الخصائص Properties Window

تعرض نافذة الخصائص مجموعة الخصائص الخاصة بالعنصر المحدد، فمثلاً إذا كان لديك ورقة عمل باسم Sheet1 وترغب بعرض خصائصها يكفي اختيار هذه الورقة من مستكشف المشاريع ليتم عرض خصائصها في نافذة الخصائص، وكذلك الأمر بالنسبة لأي عنصر آخر في مستكشف المشاريع. ومن الممكن تعديل الخصائص من خلال تلك النافذة. هذه الخصائص تُستخدم للتحكم في مظهر وسلوك العنصر. فمثلاً في الشكل 4-2 تم تحديد ورقة العمل Sheet1 وبالتالي تم اظهار خصائصها في نافذة الخصائص، من هذه الخصائص؛ هناك خاصية DisplayRightToLeft للتحكم في اتجاه ورقة العمل (وهي من الخصائص المستخدمة للتحكم في المظهر)، بينما الخاصية EnableCalculation تستخدم للتحكم في كيفية اجراء الاحتساب بداخل ورقة العمل يدوي أم تلقائي، (وهي من الخصائص المستخدمة للتحكم في سلوك العنصر).

نافذة الأوامر الفورية Immediate Window

من خلال نافذة الأوامر الفورية يمكن تنفيذ الأوامر مباشرة، أي أنك غير مضطر لإدخال الأمر الذي تريد تطبيقه ضمن إجرائية. يمكنك كتابة الجملة البرمجية ثم الضغط على مفتاح الإدخال ليتم تنفيذ الجملة مباشرة مما يعطيك إمكانية التقييم الفوري للجمل البرمجية. عادة يتم اللجوء لهذه النافذة عند اجراء

عمليات تتبع الأخطاء والتصحيح Debugging. إذا لم تكن هذه النافذة ظاهرة فيمكن اظهارها من خلال الضغط على Ctrl + G.

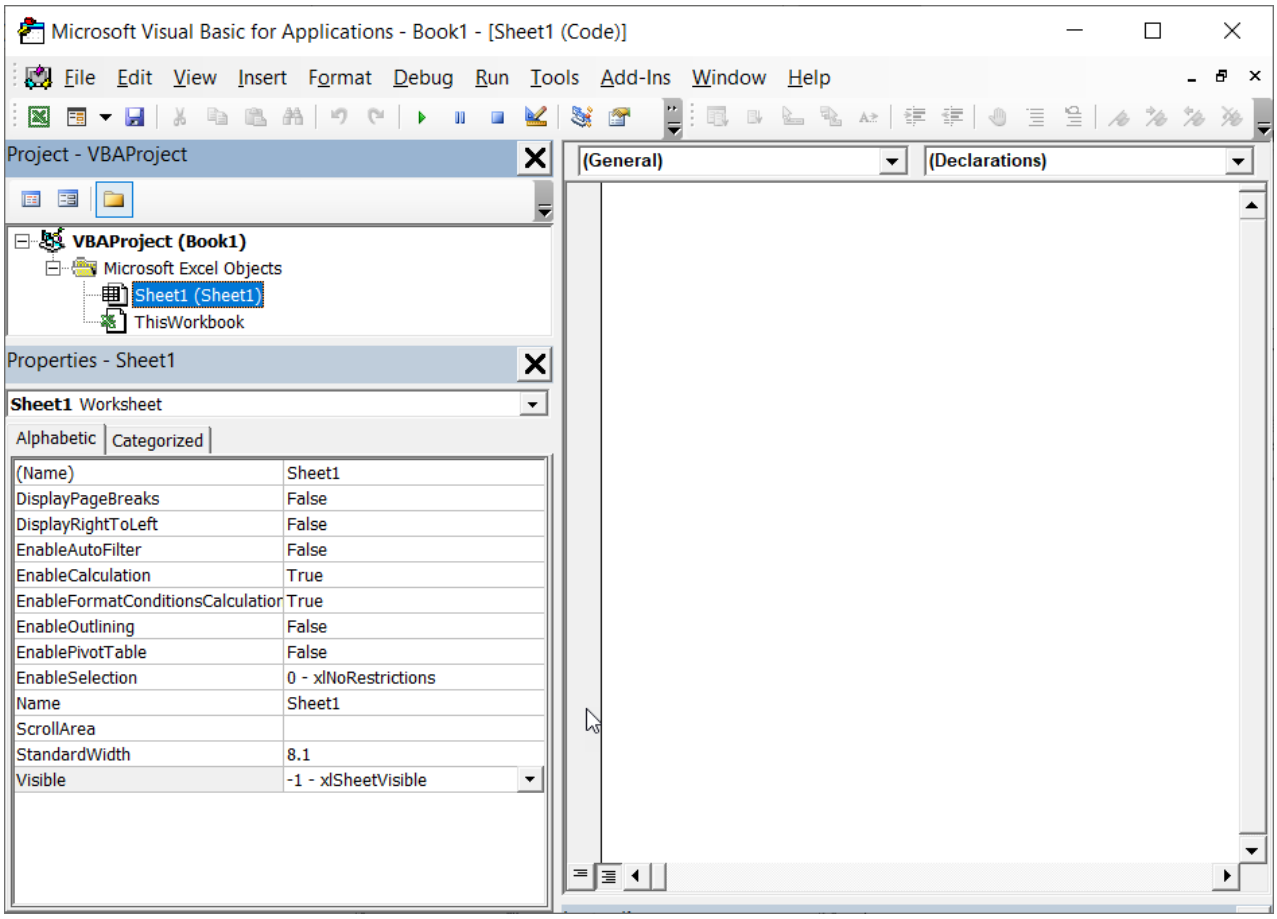


Figure2-4

كتابة برنامجك الأول بالـ VBA

الهدف من هذه الفقرة هو توضيح كيفية ادراج الكود في ملف الاكسيل ومن ثم كيفية تشغيله بأكثر من طريقة، لا تقلق إذا بدا لك الكود غير مفهوم فسوف يتم شرح وتوضيح عناصره بشكل مفصل في الفصول اللاحقة.

للبدء بكتابة برنامجك الأول في الـ VBA اتبع الخطوات التالية:

1- افتح ملف اكسيل جديد ثم اذهب إلى محرر الأكواد عن طريق الضغط على زري Alt+F11 أو من خلال الذهاب إلى تبويب المطور Developer ثم الضغط على زر Visual Basic.
2- أدرج وحدة برمجية جديدة Module من خلال الذهاب إلى قائمة "ادراج Insert" ثم Module أو من خلال النقر بزر الماوس الأيمن فوق اسم المشروع أو المصنف في نافذة مستكشف المشاريع ثم اختيار Module➔Insert أو من خلال شريط الأدوات القياسي كما هو ظاهر بالشكل 2-5 أو الشكل 2-6.

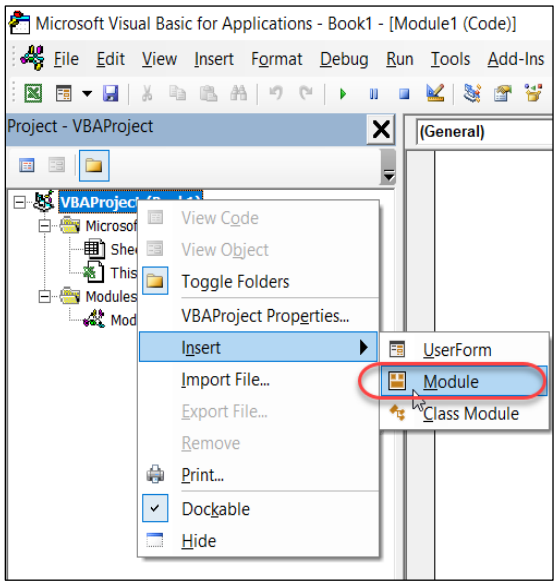


Figure2-5

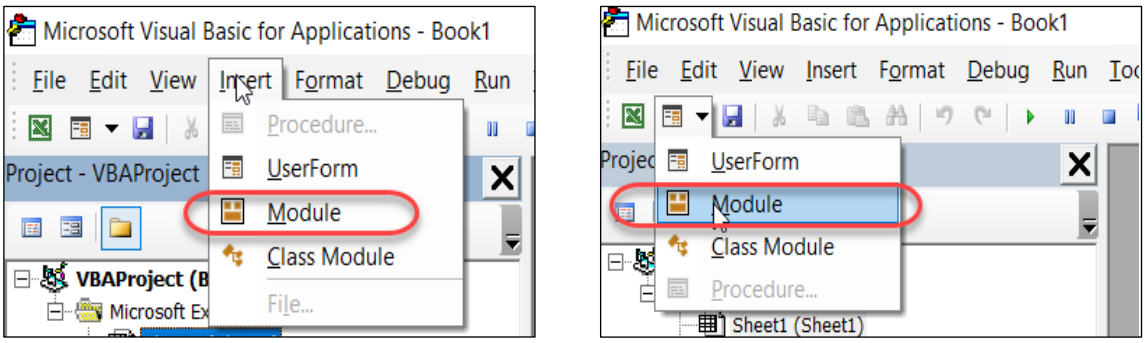


Figure2-6

3- من خلال مستكشف المشاريع اختر الوحدة البرمجية Module التي قمت ادراجها ثم حرك مؤشر الماوس إلى جزء الكود في الأيمن (إذا لم يكن جزء الكود ظاهراً لأي سبب انقر بزر الماوس الأيمن فوق الوحدة البرمجية ثم اختر View Code.

4- أدخل الكود التالي في جزء الكود

```
Sub HelloVBA()

   MsgBox ("Hello VBA!!")

End Sub
```

بهذا تكون قد أدخلت برنامجك الأول ولتشغيله اتبع إحدى الخطوات التالية:

1- من خلال محرر الأكواد، اضغط على زر Run أو اذهب إلى قائمة Run ثم اضغط Run Sub/UserForm

2- من ورقة العمل اذهب إلى تبويب Developer ثم اضغط على زر Macros ومن مربع الحوار الذي سوف يظهر اختر اسم الماكرو من مربع Macro name ثم اضغط على زر Run. شكل 2-7.

3- من الممكن إعطاء الماكرو اختصار على لوحة المفاتيح. ولعمل ذلك اختر اسم الماكرو كما في الخطوة رقم 2، ثم اضغط على زر Options ثم من نافذة Macro options حدد الاختصار

الذي تريده من خلال ادخال الأزرار المطلوبة في Shortcut key وأدخل وصف للماكرو في الخانة المخصصة للوصف Description كما في الشكل 2-7

4- من الممكن ربط الماكرو مع شكل عن طريق ادراج الشكل المطلوب ثم النقر بالزر الأيمن عليه واختيار Assign macro كما في الشكل 2-8

5- من الممكن ربط الماكرو مع زر Button من خلال الذهاب إلى تبويب Developer ثم إلى Insert ثم اختيار Button ومن ثم وضع الزر في المكان المطلوب على ورقة العمل، بمجرد وضع الزر على ورقة العمل يظهر لنا مربع حوار يطلب من تحديد الماكرو المرتبط مع هذا الزر؛ نحدد الماكرو ثم نضغط على OK. انظر شكل 2-9

Figure2-7

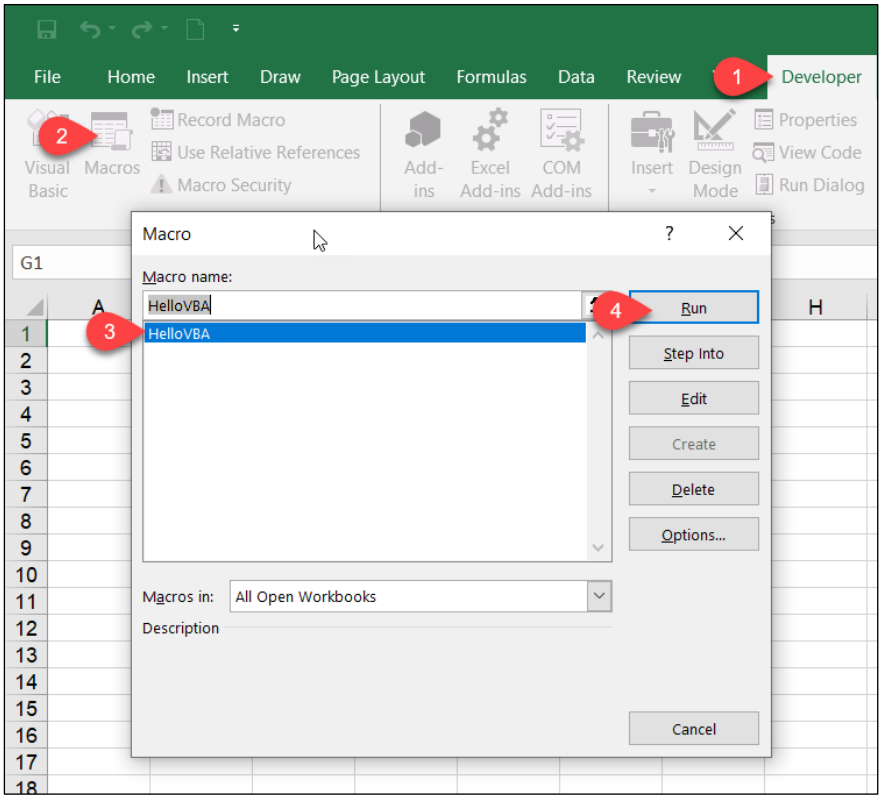


Figure2-8

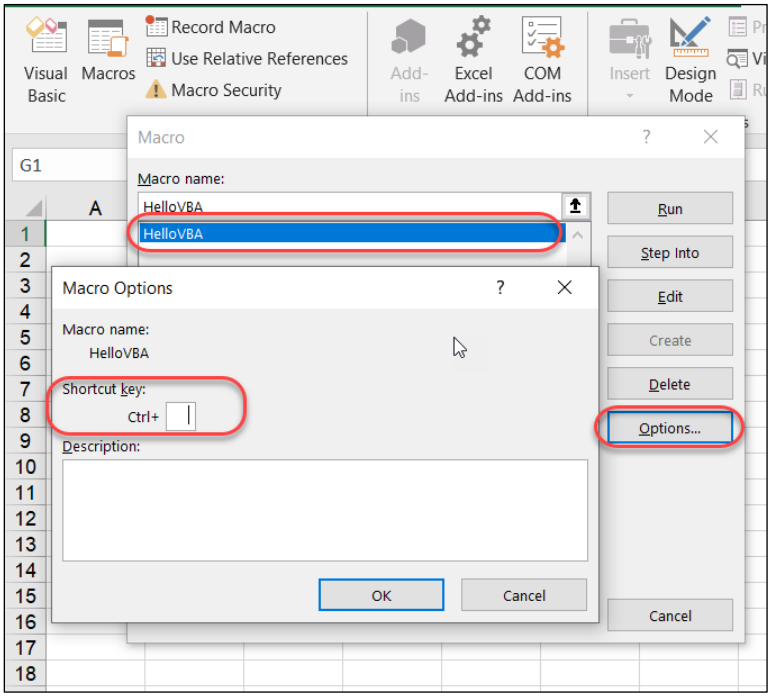
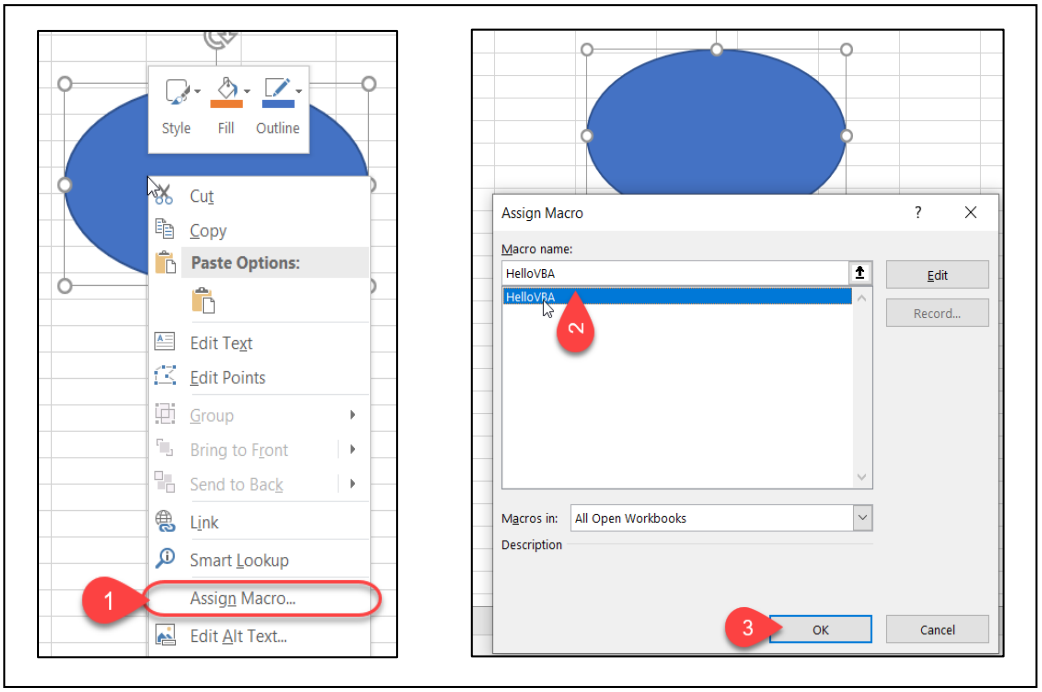


Figure2-9



أمان الماكرو Macro security

تعد لغة VBA من لغات البرمجة القوية جداً والتي يمكن استغلالها لعمل برامج ضارة فمن الممكن مثلاً انشاء برامج باستخدامها تقوم بحذف الملفات أو ارسال الملفات لأجهزة حاسوب أخرى أو تعطيل نظام التشغيل وغيرها من البرامج الخبيثة. لذلك قامت شركة مايكروسوفت – بدءاً من إكسيل 2007 – بإضافة هذه الخاصية لمساعدة المستخدمين على حماية أنفسهم من البرامج الضارة التي قد تكون مكتوبة بـ VBA.

يظهر الشكل 2-10 إعدادات الأمان الخاصة بالماكرو، والتي يمكن الوصول إليها من خلال الذهاب إلى:

File ➔ Options ➔ Trust Center ➔Macro Settings

هناك أربعة خيارات للتعامل مع الماكرو، بالوضع الافتراضي يتم تحديد الخيار الثاني والذي يقوم بتعطيل كافة وحدات الماكرو مع اعطاء اشعار للمستخدم حتى يستطيع السماح بتشغيل الماكرو إن رغب. يظهر الإشعار عادة في أعلى المصنف كشريط أصفر اللون مع وجود زر لتفعيل الماكرو (كما في الشكل 2-10).

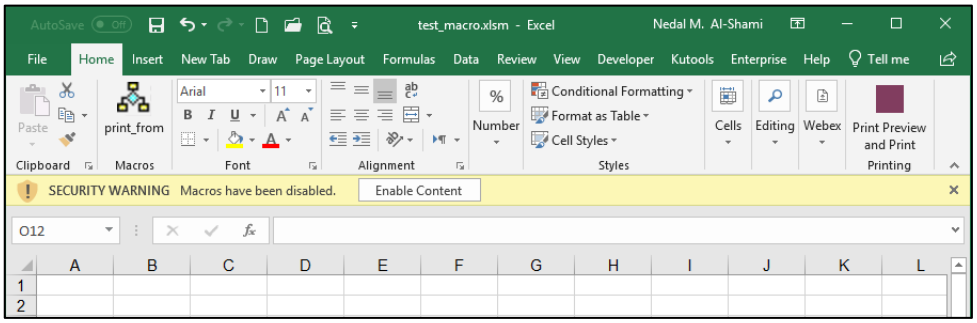


Figure2-10

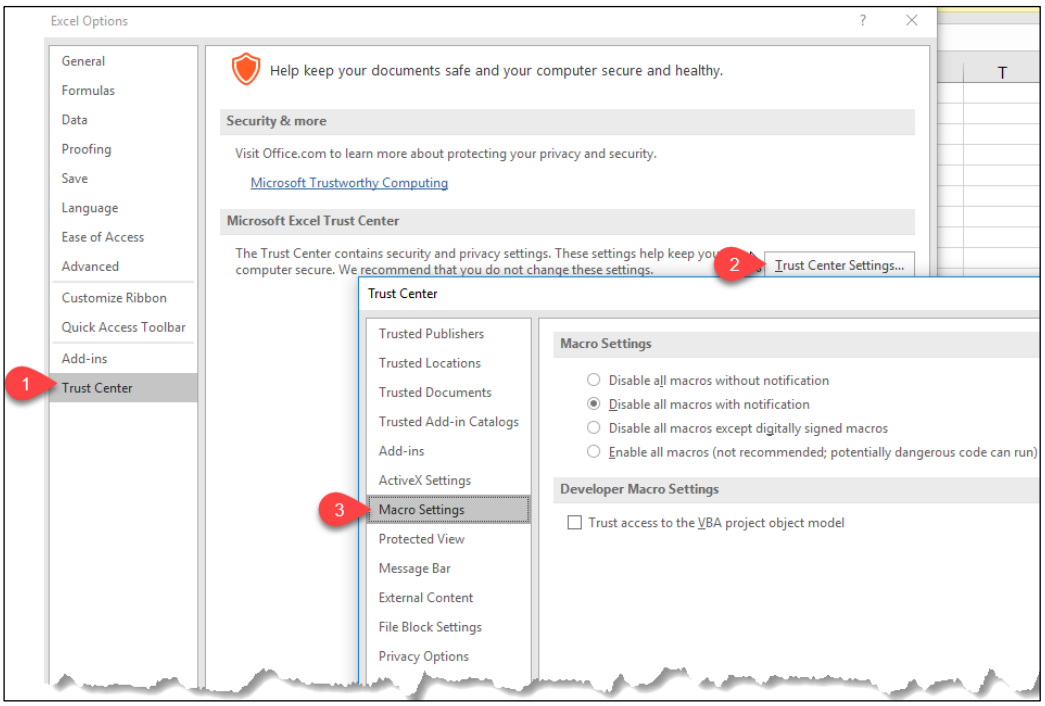


Figure2-11

حفظ ملفات الاكسيل التي تحتوي على ماكرو

لحفظ ملف اكسيل بداخله ماكرو، يجب اختيار الامتداد (Excel Macro-Enabled Workbook *.xlsm) وذلك من خلال الذهاب إلى File ثم Save as ثم اختيار مكان الحفظ وتحديد نوع الملف من خلال القائمة المنسدلة Save as type كما في الشكل 2-12. يمكنك أيضاً الضغط على زر F12 لعرض مربع الحوار الخاص بـ Save as مباشرةً.

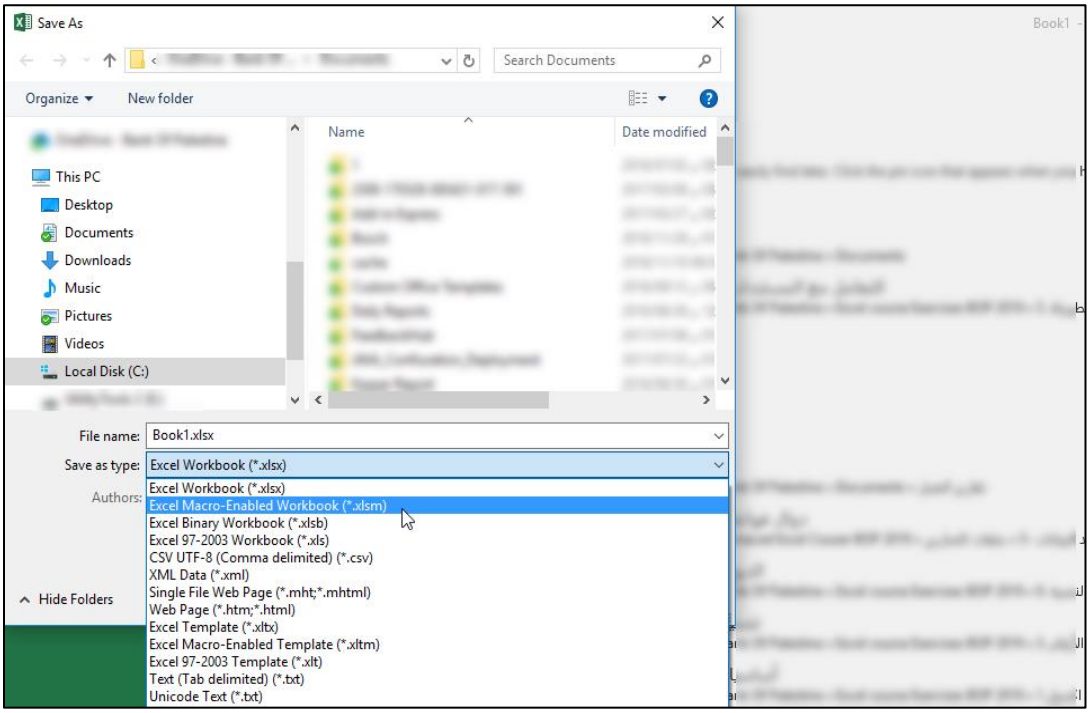

Save As
Search Documents
Organize
New folder
Name
Date modified
This PC
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
File name: Book1.xlsx
Save as type: Excel Workbook (*.xlsx)
Authors:
Hide Folders
Excel Workbook (*.xlsx)
Excel Macro-Enabled Workbook (*.xlsm)
Excel Binary Workbook (*.xlsb)
Excel 97-2003 Workbook (*.xls)
CSV UTF-8 (Comma delimited) (*.csv)
XML Data (*.xml)
Single File Web Page (*.mht;*.mhtml)
Web Page (*.htm;*.html)
Excel Template (*.xltx)
Excel Macro-Enabled Template (*.xltm)
Excel 97-2003 Template (*.xlt)
Text (Tab delimited) (*.txt)
Unicode Text (*.txt)

Figure2-12

الفصل الثالث

3. استخدام مسجل الماكرو

في هذا الفصل سوف نتعرف على أسهل الطرق للبدء في استخدام VBA وهي تسجيل وحدات الماكرو باستخدام مسجل الماكرو Macro Recorder وهي ميزة مدمجة في الاكسيل Built-In. باستخدام الماكرو سوف يكون بإمكانك أتمتة العمليات المتكررة والمضجرة مما سيسرع من انجاز أعمالك ويقلل من نسبة حدوث الأخطاء. كما أن بإمكانك أن تقوم بالوصول للكود البرمجي الناتج عن تسجيل الماكرو والتعديل فيه مما يعطيه قوة ومرونة قصوى.

بالإضافة إلى ذلك، يعد مسجل الماكرو من الوسائل الممتازة لتعلم VBA، حيث أنك إذا اردت أن تعرف كيفية اتمام تنفيذ عملية معينة باستخدام برمجة VBA، فبإمكانك تسجيل ماكرو يقوم بأداء تلك العملية ومن ثم تقوم بعرض كود الـ VBA الخاص بذلك الماكرو.

ولبدء التعامل مع الماكرو يجب أن يتم اظهار شريط المطور Developer عن طريق الضغط بزر الماوس الأيمن فوق شريط الأدوات Ribbon ثم Customize the ribbon (تخصيص الشريط) ثم نختار Developer

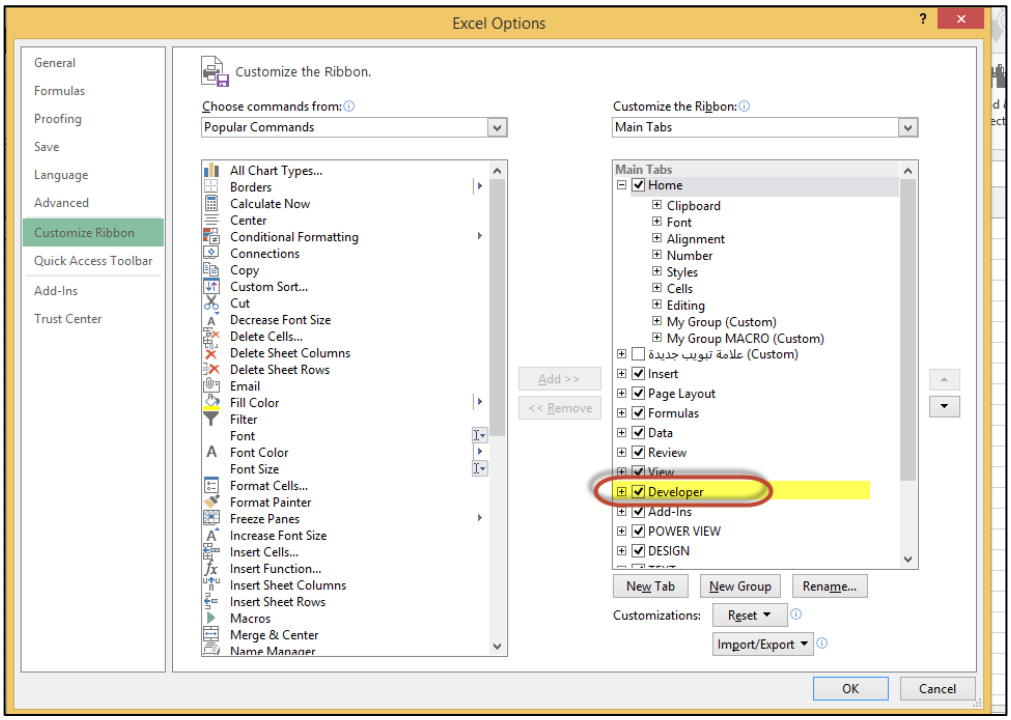


Figure3-1

خطوات انشاء ماكرو بسيط يقوم بتلوين الصف الأول باللون الأصفر

1. نذهب الى شريط المطور ثم نضغط على Record Macro (تسجيل ماكرو)
2. يظهر لنا مربع الحوار Record Macro. نعطي الماكرو اسما واختصارا لو أردنا من خلال البند Shortcut key ونحدد أين سيتم تخزين الماكرو من خلال القائمة المنسدلة Store Macro in.

تحتوي القائمة المنسدلة Store Macro in على ثلاث خيارات هي كالتالي:

- This workbook (هذا المصنف) حيث يتم تخزين هذا الماكرو في هذا المصنف فقط ويجب في هذه الحالة تخزين المصنف بامتداد xlsm

- New Workbook (مصنف جديد) حيث سيتم فتح مصنف جديد وتسجيل الماكرو بداخله.

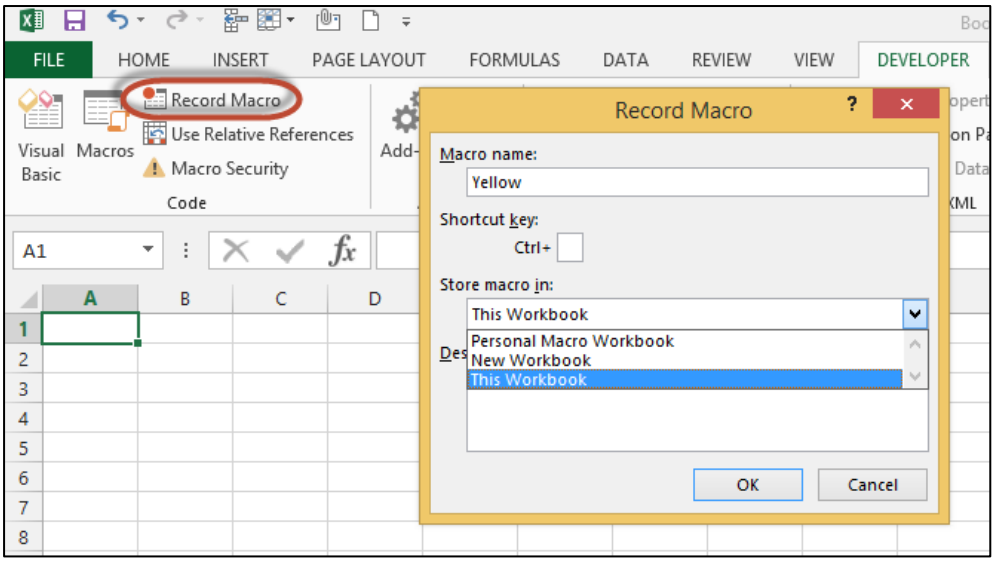


Figure3-2

3. Personal Workbook (مصنف الماكرو الشخصي). هذا المصنف موجود بالوضع الافتراضي ولكنه غير مفعل، وعند تفعيله أول مرة (عن طرق تخزين ماكرو فيه) يتم تشغيله بعدها في كل مرة يتم فتح الاكسيل بها ولكنه يكون مخفياً. اذا اردت أن يكون الماكرو متاحاً في كل المصنفات على نفس جهاز الحاسوب يجب أن يخزن في مجلد الماكرو الشخصي
4. بمجرد أن تضغط موافق يبدأ الماكرو بالتسجيل، نبدأ بإجراء العمليات التي نريد للماكرو أن يسجلها فنذهب للصف الأول ونعطيه اللون الأصفر.
5. نضغط الآن على زر إيقاف التسجيل Stop the Recording الموجود في شريط المطور أو الموجود في شريط الحالة/المعلومات status bar.

لاستدعاء الماكرو الذي تم تسجيله نستخدم اختصار الكيبورد الخاص بذلك الماكرو أو نذهب الى شريط المطور ثم نضغط زر Macros ونختار الماكرو المطلوب.

كيفية وضع زر للماكرو في شريط المهام

1. *نضغط بالزر الأيمن* للماوس فوق شريط الأدوات Ribbon ثم نختار Customize the ribbon (تخصيص الشريط)
2. ***ننشئ*** علامة تبويب جديدة New Tab أو مجموعة جديدة New Group
3. نضيف الماكرو المطلوب لهذا التبويب أو المجموعة.

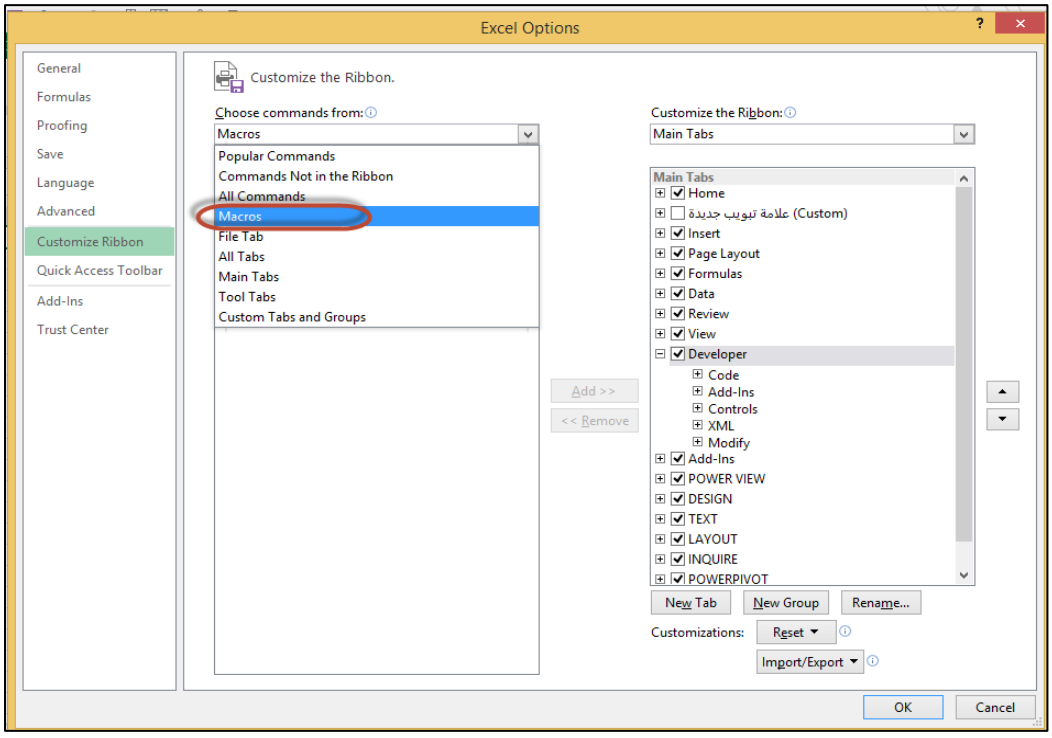


Figure3-3

كيفية ربط ماكرو مع زر Button أو شكل

1. ندرج الزر من خلال شريط المطور كما بالشكل 3-4 أو ندرج الشكل المطلوب من تبويب ادراج Insert كما في الشكل 3-5.
2. نضغط بالزر الأيمن على الزر أو الشكل ثم نختر Assign Macro (تعيين ماكرو) ونعيين الماكرو المطلوب.

Figure3-4

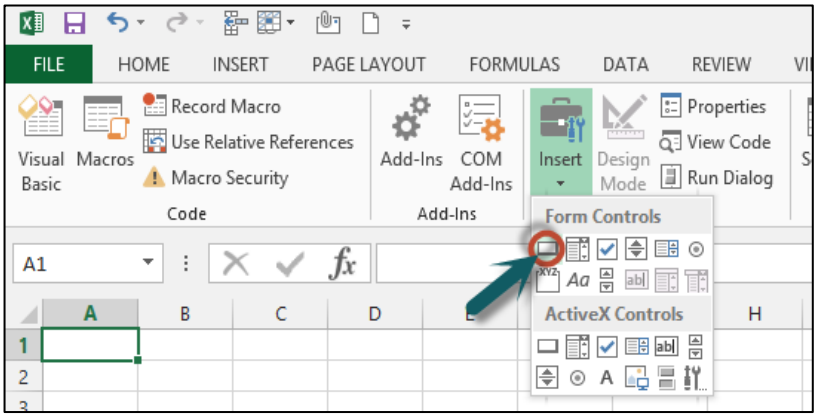
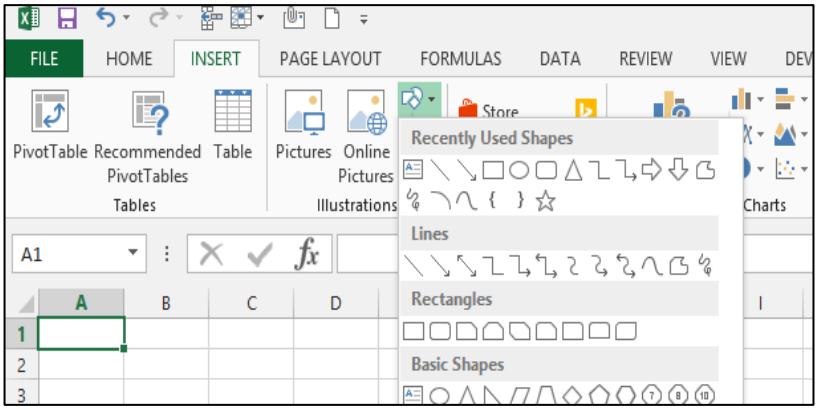


Figure3-5



استخدام مراجع الخلايا النسبية عند تسجيل الماكرو

الماكرو الذي أنشأناه في المثال السابق يستخدم مراجع الخلايا المطلقة (وهو الوضع الافتراضي) وهذا يعني أنك لو كنت في أي مكان داخل ورقة العمل ونفذت الماكرو فسوف يتم تنفيذه على نفس الخلايا التي نُفذ عليها وقت التسجيل. بالإمكان استخدام المراجع النسبية حتى يتسنى لنا تنفيذ الماكرو على الخلايا المحددة فقط وهذا الأمر يتم عن طريق الضغط على Use Relative Reference (استخدام المراجع النسبية) قبل البدء بتسجيل الماكرو.

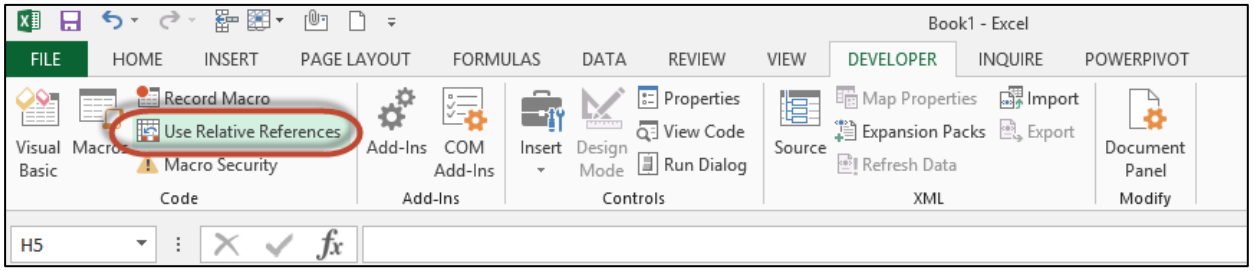


Figure3-6

فحص الكود البرمجي الموافق للماكرو

لرؤية الكود البرمجي الخاص بماكرو مُسجَل اتبع الخطوات التالية:

1- افتح محرر الأكواد VBE عن طريق الضغط على ALT + F11 أو من خلال الذهاب إلى تبويب Developer ثم Visual Basic.
2- اذهب إلى مستكشف المشاريع Project Explorer. إذا لم يكن ظاهراً فأظهره من خلال الذهاب إلى View ثم Project Explorer.
3- من مستكشف المشاريع اختر اسم المصنف الذي يحتوي على الماكرو واضغط على رمز + الموجود لعمل توسعة Expand للمصنف.
4- اضغط على + بجانب بند Modules لعمل توسعة لها. ثم انقر نقراً مزدوجاً فوق Module1.

5- سوف يتم عرض الكود الخاص بالماكرو في نافذة الكود Code Window الموجودة على يمين مستكشف المشاريع.

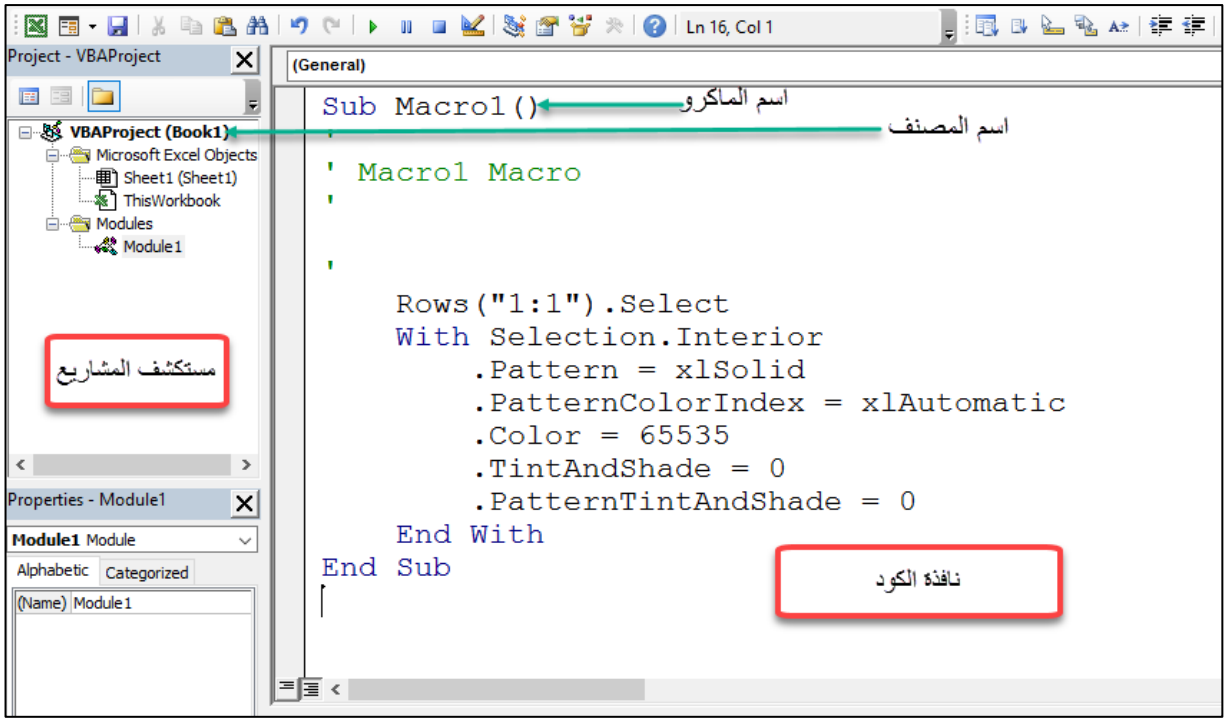


Figure3-7

من خلال عرض الكود لاحظ ما يلي:

1- الكود تم ادراجه من خلال إجرائية فرعية Sub- procedure.
2- اسم الإجرائية الفرعية هو نفس اسم الماكرو الذي تم إعطاؤه له أثناء تسجيل الماكرو.
3- هناك أسطر في الكود عبارة عن ملاحظات وهذه الأسطر ليس لها تأثير على الكود انما يتم ادراجها لتوضيح بعض الأمور التي يرغب المبرمج بإظهارها. هذه الملاحظات تبدأ بالرمز (') وتكون عادةً باللون الأخضر. كمثال على ذلك السطر التالي:

```
' Macro1 Macro
```

4- يمكنك التعديل في الكود بحيث تتغير نتيجة تنفيذه. فمثلاً خذ عندك السطر البرمجي التالي:

```
Rows("1:1").Select
```

في هذا السطر تم تحديد الصف الأول تمهيداً لتطبيق التنسيق عليه. فإذا أردت أن تغير من عمل الماكرو بحيث يقوم بتلوين الأسطر الثلاثة الأولى باللون الأصفر، فتستطيع التغيير في السطر السابق ليصبح كما يلي:

```
Rows("1:3").Select
```

حتى هذه النقطة لا تركز كثيراً على كيفية تعديل الكود الناتج عن الماكرو المُسجل، وانما ركز على فهم الفكرة العامة. وبعد انتهائك من قراءة فصول الكتاب والتعرف على أجزاء ومكونات لغة VBA وكيفية التعامل معها سوف يصبح من السهل لديك أن تقوم بقراءة وفهم الكود المسجل والتعديل فيه إن لزم.

الفصل الرابع

4. المتغيرات Variables

في هذا الفصل سوف نقوم بشرح مفهوم المتغيرات Variables وأنواعها وكيفية الإعلان عنها ونطاقها وكيفية تعيين قيم لتلك المتغيرات. كما وتم التطرق فيه للمصفوفات Arrays بأنواعها المختلفة وكيفية الإعلان عنها والتعامل معها.

ما هو المتغير؟

المتغير Variable هو عبارة عن اسم مُعطى لمكان في ذاكرة الحاسوب حيث يتم تخزين قيمة ذلك المتغير في هذا المكان، مما يسمح لك بالإشارة إلى ذلك المكان واستخدام البيانات بداخله عند الحاجة.

يمكن ان تحتوي المتغيرات على أنواع مختلفة من البيانات (سنأتي على ذكر ذلك لاحقاً). ويمكن تعيين/اعطاء قيمة للمتغير باستخدام معامل التعيين (إشارة "=").

فيما يلي بعض الأمثلة على بعض العبارات التي استخدمت في إعطاء قيمة لمتغير:

```
X = 100

Rate = 0. 05

y = y+10

name = "Ahmed"
```

```
isOK = False

StartDate = #01/18/2019#

Rate = Sheets("sheet1").Range("A1").Value
```

لاحظ أن أسماء المتغيرات تقع على الجانب الأيسر من علامة التعيين (=)، بينما تقع القيمة التي تم تعيينها على الجانب الأيمن.

تحتوي لغة VBA على العديد من الكلمات المحجوزة التي لا يمكن استخدامها كأسماء للمتغيرات، مثل Sub, Function, IF ... etc واذا حاولت استخدام إحدى هاته الكلمات فسوف تحصل على رسالة خطأ.

لمعرفة الكلمات المحجوزة في VBA انظر الرابطين التاليين:

https://bettersolutions.com/vba/syntax/keywords.htm

https://docs.microsoft.com/en-us/office/vba/language/reference/keywords-visual-basic-for-applications

أنواع المتغيرات في VBA

يدل نوع المتغير عل كيفية تخزينه في الذاكرة، مثل الأرقام الصحيحة والعشرية، النصوص، التواريخ وغيرها. بالوضع الافتراضي، تستطيع لغة VBA التعامل مع الأنواع المختلفة للمتغيرات بشكل تلقائي، فليس ضروريا أن تعلن عن المتغير (اسمه ونوعه) قبل البدء باستخدامه فيمكنك مثلا أن تكتب الجملة البرمجية التالية:

```
x = 100
```

هنا يقوم الاكسيل بشكل تلقائي بالتعرف على المتغير x وتحديد نوعه كعدد صحيح integer وذلك طبقاً للقيمة التي تم تعيينها للمتغير وهي 100 ولو كانت القيمة المعطاة للمتغير هي "Ahmed" فسوف يقوم الاكسيل بتحديد نوع البيانات للمتغير x كـ String. وعلى الرغم من المرونة التي تمنحها لنا VBA في

هذا الأمر إلا أن هناك ثمناً لذلك وهو استخداماً أقل كفاءةً لموارد الكمبيوتر وتنفيذاً قد يكون في بعض الأحيان أبطأ مما لو تم تحديد نوع البيانات المناسب للمتغير قبلاً. بشكل عام، يُوصى دوماً باستخدام نوع البيانات الذي يستخدم أقل عدد من البايتات.

أنواع البيانات في VBA

في الجدول التالي قائمة بأنواع المتغيرات في لغة VBA

نوع البيانات Data Types	نطاق القيم Range
Byte	قيمة عددية صحيحة من 0 وحتى 255
Boolean	True or False, On or Off, 0 or 1
Integer	-32,768 To 32767
Long	–2,147,483,648 To 2,147,483,647
Single	-3.402823E38 To 1.401298E45
Double (Positive)	4.94065645841247E-324 To 1.79769313486232E308
Double (Negative)	-1.79769313486232E308 To -4.94065645841247E-324
Currency	-922,337,203,685,477.5808 To 922,337,203,685,477.5807
Decimal	+/-79,228,162,514,264,337,593,543,950,335 بدون الفاصلة العشرية +/-7.9228162514264337593543950335 مع الفاصلة العشرية
Date	1/1/100 to 12/31/9999
Object	Object in VBA
String	حتى 65400 حرف
Variant	أي قيمة رقمية بنطاق الـ Double أو نصية بنطاق String

الإعلان عن المتغيرات Declaring variables

إذا لم تقم بالإعلان عن نوع المتغير فسيعطيه VBA النوع الافتراضي للمتغيرات وهو Variant. البيانات التي تخزن كـ Variant تستطيع أن تستوعب أي نوع من البيانات حيث أن النوع يتغير طبقاً للبيانات المُعينة للمتغير ولِما تريد أن تفعله بهذا المتغير. فمثلاً إذا كان لدينا متغيراً من نوع Variant وتم تعيين قيمة نصية لذلك المتغير تشابه القيم العددية (كأن تعين قيمة المتغير بـ "100200300") فإمكانك أن تقوم بإجراء العمليات النصية text manipulation (مثل أن تقوم بإعادة أول 3 رموز من النص) أو العمليات الحسابية على هذا المتغير. هنا يقوم VBA بالتحويل التلقائي لنوع البيانات حسب العملية التي طُبقت على المتغير.

انظر المثال التالي:

```
Sub test()
  var1 = "100200300"
  MsgBox (Left(var1, 3))
  MsgBox (var1 + 300)
End Sub
```

هنا استخدمنا متغيراً اسمه Var1 وأعطيناه القيمة "100200300" والآن عند تطبيق عملية نصية مثل إيجاد أول 3 رموز في السلسة من خلال الدالة Left "Left(var1, 3) " ، يتعامل الـ VBA مع البيانات كنص String. بينما عند تطبيق عملية حسابية على القيمة "Var1+300 " يتعامل الـ VBA مع البيانات كرقم. شكل 1-4 يظهر لنا نتيجة تنفيذ العمليتين

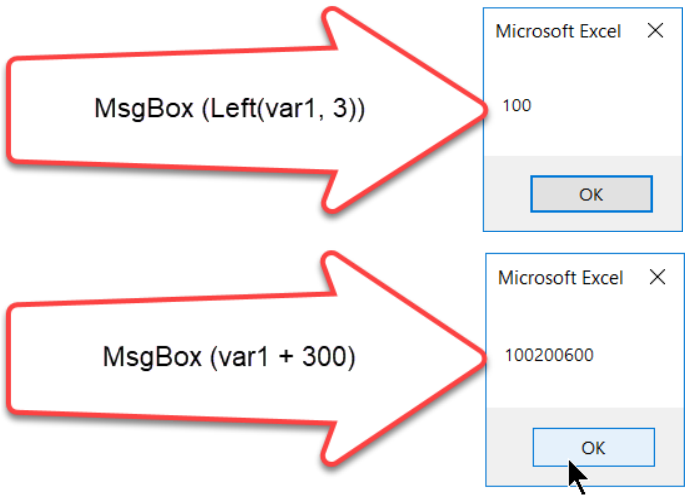


Figure4-1

ترك الـ VBA يقوم بالتعرف التلقائي على نوع البيانات قد تبدو عملية مريحة إلا أن ذلك يأتي على حساب السرعة والمساحة المُستغلة من الذاكرة حيث أن الـ VBA تقوم بحجز مساحة بالذاكرة قد تكون أكبر بكثير من المساحة المطلوبة بالإضافة أنه في حالة اجراء أي عملية على متغير من نوع Variant يقوم الـ VBA بإجراء سلسلة من عمليات التحقق حتى يختار نوع البيانات المناسب وهذا قد يسبب بطئاً في تنفيذ البرنامج.

هناك مشكلة أخرى قد تواجهك إذا لم تقم بالإعلان عن المتغيرات وهي أن الكود الخاص بك يصبح أكثر عرضة للأخطاء، خذ عندك المثال التالي: لنفترض أنك قمت باستخدام متغيراً غير معلنٍ عنه باسم CurrentValue فأعطيته القيمة 250 من خلال السطر البرمجي التالي:

```
CurrentValue = 250
```

ثم أردت أن يقوم البرنامج بإضافة 100 على قيمة CurrentValue فأدخل السطر البرمجي:

```
CurentValue = 250 + 100
```

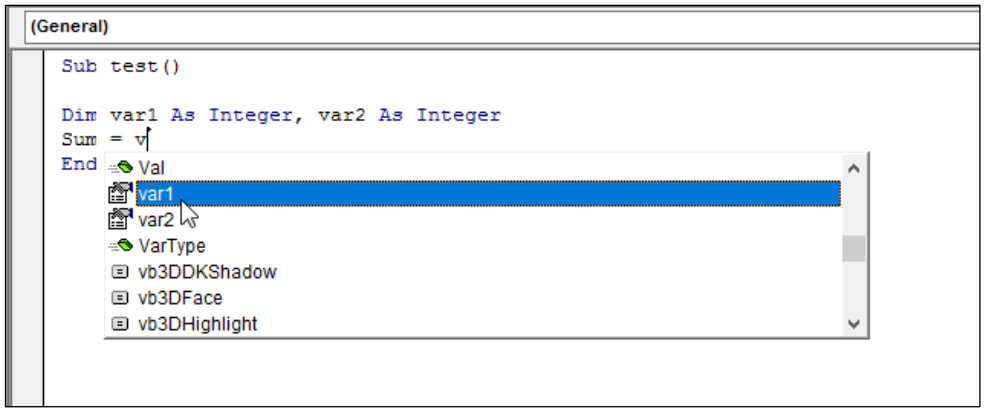
هل لاحظت الخطأ؟ في السطر البرمجي الثاني هناك حرف "r" ناقص في كلمة CurrentValue وعليه فالنسبة للـ VBA المتغير CurentValue هو متغير جديد ويختلف تماماً عن المتغير CurrentValue وبذلك فإن النتيجة النهائية لتنفيذ البرنامج ستكون خاطئة تماماً. إنه من الصعوبة بمكان

أن تكتشف هذا النوع من الأخطاء بالذات عندما يكون البرنامج كبيراً ويحتوي على عدة عشرات أو مئات من الأسطر البرمجية.

بناءً على ما ذكر سابقاً فإن الإعلان عن المتغيرات هي من الأمور المُوصى بها بشدة.

أحد الميزات التي توفرها لك عملية الإعلان عن المتغيرات هي عملية الاكمال التلقائي لأسماء المتغيرات وقت الادخال فإذا كنت تريد أن تدخل اسم أحد المتغيرات المعلن عنها فيكفي أن تكتب أول حرف أو حرفين من المتغير ثم تضغط على الاختصار Ctrl+Space ليقوم الـ VBA بإكمال الادخال أو عرض مجموعة محددة من الخيارات ليتم الاختيار من بينها.

Figure4-2



للإعلان عن المتغيرات نستخدم (غالباً) الكلمة المفتاحية Dim بالشكل التالي:

Dim <اسم المتغير> As <نوع المتغير>

أمثلة:

```
Dim Name As String

Dim Salary As Double

Dim StartDate As Date

Dim X
```

لاحظ هنا أنه في جملة الإعلان الأخيرة لم يتم إعطاء المتغير X نوع بيانات محدد ولذلك فسوف يتعامل معه VBA كمتغير من نوع Variant.

من الممكن الإعلان عن أكثر من متغير في نفس السطر كما في المثال التالي:

```
Dim Name As String, Salary As Long, StartDate As Date
```

وإذا كان لدينا أكثر من متغير من نفس النوع نستخدم نفس الطريقة كما في المثال التالي:

```
Dim FirstName As String, LasteName As String
```

مع ملاحظة أن الصيغة التالية لا يمكن استخدامها لتحديد نوع البيانات لأكثر من متغير من نفس النوع:

```
Dim FirstName, LastName As String
```

إذا استخدمت الجملة السابقة فإن المتغير الأخير فقط هو الذي سيتم تعريفه كـ String بينما سيتم تعريف باقي المتغيرات كـ Variant.

بجانب الكلمة المفتاحية Dim هناك ثلاثة كلمات أخرى يمكن استخدامها لتعريف المتغيرات هي:

- Static
- Public
- Private

وسيتم التطرق لتلك الأنواع لاحقاً في الفقرات التالية.

نطاق المتغيرات

نطاق المتغير Variable's scope يحدد الوحدات البرمجية والاجرائيات التي يمكن أن تستخدم المتغير. هناك ثلاثة أنواع لنطاق المتغير:

1- Procedure only نطاق الإجرائية فقط (سواء أكانت الإجرائية عبارة عن دالة Function أو إجرائية فرعية Sub Procedure). وفي هذه الحالة يتم تعريف النطاق للمتغير عن طريق استخدام جملة Dim أو Static بداخل الإجرائية
2- Module only نطاق الوحدة البرمجية فقط.
3- All procedures in all modules كل الاجرائيات في جميع الوحدات البرمجية.

في الفقرات التالية سنتعرض بالتفصيل لهذه الأنواع الثلاثة.

Procedure only variables المتغيرات في نطاق الإجرائية فقط

يعد هذا النطاق هو المستوى الأدنى لمتغير ما حيث ان المتغير يكون معرفاً بداخل الإجرائية ويمكن استخدامه بداخل تلك الإجرائية فقط. عندما ينتهي تنفيذ الاجرائية فإن المتغير لا يعود له وجود ويقوم الاكسيل بإخلاء مكانه في الذاكرة.

عند تنفيذ الإجرائية مرة أخرى يعود المتغير مرة أخرى ولكنه يفقد أي قيمة سابقة اكتسبها من خلال التنفيذ السابق للإجرائية.

أشهر طريقة للإعلان عن المتغير في نطاق الإجرائية فقط هو من خلال استخدام الكلمة المفتاحية Dim وذلك بالإعلان عنه بعد جملة Sub أو Function مباشرة وقبل كتابة أسطر الكود الخاصة بالإجرائية. انظر المثال التالي:

```
Sub Test( )
    Dim FirstName As Text, LastName As Text
    ……
    [أسطر الكود البرمجي]
End Sub
```

إذا أعلنت عن متغير في نطاق الإجرائية فقط فإن الاجرائيات الأخرى بداخل نفس الوحدة البرمجية تستطيع استخدام نفس اسم المتغير بداخلها ولكن يتم التعامل مع كل متغير بشكل منفصل تماماً ولا توجد علاقة أبداً بين تلك المتغيرات. يمكن القول بأن المتغيرات ذات نطاق "الإجرائية فقط" هي الأكثر فعالية في استخدام الذاكرة حيث أن المتغير يُخلى مكانه في الذاكرة فوراً بعد تنفيذ الإجرائية.

Module only variables المتغيرات في نطاق الوحدة البرمجية فقط.

إذا كنت تريد أن يكون المتغير متاحاً لجميع الاجرائيات بداخل الوحدة النمطية فبإمكانك الإعلان عن المتغير قبل أول إجرائية (سواء أكانت Function أوSub). هذا المكان الذي يسبق أي إجرائية يطلق عليه Declarations section "قسم الإعلان عن المتغيرات"

الشكل 3-4 يوضح لنا جزء الإعلان عن المتغيرات declaration. لاحظ هنا أنك بمجرد وضع مؤشر الماوس في جزء الإعلان تتغير القائمة المنسدلة على اليمين إلى Declaration. وبالإمكان استخدام القائمة المنسدلة للانتقال إلى أي جزء من الوحدة النمطية.

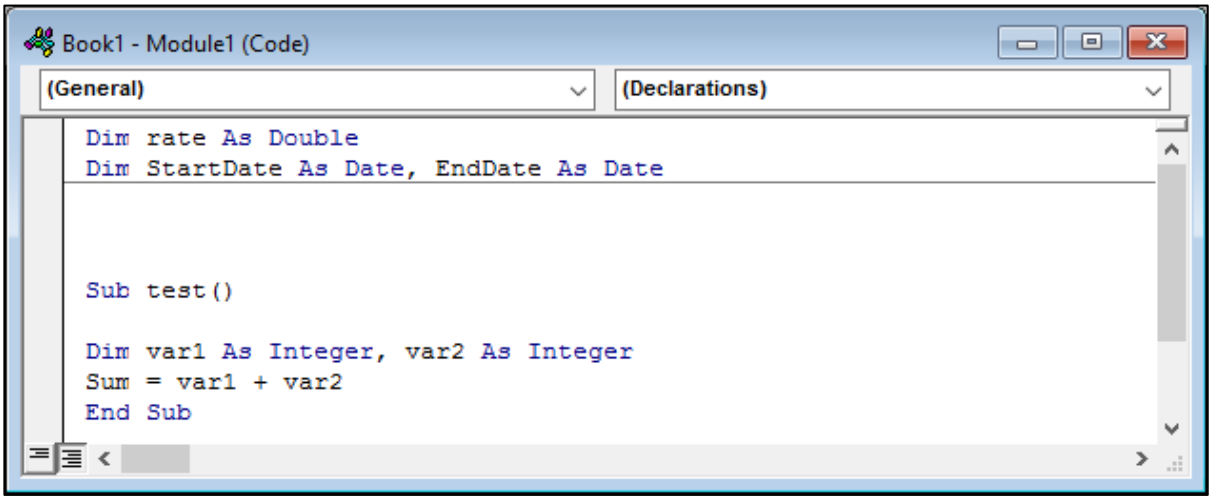


Figure4-3

بالنسبة للمتغيرات في نطاق الوحدة النمطية، يمكن استخدام هذه المتغيرات في أي إجرائية بداخل الوحدة النمطية وتبقى قيمتها كما هي عند الانتقال من إجرائية لأخرى.

المتغيرات العامة Public variables

إذا كنت تريد أن يكون المتغير متاحاً لجميع الاجرائيات في جميع الوحدات النمطية يمكنك أن تعرف المتغير كمتغير عام Public variable باستخدام الكلمة المفتاحية Public وذلك في جزء الاعلان عن المتغيرات. انظر المثال التالي:

```
Public InvestRate as Long
```

الآن يمكنك استخدام المتغير InvestRate في أي دالة أو إجرائية فرعي بداخل أي وحدة نمطية بداخل المصنف.

المتغيرات الثابتة Static Variables

عادةً، يتم عمل إعادة تعيين Reset لجميع المتغيرات في الإجرائية بعد الانتهاء من تنفيذها. المتغيرات الثابتة هي نوع خاص من المتغيرات والتي تحتفظ بقيمتها حتى بعد الانتهاء من تنفيذ الإجرائية ولا يتم عمل إعادة تعيين لها. يتم الإعلان عن المتغيرات الثابتة باستخدام الكلمة المفتاحية Static بداخل الإجرائية.

في المثال التالي استخدمنا متغيراً ثابتاً لعرض عدد المرات التي تم بها تشغيل الإجرائية الفرعية Test()

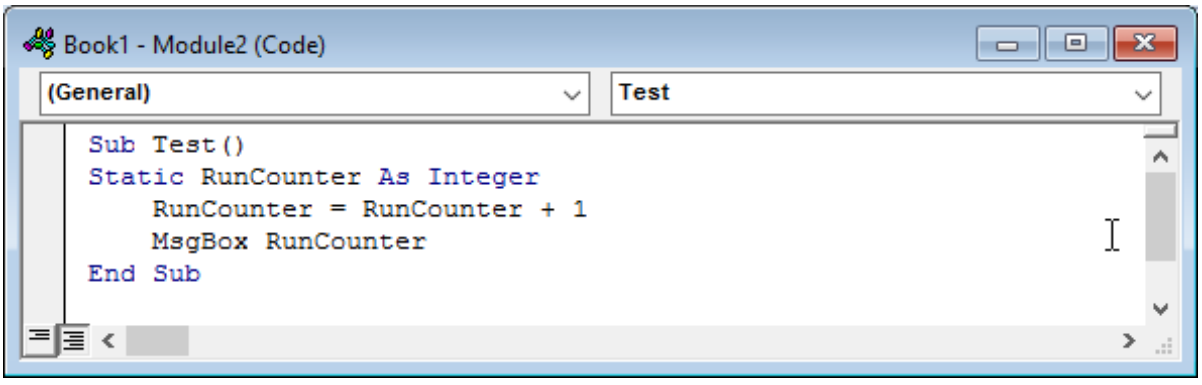


Figure 4-4

هنا تم تعريف المتغير RunCounter كمتغير ثابت من نوع Integer وفي كل مرة يتم تشغيل الإجرائية الفرعية ()Test يتم زيادة قيمة RunCounter بمقدار 1 من خلال السطر البرمجي:

```
RunCounter = RunCounter + 1
```

ويتم عرض النتيجة في مربع حوار من خلال السطر البرمجي:

```
MsgBox RunCounter
```

متى يحدث إعادة تعيين Reset للمتغيرات

نطاق المتغيرات Variables scope لا يحدد أين يمكن استخدام المتغير فقط، بل أيضاً يحدد متى تقوم VBA بحذف قيمة المتغير من الذاكرة. حيث أنه – وكما تم ذكره سابقاً- تقوم الـ VBA بحذف قيمة المتغير الموجود ضمن نطاق الإجرائية Procedure only variable بمجرد الانتهاء من تنفيذ الإجرائية. المتغيرات الأخرى (Static, Public, and Module only variables) تحتفظ بقيمتها لحين اغلاق المصنف أو الاكسيل.

يمكنك أيضاً أن تقوم بحذف قيم المتغيرات من الذاكرة باستخدام احدى الطرق التالية:

- عن طريق الضغط على زر Reset (المربع الأزرق الموجود على شريط الأدوات)
- عن طريق الضغط على زر End عند ظهور رسالة خطأ.
- عن طريق استخدام الكلمة المفتاحية "End" في أي مكان بداخل الكود

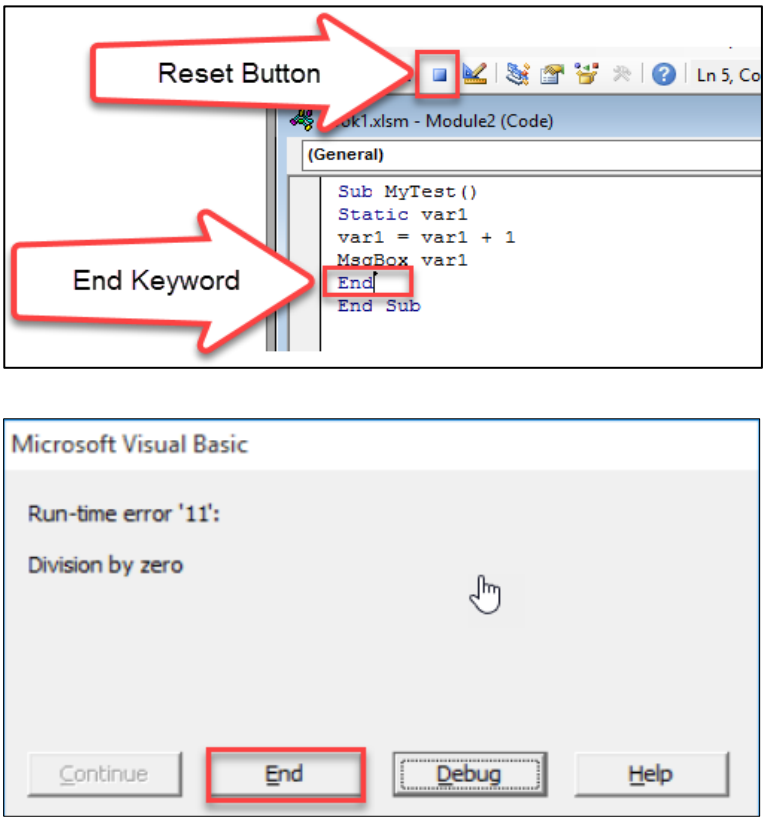


Figure4-5

الثوابت Constants

أحياناً تحتاج إلى الإشارة إلى قيمة ثابتة لا تتغير أبداً مثل أن تشير إلى ثابت الدائرة "ط Pi" والذي يساوي القيمة 3.14 تقريباً. ولهذا الغرض نستخدم الثوابت Constants حيث أن المتغيرات العادية Variable معرضة لأن تتغير قيمتها مع تشغيل البرنامج (لهذا تدعى متغيرات. أليس كذلك؟)

يتم الإعلان عن الثوابت باستخدام الكلمة المفتاحية Const. وتحتوي جملة الإعلان عن المتغير على قيمة المتغير. الجمل التالية هي أمثلة على الثوابت:

```
Const Pi = 3.14
```

```
Const ModuleName As String = "Reporting"

Public Const Rate = 0.075
```

استخدام الثوابت بدلاً من ادراج القيم مباشرة كقيم ثابتة Hard-coded في البرنامج هو من الممارسات الممتازة في البرمجة حيث أن هذا الأمر يسهل قراءة وفهم الكود لغير كاتبه، بالإضافة إلى أن هذا الأمر يسهل من عملية تغيير قيم الثوابت فتتم عملية التغيير مرة واحدة فقط. فعلى سبيل المثال اذا كان الكود يتكرر به الإشارة إلى ثابت محدد (مثل نسبة الفائدة Interest مثلاً) فمن الأفضل أن يتم الإعلان عن ثابت وتعيين قيمة تلك القيمة إليه بحيث أنه إذا تغيرت نسبة الفائدة فسنحتاج لتغيير تلك النسبة مرة واحدة عن طريق تغيير قيمة الثابت.

نطاق الثوابت Constants Scope

بالنسبة لنطاق الثوابت فما ينطبق على المتغيرات بهذا الصدد ينطبق على الثوابت.

ملاحظة/

بخلاف المتغيرات؛ قيمة الثابت لا يمكن تغييرها بعد الإعلان عنها. اذا حاولت تغييرها في أثناء الكود فستحصل على رسالة خطأ.

التعامل مع المتغيرات النصية Working with Strings

تتعامل VBA مع نوعين من السلاسل النصية:

1- **السلاسل النصية ذات الطول الثابت Fixed-length strings:** وهي التي يتم الإعلان عنها مع تحديد الحد الأقصى لعدد الحروف في السلسلة النصية. الحد الأقصى لعدد الحروف لهذا النوع هو 65526 حرف.
2- **السلاسل النصية ذات الطول المتغير Variable-length strings:** إذا لم يتم تحديد عدد الأحرف للمتسلسلة النصية وقت الإعلان عن المتغير، يتم التعامل مع المتسلسلة النصية كذات طول متغير Variable-length string. في هذه الحالة فالحد الأعلى لعدد الحروف في المتسلسلة هو عبارة عن 2 مليار حرف.

عند الإعلان عن متغير من نوع String بالإمكان تحديد عدد الأحرف في المتسلسلة في نفس جملة الإعلان، وإذا لم يتم تحديد عدد الأحرف فسوف يتعامل الـ VBA مع المتغير كـ Variable-length string.

المثالين التاليين يوضحان كيفية الإعلان عن متغير نصي مع أو بدون تحديد عدد الأحرف:

```
Dim MyName As String * 40

Dim MyName As String
```

التعامل مع متغيرات التاريخ Date variables

التاريخ Date من أنواع البيانات المفيدة التي يكثر استخدامها. إذا استخدمت متغيراً نصيا مثلاً لاحتواء قيمة تاريخ فلا يمكنك اجاء العمليات على التاريخ. بينما إذا استخدمت النوع Date فيمكنك أن تجري العمليات المختلفة على التاريخ مثل أن تعرف التاريخ بعد أو قبل عدد معين من الأيام أو معرفة عدد الأيام بعد تاريخين.

الأمثلة التالية تعرض كيفية الإعلان عن متغيرات وثوابت من نوع Date:

```
Dim FirstDate As Date

Const StartDate As Date = #1/1/2020#

Const Noon As Date = #12:00:00 PM#
```

في الـ VBA يجب أن توضع قيم الوقت والتاريخ بين علامتي # كما في الأمثلة السابقة. وهنا يجب أن تعلم شيئاً مهماً؛ الـ VBA تقوم بعرض الوقت والتاريخ كما هو معرف في الخيارات الإقليمية لجهاز الحاسوب. ولكن عند ادخال قيم الوقت والتاريخ يجب أن تلتزم بالنظام الأمريكي وهو "الشهر/اليوم/السنة dd/mm/yyyy" فمثلاً إذا أدخلت السطر البرمجي التالي:

```
Dim MyDate = #11/10/2020#
```

فهذا يعني العاشر من نوفمبر وعند عرض التاريخ فسيقوم الاكسيل بعرضه كما هو معرف في الخيارات الإقليمية (في حال عرضه في مربع حوار) أو كما تم تنسيق الخلايا (في حال عرض التاريخ بداخل خلية).

جمل التعيين Assignment Statements

جمل التعيين هي التي يتم من خلالها إعطاء القيم للمتغيرات أو الكائنات (مثل الخلايا) وهي تستخدم عادة المعامل "=".

في الأسفل بعض الأمثلة على جمل التعيين:

```
x = 100

Z = Z + 1

y = (y + x) / 20

Rate = Range("A1").Value
```

كما لاحظت من خلال الأمثلة السابقة يمكن استخدام جمل التعيين لتعيين قيمة محددة للمتغير أو تعيين نتيجة عملية ما للمتغير كما ويمكن إعطاء المتغير قيمة أحد الكائنات كأن تعطي المتغير القيمة الموجودة في الخلية A1.

من الجمل السابقة هناك جملة قد تبدو غريبة، إلا أنها من التعابير التي يكثر استخدامها في عالم البرمجة وهي:

```
Z = Z + 1
```

ومفهوم هذه الجملة هو أنه في كل مرة يتم تنفيذ هذه الجملة يتم زيادة قيمة المتغير بمقدار 1.

المصفوفات Arrays

تدعم جميع لغات البرمجة الحديثة المصفوفات ومن بينها لغة VBA منذ إصدارها الأول. المصفوفة هي مجموعة من المتغيرات التي تتشارك في نفس الاسم. تتم الإشارة إلى عنصر محدد بداخل المصفوفة من خلال استخدام اسم المصفوفة مع رقم المرجع Index number للمتغير بداخل قوسين. على سبيل المثال بإمكانك الإعلان عن مصفوفة من المتغيرات الرقمية من نوع Integer تحتوي على 10 عناصر تحت اسم MyNumbers ومن ثم يمكنك الإشارة للعنصر الأول في المصفوفة كالتالي: MyNumers(1) والعنصر الثاني: MyNumers(2) وهكذا.

الإعلان عن المصفوفات Declaring Arrays

قبل أن تشرع باستخدام مصفوفة ما يجب أن تقوم بالإعلان عنها – لا يوجد هنا أي استثناءات – بخلاف المتغيرات العادية والتي يمكن استخدامها بداخل كود الـ VBA بدون الإعلان عنها. يمكنك الإعلان عن المصفوفات باستخدام الكلمة المفتاحية Dim أو Public حسب النطاق الذي تريده لهذه المصفوفة. يجب تحديد عدد عناصر المصفوفة عند الإعلان عنها من خلال ادخال الرقم المرجعي الأول First index number ثم الكلمة المفتاحية To ثم الرقم المرجعي الأخير Last Index number. المثال التالي يوضح لنا كيفية الإعلان عن مصفوفة من الأرقام Integer تحتوي على 10 عناصر:

```
Dim MyNumbers (1 To 10) As Integer
```

عند الإعلان عن مصفوفة فلك الخيار في أن تحدد الرقم المرجعي Index الأدنى والأقصى أو أن تحدد الرقم المرجعي الأقصى فقط. في حال لم تحدد الرقم المرجعي الأدنى فإن الـ VBA يعتبره 0 وليس 1. الجملتين التاليتين لهما نفس التأثير:

```
Dim MyNumbers (0 To 10) As Integer

Dim MyNumbers (10) As Integer
```

إذا كنت تريد للـ VBA أن يفترض أن الرقم المرجعي الأدنى هو 1 وليس 0 فبإمكانك ادخال الجملة التالية في جزء الإعلان Decelerations section :

```
Option Base 1
```

المصفوفات متعددة الأبعاد Multidimensional Arrays

المصفوفات التي تم التعرض لها في الفقرات السابقة هي عبارة عن مصفوفات وحيدة البعد One-dimensional يمكن تخيل المصفوفة ذات البعد الواحد كعمود او صف من القيم. من الممكن أن تتعامل الـ VBA مع مصفوفات ذات أبعاد حتى 60 بعد على الرغم من أنه من النادر جداً أن يضطر أحد أن يتعامل مع مصفوفة ذات أبعاد أكثر من اثنين أو ثلاثة.

المثال التالي يعرض لنا مصفوفة ذات بعدين وتسعة عناصر:

```
Dim MyArray( 1 To 3, 1 To 3) As Integer
```

وللإشارة لعنصر محدد بداخل هذه المصفوفة يجب أن تحدد الرقم المرجعي Index number لكل بعد كما في التالي:

```
MyArray (1,  1 ) = 100
```

هنا عينا القيمة 100 للعنصر الواقع في الموقع 1 و 1.

في الواقع يمكن تخيل المصفوفة ثنائية البعد كمربع أو مستطيل ذات صفوف وأعمدة أما المصفوفة الثلاثية فيمكن تخيلها كمتوازي مستطيلات له طول وعرض وارتفاع.

المصفوفات الديناميكية Dynamic Arrays

بإمكانك ايضاً انشاء مصفوفات ديناميكية (ذات عدد عناصر متغير). للإعلان عن هذا النوع من المصفوفات نستخدم أقواساً فارغة بعد اسم المصفوفة كما في المثال التالي:

```
Dim MyArray () As Integer
```

وقبل أن تبدأ باستخدام هذه المصفوفة يجب أن تستخدم الكلمة المفتاحية ReDim لإخبار الـ VBA عن عدد العناصر بداخل المصفوفة. عادة يتم تحديد عدد العناصر في المجموعة في أثناء تشغيل البرنامج حيث أنه بالإمكان استخدام جملة ReDim بالعدد الذي تريده. المثال التالي يعرض لنا كيفية استخدام ReDim لتغيير عدد العناصر في المصفوفة:

```
ReDim MyArray ( 1 To ElementsNo)
```

على افتراض أن المتغير ElementsNo يحتوي على عدد العناصر المطلوب.

الفصل الخامس

5. التعامل مع كائنات النطاق

كائن النطاق يمثل نطاقاً بداخل كائن ورقة العمل Worksheet object. هذا الكائن (النطاق) له مجموعة من الخصائص Properties التي يمكن عرضها وفي بعض الأحيان تغييرها، ومجموعة من الدوال Methods التي يمكن تطبيقها عليه. كائن النطاق من الممكن أن يحتوي على خلية واحدة ومن الممكن أن يحتوي على جميع خلايا ورقة العمل.

عند الإشارة إلى كائن نطاق يجب أن يتم إحاطة العنوان بعلامتي الاقتباس Double quotes فإذا أردت أن تشير إلى النطاق A1:B10 فمن الممكن أن تستخدم الجملة التالية:

```
Range ("A1:B10")
```

وإذا كان النطاق يحتوي على خلية واحدة فقط فيجب أيضاً أن توضع بداخل علامتي اقتباس أيضاً فللإشارة للخلية A1 نستخدم الجملة التالية:

```
Range  ("A1")
```

وإذا كان للنطاق اسم Named range فيمكن استخدام اسم النطاق بين علامتي اقتباس أيضاً مثل:

```
Range ("MyRange")
```

يمكن الإشارة إلى نطاق خارج ورقة العمل النشطة Active worksheet عن طريق تضمين اسم ورقة العمل في الجملة التي تصف النطاق كما في المثال التالي:

```
Worksheet ( "Sheet1") . Range ("A1:B10")
```

كما يمكنك أن تشير إلى نطاق بداخل مصنف آخر من خلال تضمين اسم المصنف كما في المثال التالي:

```
Workbook("Book1.xlsx").Worksheet("Sheet1").Range ("A1:B10")
```

يمكن لكائن النطاق أن يتكون من صف أو عمود كامل (أو مجموعة من الصفوف أو الأعمدة)، فمثلاً يمكنك من خلال الجملة التالية الإشارة إلى الصف الأول في ورقة العمل النشطة:

```
Range("1:1")
```

كما يمكنك من خلال الجملة التالية الإشارة إلى الأعمدة الثلاثة الأولى في ورقة العمل النشطة:

```
Range("A:C")
```

وللإشارة إلى نطاقات متباعدة استخدم الفاصلة بين النطاقات كما في المثال التالي:

```
Rang ("A1:B10, D1:E10")
```

الإشارة لكائنات النطاق من خلال الخاصية Cells

من الممكن الإشارة إلى النطاقات باستخدام الخاصية Cells حيث أن هذه الخاصية تأخذ معاملين هما رقم الصف ورقم العمود. فمثلاً إذا أردنا الإشارة إلى الخلية B1 وهي الواقعة في الصف الأول والعمود الثاني نستخدم الجملة التالية:

```
Cells(2, 1)
```

حيث أن المعامل الأول يمثل رقم الصف والمعامل الثاني يمثل رقم العمود.

لإعطاء قيمة محددة للخلية A1 (100 مثلاً) يمكن استخدام أي من الجملتين التاليتين:

```
Range ("A1").Value = 100

Cells(1,1).Value = 100
```

أما إذا أردنا إعطاء القيمة 100 للخلايا في النطاق A1:B10 فيمكن استخدام أي من الجملتين التاليتين:

```
Range ( "A1:B10").Value = 100

Range(Cells (1,1) , Cells( 2 ,10)) . Value = 100
```

تظهر قوة الخاصية Cells عند استخدام المتغيرات كعوامل لها بدلاً من الأرقام مما يعطيها مرونة كبيرة.

خاصية الإزاحة Offset

من خلال هذه الخاصية بإمكانك أن تشير إلى نطاق يبعد عن النطاق الحالي بمقدار عدد محدد من الصفوف أو الأعمدة. وتأخذ هذه الخاصية معاملين المعامل الأول هو عبارة عن عدد الصفوف والثاني عدد الأعمدة (بالنسبة لعدد الصفوف، الرقم الموجب هو للسير من اليسار لليمين أما السالب فللسير من اليمين لليسار، أما بالنسبة لعدد الأعمدة، فالرقم الموجب للسير من الأعلى للأسفل والسالب من الأسفل للأعلى). فمثلاً للإشارة إلى الخلية التي تبعد بمقدار صفين وثلاثة أعمدة عن الخلية A1 (وهي الخلية D3) نستخدم الجملة التالية:

```
Range ("A1").Offset (2, 3)
```

خاصية القيمة Value

تمثل الخاصية Value القيمة بداخل خلية ما. وهي من الخصائص التي يمكن قراءتها وتغييرها -Read write. ولذلك فمن الممكن لكود VBA أن يقرا القيمة فقط بدون تغييرها كما في السطر البرمجي التالي والذي يقوم بعرض قيمة الخلية A1 في مربع حوار Message Box:

```
MsgBox Range("A1").Value
```

ومن الجدير بالذكر أنه يمكن قراءة قيمة خلية واحدة فقط ولذلك فالجملة التالية لا تعمل وتعرض خطأ عند تنفيذ البرنامج:

```
MsgBox Range("A1:B10").Value
```

ملاحظة

إذا كنت تريد أن تقوم بقراءة محتويات مجموعة من الخلايا فيمكنك أن تعرف متغير كـ Variant ومن ثم تضع القيم بداخل هذا المتغير. وذلك لأن المتغير من نوع Variant من الممكن أن يعمل كمصفوفة

Array. ومن ثم يمكنك كما في أي مصفوفة أن تقوم بعرض قيمة الخلية المطلوبة من خلال تحديد موقعها في المصفوفة. في المثال التالي قمنا بتعريف متغير كـ Variant وأعطيناه قيم النطاق A1:B10 ومن ثم قمنا بعرض قيمة الخلية B2:

```
Dim x As Variant
X = Range("A1:B10") . Value
Msgbox x(2 ,2)
```

ومن الممكن أن نقوم بتغيير قيمة خلية ما من خلال هذه خاصية Value كما في المثال التالي:

```
Range ("A1") . Value = 150
```

وبعكس عملية قراء القيم من مجموعة من الخلايا التي لا تجوز، فإنه بالإمكان استخدام الخاصية Value لإعطاء قيمة محددة لمجموعة من الخلايا دفعة واحدة كما في المثال التالي:

```
Range("A1: B10").Value = 150
```

ملاحظة////

خاصية القيمة Value هي الخاصية الافتراضية لكائن النطاق Range object ولذلك يمكن استخدام أي من الجملتين التاليتين لإعطاء قيمة للخلية A1:

```
Range ("A1") . Value = 150
Range ("A1") = 150
```

خاصية Text

خاصية Text ترجع القيمة التي تعرضها الخلية كنص. فمثلاً إذا تم تطبيق التنسيق الرقمي عملة على الخلية B2 كما في الشكل 1-5 فإن الجملة:

```
MsgBox Range("B2").Value
```

ترجع قيمة الخلية الرقمية وهي 160. بينما ترجع الجملة التالية القيمة كما تظهر في الخلية:

```
MsgBox Range("B2").Text
```

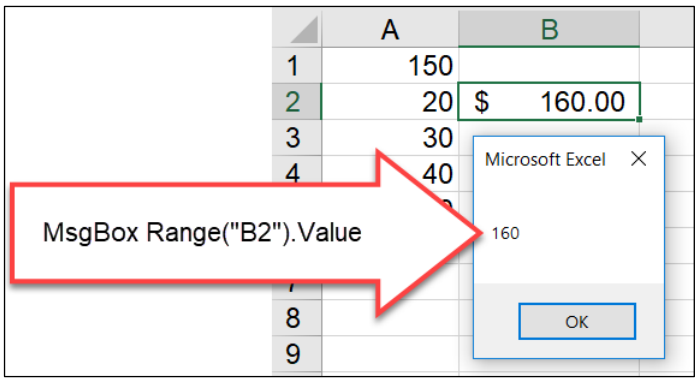


Figure 5-1

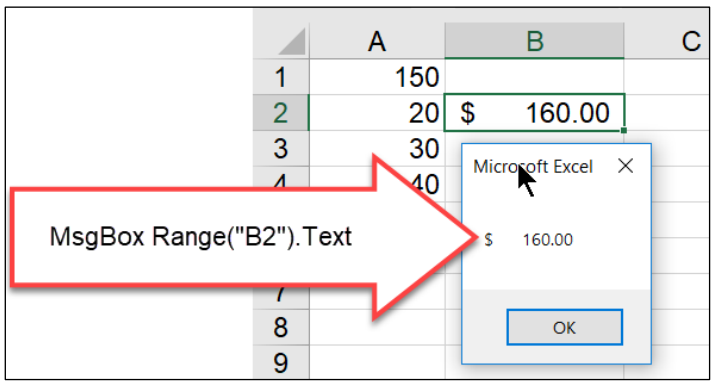


Figure 5-2

إذا كانت الخلية تحتوي على معادلة فإن خاصية Text تعيد نتيجة تنفيذ المعادلة، أما اذا كانت تحتوي على نص فإن خاصية Text وخاصية Value ترجعان نفس النتيجة حيث أن النص بخلاف الأرقام لا يمكن تنسيقه ليظهر بشكل مختلف.

خاصية العد Count

الخاصية Count ترجع عدد الخلايا في نطاق محدد بغض النظر عما إذا كانت الخلايا تحتوي على بيانات أم لا. هذه الخاصية هي للقراءة فقط Read-only ولا يمكن تعديل قيمتها.

السطر البرمجي التالي يعرض عدد الخلايا في النطاق A1:B10 في مربع حوار:

```
MsgBox Range("A1:B10").Count
```

خاصية HasFormula

هذه الخاصية هي للقراءة فقط وهي ترجع القيمة True إذا كانت الخلية تحتوي على معادلة وFalse بخلاف ذلك. إذا كان النطاق يحتوي على أكثر من خلية فإن هذه الخاصية ترجع True في حال أن جميع خلايا النطاق تحتوي على معادلات و False إذا لم يكن هناك أي خلية تحتوي على معادلة، أما اذا كانت بعض خلايا النطاق تحتوي على معادلات و البعض الآخر لا يحتوي فإن هذه الخاصية ترجع القيمة Null.

السطر البرمجي التالي يعرض القيمة True إذا كانت الخلية A1 تحتوي على معادلة و False فيما عدا ذلك:

```
MsgBox Range("A1").HasFormual
```

يجب أن تكون شديد الحذر عند التعامل مع القيمة Null حيث أن المتغير الوحيد الذي يستطيع التعامل مع هذا النوع من القيم هو Variant.

خاصية Font

من خلال هذه الخاصية يمكن التحكم بالخط بداخل نطاق من الخلايا، مثل اختيار نوع وحجم الخط، جعل الخط غامق Bold أو مائل Italic، تغيير لون الخط وغيرها من خصائص الخط.

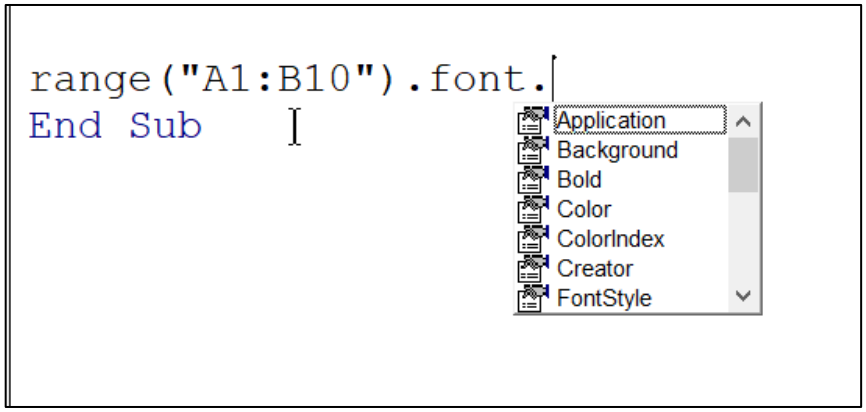
المثال التالي يوضح لنا كيفية استخدام هذه الخاصية لجعل الخط في خلايا النطاق A1:B10 مائلاً Italic:

```
Range("A1:B10").Font.Italic = True
```

وبالمثل، لو أردنا جعل الخط غامقاً نستخدم السطر التالي:

```
Range("A1:B10").Font.Bold = True
```

تساعدك بيئة البرمجة VBE في اختيار خاصية الخط المطلوبة من خلال ميزة الاكمال التلقائي، ففي المثالين السابقين بمجرد أن تكتب " Range("A1:B10").Font. " يقوم VBE بعرض الخيارات المتاحة مما يساعدك ويسهل عليك كتابة الكود.



الخاصية Interior

يمكن استخدام هذه الخاصية للتحكم بلون الخلفية ونمطها Color and pattern لخلايا النطاق المحدد. لتلوين الخلايا في النطاق A1:B10 باللون الأحمر نستخدم الكود التالي:

```
Range("A1:B10").Interior.Color = vbRed
```

يمكن استخدام ألوان أخرى مثل vbGreen, vbBlack وغيرها ويمكن استخدام رقم اللون كما في الشكل 5-3

5-3 Figure

Color Constant	Color Code	Color
vbBlack	0	
vbBlue	16711680	
vbCyan	16776960	
vbGreen	65280	
vbMagenta	16711935	
vbRed	255	
vbWhite	16777215	
vbYellow	65535	

وللحصول على ألوان أكثر يمكن استخدام الدالة RGB والتي يمكن ن خلالها المزج بين الألوان الأساسية (الأحمر، الأخضر والأزرق للحصول على 16 مليون لون). المثال التالي يوضح لنا كيفية استخدام الدالة RGB لاعطاء اللون الرمادي لنطاق من الخلايا:

```
Range("A1:B10").Interior.Color = RGB(128, 128, 128)
```

انظر الشكل 5-4 لمعرفة كيفية تمثيل بعض الألوان الشائعة باستخدام الدالة RGB.

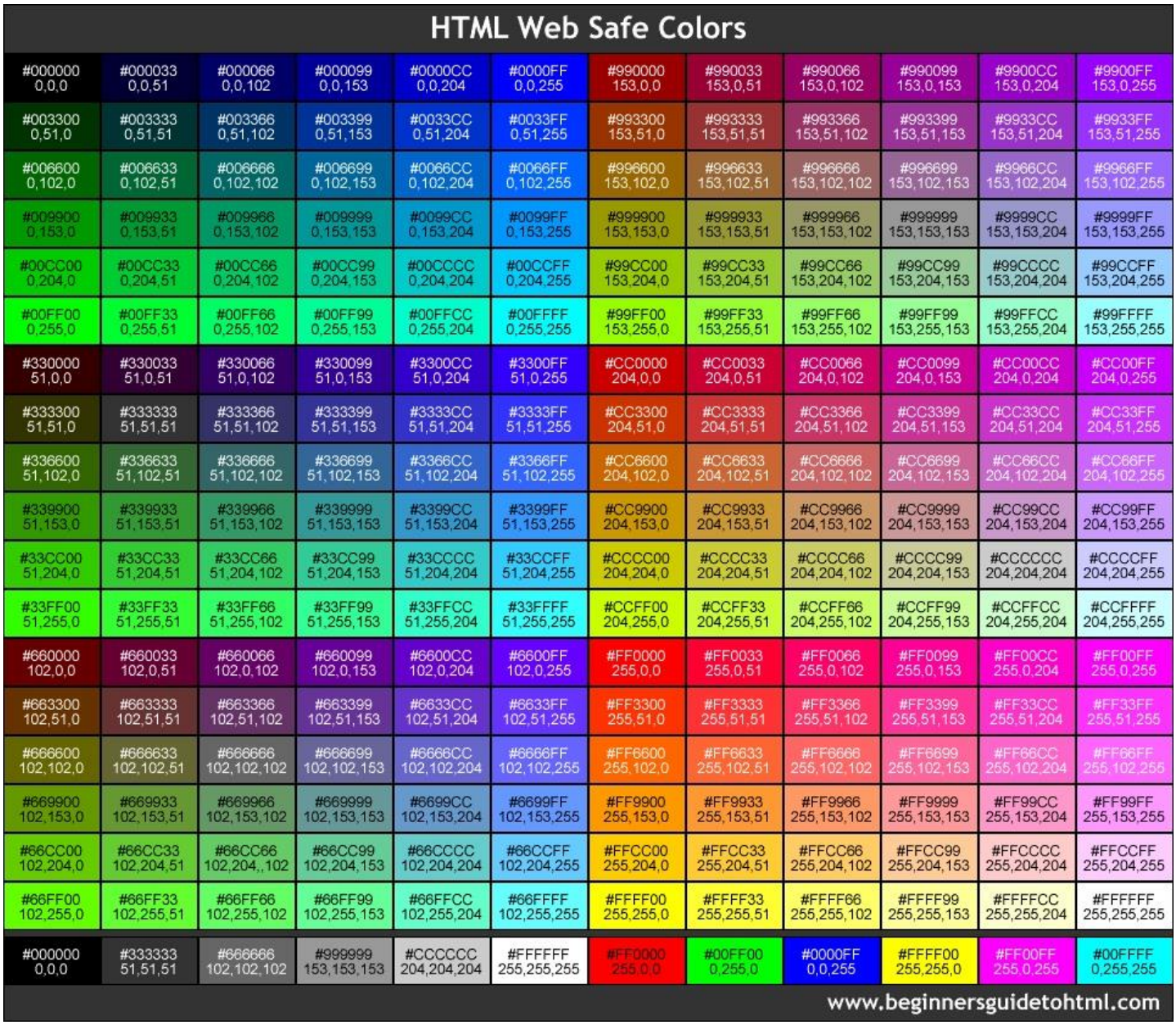


Figure5-4

بعض الوظائف Methods التي يمكن تطبيقها على كائنات النطاق

الوظائف هي مجموعة من الإجراءات التي يمكن تطبيقها على الكائن. هناك العشرات من الوظائف التي يمكن تطبيقها على كائنات النطاق وفي هذا الجزء سوف نستعرض بعضاً من أهمها.

وظيفة التحديد Select

تستخدم هذه الوظيفة لتحديد نطاق من الخلايا. الكود التالي يحدد النطاق A1:B10 في ورقة العمل النشطة:

```
Range("A1:B10").Select
```

قبل استخدام Select لتحديد نطاق ما، من الجيد أن تستخدم سطراً برمجياً آخر للتأكد أن ورقة العمل المطلوب تحديد النطاق بها هي الورقة النشطة Active sheet. يتم ذلك من خلال الوظيفة Activate. مثلاً إذا كان النطاق المطلوب تحديده ضمن ورقة العمل Sheet1 نستخدم الكود التالي:

```
Sheets( "Sheet1" ) . Activate

Range("A1:B10").Select
```

لاحظ أن الجملة التالية لا تعتبر صحيحة وترجع قيمة خطأ عند تنفيذ البرنامج إذا لم تكن Sheet1 هي الورقة النشطة.

```
Sheets("Sheet1").Range("a1").Select
```

بإمكانك اجراء عمليات النسخ واللصق من خلال وظيفتي Copy and Paste. مع ملاحظة أن كل وظيفة من هاتين الوظيفتين تُطبق على كائن مختلف، فوظيفة Copy تطبق على كائن النطاق بينما تطبق وظيفة اللصق على كائن ورقة العمل Worksheet.

الإجرائية الفرعية التالية تقوم بنسخ محتويات النطاق A1:B10 إلى النطاق الذي يبدأ بالخلية D1:

```
Sub CopyPaste()
  Range("A1:B10").Select
  Selection.Copy
  Range("D1").Select
  ActiveSheet.Paste
End Sub
```

لاحظ أننا في هذا المثال استخدمنا الكائن ActiveSheet وهو نوع خاص من الكائنات يشير إلى ورقة العمل النشطة.

ويمكن اختصار الكود السابق بالكود التالي:

```
Sub CopyPaste()
  Range("A1:B10").Copy Range("D1")
End Sub
```

يستفيد هذا الكود من حقيقة أن الوظيفة Copy تستطيع أن تأخذ معاملاً يحدد وجهة النسخ Destination وفي حالتنا الوجهة هي Range("D1").

وظيفة المسح Clear Method

تمحو هذه الوظيفة محتويات الخلايا بالإضافة إلى أي تنسيقات مطبقة على الخلايا. الكود التالي يستخدم هذه الوظيفة لمحو محتويات الخلايا في النطاق (A1:B10):

```
Range ("A1:B10").Clear
```

بالإضافة إلى هذه الوظيفة هناك الوظيفة ClearContents التي تمحو محتويات الخلايا بدون أن تمحو التنسيق، والوظيفة ClearFormats التي تمحو التنسيق بدون أن تمحو المحتويات.

وظيفة الحذف Delete method

تختلف وظيفة الحذف Delete عن وظيفة المسح Clear فعندما تحذف نطاقاً من الخلايا يتم إزاحة الخلايا المحيطة لسد مكان الخلايا المُزاحة. لذلك يجب أن يتم تحديد اتجاه الازاحة كمعامل لوظيفة Delete. أما إذا أردت حذف صف أو عمود فلا يلزم تحديد اتجاه الازاحة ويتم حذف الصف أو العمود مباشرة.

كي تتخيل عمل هذه الوظيفة حدد نطاق بداخل ورقة العمل ثم انقر بزر الماوس الأيمن واختر Delete حذف، سوف يظهر مربع حوار حينها يطلب منك تحديد اتجاه الازاحة إذا كنت تريد حذف النطاق أو تحديد إذا ما كنت تريد حذف صفوف أو أعمدة النطاق بأكملها.

الكود التالي يحذف العمود D:

```
Columns("D:D").Delete
```

أما الكود التالي فيحذف خلايا النطاق (A1:B10):

```
Range("A1:B10").Delete xlToLeft
```

لاحظ هنا اننا استخدمنا الكلمة المفتاحية xlToLeft لإزاحة الخلايا التي على يسار الخلايا المحذوفة ويمكن استخدام الكلمة xlUp لازاحة الخلايا التي بأعلى الخلايا المحذوفة.

الفصل السادس

6. جمل التحكم

تعد جمل التحكم من أهم الجمل البرمجية حيث أنك تستطيع من خلال جمل التحكم Control statements، التحكم بسير البرنامج بناءً على تحقق شرط أو مجموعة من الشروط. وفي هذا الفصل سوف نتعرف على العديد من جمل التحكم التي توفرها لنا لغة VBA وكيفية استخدامها.

جملة IF – Then

تعد هذه الجملة هي أشهر وأهم جمل التحكم في لغات البرمجة قاطبةً. من خلال هذه الجملة يمكن تنفيذ أوامر محددة بناءً على تحقق شرط أو شروط محددة. إذا تم استخدام الكلمة Else مع جملة IF فهذا يسمح لنا بتنفيذ أمر أو مجموعة من الأوامر في حال عدم تحقق الشرط. في احدى حالاتها؛ تأخذ جملة IF-Then الشكل التالي (لاحظ هنا أنه في هذه الحالة يجب أن تكون كلمتي IF وThen على نفس السطر):

IF [الشرط] Then [الأمر في حال تحقق الشرط]

في حال وجود كلمة Else يصبح شكل الدالة كالتالي (وهنا لاحظ أيضاً أن IF, Then, and Else يجب أن يكونوا على نفس السطر):

IF [الشرط] Then [الأمر في حال تحقق الشرط] Else [الأمر في حال عدم تحقق الشرط]

مثال على IF- Then

في هذا المثال تقوم جملة IF بالتحقق من القيمة في الخلية A1، وإذا كانت القيمة أكبر من 60 تعيّن القيمة Pass للمتغير result. ومن ثم يتم تعيين قيمة المتغير result للخلية B1.

```
Sub PassFail()
   Dim score As Integer, result As String
   score = Range("A1").Value
   If score >= 60 Then result = "Pass"
   Range("B1").Value = result
End Sub
```

جملة IF – Then – Else

في المثال السابق، إذا كانت القيمة في الخلية A1 أقل من 60 فإن الخلية B1 لا تظهر أي قيمة. إذا أردنا أن تظهر القيمة Fail في حال كانت قيمة الخلية A1 أقل من 60؛ فمن الممكن استخدام كلمة Else مع جملة IF، كما في الكود التالي الذي تم تعديله على المثال السابق.

```
Sub PassFail()
   Dim score As Integer, result As String
   score = Range("A1").Value
   If score >= 60 Then result = "Pass" Else result = "Fail"
   Range("B1").Value = result
End Sub
```

جملة IF- Then – [Else]- Endif

في حال كان لديك أكثر من أمر لتنفيذه سواءً عند تحقق الشرط أو عند عدم تحققه، فمن الممكن استخدام الشكل التالي من جملة IF:

```
If [الشرط] Then
   [الأمر البرمجي في حال تحقق الشرط]
End If
```

وفي حال If - Else يصبح الشكل كالتالي:

```
If [الشرط] Then
   [الأمر البرمجي في حال تحقق الشرط]
Else
   [الأمر البرمجي في حال عدم تحقق الشرط]
End If
```

مثال

في هذا المثال، المطلوب هو أنه إذا كانت القيمة في الخلية A1 أقل من 60 أن يتم كتابة Fail في الخلية B1 بالإضافة إلى عرض مربع حوار يعرض الجملة "Hard luck!!".

لهذا الهدف نستخدم الكود التالي:

```
Sub PassFail()
   Dim score As Integer, result As String
   score = Range("A1").Value
   If score >= 60 Then
      result = "Pass"
   Else
      result = "Fail"
      MsgBox "Hard luck!!"
   End If
```

```
    Range("B1").Value = result
End Sub
```

في هذا الشكل من أمر If يمكنك إضافة أي عدد من الأسطر البرمجية (التي تمثل الشروط) تحت جزء If وأي عدد من الأسطر البرمجية تحت جزء Else. هذه الطريقة في كتابة جملة IF تعد في نظر الكثيرين أفضل من الطريقة التي تم ذكرها أولاً والتي يتم فيها كتابة جملة If على سطر واحد حيث أنها بالإضافة إلى أنك تستطيع أن تضيف أي عدد من الشروط وأي عدد من الجمل البرمجية في حال تحقق الشرط/ أو الشروط المطلوبة، أسهل في القراءة والتتبع بالإضافة إلى أنك في هذه الحالة ستكتب جمل برمجة أقصر وبالتالي سوف تكون أقل عرضة للأخطاء.

استخدام If – Then -Elseif – Endif

في حال كان لديك مجموعة من الحالات / الشروط للاختيار فيما بينها يمكنك استخدام Elseif مع جملة If. تأخذ جملة If مع Elseif الشكل العام التالي:

```
If [الشرط الأول] Then
   [الكود البرمجي في حال تحقق الشرط الأول]
ElseIf [ الشرط الثاني] Then
   [الكود البرمجي في حال تحقق الشرط الثاني]
ElseIf
.
.
ElseIf [ الشرط الأخير] Then
   [الكود البرمجي في حال تحقق الشرط الأخير]
End If
```

بفحص الشرط الأول، فإذا تحقق يتم تنفيذ الكود البرمجي الذي يقع تحته مباشرة IF في هذه الحالة تقوم بفحص الشرط الثاني، واذا تحقق ينفذ الكود تحته If ، واذا لم يتحقق تقوم Ifثم ثم يتم الخروج من دالة وهكذا. Ifويتم الخروج من دالة

هنا ملاحظة مهمة جداً وهي أن جملة If – ElseIf لا تقوم بفحص كافة الشروط وانما تتوقف عند أول شرط يتحقق، فتنفذ الكود البرمجي الخاص بذلك الشرط ثم يخرج البرنامج من دالة If ويستمر في الكود الذي يلي الدالة.

مثال

في هذا المثال المطلوب فحص القيمة في الخلية A1 (علامة الطالب) ومن ثم عرض مربع حوار يظهر التقدير الذي سوف يكون كالتالي:

الدرجة	التقدير
أكبر من 90	ممتاز Excellent
من 80 وحتى أقل من 90	جيد جداً Very Good
من 70 وحتى أقل من 80	جيد Good
من 60 وحتى أقل من 70	مقبول Pass
أقل من 60	راسب Fail

للوصول لذلك الهدف نستخدم جملة If – ElseIf كما في الإجرائية بالأسفل، وكما تم ذكره سابقاً لا تقوم جملة If – ElseIF بفحص جميع الشروط وانما تنتهي عند تحقق أي من الشروط ففي هذا المثال اذا كانت علامة الطالب أكبر من 90، يتم فحص الشرط الأول ومن ثم تنفيذ الأوامر البرمجية الخاصة بذلك الشرط وهي جملة **MsgBox "Excellent" بعد ذلك يخرج البرنامج من دالة If** ويستمر في الكود الذي يلي الدالة (في هذا المثال لايوجد أي كود بعد دالة If لذلك فبمجرد الخروج من دالة If سوف تنتهي الإجرائية):

```
Sub StudentsGrades()
        If Range("A1").Value >= 90 Then
           MsgBox "Excellent"
        ElseIf Range("A1").Value >= 80 Then
           MsgBox "Very Good"
        ElseIf Range("A1").Value >= 70 Then
           MsgBox "Good"
        ElseIf Range("A1").Value >= 60 Then
           MsgBox "Pass"
        ElseIf Range("A1").Value < 60 Then
           MsgBox "Fail"
        End If
End Sub
```

استخدام If – ElseIf – Else

من الممكن استخدام Else مع جملة If – ElseIf، وفي هذه الحالة يكون موقع Else في النهاية حيث أنه في حال عدم تحقق أي من الشروط المعطاة؛ يتم تطبيق الكود الذي يقع بعد كلمة Else.

كمثال على ذلك، تم إعادة كتابة الإجرائية السابقة باستخدام If – ElseIf – Else كما هو بالأسفل:

```
Sub StudentsGrades()
        If Range("A1").Value >= 90 Then
           MsgBox "Excellent"
        ElseIf Range("A1").Value >= 80 Then
           MsgBox "Very Good"
        ElseIf Range("A1").Value >= 70 Then
           MsgBox "Good"
        ElseIf Range("A1").Value >= 60 Then
           MsgBox "Pass"
        Else
           MsgBox "Fail"
        End If
End Sub
```

غني عن الذكر أنه يمكنك استخدام If بداخل If (Nested IFs)

عوامل المقارنات التي يمكن استخدامها Comparison operators

العامل	نوع المقارنة
>	أكبر من
<	أصغر من
=	يساوي
=< , <=	أكبر من ويساوي، أصغر من ويساوي
<>	لا يساوي

فهم العوامل المنطقية

تستخدم العوامل المنطقية عادةً في اجراء المقارنات بين نتائج تنفيذ الجمل البرمجية المنطقية. نتيجة تطبيق العوامل المنطقية هي إما صحيح Trueأو خطأ False. ومن أشهر العوامل المنطقية ,AND OR, and NOT

الجمل المنطقية هي الجمل البرمجية التي يكون نتيجة تنفيذها اما صحيح True أو خطأ False فمثلاً الجملة التالية هي جملة منطقية قد يكون نتيجة تنفيذها True أو False طبقاً للقيمة الموجودة بداخل الخلية A1:

```
Range("A1").Value >= 70
```

العامل AND

يستخدم العامل AND للمقارنة بين نتيجتين منطقيتين بحيث أنه يرجع القيمة True اذا كانت كلا النتيجتين True. أما إذا كانت احدي النتيجتين False فإن النتيجة النهائية سوف تكون False. فمثلاً الجملة البرمجية التالية تكون نتيجة تنفيذها True، فقط في حال كانت نتيجة تنفيذ كلا الجملتين المنطقتين على جانبي المعامل AND هي True (في حالة هذا المثال يجب أن تكون القيمة في الخلية A1 اكبر من 70 والقيمة في الخلية B1 أكبر من 100):

```
Range("A1").Value >70 AND Range("B1").Value >100
```

العامل OR

يستخدم العامل OR للمقارنة بين نتيجتين منطقيتين بحيث أنه يرجع القيمة True اذا كانت احدى النتيجتين أو كلاهما True. أما إذا كانت كلتا النتيجتين False فإن النتيجة النهائية سوف تكون False. فمثلاً الجملة البرمجية التالية تكون نتيجة تنفيذها True في حال كانت نتيجة تنفيذ أي من الجملتين المنطقيتين على جانبي OR هي True:

```
Range("A1").Value >70 AND Range("B1").Value >100
```

ملاحظة مهمة

العوامل المنطقية لا تقارن بين قيم الخطأ فمثلاً الاجرائية التالية سوف ترجع خطأ عند تنفيذها إذا لم يتم تعيين قيمة أكبر من صفر للمتغير rate، حيث أن القسمة على صفر هي عملية غير مسموح بها. وبالرغم من أن الجزء الواقع على يسار OR من الجملة

```
(Range("A1").Value > 70 OR Range("B1").Value / rate > 1)
```

قد يكون صحيحاً True إلا أن VBA سوف يظهر خطأ Overflow عند تشغيل البرنامج لأن الطرف الأيمن من جملة المقارنة يحتوي على عملية غير مسموح بها.

```
Sub test()
  Dim rate As Integer
  If Range("A1").Value > 70 OR Range("B1").Value / rate > 1 Then
    MsgBox "Right!!!)"
  End If
End Sub
```

العامل NOT

يستخدم العامل NOT لعكس النتيجة المنطقية. فإذا كانت نتيجة تطبيق الجملة التالية:

```
Range("A1").Value > 70
```

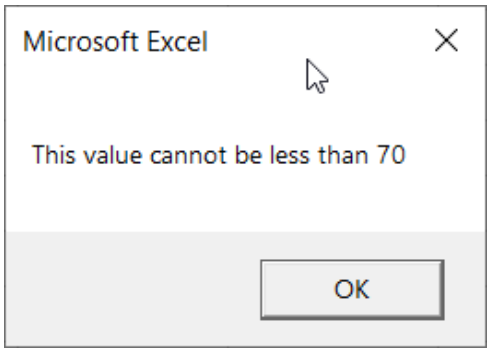
هي False فإن النتيجة تصبح True إذا تم تطبيق المعامل NOT على الجملة كالتالي:

```
Not Range("A1").Value > 70
```

مثال

الإجرائية التالية تظهر مربع الحوار التالي إذا لم تكن قيمة الخلية A1 أكبر من 70.

Figure6-1




```
Sub Test2()
   If Not Range("A1").Value > 70 Then
      MsgBox "This value cannot be less than 70"
   End If
End Sub
```

مثال 2

الإجرائية التالية تقوم بالتبديل بين عرض أو عدم عرض خطوط الشبكة Grid lines في ورقة العمل الحالية:

```
Sub ToggleGridLines()
  ActiveWindow.DisplayGridlines = Not ActiveWindow.DisplayGridlines
End Sub
```

يمكن إضافة زر لورقة العمل، وعند الضغط عليه يقوم بإخفاء خطوط الشبكة إذا كانت ظاهرة والعكس.

خطوط الشبكة GridLines هي احدى خصائص الكائن Window وليس Worksheet كما قد يعتقد البعض. لذلك استخدمنا ActiveWindow.DisplayGridlines وليس ActiveSheet. DisplayGridlines.

جملة Select Case statement

إذا كان لديك العديد من الخيارات للاختيار فيما بينها فإن أنسب جملة لهذا الغرض هي جملة Select Case (في كثير من الأحيان تكون أفضل حتى من جملة If – Elseif). تأخذ هذه الدالة جملة شرطية واحدة Expression والعديد من الخيارات Options (النتائج المحتملة لتلك الجملة) وعند تحقق أحد الخيارات يتم تنفيذ الكود المرافق لذلك الخيار ويتم الخروج من الدالة.

تأخذ هذه الجملة الشكل العام التالي:

```
Select Case Test_Expression تعبير الفحص

الخيار الأول Case
الكود حال تطابق هذا الخيار مع نتيجة الفحص Code

الخيار الثاني Case
الكود حال تطابق هذا الخيار مع نتيجة الفحص Code
.
.
أخرى Selectجمل
.
```

```
.


Case Else
 الكود حال عم وجود تطابق
End Select
```

مثال

في هذا المثال تم انشاء دالة تقوم بأخذ مؤشر الأداء للموظف (قيمة من 1 إلى 10 حيث 1 هو الأعلى تقييماً) ومقدار راتب الموظف، وترجع لنا مقدار مكافئة الموظف. فلدينا الآن التعبير Expression وهو عبارة عن قيمة المتغير performance، وهذا التعبير له العديد من الخيارات (من 1 حتى 10). عند تطابق نتيجة التعبير مع أي من الخيارات يتم تنفيذ الكود لذلك الخيار ثم يتم الخروج من جملة Case Select واذا لم يحدث أي تطابق يتم تنفيذ الكود الواقع تحت Case Else.

```
Function Bonus(performance, salary)
 Select Case performance
  Case 1
          Bonus = salary * 0.1
  Case 2, 3
    Bonus = salary * 0.09
  Case 4 To 6
    Bonus = salary * 0.07
  Case Is > 8
```

```
    Bonus = 100
  Case Else
    Bonus = 0
 End Select
End Function
```

بالنسبة للخيارات لاحظ أنه هناك أكثر من شكل لفحص الخيارات:

1- إذا كنا نريد نطابق مع قيمة محددة نكتب القيمة المحددة بعد كلمة Case مباشرة (مثل Case 1)
2- إذا كنا نريد أن نطابق مع أي قيمة ضمن نطاق من القيم المتجاورة نستخدم كلمة To (مثل Case 4 To 6)
3- إذا كنا نريد أن نطابق مع أي قيمة ضمن مجموعة من القيم نستخدم الفاصلة مثل (Case 2,3)
4- إذا كنا نريد أن نستخدم عوامل المقارنات نستخدم Is (مثل Case Is > 8)

مثال 2

في هذا المثال سوف نقوم بإعادة كتابة الإجرائية الخاصة بالتقدير والتي تم كتابتها سابقاً باستخدام جملة If – ElseIf. هنا سوف نستخدم جملة Case Select كبديل عن جملة If – ElseIf. لاحظ هنا أن الكود أكثر سهولة ووضوحاً.

```
Sub StudentsGrades()
  Select Case Range("A1").Value
    Case Is >= 90
      MsgBox "Excellent"
    Case Is >= 80
      MsgBox "Very Good"
```

```
    Case Is >= 70
       MsgBox "Good"
    Case Is >= 60
       MsgBox "Pass"
    Case Else
       MsgBox "Fail"
  End Select
End Sub
```

مثال 3

تقوم الإجرائية بالأسفل بعرض الربع السنوي للتاريخ الحالي في مربع حوار. تقوم جملة تعبير الفحص وهي Month(Date) بإرجاع رقم الشهر للتاريخ الحالي (الذي تم الحصول عليه من خلال الدالة Date)، قد ينتج عن تعبير الفحص 12 قيمة مختلفة (خيارات)، تم تجميع كل 3 خيارات مع بعضها من خلال كلمة To لفحص التطابق بين نتيجة تعبير الفحص وإحدى القيم ضمن المجموعات الأربع، ومن ثم ارجاع رقم الربع السنوي الحالي.

```
Sub ShowCurrentQuarter()
  Select Case Month(Date)
    Case 1 To 3
       MsgBox "Quarter 1"
    Case 4 To 6
```

```
        MsgBox "Quarter 2"
    Case 7 To 9
        MsgBox "Quarter 3"
    Case 10 To 12
        MsgBox "Quarter 4"
  End Select
End Sub
```

من الممكن كتابة الإجرائية السابقة كما في الكود أدناه، حيث استخدمنا النقطتان الرأسيتان Colon كبديل لإدراج سطر جديد بعد كل كلمة Case. ذلك قد يساهم في جعل الكود أكثر سهولة في القراءة والتتبع.

```
Sub ShowCurrentQuarter()
  Select Case Month(Date)
    Case 1 To 3: MsgBox "Quarter 1"
    Case 4 To 6: MsgBox "Quarter 2"
    Case 7 To 9: MsgBox "Quarter 3"
    Case 10 To 12: MsgBox "Quarter 4"
  End Select
End Sub
```

الفصل السابع

7. جمل التكرار Looping Statements

من خلال جمل التكرار تستطيع تكرار تنفيذ أوامر محددة حتى يتحقق شرط معين. من الأمثلة على استخدام جمل التكرار:

1- تنفيذ عمليات معينة على جميع المصنفات أو أوراق العمل المفتوحة. مثل حفظ واغلاق جميع المصنفات المفتوحة.
2- تنفيذ عمليات معينة على جميع عناصر مصفوفة محددة.
3- تنفيذ عمليات معينة على جميع الحروف في جملة محددة
4- تنفيذ عمليات معينة على جميع عناصر مجموعة معينة Collection مثل عناصر مجموعة المخططات Charts collection

هناك العديد من جمل التكرار التي توفرها لنا لغة VBA. وفي الفقرات القادمة سوف نتعرف عليها بالتفصيل.

جملة For….Next

تعد هذه الجملة أبسط جمل التكرار. يتم التحكم بالتكرار Looping من خلال متغير يعمل كعداد Counter variable حيث أن هذا المتغير يبدأ بقيمة محددة وينتهي عند قيمة أخرى. يتم تكرار الجمل البرمجية التي تقع ما بين كلمتي For و Next حتى وصول متغير العداد إلى قيمته النهائية.

مثال 1

هذا مثال بسيط يقوم بجمع الأعداد من 1 إلى 100 بشكل تراكمي.

```
Sub AddFirst100()
   Dim Total As Integer
   Dim Count As Integer
   Total = 0
   For Count = 1 To 100
      Total = Total + Count
   Next Count
   MsgBox Total
End Sub
```

في هذا المثال تم انشاء المتغير Count من نوع Integer لاستخدامه كعداد Counter variable، ومن خلال جملة For تم تعيين القيمة الدنيا والعليا له من خلال الجملة: For Count = 1 To 100. بالوضع الافتراضي يزيد العداد بمقدار 1 عند كل تنفيذ لجملة For حتى يصل لقيمته النهائية.

تم انشاء المتغير Total من نوع Integer ليتم تخزين حاصل الجمع التراكمي به. في كل مرة يتم فيها تشغيل جملة If يتم زيادة قيمة هذا المتغير بمقدار المتغير Count.

استخدام Step مع جملة for

كما تم ذكره سابقاً، يزيد العداد بالوضع الافتراضي بمقدار 1 عند كل تنفيذ لجملة for. إذا كنت تريد أن تكون الزيادة بمقدار قيمة أخرى من الممكن استخدام الكلمة المفتاحية Step مع جملة If.

مثال 2

إذا أردنا أن نقوم بجمع الأعداد الفردية الواقعة ضمن النطاق من 1 حتى 100 بشكل تراكمي نستخدم الإجرائية التالية:

```
Sub AddFirst100_Odd()
   Dim Total As Integer
   Dim Count As Integer
   Total = 0
   For Count = 1 To 100 Step 2
      Total = Total + Count
   Next Count
   MsgBox Total
End Sub
```

هنا استخدمنا كلمة Step مع جملة for لجعل العداد يزيد بمقدار 2 في كل مرة وبالتالي يتم في كل مرة جمع الأرقام الفردية فقط.

من الممكن أن تكون قيمة Step بالسالب، فمثلاً نستطيع كتابة جملة for في المثال الأول كما يلي:

```
Sub AddFirst100()
   Dim Total As Integer
   Dim Count As Integer
   Total = 0
   For Count = 100 To 1 Step -1
      Total = Total + Count
   Next Count
```

```
  MsgBox Total
End Sub
```

في هذا الكود جعلنا العداد يسير بالعكس من آخر قيمة حتى أول قيمة له وفي كل تنفيذ لجملة for يتم انقاص قيمة العداد بمقدار 1 من خلال استخدام Step -1.

مثال 3

في هذا المثال يقوم الكود يقوم بإدراج عدد محدد من أوراق العمل بشكل اوتوماتيكي. بحيث أن عدد أوراق العمل المطلوب اضافته يتم إدخاله من قبل المستخدم عن طريق مربع ادخال Input box. في داخل جملة for تم استخدام الجملة Sheets.Add لاضافة ورقة عمل في كل مرة يتم فيها تنفيذ جملة for.

```
Sub AddSheets()
Dim SheetsNumber As Integer, counter As Integer
SheetsNumber = InputBox("Please add the number of sheets you would like to add ")
For counter = 1 To SheetsNumber
  Sheets.Add
Next
End Sub
```

مثال 4

يقوم الكود التالي بتظليل سطر كل ثلاثة أسطر. نتيجة تنفيذ هذه الإجرائية هي كما في الشكل 1-7

```
Sub ShadeEveryThirdRow()
Dim i As Long
For i = 1 To 100 Step 3
   Rows(i).Interior.Color = RGB(200, 200, 200)
Next i
End Sub
```

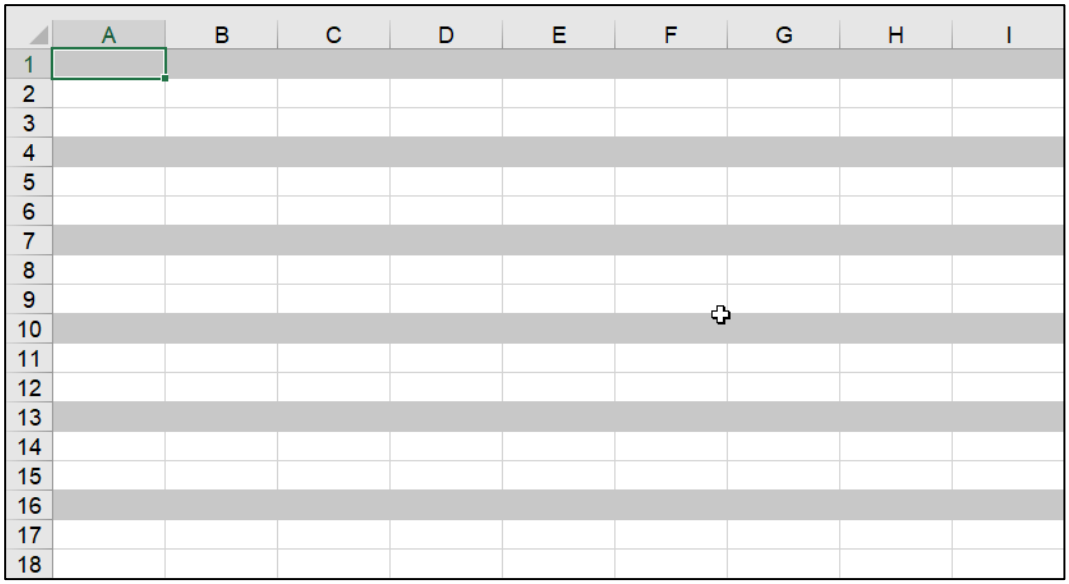


Figure 7-1

الخروج من جملة for من خلال جملة Exit For

من الممكن استخدام الجملة Exit For لإنهاء جملة التكرار والخروج منها. عادةً ما تستخدم هذه الكلمة بعد أن يتم التحقق من وجود شرط معين، فإذا تحقق هذا الشرط يتم الخروج من جملة for.

مثال

في هذا المثال تم انشاء جملة تكرار تقوم بإدخال القيم من 1 حتى 10 في الخلايا بدءاً من الخلية A1 وعمودياً باتجاه الأسفل بمقدار 10 خلايا - من خلال جملة Cells(counter, 1) = counter. تم استخدام دالة If لفحص قيمة العداد وفي حال أصبحت قيمة العداد counter 4 يتم الخروج من جملة For.

```
Private Sub Exit_For_Demo()
  Dim counter As Integer
   For counter = 1 To 10
      Cells(counter, 1) = counter
      If counter = 4 Then
         Cells(counter, 1) = counter * 10
         Exit For
      End If
  Next
End Sub
```

مثال 2

هنا مثال أكثر تطوراً عن استخدام Exit For مع دالة For. في هذا المثال تم استخدام جملة For مع Exit For لإنشاء دالة تأخذ معامل واحد هو عبارة عن جملة ثم تقوم باستخراج الحروف الواقعة على

يسار أول رقم من تلك الجملة فمثلاً إذا تم ادخال الجملة التالية على الدالة ABC123 تقوم الدالة بإرجاع ABC أما إذا تم ادخال AB1245 تقوم بإرجاع AB وهكذا.

تعمل هذه الدالة بالشكل التالي:

1- تقوم الدالة بتخزين قيمة الجملة المدخلة (معامل الدالة) في متغير باسم Str.
2- تعطي المتغير TextPart قيمة فارغة مبدئياً.
3- تم استخدام الدالة المدمجة Len لإرجاع عدد الأحرف في النص المدخل. من خلال Len(Str)
4- تم استخدام جملة For مع الدوال IsNumber و Mid لفحص كل رمز Charachter من النص بدءاً من اليسار هل الرمز عبارة عن رقم أم لا. حيث أن الدالة Mid تقوم باستخراج رمز واحد من النص المعطى في كل مرة يتم فيها تنفيذ جملة For -بدءاً من اليسار- بينما تقوم الدالة IsNumber بفحص هل الرمز رقم أم لا.
5- بالنسبة لجملة For، تم استخدام المتغير i كعداد لجملة التكرار، وتم تحديد القيمة الدنيا للعداد بـ 0 والعليا بعدد الرموز في النص المعطى (الذي تم الحصول عليه من خلال Len(Str).
6- تقوم جملة If الموجودة بداخل جملة التكرار كل مرة بفحص نتيجة تنفيذ الجملة IsNumeric(Mid(Str, i, 1)). وفي حال كانت النتيجة True (أي أن الرمز عبارة عن رقم). تقوم بإنهاء جملة التكرار من خلال تطبيق جملة Exit For. وإلا فإنها تقوم بإضافة الرمز بجانب الرموز السابقة من خلال معامل الربط & ووضع النتيجة في المتغير TextPart. وهي القيمة التي تقوم الدالة بإرجاعها في النهاية.

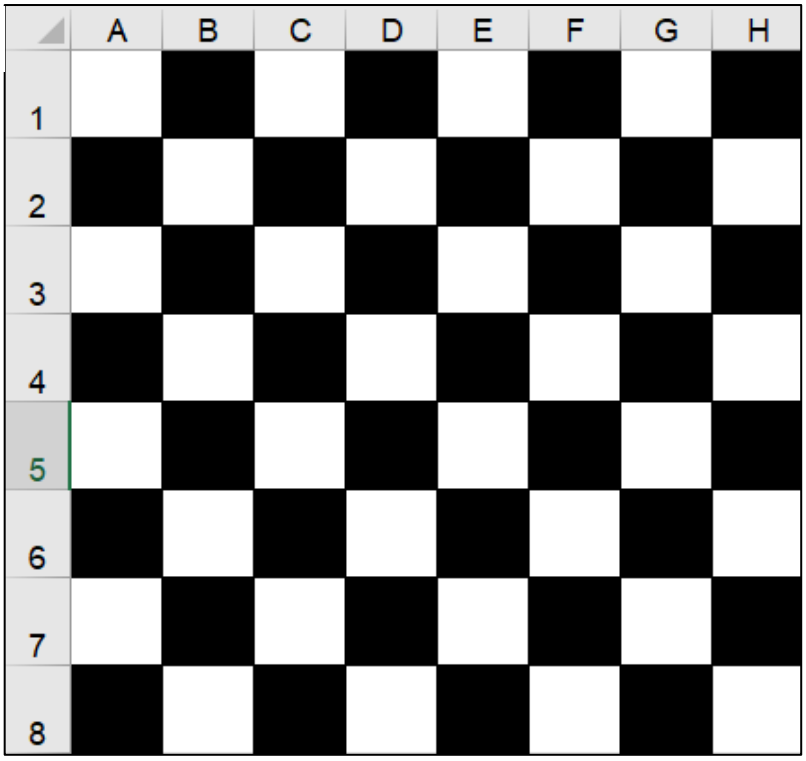
```
Function TextPart(Str)
TextPart = ""
  For i = 1 To Len(Str)
    If IsNumeric(Mid(Str, i, 1)) Then
      Exit For
    Else
```

```
        TextPart = TextPart & Mid(Str, i, 1)
    End If
  Next i
End Function
```

جملة For المتداخلة

من الممكن استخدام جملة For بداخل جملة أخرى لإنشاء جمل تكرار أكثر تعقيداً. في المثال التالي استخدمنا هذه التقنية لعمل شكل لوحة الشطرنج على ورقة العمل، كما في الشكل 7-2

Figure7-2



```
Sub MakeCheckerboard()
Dim Row As Integer, Col As Integer
For Row = 1 To 8
    If WorksheetFunction.IsOdd(Row) Then
        For Col = 2 To 8 Step 2
            Cells(Row, Col).Interior.Color = 0
        Next Col
    Else
        For Col = 1 To 8 Step 2
            Cells(Row, Col).Interior.Color = 0
        Next Col
    End If
Next Row
End Sub
```

الفكرة في هذا المثال هي أن لوحة الشطرنج تتكون من ثمانية أعمدة وثمانية صفوف، في الصفوف الفردية تكون الخلايا الزوجية ملونة بالأسود أما في الصفوف الزوجية تكون الخلايا الفردية ملونة بالأسود.

لذلك أنشأنا جملة For الخارجية لإنشاء تكرار على مدى الثمانية صفوف. بداخل جملة For الخارجية هناك جملتي For ، واحدة للتكرار عبر أعمدة الصفوف الفردية والأخرى للتكرار عبر أعمدة الصفوف الزوجية. هنا استخدمنا دالة ورقة العمل IsOdd لفحص هل الصف فردي أم زوجي.

بالنسبة للصفوف الفردية (التي تكون نتيجة تنفيذ دالة IsOdd فيها True) تقوم جملة For بالتكرار عبر الأعمدة الزوجية من 2 حتى 8 من خلال جملة

```
For Col = 2 To 8 Step 2
```

لاحظ هنا أننا استخدمنا Step 2 للقفز بقيمة العداد بمقدار 2 في كل تكرار. وفي كل تكرار يتم تلوين الخلية الناتجة عن تقاطع ذلك الصف الفردي مع العمود الزوجي باللون الأسود من خلال الكود:

```
Cells(Row, Col).Interior.Color = 0
```

وبالمثل يتم تلوين الخلايا الفردية في الصفوف الزوجية.

جملة For…Each…Next

تستخدم جملة For…Each…Next لتكرار تنفيذ مجموعة من الأوامر على كل عنصر من عناصر مجموعة Collections أو مصفوفة Array، مثل أن تستخدم هذه الجملة لتنفيذ أوامر معينة على أوراق العمل ضمن مصنف ما أو على خلايا ضمن أحد النطاقات.
في هذا النوع من جملة For لا يتم استخدام متغيراً كعداد Counter variable حيث أن الهدف هو تطبيق الأوامر البرمجية على جميع عناصر المجموعة Collection.

للمزيد حول المجموعات Collections راجع الفصل الثاني (محرر الأكواد) وحول المصفوفات Arrays راجع الفصل الرابع (المتغيرات)

مثال
يقوم الكود بالأسفل باستخدام For…Each…Next بإغلاق كل المصنفات المفتوحة ما عدا المصنف الحالي.

```
Sub CloseWorkbooks()
Dim wb As Workbook
  For Each wb In Workbooks
    If wb.Name <> ThisWorkbook.Name Then
    wb.Save
    wb.Close
    End If
  Next wb
End Sub
```

من هذا المثال يتضح لنا أن هناك أربع خطوات لكتابة جملة For Each Next هي كالتالي:
1- الإعلان عن متغير لكائن معين من نفس نوع العناصر التي نرغب بتكرار العمليات عليها. في هذا المثال تم الإعلان عن متغير باسم wb من نوع Workbook .

2- كتابة جملة For Each Next باستخدام العنصر الذي تم الإعلان عنه في الخطوة 1 والتجميعة Collection التي ينتمي لها العنصر. هنا استخدمنا السطر التالي للبدء بجملة For Each Next:

```
For Each wb In Workbooks
```

تبدأ الجملة بـ For Each ثم اسم المتغير (هنا اسمه wb) ثم الكلمة المفتاحية In ثم اسم التجميعة التي ينتمي اليها المتغير (وهي هنا Workbooks).

3- كتابة أسطر الأوامر التي نريد تكرارها عبر عناصر التجميعة المطلوبة. هنا استخدمنا جملة If لفحص اسم كل عنصر من عناصر التجميعة (في هذا المثال عناصر التجميعة عبارة عن مصنفات) ومن ثم حفظ وإغلاق أي مصنف اسمه مختلف عن اسم المصنف الحالي.

```
If wb.Name <> ThisWorkbook.Name Then
    wb.Save
    wb.Close
End If
```

مثال 2

تقوم الإجرائية التالية بإخفاء جميع أوراق العمل في المصنف الحالي ما عدا ورقة العمل الحالية.

```
Sub HideSheets()
  Dim Sh As Worksheet
  For Each Sh In ActiveWorkbook.Worksheets
    If Sh.Name <> ActiveSheet.Name Then
      Sh.Visible = xlSheetHidden
    End If
  Next Sh
End Sub
```

مثال 3

تقوم الإجرائية التالية بإظهار جميع أوراق العمل المخفية مرة واحدة (وهذا شيء لا يمكن عمله من خلال الاكسيل حيث أنك من خلال الاكسيل تستطيع اظهار ورقة عمل واحدة فقط في كل مرة ولا يمكن اظهار جميع الأوراق المخفية مرة واحدة)

```
Sub UnhideSheets()
Dim ws As Worksheet
    For Each ws In Worksheets
    ws.Visible = xlSheetVisible
    Next ws
End Sub
```

مثال 4

في هذا المثال تقوم الإجرائية التالية بتلوين الخلايا الفردية في النطاق A1:A20 باللون الأسود. في البداية تم تعريف متغيرين من نوع Range وهما Rng و cell. ثم تم تعيين النطاق A1:A20 للمتغير Rng من خلال السطر البرمجي:

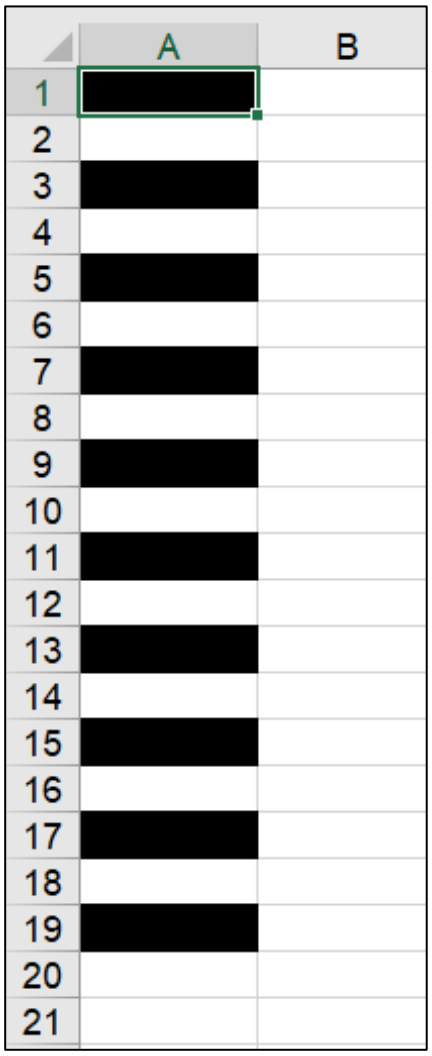
```
Set Rng = Range("A1:A20")
```

ثم تم كتابة جملة For Each Next للتكرار عبر عناصر مجموعة النطاق Range collection (الموجودة داخل المتغير Rng) وذلك من خلال المتغير cell. حيث يتم فحص المتغير cell (في هذا المثال يمثل المتغير cell خلية واحدة أو بالأحرى نطاقاً يتكون من خلية واحدة) في كل مرة يتم فيها التكرار من خلال دالة ورقة العمل IsOdd، إذا كانت الخلية تقع ضمن صف فردي يتم تلوين الخلية باللون الأسود.

```
Sub PaintOddCells()
Dim Rng As Range, cell As Range
Set Rng = Range("A1:A20")
For Each cell In Rng
```

```
    If WorksheetFunction.IsOdd(cell.Row) Then
        cell.Interior.Color = 0
    End If
Next
End Sub
```

Figure 7-3



مثال 5

تقوم الإجرائية التالية بحذف جميع أوراق العمل الفارغة من المصنف الحالي.

تعمل هذه الإجرائية كالتالي:

1- تم تعريف متغير من نوع Worksheet باسم ws
2- تم انشاء جملة For Each Next للتكرار عبر عناصر مجموعة أوراق العمل Worksheets Collection من خلال المتغير ws.
3- في كل مرة يتم فيها التكرار يتم تطبيق جملة If على المتغير ws والذي يحتوي على إحدى أوراق العمل. تقوم جملة If باستخدام دالة ورقة العمل CouontA لفحص ما إذا كانت خلايا ورقة العمل تحتوي على أية بيانات. حيث أنه وكما نعلم تقوم دالة CountA بحساب عدد الخلايا التي تحتوي على بيانات في نطاق محدد. تأخذ الدالة CountA المعامل ws.cells والذي هو عبارة عن نطاق يشمل جميع خلايا ورقة العمل.
4- في كل مرة تكون فيها نتيجة تنفيذ الدالة CountA صفر (أي أن الورقة فارغة) يتم حذف ورقة العمل من خلال السطر البرمجي: Ws.Delete
5- لإلغاء رسائل التحذير التي قد تظهر عند حذف أوراق العمل استخدمنا الجملة في بداية الإجرائية:

```
  Application.DisplayAlerts = False
```

ثم أرجعنا الوضع لما كان عليه في نهاية الإجرائية من خلال إعادة تفعيل رسائل التحذير عن طريق الجملة التالية:

```
  Application.DisplayAlerts = True
```

إذا لم نعطل اظهار رسائل التحذير فسوف يقوم الاكسيل بإظهار رسالة تحذير في كل مرة تقوم هذه الإجرائية بمحاولة حذف إحدى أوراق العمل.

```
Sub DeleteEmptySheets()
Dim Ws As Worksheet
Application.DisplayAlerts = False
    For Each Ws In ActiveWorkbook.Worksheets
```

```
        If  WorksheetFunction.CountA(Ws.Cells)  =  0
Then
        Ws.Delete
        End If
    Next Ws
Application.DisplayAlerts = True
End Sub
```

التكرار باستخدام الجملة Do...While...Loop

عند تطبيق هذه الجملة يتم فحص شرط معين وطالما أن هذا الشرط متحقق True، يتم تنفيذ الأوامر بداخل جملة التكرار. تأخذ هذه الجملة الشكل العام التالي:

```
Do While [الشرط]
        [مجموعة من الأوامر البرمجية]
Loop
```

كمثال بسيط على كيفية استخدام هذه الجملة، تقوم الإجرائية بالأسفل بعرض مربع حوار يظهر الجملة "Hello VBA " خمس مرات.

```
Sub DoWhileDemo()
    Dim counter As Integer
    counter = 1
    Do While counter <= 5
        MsgBox ("Hello VBA, " & counter)
        counter = counter + 1
  Loop
End Sub
```

تعمل هذه الإجرائية كالتالي:

1. تم الإعلان عن متغير ليعمل كعداد وأُعطي الاسم counter وتم تعيين قيمة مبدئية له بـ 1.

2. في جملة Do While تم استخدام الشرط counter <= 5 حيث أنه طالما أن هذا الشرط متحقق True يتم تنفيذ ما بداخل جملة التكرار.
3. بداخل جملة التكرار هناك سطران برمجيان (أمران) وهما:

```
MsgBox ("Hello VBA, " & counter)
counter = counter + 1
```

في السطر الأول يتم اظهار مربع حوار بالجملة "Hello VBA " مع رقم العداد، أما في السطر الثاني فيتم زيادة العداد counter بمقدار 1 في كل مرة يتم فيها التكرار.

4. عند وصول العداد counter إلى الرقم 6 تصبح نتيجة الجملة الشرطية False وبالتالي يتم الخروج من التكرار.

مثال 2

تقوم الإجرائية التالية بالتكرار عبر مجموعة من الخلايا التي تقع في نفس العمود بدءاً من الخلية النشطة، وشرط التكرار هنا أن لا تكون الخلية فارغة (من خلال ActiveCell.Value <> Empty). في حال أن الخلية تحتوي على قيمة يتم ضرب القيمة في 2 من خلال الأمر ActiveCell.Value = ActiveCell.Value * 2، ثم يتم التحرك بالخلية النشطة للأسفل بمقدار صف واحد من خلال الأمر ActiveCell.Offset(1, 0).Select. يتم تكرار العملية حتى تصبح الخلية النشطة فارغة وعندها يتم الخروج من جملة التكرار.

```
Sub DoWhileDemo()
    Do While ActiveCell.Value <> Empty
        ActiveCell.Value = ActiveCell.Value * 2
        ActiveCell.Offset(1, 0).Select
    Loop
End Sub
```

جملة Do Until...Loop

تشابه هذه الجملة جملة Do While إلا أن التكرار يتم هنا طالما أن الشرط غير متحقق False بعكس جملة Do While. فمثلاً إذا أردنا كتابة المثال الأول باستخدام Do Until تصبح الإجرائية كالتالي:

```
Sub DoWhileDemo()
   Dim counter As Integer
   counter = 1
   Do until counter > 5
      MsgBox ("Hello VBA, " & counter)
      counter = counter + 1
   Loop
End Sub
```

أما المثال الثاني فيمكن إعادة كتابته باستخدام Do Until كالتالي:

```
Sub DoWhileDemo()
   Do until  IsEmpty(ActiveCell.Value )
      ActiveCell.Value = ActiveCell.Value * 2
      ActiveCell.Offset(1, 0).Select
   Loop
End Sub
```

الفصل الثامن

8. استخدام دوال ورقة العمل ودوال VBA

في السابق ذكرنا أنه بإمكانك استخدام الدوال Function بداخل الكود البرمجي الخاص بـ VBA. في هذا الفصل سوف نتعرف على أنواع الدوال الثلاثة التي يمكن للغة VBA التعامل معها؛ وهي الدوال المدمجة في VBA والدوال المدمجة في الإكسيل (تسمى دوال ورقة العمل Worksheet functions)، والدوال المخصصة Custom functions وهي التي تقوم أنت بكتابتها.

ما هي الدالة؟

الدالة Function هي عبارة عن مجموعة من الأسطر البرمجية تم تنظيمها بشكل معين بحيث تقوم ببعض الحسابات أو الإجراءات ومن ثم ترجع قيمة واحدة. فمثلاً دالة SUM وهي أشهر دوال ورقة العمل تأخذ مجموعة من القيم (سواءً أكانت في نطاقات أم لا) ثم تحسب حاصل جمع المعطيات وترجع قيمة واحدة فقط هي قيمة حاصل الجمع.

الدوال المدمجة في VBA (Built-In VBA Functions)

توفر لنا لغة VBA العديد من الدوال المدمجة التي تسهل حياة المبرمج وتجعل الكود أكثر سرعة وسهولة. بعض تلك الدوال يأخذ معاملاً أو مجموعة من المعاملات Arguments وبعضها لا يأخذ. في هذا الجزء سوف نقوم باستعراض بعض الدوال. بطبيعة الحال هناك عدد كبير جداً من الدوال المدمجة في

VBA وهدف هذا الفصل ليس شرح جميع هذه الدوال وانما إعطاء بعض الأمثلة التي نستطيع من خلالها أن نوضح كيفية التعامل مع هذا النوع من الدوال.

الدوال Date, Time, and Now

تقوم دالة Date بإرجاع تاريخ اليوم أما الدالة Time فترجع الوقت الحالي، بينما ترجع الدالة Now التاريخ والوقت الحاليين. المثال التالي يقوم بعرض الوقت الحالي في مربع حوار:

```
Sub ShowCurrentDate()

  MsgBox "Today is: " & Date

End Sub
```

لاحظ هنا أن هذه الدوال لا تأخذ معاملات ولا حتى أقواس فارغة بخلاف دالتي ورقة العمل Date and Time اللتان تأخذان مجموعة من المعاملات ودالة ورقة العمل Now التي تأخذ أقواس فارغة. وفي الواقع لو أنك أضفت أقواساً فارغة للدوال السابقة فإن محرر VBE سوف يقوم بإزالتها.

دالة Len

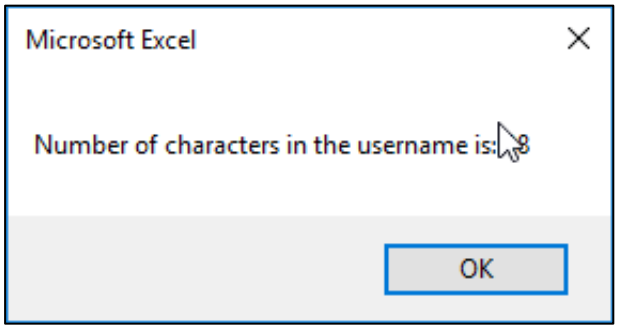
تقوم دالة Len بحساب عدد الأحرف في الجملة، وتأخذ معاملاً واحداً وهو عبارة عن الجملة المراد معرفة عدد الأحرف بها. تقوم الإجرائية التالية بحساب عدد الأحرف المكونة لاسم المستخدم لبرنامج الاكسيل:

```
Sub GetNameLength()

    Dim UserName As String

        MsgBox "Number of characters in the username is: " &
    Len(Application.UserName)

End Sub
```

لاحظ هنا أننا استخدمنا أحد خصائص تطبيق الاكسيل وهي خاصية UserName لمعرفة اسم المستخدم لبرنامج الاكسيل من خلال Application.UserName. حيث استخدمناه كمعامل لدالة Len.

عند تنفيذ الإجرائية نحصل على النتيجة كما بالشكل 1-8

Figure8-1



دالتي Month and MonthName

تقوم الدالة Month بعرض رقم الشهر لتاريخ معطى، بينما تقوم الدالة MonthName بأخذ رقم الشهر وعرض اسمه.

السطر البرمجي التالي يقوم بعرض رقم الشهر للتاريخ المعطى:

```
Month(#1/12/2019#)
```

تذكر أن تضع التاريخ بين علامتي السلم (#)، ولاحظ هنا أن التاريخ يجب أن تتم كتابته بالنظام الأمريكي (Month/Day/Year) بناءً على ذلك فنتيجة تنفيذ السطر السابق هي (1).

مثال 2:

الإجرائية التالية تقوم بعرض اسم الشهر للتاريخ الحالي:

```
Sub ShowMonthName()
   Dim MonthNumber As Integer
   MonthNumber = Month(Date)
   MsgBox MonthName(MonthNumber)
End Sub
```

ويمكن اختصار الكود السابق كما يلي:

```
Sub ShowMonthName()
    MsgBox MonthName(Month(Date))
End Sub
```

لاحظ هنا اننا استخدمنا الدوال المتداخلة Nested functions لاختصار السطر البرمجية. حيث أننا استخدمنا الدالة Date كمعامل للدالة Month والدالة Month كمعامل للدالة MonthName.

استخدام الدالة TypeName لتحديد نوع الكائن المحدد

تستخدم الإجرائية التالية الدالة TypeName لعرض نوع الكائن المحدد:

```
Sub ShowType()
  Dim ObjectType As String
  ObjectType = TypeName(Selection)
  MsgBox ObjectType
End Sub
```

إذا كان العنصر المحدد عبارة عن صورة فإن نتيجة تنفيذ الإجرائية السابقة هي كما في الشكل 2-8

Figure8-2

دوال VBA التي لا ترجع قيمة

هناك بعض دوال الـ VBA لا تقوم بإرجاع قيم عند تنفيذها ولكنها تقوم بفعل معين وهي كالتالي:

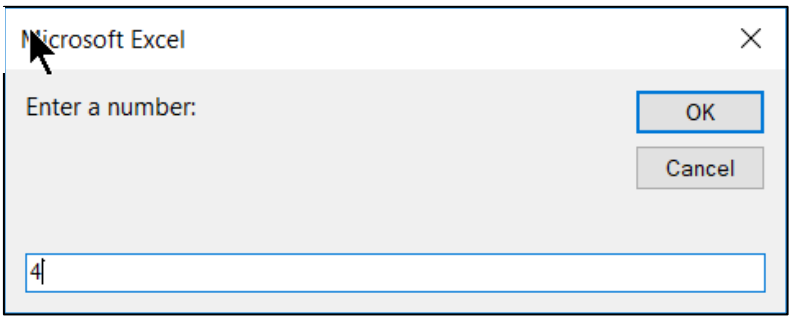
1- **دالة MsgBox:** تم التعرض لهذه الدالة مرات عديدة خلال هذا الفصل والفصول السابقة، وهي تقوم بعرض مربع حوار يحتوي على نص و/أو أزرار تحكم. إذا كان مربع الحوار الناتج عن هذه الدالة يحتوي على أزرار تحكم وضغط المستخدم على أحد هذه الأزرار فإن الدالة تقوم بإرجاع قيمة محددة حيث أنه من الممكن استخدام هذه القيمة في البرنامج لتحديد كيفية سيره (كأن يضغط المستخدم زر Cancel فيتم الغاء العملية أو OK للسير في البرنامج).

2- **دالة InputBox:** تقوم هذه الدالة بعرض مربع حوار بسيط يطلب من المستخدم ادخال قيمة معينة، ومن ثم يتم استخدام هذه القيمة في البرنامج. المثال التالي يوضح كيفية استخدام هذه الدالة حيث يطلب البرنامج من المستخدم ادخال رقم ومن ثم يرجع مربع الرقم:

```
Sub Power2()
   Dim Number As Double
   Number = InputBox("Enter a number: ")
   MsgBox Number * Number
End Sub
```

عند تشغيل البرنامج، سوف يظهر لنا مربع الادخال الخاص بدالة InputBox كما في الشكل 3-8. بعد أن يتم ادخال الرقم المطلوب في مربع الادخال سوف يتم تعيين تلك القيمة للمتغير Number ومن ثم ستقوم الدالة MsgBox بعرض مربع القيمة في مربع حوار.

Figure8-3



3- الدالة Shell: تستطيع من خلال هذه الدالة استدعاء أوامر نظام التشغيل ويندوز وبالتالي تشغيل أي تطبيق مثبت على النظام. تأخذ الدالة الشكل التالي:

```
Shell (Program,WindowStyle)
```

حيث أن Program هو اسم البرنامج المطلوب تشغيله، و WindowStyle هو كيفية اظهار نافذة البرنامج الذي تم استدعاؤه (كأن تكون مكبرة Maximize أو مصغرة Minimized، تم التركيز عليها Focused أو لم يتم التركيز عليها Not Focused، ....الخ). يظهر لنا الشكل 4-8 خيارات

اظهار النافذة، لاحظ هنا أن محرر الأوامر VBE يساعدك في الاختيار بين الخيارات المختلفة؛ فبمجرد وضع الفاصلة التي تلي اسم البرنامج المطلوب يقوم VBE بعرض قائمة بالخيارات للاختيار منها. ولاحظ أيضاً أن اسم البرنامج يجب أن يوضع بداخل علامات الاقتباس.

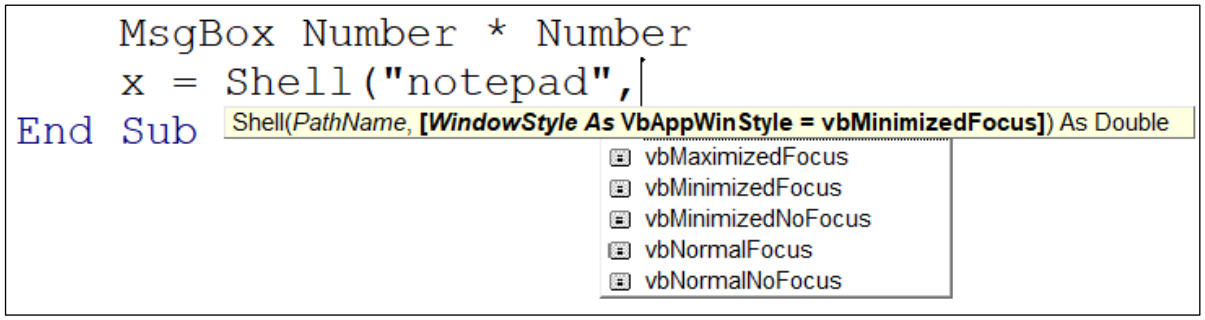


Figure8-4

من التطبيقات المفيدة لهذه الدالة استدعاء برنامج الآلة الحاسبة لإجراء الحسابات أثناء استخدام البرنامج أو برنامج التقاط الشاشة Snipping Tool لأخذ صور لشاشة الحاسب أثناء تشغيل البرنامج.

تقوم هذه الدالة بإرجاع رقم العملية الخاص بالبرنامج المُستدعَى Process ID (PID) ويجب تعيين هذه القيمة لمتغير حتى يمكن تشغيل البرنامج الذي تم استدعاؤه.

الإجرائية التالية تقوم باستدعاء برنامج الآلة الحاسبة، هنا تم الإعلان عن المتغير PID (من الأفضل أن يكون من نوع Variant في حالة هذه الدالة) ومن ثم تم تعيين قيمة تنفيذ الدالة Shell إليه . وبالطبع يمكن ربط هذه الإجرائية مع زر أوامر أو شكل لتشغيلها كما رأينا في الفصل الثالث.

```
Sub LaunchCalc()
  Dim PID As Variant
  PID = Shell("calc", vbNormalFocus)
End Sub
```

طلب المساعدة من محرر الأكواد في كتابة الدوال

لرؤية جميع الدوال المدمجة في VBA؛ يمكنك بالطبع اللجوء إلى نظام المساعدة الخاص بـ VBA عن طريق الضغط على F1 او من خلال الذهاب لقائمة Help. كما يمكنك أيضاً كتابة vba في أي مكان بداخل محرر الأكواد متبوعاً بالنقطة (.) فيقوم محرر الأكواد بعرض قائمة بجميع الدوال المتاحة في VBA وهنا يمكنك الاستفادة من ميزة الاكمال التلقائي حيث أنك بمجرد كتابة الأحرف الأولى من الدالة المطلوبة حتى يقوم VBE باختصار القائمة ليعرض فقط الدوال التي تبدأ بالأحرف التي تم كتابتها. شكل 8-5

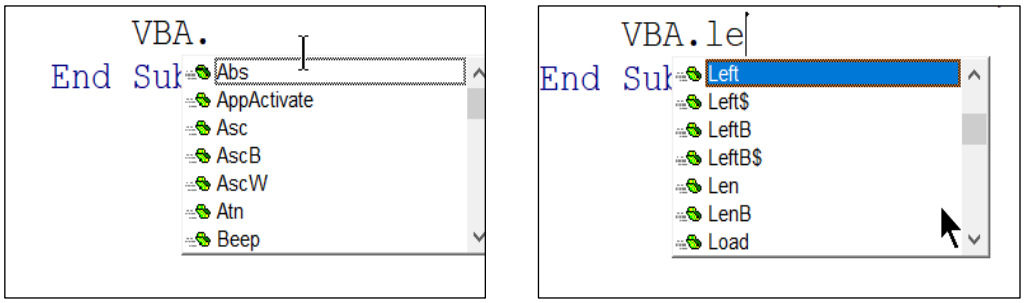


Figure8-5

دوال ورقة العمل Worksheet Functions

بالرغم من العدد الكبير للدوال المدمجة في VBA إلا أنك أحياناً قد لا تجد ما تبحث عنه ضمن تلك الدوال. ولكي تعطيك VBA خيارات أخرى فقد سمحت لك لغة VBA لاكسيل باستخدام معظم دوال ورقة العمل (دوال ورقة العمل التي لا يسمح لك محرر VBE باستدعائها هي الدوال التي يوجد ما يكافئها في الدوال المدمجة في VBA مثل دالة Now على سبيل المثال).

لاستخدام دوال ورقة العمل من الممكن استخدام الكائن WorksheetFunction الموجود ضمن كائن Application متبوعاً بدالة ورقة العمل. كمثال على استخدام دوال ورقة العمل SUM انظر الكود التالي والذي يقوم بإيجاد حاصل جمع خلايا النطاق A1:A10.

```
total = Application.WorksheetFunction.Sum(Range("A1:A10"))
```

ومن الممكن حذف جزء Application أو جزء WorksheetFunction من السطر البرمجي السابق والحصول على نفس النتيجة. فالجمل الثلاثة التالية تؤدي إلى نفس النتيجة:

```
total = Application.WorksheetFunction.Sum(Range("A1:A10"))
total = WorksheetFunction.Sum(Range("A1:A10"))
total = Application.Sum(Range("A1:A10"))
```

ربما كان من الأفضل استخدام الجملة البرمجية التي يظهر بها WorksheetFunction حيث أن الكود في هذه الحالة يكون أكثر وضوحاً.

بعض الأمثلة على استخدام دوال ورقة العمل

إيجاد الحد الأعلى والأدنى وثاني أكبر قيمة في نطاق محدد

من الممكن استخدام الإجرائية التالية لإيجاد المطلوبات أعلاه.

```
Sub MinMax()
  Dim TheMax As Double
  Dim TheMin As Double
  Dim Second_Large As Double
  TheMax = WorksheetFunction.Max(Range("A1:A10"))
  TheMin = WorksheetFunction.Min(Range("A1:A10"))
  Second_Large = WorksheetFunction.Large(Range("A1:A10"), 2)
  MsgBox "Maximum number in the range is " & TheMax
  MsgBox "Minimum number in the range is " & TheMin
  MsgBox "second large number in the range is " & Second_Large
End Sub
```

في هذه الإجرائية يقوم السطر

```
TheMax = WorksheetFunction.Max(Range("A1:A10"))
```

بإيجاد القيمة القصوى وتخزين الناتج في المتغير TheMax. بينما يقوم السطر

```
MsgBox "Minimum number in the range is " & TheMin
```

بإيجاد القيمة الدنيا وتخزين الناتج في المتغير TheMin. ويقوم السطر التالي بإيجاد ثاني أكبر قيمة وتخزين الناتج في المتغير Second_Large

```
Second_Large = WorksheetFunction.Large(Range("A1:A10"), 2)
```

من ثم تقوم الإجرائية بعرض النتائج من خلال ثلاث مربعات حوار متتالية.

مثال على استخدام الدلة VLOOKUP

هذا المثال يفترض أن لديك جدول بالأصناف يحتوي على اسم الصنف والسعر. **تأخذ** الإجرائية التالية رقم المنتج عن طريق مربع ادخال InputBox ومن ثم تقوم بالبحث عن السعر الموافق لذلك المنتج عن طريق دالة ورقة العمل VLOOKUP وترجع النتيجة من خلال مربع حوار MsgBox.

```
Sub ShowPrice()
    Dim PartNo As Variant
    Dim price As Double
    PartNo = InputBox("Enter the part number to get its price")
    Sheets("Prices").Activate
    price = WorksheetFunction.VLookup(PartNo, Range("A1:B10"), 2, False)
    MsgBox price
End Sub
```

تعمل هذه الإجرائية كالتالي:

1- تطلب من المستخدم ادخال رقم الصنف من خلال مربع ادخال عن طريق السطر البرمجي التالي:

```
PartNo = InputBox("Enter the part number to get its price")
```

2- تتأكد الإجرائية من أن ورقة العمل الفعالة هي الورقة التي تحتوي عل جدول الأسعار (وهي ورقة Prices) من خلال السطر البرمجي التالي:

```
Sheets("Prices").Activate
```

3- تقوم الإجرائية باستخدام دالة ورقة العمل VLOOKUP للبحث عن سعر الصنف بدلالة رقمه الذي تم الحصول عليه من الخطوة رقم 1 من خلال السطر التالي:

```
price = WorksheetFunction. VLookup(PartNo, Range("A1:B10"), 2, False)
```

4- تقوم الإجرائية بعرض السعر من خلال مربع حوار.

الفصل التاسع

9. التعامل مع الأخطاء في VBA

الأخطاء من الأمور الشائعة جداً عند التعامل مع VBA، تنقسم الأخطاء في VBA إلى ثلاثة أقسام:

1- الأخطاء التشغيلية
2- أخطاء الصياغة Syntax errors
3- الأخطاء المنطقية

الأخطاء التشغيلية Runtime time errors

الأخطاء التشغيلية هي الأخطاء التي تحدث أثناء تشغيل البرنامج وهي تحدث عندما يحاول البرنامج اجراء عملية غير شرعية، اما بسبب ادخال خاطئ مثل أن يقوم المستخدم بإدخال قيمة نصية في حين يتوقع البرنامج قيمة رقمية، أو أن يحاول البرنامج اجراء عملية ما على عنصر غير موجود؛ مثل أن يكون البرنامج مبرمجاً لإجراء عملية ما على ورقة العمل Sheet2 بينما لا يحتوي المصنف على ورقة العمل هذه، أو أي عملية أخرى غير شرعية مثل القسمة على صفر. عند حدوث هذا النوع من الأخطاء يتوقف البرنامج عن العمل ويُظهر رسالة خطأ. في الشكل 1-9 يظهر خطأً تشغيلياً ناتج عن محاولة البرنامج القسمة على صفر.

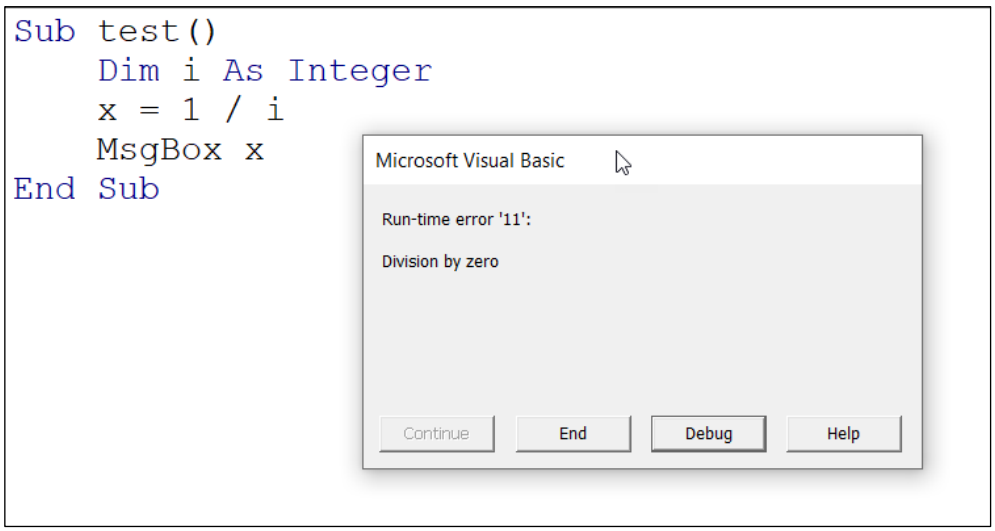


Figure9-1

من الوسائل التي يمكن استخدامها لتلافي الأخطاء التشغيلية، اجراء عملية تحقق للبيانات والسماح للمستخدم بإدخال النوع المناسب. كمثال على ذلك انظر الكود التالي:

```
Sub test()
  Dim i As Double
  i = InputBox("Enter a value")
  x = Sqr(i)
  MsgBox x
End Sub
```

تقوم هذه الإجرائية بعرض الجذر التربيعي للرقم الذي يقوم المستخدم بإدخال من خلال مربع رسائل، ثم تقوم بكتابة تلك القيمة في الخلية النشطة. يعمل البرنامج بشكل ممتاز طالما أن المستخدم يدخل قيماً موجبة، أما إذا حاول المستخدم ادخال رقم أقل من صفر فإن البرنامج سيُظهر رسالة خطأ.

ولتلافي هذه الخطأ من الممكن استخدام اجراء عملية تحقق للبيانات التي يقوم المستخدم بإدخالها كما في الكود التالي:

```
Sub test()
  Dim i As Double
  i = InputBox("Enter a value")
  If i < 0 Then
    MsgBox "You must enter a positive number."
    Exit Sub
  End If
  x = Sqr(i)
  MsgBox x
End Sub
```

هنا استخدمنا جملة If لإجراء عملية التحقق من صحة البيانات من خلال فحص قيمة الرقم المدخل فإذا كان ذو قيمة أقل من صفر تقوم جملة If بعرض رسالة خطأ، ومن ثم يتم الخروج من الإجرائية من خلال جملة Exit Sub.

هل انتهينا؟ جرب أن تدخل قيمة نصية. ماذا لو كانت ورقة العمل النشطة هي عبارة عن ورقة مخطط؟ سيظهر لك خطأً تشغيلياً أيضاً. من الممكن أن تكتب جمل أخرى اضافية للتحقق من صحة البيانات المدخلة، كما في الكود التالي (واحدة للتحقق من أن القيمة المدخلة هي قيمة رقمية وأخرى للتحقق من أن التحديد هو على نطاق Range وليس على عنصر آخر مثل الصور أو المخططات).

```
Sub test()
  Dim i As Variant
  i = InputBox("Enter a value")
```

```
  If i < 0 Then
     MsgBox "You must enter a positive number."
     Exit Sub
  End If
  If TypeName(Selection) <> "Range" Then
     MsgBox "Select a cell for the result."
     Exit Sub
  End If
  If Not IsNumeric(i) Then
     MsgBox "You must enter a number."
     Exit Sub
  End If
  x = Sqr(i)
  MsgBox x
  ActiveCell.Value = x
End Sub
```

هنا استخدمنا الجملة If TypeName(Selection) <> "Range" للتحقق من أن التحديد على نطاق من الخلايا وجملة If Not IsNumeric(i) Then للتحقق من أن الادخال ليس نصياً.

هل انتهينا بعد؟ جرب أن تكون ورقة العمل النشطة محمية وسيظهر لك خطأً تشغيلياً.

لسوء الحظ قد لا يمكنك تخمين جميع الأخطاء التشغيلية التي قد تظهر أثناء تشغيل البرنامج، إضافة إلى أن استخدام مجموعة كبيرة من جمل التحقق من الصحة يجعل البرنامج أكثر تعقيداً. وهنا يعطيك الاكسيل خياراً آخر للتعامل مع الأخطاء التشغيلية ألا وهو جملة On Error.

جملة On Error GoTo

تقوم جملة On Error باعتراض جميع أنواع الأخطاء التشغيلية التي قد تظهر ومن ثم يمكن من خلالها تحديد نقطة محددة في البرنامج يستمر البرنامج بعدها بعد أن يكون قد تجاوز الخطأ التشغيلي. في الإجرائية التالية أعدنا كتابة المثال السابق الخاص بالجذر التربيعي انما باستخدام جملة On Error كوسيلة لتلافي ظهور الأخطاء التشغيلية.

```
Sub test()
    Dim i As Double
    On Error GoTo bad_entry
    i = InputBox("Enter a value")
    x = Sqr(i)
    MsgBox x
    Exit Sub
    bad_entry:
    msg = "Bad entry. Please make sure that entry is number greater than zero,"
    msg = msg & " you select a cell, and worksheet is not protected"
    MsgBox msg
End Sub
```

لاحظ هنا أننا استخدمنا تسمية Label مع جملة On Error لإخبار البرنامج بأن يستمر من بعد تلك التسمية. في هذا المثال تم استخدام التسمية :bad_entry (لاحظ هنا أن التسمية يجب أن تكون متبوعة بالنقطتين الرأسيتين)، بينما تم استخدام الكلمة المفتاحية GoTo مع جملة On Error متبوعة باسم التسمية التي سوف يستمر البرنامج من بعدها عند حدوث خطأ ما.

جملة Exit Sub

لاحظ أنه في هذا المثال أننا استخدمنا الجملةُ Exit Sub قبل التسمية bad_entry مباشرة، حيث أن هذا الأمر يعمل على الخروج من البرنامج عند الوصول لتلك النقطة (بمعنى أدق، إذا اشتغل البرنامج بدون أي أخطاء فإنه سيتم الخروج من البرنامج قبل التسمية bad_entry وإلا فإن رسالة الخطأ ستظهر على الرغم من أنه لا توجد أية أخطاء في هذه الحالة.)

جملة On Error Resume Next

تقوم هذه الجملة بتجاهل الخطأ، والاستمرار بعد الجملة التي ظهر بها الخطأ. في المثال التالي تحاول الجملة (x = 20 / i) اجراء عملية غير شرعية وهي القسمة على صفر، إلا أن وضع جملة On Error Resume Next في بداية الإجرائية حال دون ظهور الخطأ التشغيلي. فهنا البرنامج تجاوز الجملة المسببة للخطأ كأنها لم تكن وأكمل من بعدها (جملة MsgBox "This is a test")

```
Sub test()
  On Error Resume Next
  Dim i As Integer
  i = 0
  x = 20 / i
  MsgBox "This is a test"
End Sub
```

جملة Resume *Label*

تقوم هذه الجملة بإلغاء حالة الخطأ Error condition ومن ثم الاستمرار في البرنامج من عند التسمية Label الموجودة في الجملة. كمثال على ذلك، انظر الإجرائية التالية (وهي نفس الإجرائية الخاصة بالجذر التربيعي مع جملة On Error GoTo وجملة Resume *Label*).

في هذه الإجرائية تم وضع تسمية Label في بداية البرنامج بإسم try_again. وفي نهاية الإجرائية تم عرض رسالة الخطأ مع أزرار Yes, No واستخدمنا الجملة الشرطية If لإخبار البرنامج بأن يستمر من عند التسمية try_again في حال ضغط المستخدم على زر Yes من خلال السطر البرمجي:

```
If ans = vbYes Then Resume try_again
```

```
Sub test()
   Dim i As Double
try_again:
   On Error GoTo bad_entry
   i = InputBox("Enter a value")
   x = Sqr(i)
   MsgBox x
    Exit Sub
bad_entry:
   msg = "Bad entry. Please make sure that entry is number greater than zero,"
```

```
  msg = msg & " you select a cell, and worksheet is not protected"
  ans = MsgBox(msg, vbYesNo + vbCritical)
  If ans = vbYes Then Resume try_again
End Sub
```

جملة Resume

تستخدم هذه الجملة فقط في حال ما إذا كان الكود الذي يعالج الأخطاء Error handling يحتوي على آلية لمعالجة الأخطاء، فبعد أن تتم معالجة الخطأ يتم استخدام كلمة Resume للرجوع مرة أخري للجملة التي أنتجت الخطأ. فمثلاً تقوم الإجرائية التالية بطلب أن يقوم المستخدم بإدخال قيمة ومن ثم اجراء عملية معينة على تلك القيمة وارجاع الناتج في الخلية A1 الموجودة ضمن ورقة العمل test. إذا لم تكن ورقة العمل test موجودة ضمن المصنف فإن خطأً تشغيلياً سوف يظهر. لتلافي ظهور هذا الخطأ تم إضافة جزء معالجة الأخطاء Error handler إلى الكود تحت التسمية error_handler.

في جزء معالجة الأخطاء تم إضافة كود لإدراج ورقة عمل بإسم test من خلال السطر البرمجي:

```
Worksheets.Add(after:=Worksheets(1)).Name = "test"
```

ومن ثم تم إضافة الكلمة Resume للرجوع مرة أخرى للجملة التي أنتجت الخطأ والمحاولة مرة أخرى.

```
Sub test()
  Dim i As Integer
  i = InputBox("Please enter a value")
  On Error GoTo error_handler
```

```
  Sheets("test").Range("A1") = i * 150
  Exit Sub
error_handler:
  Worksheets.Add(after:=Worksheets(1)).Name = "test"
  Resume
End Sub
```

إزالة الشوائب من البرامج Debugging Techniques

الشوائب Bugs هو مصطلح يشير إلى المشاكل التي تظهر في البرامج. بمعني أدق، إذا لم يعمل البرنامج كما يجب سواء أكانت النتائج التي يخرجها غير منطقية مثل أن تكون النتائج الرقمية صغيرة جداً أو كبيرة جداً أو إذا كان البرنامج يخرج رسائل خطأ عند تنفيذ عمليات محددة أو غيرها من الأخطاء، يقال أن هذا البرنامج به شائبة Bug. والحقيقة أن معظم البرامج - بما فيها البرامج التي تنتجها شركات كبيرة مثل مايكروسوفت – تحتوي على شوائب. فالشوائب هي من الأمور الواردة جداً عند كتابة البرامج. في هذا الفصل سوف نتعرف على أهم الطرق التي يمكن استخدامها لاكتشاف الشوائب ومواضع الخلل في البرامج Debugging.

استخدام مربع الحوار Message Box لاكتشاف الأخطاء

من الأخطاء الشائعة في البرمجة، أن تظهر قيمة متغير ما بشكل غير المتوقع كأن تكون كبيرة جداً أو صغيرة جداً. ومن الممكن في هذه الحالة استخدام مربعات الحوار لاكتشاف مكمن الخطأ.

من خلال هذا التكنيك يتم ادراج مربعات حوار في أماكن محددة بداخل الكود لتعرض قيمة المتغير عند كل مكان، بحيث تتم متابعة تغير قيمة المتغير أولاً بأول، مما قد يساعد في اكتشاف النقطة التي بدأ عندها الخلل.

كمثال على ذلك، انظر البرنامج التالي – وهو برنامج بسيط ليس له أي فائدة عملية باستثناء توضيح الفكرة – تقوم فكرة البرنامج على أخذ قيمتين من المستخدم i و a ، اذا كانت قيمة a أقل من 1 يتم

ضرب قيمة i في a. هنا استخدمنا مربعي حوار لمتابعة قيمة i الأول قبل ادخال قيمة a والآخر بعد إدخالها وبعد تنفيذ جملة If.

```
Sub test ()
  Dim i As Double
  i = InputBox("Enter a value of i")
  MsgBox "Value of i in the beginning is: " & i
  a = InputBox("Enter a value of a")
  If a < 1 Then i = i * a
  MsgBox "Value of i after If statment is: " & i
End Sub
```

لإظهار جميع القيم في مربع واحد يمكن استخدام الكود التالي:

```
Sub test()
  Dim i As Double
  i = InputBox("Enter a value of i")
  msg = "Intial value of i is: " & i & vbNewLine
  a = InputBox("Enter a value of a")
  If a < 1 Then i = i * a
  msg = msg & "Value of i after If statment is: " & i & vbNewLine
```

```
  MsgBox msg & "Value of a: " & a
End Sub
```

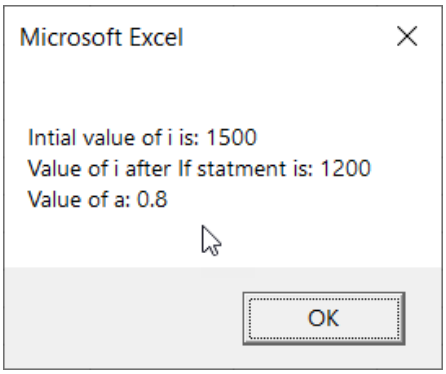


Figure9-2

لاحظ هنا أننا استخدمنا الكلمة المفتاحية vbNewLine مع دالة مربع الحوار MsgBox للنزول بالنص سطراً واحداً.

لا تنسَ أن تقوم بحذف مربعات النص بعد أن يتم تدقيق الكود وتصحيحه من الأخطاء

مثال 2

في المثال التالي يقوم الكود بالتكرار عبر أوراق العمل في المصنف وإعطاء الصف الأول في كل ورقة اللون الأحمر. من الممكن هنا استخدام مربعات الحوار لمعرفة الورقة التي يتم العمل عليها حالياً.

```
Sub test()
Dim ws As Worksheet
For Each ws In Worksheets
   ws.Activate
   MsgBox "Current sheet is: " & ws.Name
   ws.Range("1:1").Interior.Color = vbRed
Next ws
End Sub
```

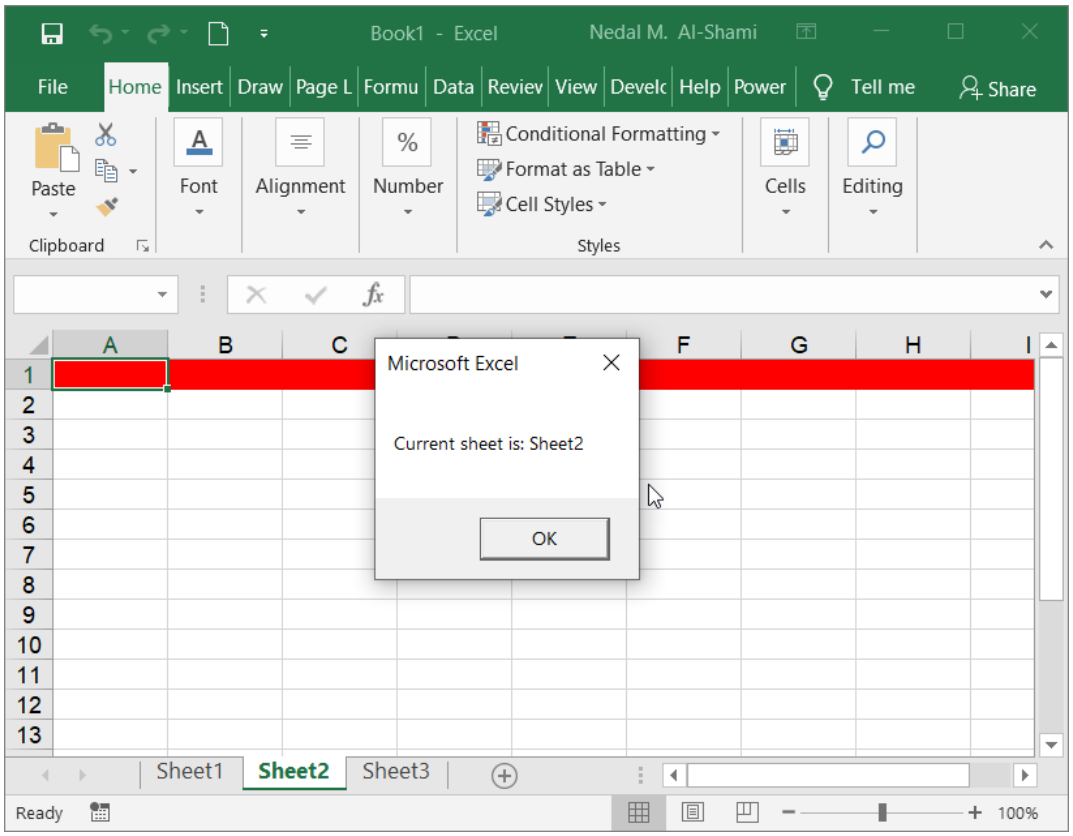


Figure9-3

استخدام جملة Debug.Print في تتبع الأخطاء

من الممكن استخدام Debug.Print كبديل لمربع الحوار حيث أن هذه الجملة تقوم بعرض القيم ولكن في نافذة التنفيذ الفوري Immediate Window التي يجب اظهارها إذا أردت استخدام هذا التكنيك من خلال استخدام الاختصار Ctrl + G.

في الكود التالي تم استخدام Debug.Print كبديل عن مربعات الحوار التي استخدمت في المثال السابق. في الشكل4-9 يظهر لنا كيف تظهر القيم في نافذة التنفيذ الفوري.

```
Sub test()
   Dim i As Double
   i = InputBox("Enter a value of i")
   Debug.Print "Intial value of i is: " & i
   a = InputBox("Enter a value of a")
   If a < 1 Then i = i * a
   Debug.Print "Value of i after If statment is: " & i
   Debug.Print "Value of a: " & a
End Sub
```

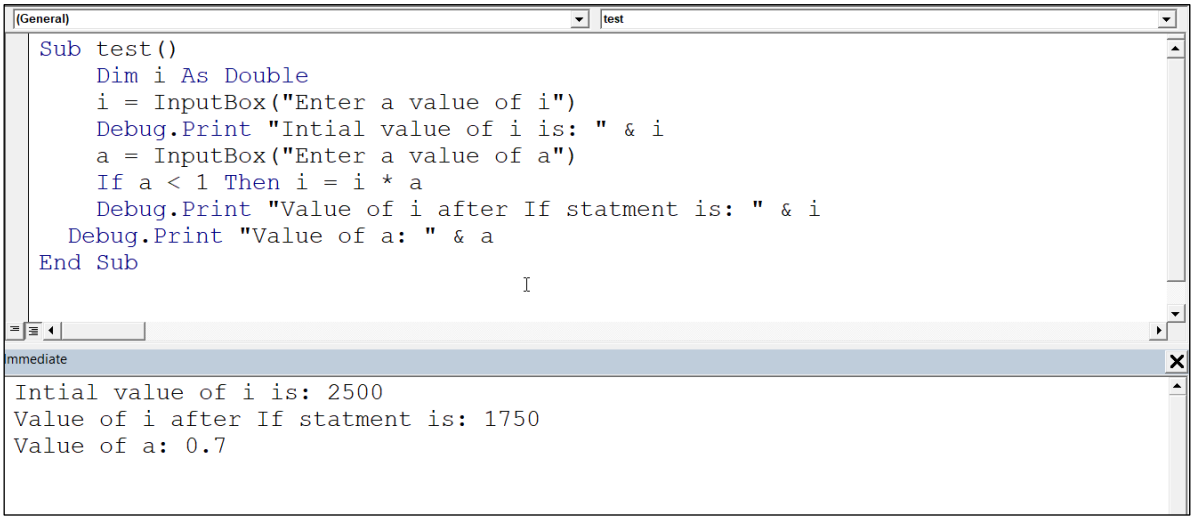


Figure9-4

استخدام VBA Debugger

يقدم الاكسيل مجموعة من الأدوات الت تساعد في عملية اكتشاف وتنقيح الأخطاء من خلال ما يعرف بـ VBA Debugger. هذه الأدوات تقدم لنا طرقاً أكثر قوة في اكتشاف الأخطاء من الوسيلتين اللتين تم التطرق لهما سابقاً (مربعات الحوار و Debug.Print). تالياً سوف نتطرق لتلك الأدوات تباعاً.

نقاط التوقف Breakpoints

تسمح لك نقاط التوقف بإيقاف تنفيذ البرنامج عند نقاط محددة ومن ثم النظر عن كثب وتدقيق البرنامج بحثاً عن الأخطاء. عندما يصل البرنامج إلى نقطة التوقف Breakpoint يدخل البرنامج في طور التوقف Break. وعندما يكون البرنامج في هذه الحالة يمكنك اجراء عمليات التدقيق التالية:

- تنفيذ أوامر VBA في نافذة التنفيذ الفورية (سيتم التطرق لهذا الموضوع لاحقاً)
- الاستمرار بتنفيذ الكود سطراً بسطر حيث أن الضغط على زر F8 والبرنامج في حالة التوقف يسمح لك بتنفيذ سطراً برمجياً واحداً في كل مرة.

- معرفة قيم المتغيرات بصورة سهلة وسريعة من خلال التحرك بمؤشر الماوس فوق المتغير، فيتم عرض قيمة المتغير في نافة صغيرة.
- تحرير وتعديل الكود ومن ثم الاستمرار في التنفيذ.
- تجاوز تنفيذ بعض الأسطر والانتقال مباشرة إلى السطر المطلوب (من الممكن السير في هذه العملية للأمام وللخلف)

ادراج نقاط التوقف

يمكن ادراج نقاط التوقف باستخدام أي من الطرق التالية:

- من خلال النقر بزر الماوس الأيمن فوق السطر الذي نريد ادراج نقطة التوقف عنده، ثم اختيار Toggle ثم Breakpoint
- من خلال النقر بزر الماوس الأيسر على المنطقة الرمادية الموجودة على يسار السطر الذي نريد ادراج نقطة التوقف عنده.
- من خلال وضع مؤشر الماوس في السطر المطلوب، ومن ثم الضغط على F9.

عند ادراج نقطة توقف عند سطر ما، يتم تلوين ذلك السطر باللون الأحمر، وتُوضع نقطة حمراء في الهامش الرمادي الموجود يسار الكود، مقابل السطر المطلوب.

Figure9-5

```
(General)
Sub test()
    Dim i As Double, a As Double
    i = InputBox("Enter a value of i")
    a = InputBox("Enter a value of a")
    If a < 1 Then i = i * a
    MsgBox "Value of i is: " & i
End Sub
```

لإزالة نقطة توقف، انقر بزر الماوس الأيسر فوق النقطة الحمراء الموجودة على يسار السطر.

عند تشغيل برنامج يحتوي على نقطة توقف، فإن تنفيذ البرنامج يتوقف قبل السطر الذي يحتوي على نقطة لتوقف، ويتم تظليل السطر باللون الأصفر ويظهر سهم أصفر على يسار السطر، وتظهر كلمة Break في أعلى النافذة.

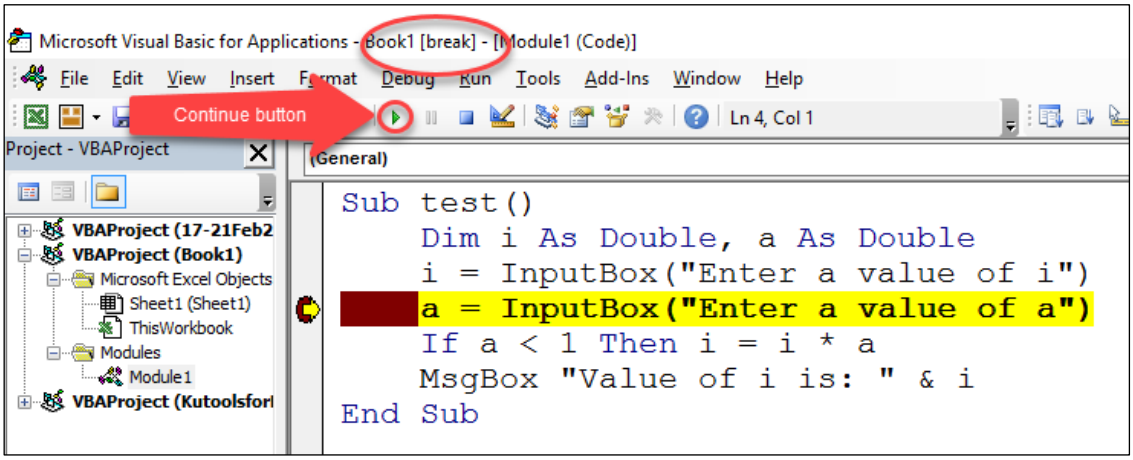


Figure9-6

بعد أن يدخل البرنامج في وضع الإيقاف:

- تحرك بمؤشر الماوس فوق المتغيرات لإظهار قيمها الحالية.
- أو اضغط على زر Continue (F5) للاستمرار في تنفيذ البرنامج.
- لتنفيذ البرنامج سطراً بسطر اضغط على F8.
- لتجاوز سطر أو أكثر، انقر فوق السهم الأصفر بزر الماوس الأيسر وباستمرار الضغط حرك السهم للمكان الذي تريد الاستمرار من عنده.

من الممكن استخدام الكلمة المفتاحية Stop لإدراج نقطة توقف، حيث تُكتب هذه الجملة قبل السطر الذي تريد ادراج نقطة التوقف عنده مباشرة.

نافذة التنفيذ الفوري Immediate Window

من خلال نافذة التنفيذ الفوري، يمكن تنفيذ أي أمر من أوامر VBA الصالحة. تظهر هذه النافذة عادة في أسفل محرر الكود، إذا لم تكن هذه النافذة ظاهرة فيمكن اظهارها من خلال الاختصار Ctrl + G.

لتنفيذ أي أمر في نافذة التنفيذ الفوري يجب كتابة كلمة Print قبل الأمر (يمكن الاستعاضة بـ ؟ بدلاً من Print). على سبيل المثال لعرض قيمة المتغير i نستخدم: Print i (أو ببساطة ? i). انظر الشكل9-7

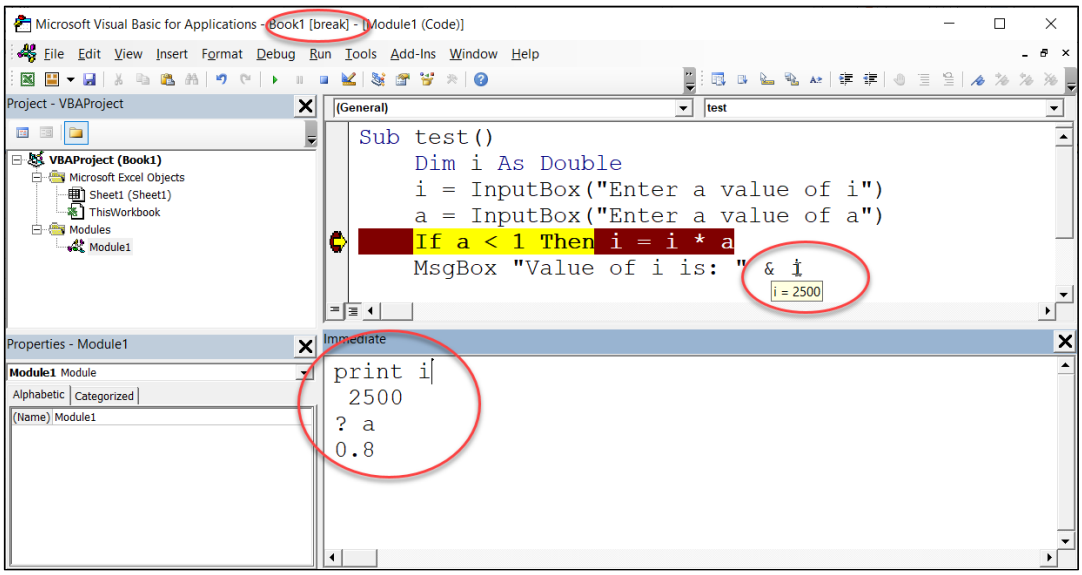


Figure9-7

استخدام نافذة المراقبة Watch Window

تتيح لك نافذة المراقبة معرفة قيمة متغير ما أو مجموعة من المتغيرات عند تحقق شرط معين. فمثلاً في الكود التالي، إذا أردت معرفة قيم المتغير x عند وصول قيمة i لـ 500، فيمكنك استخدام نافذة المراقبة من خلال الذهاب إلى قائمة Debug ثم اختيار Add Watch.

```
Sub test()

  Dim i As Double

    For i = 1 To 1000

      x = i + (i - 1) ^ 2
```

```
    Next i
  MsgBox x
End Sub
```

يظهر مربع الحوار الخاص بإضافة نافذة مراقبة كما في الشكل 8-9. أدخل الشرط /التعبير المطلوب في خانة Expression، ثم حدد الإجرائية المطلوب مراقبتها من خلال الخانة Procedure والوحدة البرمجية Module.

في الجزء الخاص بنوع المراقبة Watch Type هناك ثلاث خيارات:

1- Watch Expression: هذا الخيار يقوم بعرض نتيجة التعبير (الجملة البرمجية) الذي تم إدخاله في خانة Expression عند الوصول إلى نقطة توقف Breakpoint
2- Break when value is True: هذا الخيار يقوم بإدخال البرنامج في طور الإيقاف Breakpoint عند تحقق الشرط المدخل في خانة Expression من ثم يمكنك تفحص قيم المتغيرات من خلال نافذة Immediate Window أو من خلال التحرك بمؤشر الماوس فوق المتغير المطلوب.
3- Break When Value Changes: هذا الخيار يقوم بإدخال البرنامج في طور الإيقاف Breakpoint عند تغير قيمة المتغير/ نتيجة التعبير المدخل في خانة Expression

الشكل 8-9 يظهر لنا كيفية ادراج شاشة مراقبة، تقوم بإيقاف البرنامج عند وصول المتغير i للقيمة 500، بينما يقوم الشكل 9-9 بعرض شكل نافذة المراقبة والبرنامج عند تحقق الشرط المكتوب في خانة Expression.

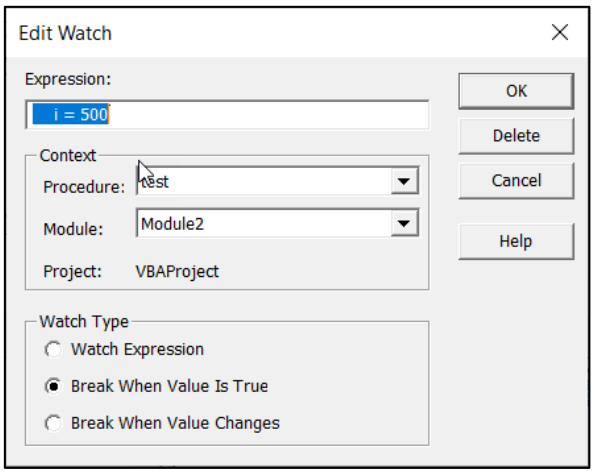


Figure9-8

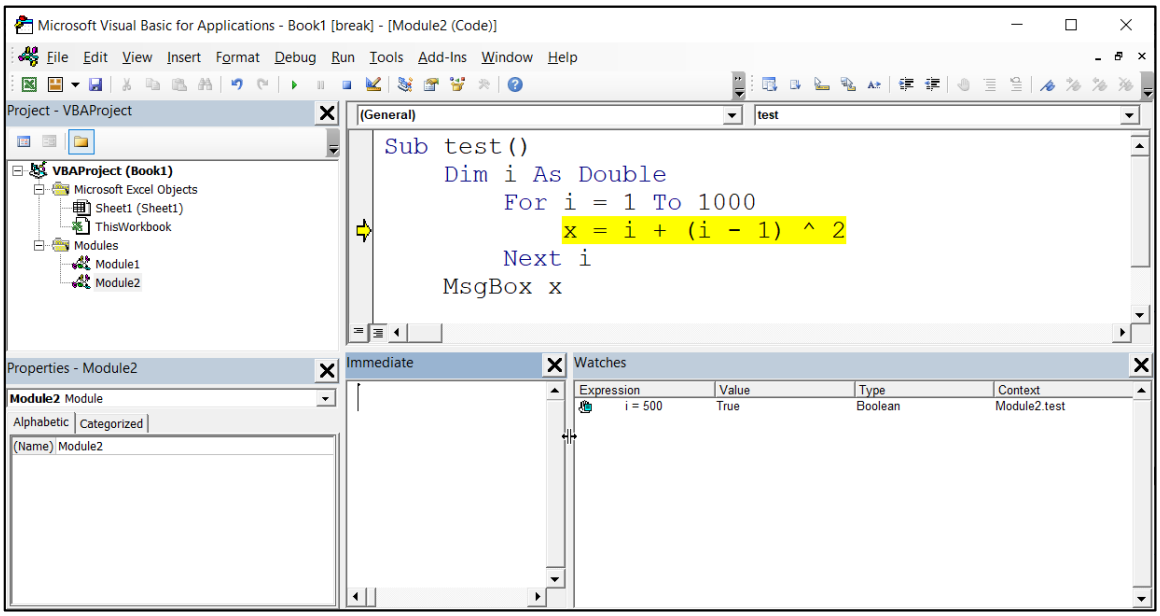


Figure9-9

استخدام نافذة Locals Window

تقوم نافذة Locals بعرض قيم جميع المتغيرات الواقعة ضمن الإجرائية الحالية، وذلك عند دخول البرنامج طور الإيقاف Breakpoint. يتم ادراج هذه النافذة من خلال الذهاب لقائمة View ثم Locals Window.

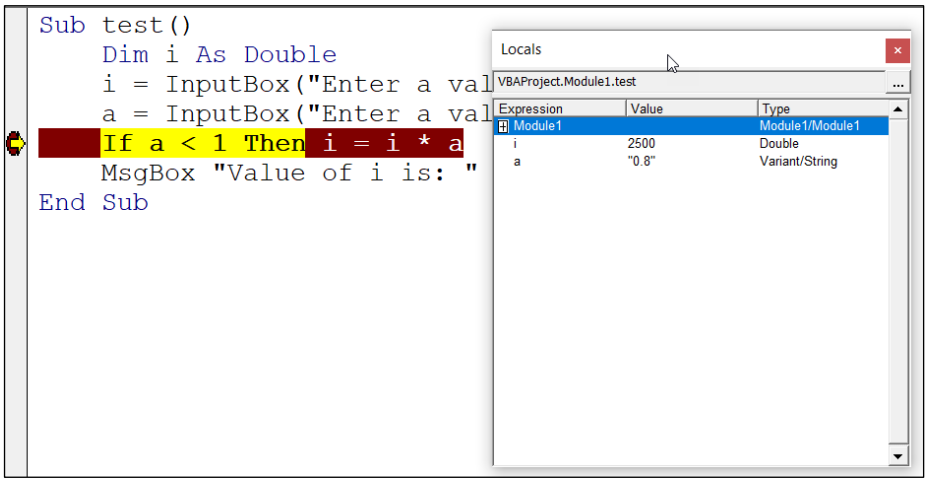


Figure9-10

تلميحات لتقليل الأخطاء البرمجية

1- حاول أن تستخدم الخيار Option Explicit في بداية البرامج دائما. استخدام هذا الخيار يقلل من احتمالية حدوث أخطاء بسبب خطأ مطبعي قي كتابة اسم متغير ما.
2- استخدم الكثير من التعليقات. وجود التعليقات يساعدك في فهم الكود عند الرجوع له بعد فترة من كتابة البرنامج. ويساعد أيضاً من يريد العمل على البرنامج من بعدك.
3- كن حذراً عند استخدام جملة On Error Resume. وجود هذه الجملة يجعل البرنامج يتجاوز الخطأ بدون أن يعطيك فرصة لملاحظة الخطأ. لذلك عند استخدام هذه الجملة تأكد مما تريده جيداً.
4- نسق الكود بشكل واضح عن طريق محاذاة كل مجموعة من الجمل ذات العلاقة مع بعضها البعض (لاحظ كيف تمت محاذاة الأوامر في الشكل 11-9)

5- أبق على كودك سهلاً قدر الإمكان واسلك دوماً أقصر الطرق.

```
Sub test()
    Dim i As Double
        For i = 1 To 1000
            x = i + (i - 1) ^ 2
        Next i
    MsgBox x
End Sub
```

Figure9-11

الفصل العاشر

10. الأحداث Events

احدى طرق تنفيذ الاجرائيات هي تنفيذها تلقائياً عند حدوث حدث معين. الحدث Event، هو ببساطة عبارة عن شيء حدث للاكسيل. بعض الأحداث هي كما يلي:

- فتح مصنف جديد
- حفظ المصنف
- اغلاق المصنف
- ادراج أو حذف ورقة عمل
- التعديل على ورقة عمل
- طباعة ورقة عمل
- دخول وقت معين
- حدوث خطأ معين

في هذا الفصل سوف نتعرف على الأحداث Events الأكثر شيوعاً في الاكسيل وكيفية الاستفادة منها في تنفيذ الاجرائيات. في الجدولين التاليين قائمة بالأحداث الأكثر شيوعاً. بالطبع هناك أحداث أكثر بكثير، إذا كنت تريد معرفة القائمة الكاملة فيمكنك أن تبحث في نظام المساعدة الخاص بمحرر الأكواد VBE أو المساعدة عبر الانترنت.

الحدث	**متى يتم تفعيله**
Activate	عند تنشيط المصنف
BeforeClose	عند اغلاق المصنف
BeforePrint	عند طباعة المصنف
BeforeSave	عند حفظ المصنف
Deactivate	عند إزالة التنشيط عن المصنف
NewSheet	عند إضافة ورقة عمل جديدة للمصنف
Open	عند فتح المصنف
SheetActivate	عند تنشيط ورقة عمل في المصنف
SheetBeforeDoubleClick	عند النقر نقراً مزدوجاً على احدى الخلايا في ورقة العمل
SheetBeforeRightClick	عند النقر بالزر الأيمن على احدى الخلايا في ورقة العمل
SheetChange	عند اجراء تغيير على احدى الخلايا
SheetDeactivate	عند الغاء التنشيط عن احدى أوراق العمل
SheetSelectionChange	عند اجراء تغيير على التحديد

الحدث	**متى يتم تفعيله**
Activate	عند تنشيط ورقة عمل
BeforeDoubleClick	عند النقر المزدوج على احدى الخلايا في ورقة عمل
BeforeRightClick	عند النقر بالزر الأيمن على احدى الخلايا في ورقة عمل
Change	عند اجراء تغيير على احدى الخلايا
Deactivate	عند إزالة التنشيط عن ورقة العمل
SelectionChange	عند اجراء تعديل على التحديد

ما هو معالج-الحدث Event-Handler

الكود الذي يتم تنفيذه كرد فعل على حدث معين يسمى معالج الحدث Event-Handler. عادة يكون عبارة عن إجرائية فرعية Sub-Procedure وليس Function.

لكتابة Event-Handler اتبع الخطوات التالية:

- حدد الحدث الذي تريد أن يتم تنفيذ الكود عند تحققه.
- في محرر الأكواد VBE اذهب إلى العنصر Object المناسب للحدث وانقر عليه نقراً مزدوجاً. تكون العناصر تحت Microsoft Excel Objects. للأحداث الخاصة بالمصنف انقر نقراً مزدوجاً على ThisWorkbook أما بالنسبة للأحداث الخاصة بأوراق العمل فانقر نقراً مزدوجاً على ورقة العمل المطلوبة (مثل Sheet1 مثلاً) وهكذا.
- في نافذة الكود الخاصة بالعنصر، اكتب معالج-الحدث المطلوب. في أعلى نافذة الكود هنا قائمتان منسدلتان:
  - قائمة العنصر Object drop-down list
  - قائمة الإجرائية Procedure drop-down list
- إذا كنت تريد كتابة Event-Handler للعنصر ThisWorkbook ، انقر نقراً مزدوجاً فوق ThisWorkbook، ثم من نافذة الكود اختر Workbook من القائمة المنسدلة Object drop-down list.
- بالمثل إذا كنت تريد كتابة Event-Handler لورقة عمل Sheet، انقر نقراً مزدوجاً فوق ورقة العمل المطلوبة ثم من نافذة الكود اختر Worksheet من القائمة المنسدلة Object drop-down list
- اختر الحدث المطلوب من قائمة Procedure drop-down list. الشكل 1-10 يعرض بعض الأحداث المتاحة لورقة العمل.
- عند اختيار الحدث من القائمة المنسدلة يقوم VBE تلقائياً بإدراج إجرائية معالجة الحدث Event-Handler procedure للحدث المطلوب.
- أدخل الكود المطلوب في إجرائية معالجة الحدث التي تم فتحها في الخطوة السابقة.

ملاحظة

عند اختيار Workbook من القائمة المنسدلة Object drop-list فإن الاكسيل يفترض أن إجرائية الـ Event-Handler هي للحدث Open وبالتالي يفتح الإجرائية Workbook_Open وهذا جيد اذا

كنت فعلاً تريد استخدام هذا الحدث. أما إذا لم يكن هذا الحدث المطلوب فعليك حذف إجرائية Workbook_Open. وبالمثل اذا اخترت Worksheet من القائمة Object فإن إجرائية معالجة الحدث الافتراضية سوف تكون Worksheet_SelectionChange فإذا لم تكن بحاجة لها فقم بحذفها.

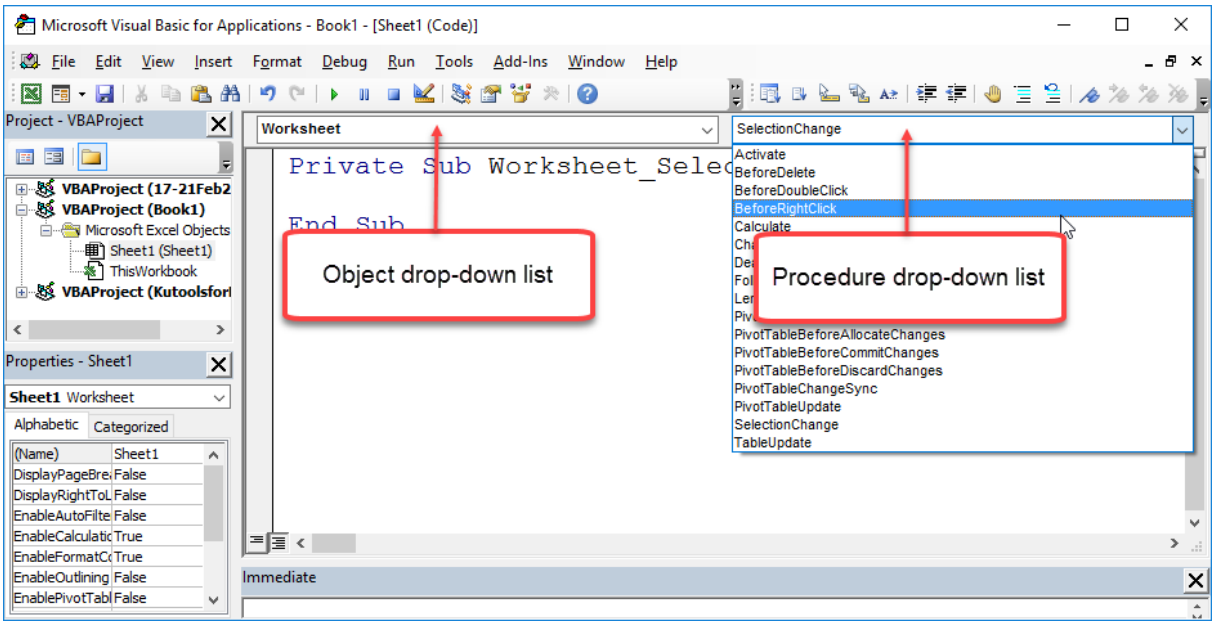


Figure10-1

يجب أن يوضع كل Event-Handler في مكانه الصحيح، فإذا تم مثلاً كتابة معالج حدث خاص بورقة العمل في نافذة الكود الخاصة بالمصنف فلن يعمل.

أمثلة على استخدام الأحداث لتنفيذ أوامر VBA

فيما يلي مجموعة من الأمثلة على أتمتة تنفيذ الاجرائيات من خلال الأحداث. سواء أكانت أحداث متعلقة بالمصنف أو بورقة العمل.

أمثلة على أحداث المصنف Workbook Events

الحدث Open Workbook

في هذا المثال مطلوب كتابة Event-Handler يقوم بعرض رسالة تذكيرية كل يوم ثلاثاء عند فتح ملف الاكسيل المحدد. لكتابة معالج الحدث هذا اتبع الخطوات التالية:

1- انقر نقراً مزدوجاً فوق ThisWorkbook الموجود ضمن مستكشف المشاريع على اليسار، لعرض نافذة الكود الخاصة بالمصنف.
2- من نافذة الكود، اختر Workbook من القائمة المنسدلة Object drop-down list و Workbook_Open من القائمة المنسدلة Procedure drop-down list.
3- اكتب الكود التالي في الإجرائية التي فتحها باسم Workbook_Open.

```
Private Sub Workbook_Open()
 If Weekday(Now) = 3 Then
    MsgBox "Today is Tuseday, Don't forget to submit sales report"
 End If
End Sub
```

في النهاية سيظهر الكود كما في الشكل 2-10

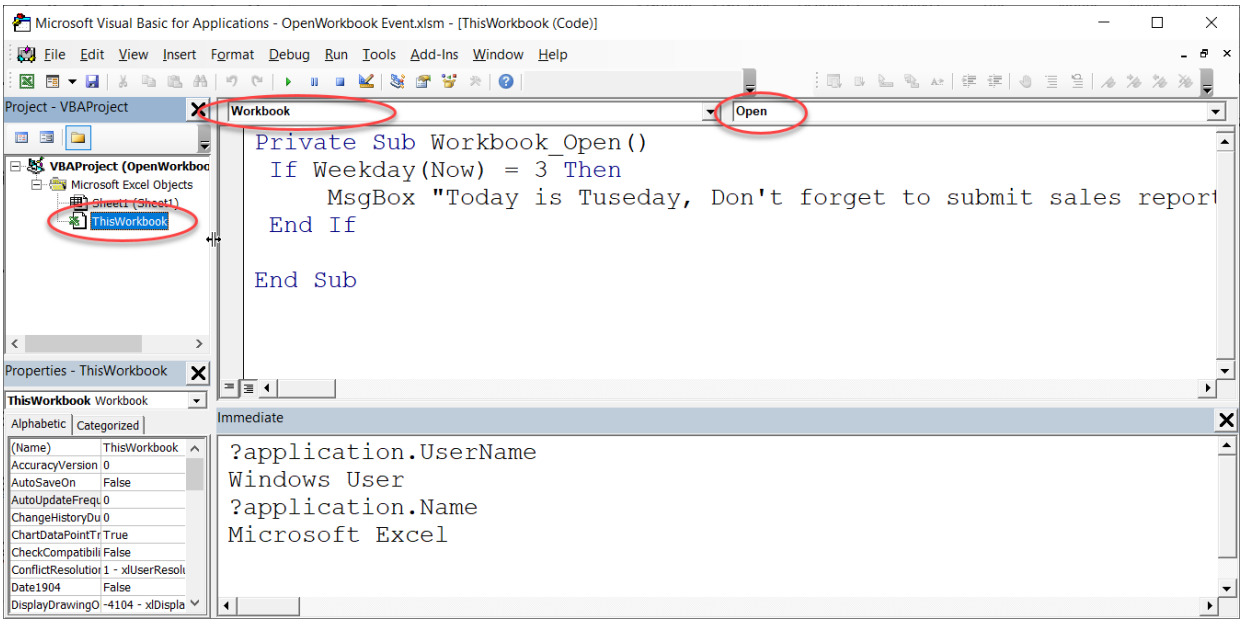


Figure10-2

الحدث Before Close

الكود التالي يقوم بفحص المصنف قبل اغلاقه، إذا كانت هناك تعديلات على المصنف بدون أن يكون قد تم حفظ المصنف، فإن معالج الحدث هذا يقوم بحفظ المصنف بدون اظهار أية رسائل. (ذلك يلغي السلوك الافتراضي للاكسيل والذي يقوم بعرض رسالة تفيد بأن هناك تعديلات لم يتم حفظها، هل ترغب بالحفظ أم لا؟)

انقر نقراً مزدوجاً فوق ThisWorkbook ثم من نافذة الكود اختر Workbook من قائمة Object و Workbook_BeforeClose من نافذة Procedure. وفي الإجرائية Workbook_BeforeClose اكتب التالي:

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)

    If ThisWorkbook.Saved = False Then ThisWorkbook.Save

End Sub
```

لاحظ أنك من الممكن أن تكب جملة If بالشكل التالي اختصاراً حيث تم استبدال ThisWorkbook بـ Me:

```
If Me.Saved = False Then Me.Save
```

مثال على الحدث Activate

الحدث Activate يحصل عند تنشيط المصنف. تقوم الإجرائية التالية بتكبير نافذة المصنف وعرض رسالة تحتوي على اسم المصنف. هذا الأمر مفيد إذا كنت تعمل على عدد كبير من المصنفات المفتوحة وكنت تخاف أن تختلط عليك الأمور فتجري تعديلاً على المصنف الخطأ.

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

الحدث Deactivate

الحدث Deactivate يحصل عند الغاء التنشيط عن المصنف. إجرائية معالجة الحدث التالية هي مثال على استخدام الحدث Deactivate. تقوم هذه الاجرائية بنسخ النطاق المحدد ووضعه في الحافظة وذلك عند الغاء التنشيط عن المصنف. وهذا الأمر مفيد إذا كان لديك مصنف تريد أن تنسخ مجموعة من النطاقات من أوراق العمل بداخله إلى مصنفات أخرى حيث أن تلك الإجرائية تسهل الأمر عليك فبدلاً من تحديد النطاق ثم اختيار الأمر نسخ Copy فيكفي اختيار النطاق فقط ثم الذهاب للمصنف الهدف واجراء عملية اللصق.

```
Private Sub Workbook_Deactivate()
   ThisWorkbook.Windows(1).RangeSelection.Copy
End Sub
```

مثال على الحدث New Sheet

الإجرائية التالية تقوم بنسخ ترويسة ورقة العمل الموجودة في ورقة العمل Sheet1 ضمن النطاق A1:I1 إلى أي ورقة عمل جديدة.

```
Private Sub Workbook_NewSheet(ByVal Sh As Object)
  Sheets("sheet1").Range("A1:I1").Copy
  Sh.Paste
End Sub
```

مثال على الحدث Before Save

الإجرائية التالية تقوم بتسجيل وقت وتاريخ آخر مرة تم فيها حفظ المصنف في الخلية A1 من ورقة العمل Sheet1.

```
Private Sub Workbook_BeforeSave(ByVal SaveAsUI As Boolean, _
Cancel As Boolean)
  Worksheets("Sheet1").Range("A1") = Format(Now, _
  "dd-mm-yyyy hh:mm:ss")
End Sub
```

بعض الأمثلة على أحداث أوراق العمل Worksheets Events

الحدث Activate

يحص هذا الحدث عند تنشيط ورقة العمل. كمثال تقوم إجرائية معالجة الحدث التالية بتنشيط الخلية A1 لورقة العمل المحددة في كل مرة يتم فيها تنشيط ورقة العمل.

لكتابة هذه الإجرائية (أو أي إجرائية أخري تتعامل مع أحداث ورقة العمل) اتبع الإجراءات التالية:

1- انقر نقراً مزدوجاً على ورقة العمل المطلوبة من مستكشف المشاريع
2- من نافذة الكود اختر Worksheet من القائمة المنسدلة Object drop-down list و Worksheet_Activate من Procedure drop-down list.
3- أدخل الكود كما بالأسفل.

```
Private Sub Worksheet_Activate()
   Range("A1").Activate
End Sub
```

الحدث Deactivate

يحصل هذا الحدث عند الغاء التنشيط عن ورقة العمل. كمثال يقوم الكود بالأسفل بفحص محتويات الخلية A1 عند الغاء التنشيط عن ورقة العمل Sheet1، فإذا كانت فارغة يعرض رسالة خطأ تذكر المستخدم بأن يدخل قيمة في تلك الخلية ويمنع المستخدم من الانتقال لورقة عمل أخرى قبل ادخال قيمة في الخلية A1 في الورقة Sheet1.

```
Private Sub Worksheet_Deactivate()
   If Len(Me.Range("A1").Value) = 0 Then
      MsgBox "Reminder!! Please enter a value in A1 "
```

```
    Sheets("sheet1").Activate
  End If
End Sub
```

الحدث Select Change

يحصل الحدث Select_Change عند تغير الخلايا المحددة. كمثال على هذا الحدث تقوم الإجرائية بالأسفل بتلوين صف وعمود الخلية النشطة لتسهيل عملية قراءة البيانات. يكون نتيجة تنفيذ هذه الإجرائية كما في الشكل 3-10.

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
  Cells.Interior.ColorIndex = xlNone
  With ActiveCell
  .EntireRow.Interior.Color = RGB(248, 203, 173)
  .EntireColumn.Interior.Color = RGB(180, 198, 231)
  End With
End Sub
```

تبدأ الإجرائية بإزالة اللون عن كافة خلايا ورقة العمل من خلال السطر:

```
Cells.Interior.ColorIndex = xlNone
```

ثم تقوم بتلوين صف وعمود الخلية النشطة من خلال كتلة الأوامر Command Block:

```
With ActiveCell
  .EntireRow.Interior.Color = RGB(248, 203, 173)
  .EntireColumn.Interior.Color = RGB(180, 198, 231)
  End With
```

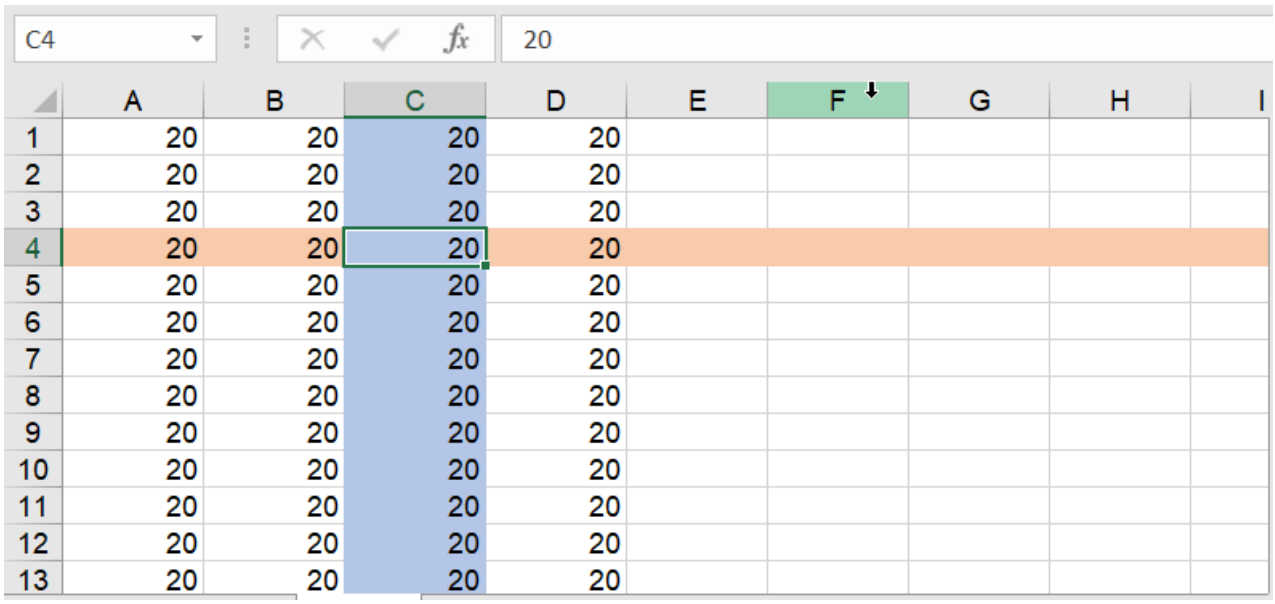


Figure10-3

الحدث Worksheet Calculate

يحصل هذا الحدث عند اجراء احتساب لورقة العمل Worksheet Calulate سواءً أكان الاحتساب تلقائياً أو يدوياً. في الكود بالأسفل يتم عمل احتواء تلقائي للأعمدة A:F في كل مرة يتم فيها إعادة احتساب ورقة العمل.

```
Private Sub Worksheet_Calculate()
    Columns("A:F").AutoFit
End Sub
```

الحدث Worksheet Change

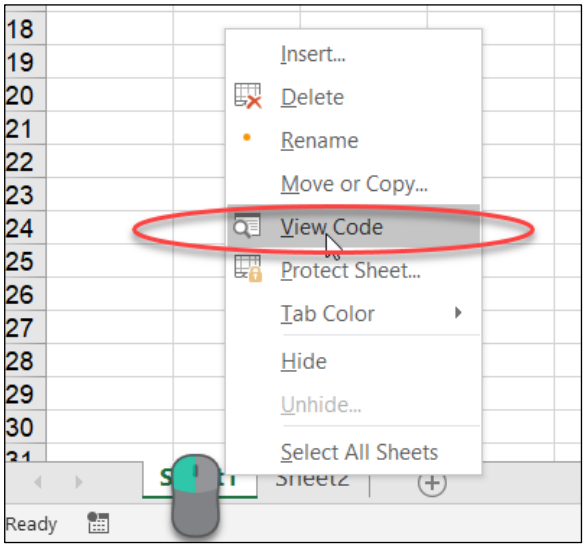
يحصل هذا الحدث عند حدوث تغيير على خلايا ورقة العمل. كمثال على استخدام هذا الحدث؛ تقوم الإجرائية بالأسفل بإدراج وقت وتاريخ تعديل أي خلية من خلايا العمود B وذلك في الخلية المجاورة (التي تقع في العمود C)

```
Private Sub Worksheet_Change(ByVal Target As Range)
  If Target.Column <> 2 Then Exit Sub
  Target.Offset(0, 1).Value = Format(Now, "MM / DD / YYYY hh:mm")
End Sub
```

الانتقال إلى الكود الخاص بورقة عمل من خلال اكسيل

للانتقال بشكل مباشر إلى الكود الخاص بورقة عمل معينة من خلال الاكسيل (وليس VBE) انقر بزر الماوس الأيمن فوق ورقة العمل ثم اختر View Code كما في الشكل 4-10

Figure10-4


18
19
20
21
22
23
24
25
26
27
28
29
30
Insert...
Delete
Rename
Move or Copy...
View Code
Protect Sheet...
Tab Color
Hide
Unhide...
Select All Sheets
Sheet2
Ready

الفصل الحادي عشر

11. التفاعل مع المستخدم من خلال مربعات الحوار

في هذا الفصل سيتم التعرض لحالات مختلفة لكل من دالة MsgBox التي تقوم بعرض البيانات ضمن مربع حوار مع إمكانية أخذ تغذية راجعة من المستخدم، ودالة InputBox التي تستخدم لإدخال البيانات من خلال مربع حوار. إضافة إلى ذلك فسيتم التطرق لدالة Application.InputBox والتي تُستخدم لإدخال أشياء مثل المعادلات أو مراجع النطاقات.

الدالة MsgBox

تقوم الدالة MsgBox بعرض مربع حوار بسيط يمكن استخدامه لعرض معلومات للمستخدم أو أخذ تغذية راجعة منه مثل أن يضغط المستخدم على زر موافق أو الغاء الأمر.

تأخذ هذه الدالة الشكل التالي:

```
MsgBox(prompt[, buttons][, title])
```

حيث أن:

- Prompt: عبارة عن نص الرسالة الذي سوف يظهر على مربع الحوار. وهو معامل اجباري.
- Buttons: هذا المعامل يوضح نوع الأزرار التي ترغب بإظهارها على مربع الحوار بالإضافة إلى أنه يمكن استخدامه أيضاً لعرض احدى الأيقونات التي قد تساعد في توضيح الهدف من مربع الحوار (مثل أيقونة التحذير أو الإعلام).

- Title: عنوان مربع الحوار.

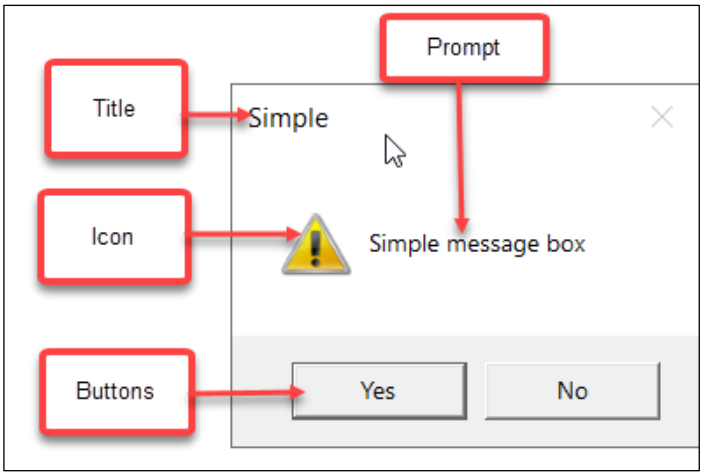


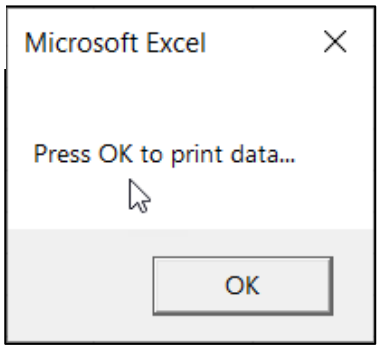
Figure11-1

عرض مربع حوار بسيط

المثال التالي يقوم بعرض مربع حوار بسيط يقوم بعرض رسالة تطلب من المستخدم أن يضغط على زر OK للبدء بطباعة نطاق البيانات المسمى data. لاحظ أن سير البرنامج يتوقف لحين الاستجابة لمربع الحوار.

```
Sub test()
   MsgBox "Press OK to print data..."
   Range("data").PrintOut
End Sub
```

Figure11-2



أخذ استجابة من مربع حوار

إذا كان مربع الحوار يحتوي على ما هو اكثر من زر OK فيمكنك أن تحدد مسار البرنامج بناءً على الزر الذي قام المستخدم بضغطه. تقوم الإجرائية التالية بعرض مربع حوار مع زري OK و Cancel بحيث إذا تم الضغط على OK يتم طباعة البيانات أما إذا تم الضغط على Cancel فيتم الغاء العملية ويتم عرض رسالة تفيد بالإلغاء.

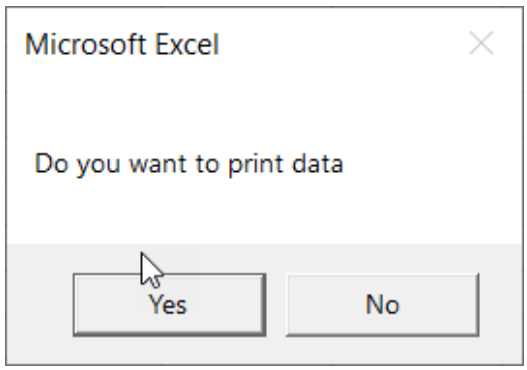
```
Sub test_msgbox()
   Dim Ans As Integer
   Ans = MsgBox ("Do you want to print data", vbYesNo)
   Select Case Ans
      Case vbYes
      Sheets("sheet1").Range("data").PrintOut
      Case vbNo
      MsgBox "Printing canceled"
   End Select
End Sub
```

لاحظ أننا في هذا المثال استخدمنا المعامل الثاني (وهو اختياري) لدالة MsgBox والذي من خلاله تم تحديد نوع الأزرار التي نرغب بإظهارها على مربع الحوار (في هذا لمثال زري Yes و No). ثم استخدمنا جملة Select لتنفيذ الأوامر بناء على الزر الذي سوف يتم الضغط عليه.

ولاحظ هنا أننا استخدمنا متغير (Ans) لتخزين نتيجة الضغط على أحد الأزرار فيه، قبل استخدام هذا المتغير في جملة Select. ومن الممكن عدم استخدام المتغير Ans كما في المثال التالي:

```
Sub test_msgbox()
        If MsgBox("Do you want to print data", vbYesNo)= vbYes Then
        ...[code if Yes is clicked]...
        Else
         ...[code if Yes is not clicked]...
        End If
End Sub
```

Figure11-3



في الجدول التالي القائمة الكاملة للخيارات التي يمكن استخدامها في المعامل الثاني لدالة MsgBox (Buttons). يمكنك استخدام الكلمة مثل vbOKCancel أو القيمة الرقمية المكافئة كما في الجدول.

الكلمة	القيمة	ماذا تفعل
vbOKOnly	0	عرض زر OK فقط
vbOKCancel	1	عرض زري OK و Cancle
vbAbortRetryIgnore	2	عرض الأزرار Abort, Retry, and Ignore
vbYesNoCancel	3	عرض الأزرار Yes, NO, and Cancle
vbYesNo	4	عرض زري Yes، NO
vbRetryCancel	5	عرض زري Retry, Cancle
vbCritical	16	عرض أيقونة الخطر Critical
vbQuestion	32	عرض أيقونة الاستفهام Query
vbExclamation	48	عرض أيقونة التحذير Warning
vbInformation	64	عرض أيقونة المعلومات Information
vbDefaultButton1	0	الزر الأول هو الزر الافتراضي
vbDefaultButton2	256	الزر الثاني هو الزر الافتراضي
vbDefaultButton3	512	الزر الثالث هو الزر الافتراضي
vbDefaultButton4	768	الزر الرابع هو الزر الافتراضي

لاستخدام أكثر من قيمة للمعامل مثل أن تعرض الأزرار OK, Cancel مع علامة المعلومات Information يمكنك استخدام إشارة +:

```
vbYesNO + Information
```

في المثال السابق إذا تم تغيير الجملة

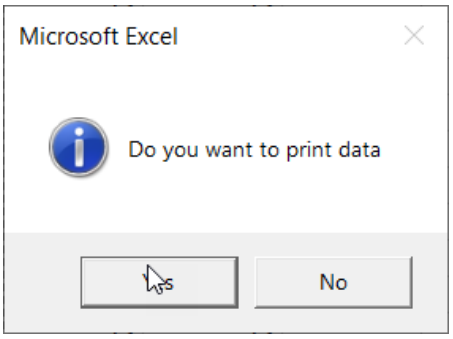
```
Ans = MsgBox("Do you want to print data", vbYesNo)
```

إلى

```
Ans = MsgBox("Do you want to print data", vbYesNo + vbInformation)
```

فسيظهر لنا مربع الحوار كما في الشكل 4-11 . لاحظ علامة المعلومات على اليسار.

Figure11-4



مربع الادخال InputBox

دالة InputBox هي من الدوال المفيدة التي يمكن استخدامها لاستقبال ادخال معين من قبل المستخدم. تأخذ الدالة الشكل العام التالي:

```
InputBox(prompt[, title][, default])
```

حيث أن:

- Prompt: هي الرسالة التي سوف تظهر على مربع الحوار. عادة ما تحتوي على توجيهات للمستخدم حول طبيعة الادخال المتوقع. وهي قيمة اجبارية
- Title: عنوان مربع الحوار. وهي قيمة اختيارية.
- Default: النص الافتراضي للإدخال. وهي قيمة اختيارية.

مثال على دالة InputBox

تستخدم الإجرائية التالية دالة InputBox لأخذ الاسم من المستخدم ثم تقوم بعرض جملة ترحيبية باسم المستخدم من خلال مربع MsgBox.

```
Sub test()
  Dim name As String
```

```
  name = InputBox("Please enter your name...", "Name")
  MsgBox "Helo " & name
End Sub
```

تم تعريف متغير من نوع String باسم name لتخزين قيمة الادخال فيه. عند تنفيذ الإجرائية سيظهر لنا مربع الادخال كما بالشكل 11-5

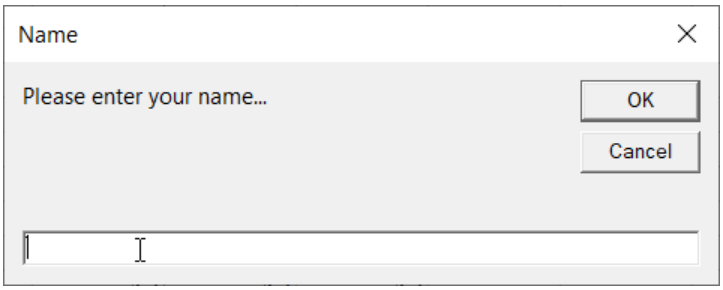
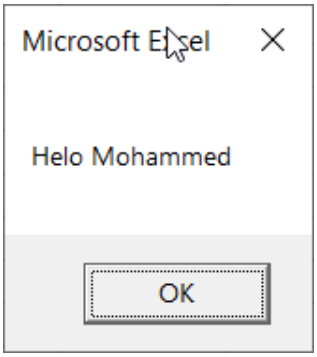


Figure11-5

إذا أدخلنا الاسم Mohammed فإن البرنامج سيقوم بعرض الرسالة الترحيبية كما بالأسفل

Figure11-6



الدالة Application.InputBox method

يوجد لعنصر البرنامج Application object وظيفة Method تسمى InputBox (بخلاف الدالة InputBox) وهي تظهر مربع حوار شبيه بمربع الحوار الخاص بالدالة InputBox إلا أنها أكثر مرونة منها حيث أنه من الممكن استخدام هذه الـ Method لإدخال أشياء مثل المصفوفات أو المعادلات إلا أن أكثر فائدة لهذه الـ Method هو أنه يمكن استخدامها لتحديد نطاق من البيانات.

تأخذ هذه الـ Method الشكل العام التالي:

```
Application.  InputBox (_Prompt_, _Title_, _Default_,
      _Left_,_Top_,_HelpFile_,_HelpContextID_,_Type_)
```

الجدول التالي يوضح معاملات الدالة:

الاسم	مطلوب/ اختياري	الوصف
Prompt	مطلوب	**الرسالة التي سوف تعرض على مربع الحوار**
Title	اختياري	**عنوان مربع الحوار**
Default	اختياري	**القيمة الافتراضية**
Left	اختياري	**موقع مربع الحوار بالنسبة لشاشة الاكسيل**
Top	اختياري	**موقع مربع الحوار بالنسبة لشاشة الاكسيل**
HelpFile	اختياري	**اسم ملف المساعدة**
HelpContextID	اختياري	**الرقم المرجعي للعنوان الذي تريد مساعدة بالنسبة له.**
Type	اختياري	**نوع البيانات المدخلة. إذا لم يكن هذا الخيار موجوداً فإن نوع البيانات الافتراضي هو نص Text**

الجدول التالي يوضح القيم التي يمكن استخدامها مع المعامل Type:

القيمة	ما تعنيه
0	صيغة Formula
1	رقم
2	نص
4	قيمة منطقية True or False
8	مرجع خلايا (كعنصر نطاق Range object)
16	قيمة خطأ (مثل #N/A)
64	مصفوفة

مثال

يقوم البرنامج بالأسفل باستخدام Application.InputBox method بعرض مربع حوار يقوم المستخدم من خلاله بتحديد خلية ومن ثم يتم فحص قيمة الخلية فإذا كانت أكبر من صفر يتم تلوين الخلية باللون الأخضر.

لاحظ هنا أننا استخدمنا الكلمة المفتاحية Set لتعيين قيمة للمتغير mycell حيث أن هذا المتغير يتم استخدامه هنا ليحتوي على عنصر نطاق Range (من المعروف أنه لتعيين قيمة لمتغير من نوع نطاق يجب استخدام الكلمة المفتاحية Set. راجع الفصل الرابع)

```
Sub test_cell()
   Worksheets("Sheet1").Activate
   Set mycell = Application.InputBox( prompt:="Select a cell", Type:=8)
   If mycell.Value > 0 Then mycell.Interior.Color = vbGreen
End Sub
```

الفصل الثاني عشر

12. النماذج UserForms

تعد النماذج من الوسائل الجيدة لإدخال البيانات وعرضها. حيث أنك من خلالها تستطيع التحكم بكيفية ادخال البيانات من حيث ترتيب ادخال البيانات أو اختيار الوسيلة المناسبة لكل نوع من البيانات المطلوبة أو اجراء عملية التحقق من الادخال للبيانات المدخلة وغيرها من الأشياء. ويمكنك أيضاً التحكم بكيفية عرض البيانات بالشكل الذي يلائم طبيعة البيانات المعروضة مما يعطيك مرونة عالية جداً في التعامل مع البيانات.

يتم ادراج النماذج مع عناصر التحكم الخاصة بها والكود المرتبط بالنموذج والعناصر عليه من خلال محرر الأكواد VBE.

انشاء نموذج المستخدم

الخطوة الأولى في عملية انشاء نموذج المستخدم UserForm هي ادراج نموذج من خلال محرر الأكواد من خلال النقر بزر الماوس الأيمن على اسم المصنف الذي نريد انشاء النموذج بداخله ثم اختيار Insert ➔ UserForm أو من خلال تحديد اسم المصنف ثم الضغط على زر Insert UserForm على شريط الأدوات.

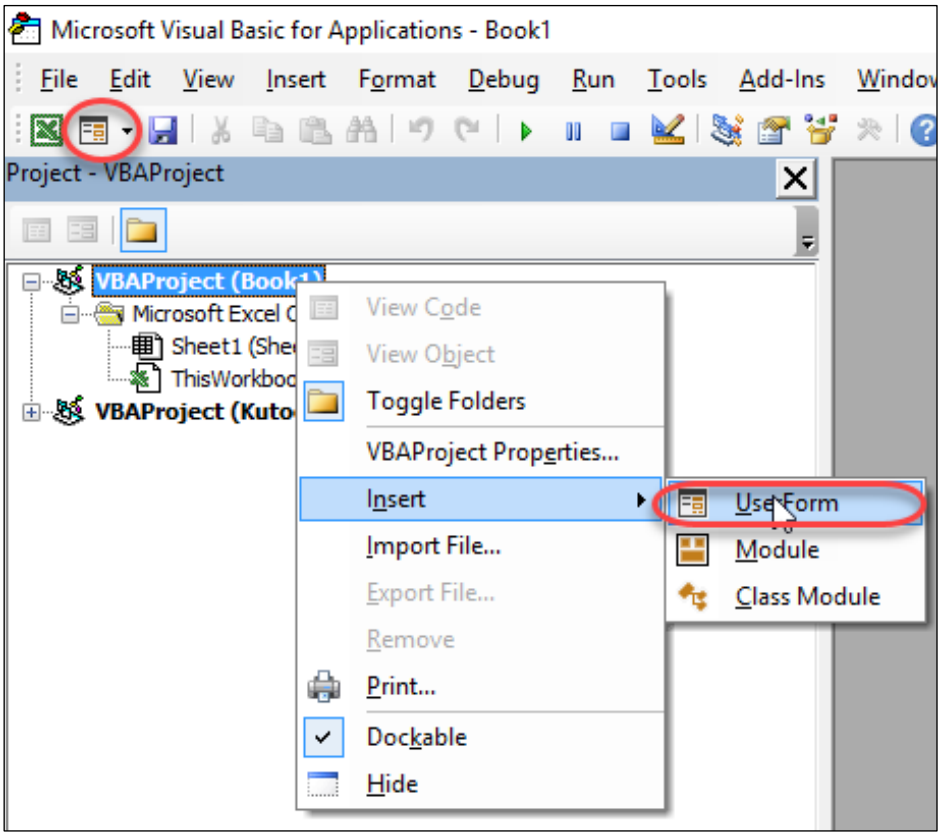


Figure12-1

سوف يتم ادراج نموذج Form كما في الشكل 12-1 . بعد أن يتم ادراج النموذج من الممكن أن تتحكم بخصائصه من خلال الجزء الخاص بخصائص النموذج UserForm properties فمثلا يمكن تغيير لون الخلفية من خلال تعديل الخاصية BackColor ويمكن تغيير عنوان النموذج من خلال الخاصية Caption. يتم تعديل الخصائص من خلال الضغط على الحقل الموجود على يمين اسم الخاصية فيتم فتح قائمة منسدلة (في أغلب الأحيان) نختار منها الاعداد المطلوب، وإذا لم تكن هناك قائمة منسدلة (كما في خاصية اسم النموذج) نكتب قيمة الاعداد المطلوب. اذا لم تكن نافذة الخصائص ظاهر فيمكن اظهارها من خلال الذهاب إلى قائمة View ثم اختيار Properties Window.

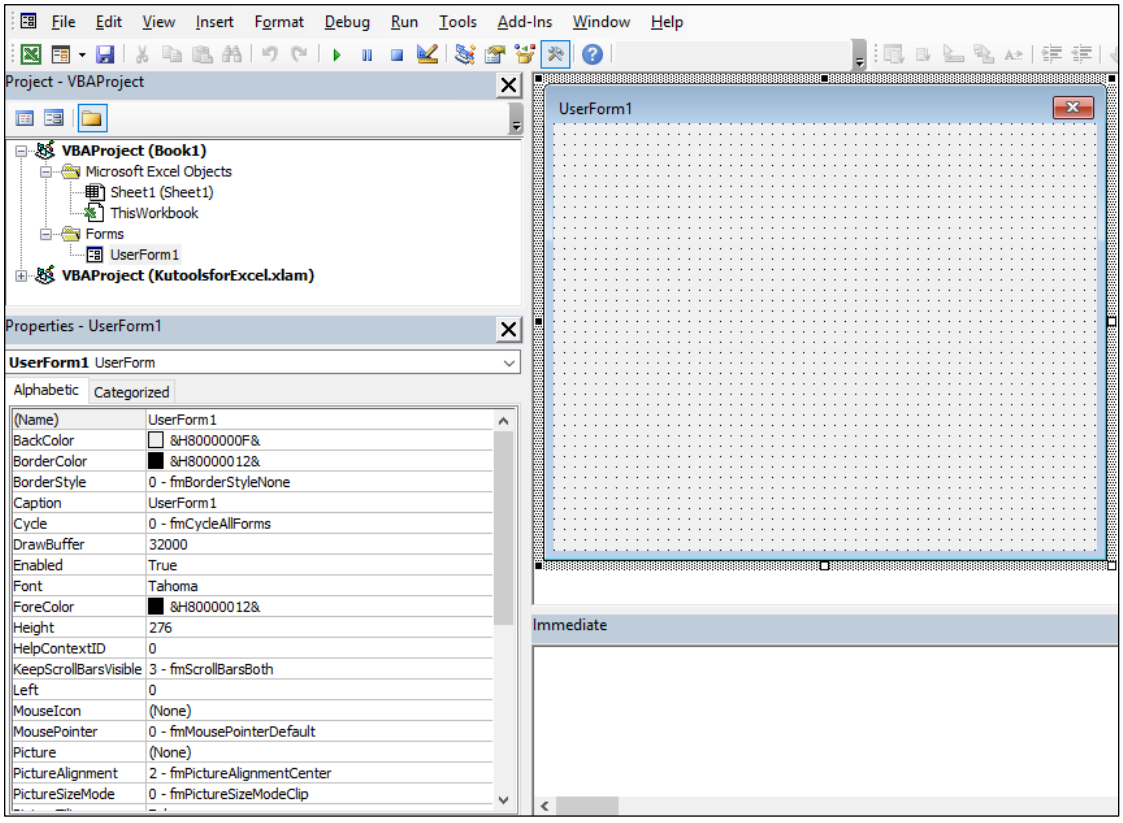


Figure12-3

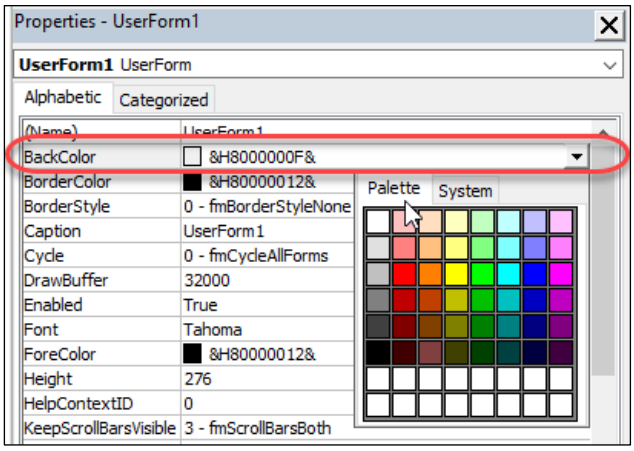


Figure12-2

إضافة عناصر تحكم للنموذج

عنصر التحكم Control هو عبارة عن عنصر يوضع على النموذج (مثل الأزرار أو مربع النص أو القوائم المنسدلة) يتم من خلاله ادخال أو عرض البيانات. يتم ادراج عناصر التحكم من خلال صندوق الأدوات Toolbox والذي يظهر عادة بمجرد ادراج النموذج. إذا لم يكن صندوق الأدوات ظاهراً فيمكن ادراجه من خلال الذهاب إلى قائمة View ثم اختيار Toolbox.

لإضافة عنصر تحكم على النموذج اختر العنصر المطلوب ثم أدرجه في المكان المطلوب على النموذج. يمكنك بعد ادراج عنصر التحكم، التحكم بحجمه أو مكانه باستخدام مهارات الحاسوب الأساسية مثل السحب والإفلات أو غيرها.

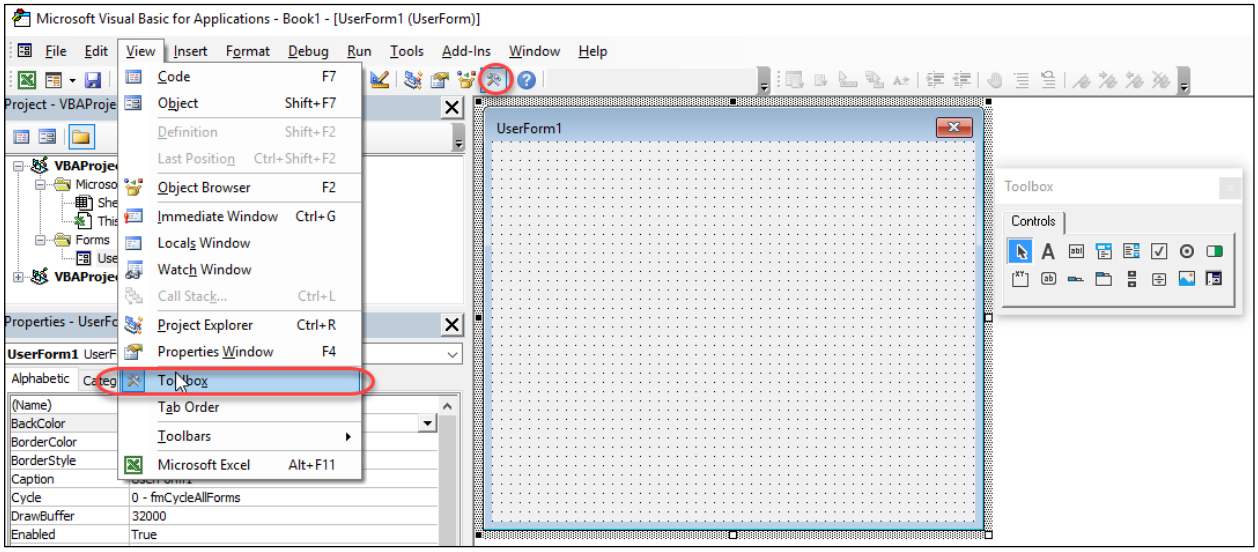


Figure12-4

الجدول التالي يوضح عناصر التحكم الشائعة ووظيفة كل منها

عنصر التحكم	وظيفته
التسمية التوضيحية Label	لاظهار نص.
مربع النص TextBox	لادخال أو عرض نص.
ComboBox	يعرض مجموعة من القيم على شكل قائمة منسدلة للاختيار من بينها. ويمكن إدخال قيمة جديدة غير موجودة من ضمن القيم التي تعرضها القائمة المنسدلة.
القائمة المنسدلة ListBox	تعرض مجموعة من القيم على شكل قائمة منسدلة للاختيار من بينها.
مربع الاختيار CheckBox	يستخدم للاختيار ما بين on/off أو نعم/لا
زر الاختيار OptionButton	يستخدم ضمن مجموعات من زرين أو أكثر حيث يسمح للمستخدم باختيار زر واحد فقط في المرة الواحدة.
زر التبديل ToggleButton	زر تشغيل/إيقاف
الاطار Frame	حاوية لعناصر أخرى
زر الأوامر CommandButton	زر قابل للضغط
شريط الألسنة TabStrip	يعرض مجموعة من ألسنة التبويب التي يمكن التنقل ما بينها
شريط التمرير ScrollBar	شريط قابل للتمرير
صورة Image	لعرض الصور
عنصر RefEdit	يسمح للمستخدم بإدخال نطاق Range

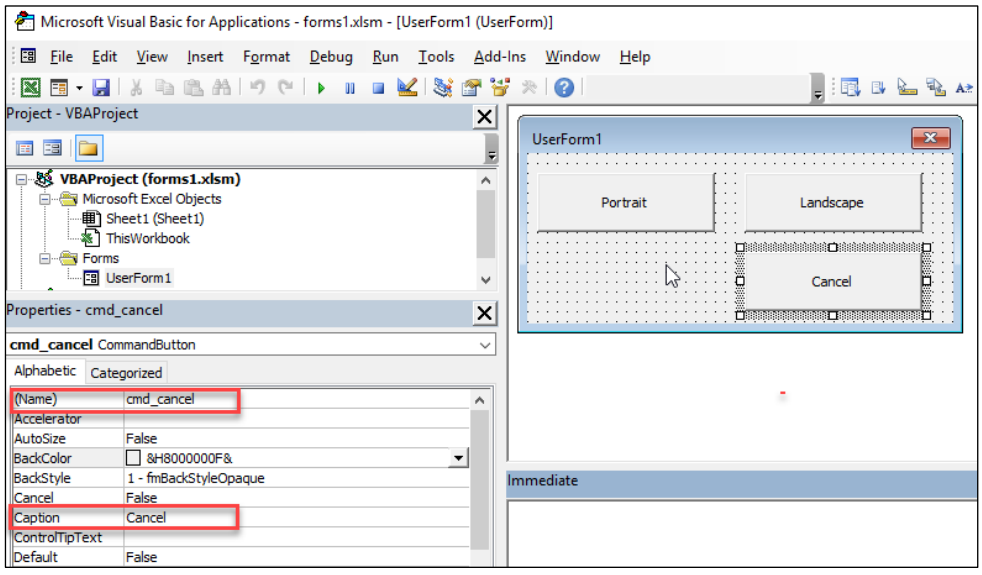
عنصر التحكم زر الأوامر Command Button

نستطيع من خلال هذا العنصر تنفيذ مجموعة من الأوامر عند الضغط عليه. لإدراج زر أمر، اختر زر الأمر من صندوق الأدوات ثم أدرجه في المكان المناسب على النموذج Form. عدل الحجم بالشكل المناسب ثم غير أي مواصفات أخرى للزر على سبيل المثال من الأمور الجيدة أن تعطي كل زر اسم خاص به Name و عنوان Caption وذلك من خلال تعديل خصائص الزر من جزء الخصائص Properties part. بعد ذلك يمكنك إضافة الكود المطلوب للزر.

كمثال على استخدام أزرار الأوامر افترض أنك تريد انشاء برنامج بسيط يقوم بإجراء عملية طباعة لورقة العمل مع إعطاء المستخدم الخيار للطباعة الطولية أو العرضية. ابدأ بإدراج نموذج فارغ ثم أضف له ثلاث أزرار كما في الشكل 12-5. من جزء الخصائص الخاص بكل زر عدل خصائص الأزرار كالتالي (لاحظ أنه من الأمور المُوصى بها أن تبدأ أسماء أزرار الأوامر بالبادئة cmd_، ولكن هذا الأمر ليس ملزما ويمكنك إعطاء أي اسم للزر بدون مشاكل):

عنوان الزر Caption	اسم الزر Name
Portrait	cmd_portrait
Landscape	cmd_landscape
Cancel	cmd_cancel

Figure12-5



اظهار النموذج Show UserForm

لإظهار نموذج ما يمكنك كتابة جملة برمجية تستخدم أمر Show مع اسم النموذج المطلوب. فمثلاً إذا أردنا أن نكتب إجرائية تقوم بعرض النموذج السابق (على أساس أن اسم النموذج UserForm1) فمن الممكن استخدام الكود التالي:

```
Sub print_from()
   UserForm1.Show
End Sub
```

عند تنفيذ هذه الإجرائية تقوم بعرض النموذج كما في الشكل 6-12.

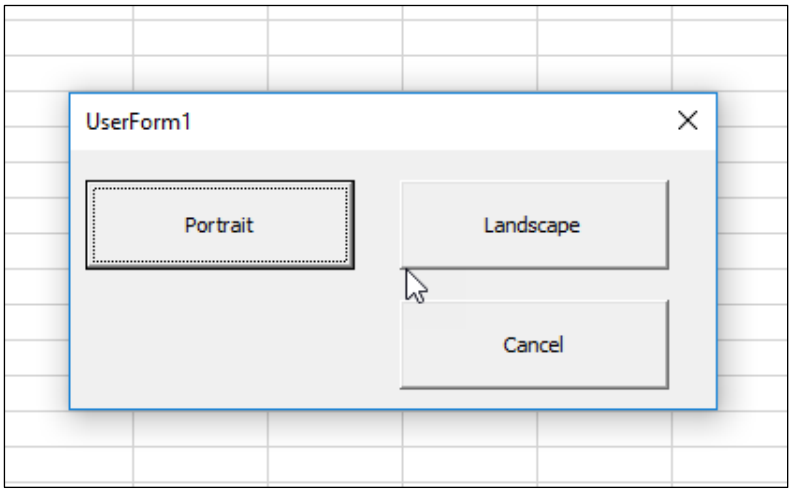


Figure12-6

حتى هذه اللحظة لا يقوم النموذج بإجراء أي عملية فعند الضغط على أي من الأزرار عليه لا يحدث شيء حيث أنه لم يتم بعد كتابة أي كود خاص بالنموذج أو العناصر الموجودة عليه. في الفقرات التالية سوف نتعرف على كيفية إضافة الأكواد الخاصة بالنماذج.

كتابة الأكواد الخاصة بالنماذج UserForms Codes

لعرض الكود الخاص بنموذج ما انقر بزر الماوس الأيمن فوق اسم النموذج في مستكشف المشاريع ثم اختر View Code. يمكنك أيضاً أن تنقر نقراً مزدوجاً فوق النموذج لعرض الكود الخاص به. وبالمثل، لعرض الكود الخاص بأحد العناصر على النموذج انقر نقراً مزدوجاً عليه أو انقر عليه بزر الماوس الأيمن ثم اختر View Code.

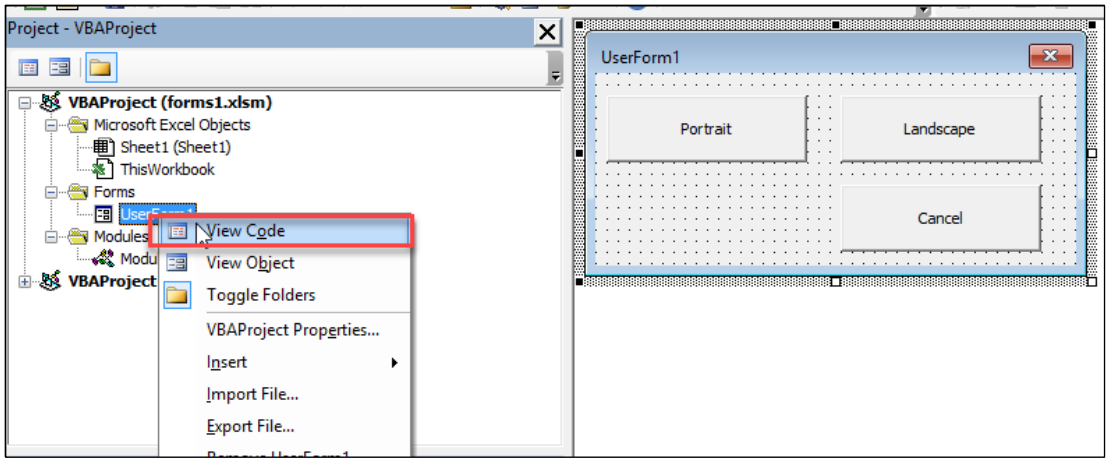


Figure12-7

اغلاق النموذج Unloading a Form

لإغلاق نموذج ما استخدم الكود Unload me. وأنسب مكان لوضع هذا الكود هو ارفاقه مع زر Cancel عن طريق التالي:

- انقر نقراً مزدوجاً فوق زر Cancel (أو انقر بالزر الأيمن واختر View Code). سوف يظهر لك الكود التالي:

```
Private Sub cmd_cancel_Click()

End Sub
```

- أدخل السطر Unload me بداخل الإجرائية أعلاه كما يلي:

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

الآن عند الضغط على زر Cancel فسيتم اغلاق النموذج ومحو بياناته من ذاكرة الحاسوب.

إخفاء النموذج Hiding a form

تقوم الـ Hide method بجعل النموذج غير مرئي. مع الاحتفاظ بالبيانات الموجودة فيه. ويمكن في أي لحظة اظهار النموذج المخفي من خلال Show method. إخفاء النموذج قد يكون مفيداً في بعض الحالات، أحد الأمثلة على ذلك هو عندما تريد أن تظهر نموذجاً فرعياً من آخر أساسي فيمكنك أن تخفي النموذج الأساسي كي تجعل المستخدم يركز على النموذج الفرعي وبعد ادخال بيانات الفرعي يتم اظهار النموذج الأساسي مرة أخرى.

لإخفاء النموذج UserForm1 مثلاً، قم بإدخال الكود UserForm1.Hide في المكان المناسب (حسب تسلسل سير البرنامج).

تذكر !! الفرق بين Unload و Hide هو أن VBA يقوم في الحالة الأولى بإغلاق النموذج ومحو بياناته من الذاكرة بينما في الحالة الثانية يقوم بإخفاء النموذج مع الإبقاء على البيانات بداخله كما هي.

كتابة الأكواد الخاصة بأزرار الأوامر

لكتابة الكود الخاص بزر الأمر Portrait انقر نقراً مزدوجاً فوقه فتظهر لك إجرائية الحدث الخاصة بالنقر على الزر كما يلي:

```
Private Sub cmdPortrait_Click()
End Sub
```

أدخل الكود التالي بداخل الإجرائية كما يلي:

```
Private Sub cmdPortrait_Click()
Unload Me
With ActiveSheet
        .PageSetup.Orientation = xlPortrait
        .PrintPreview
End With
End Sub
```

لاحظ أننا في هذه الإجرائية استخدمنا With.....End With لتبسيط الكود ومنع التكرار. يقوم الكود أولاً بتغيير اتجاه الورقة إلى الاتجاه الطولي Portrait من خلال الكود:

```
        .PageSetup.Orientation = xlPortrait
```

ثم عرض معاينة قبل الطباعة لورقة العمل من خلال السطر البرمجي:

```
        .PrintPreview
```

لاحظ أننا في بداية الإجرائية أدخلنا السطر البرمجي Unload Me لإغلاق النموذج بعد الضغط على زر Portrait. بدون هذا السطر سوف يبقى النموذج ظاهراً ولن يكون بإمكانك اغلاقه حيث أن الاكسيل سوف يكون ينتظر منك أن تتعامل مع نافذة المعاينة قبل الطباعة، في نفس الوقت لن يمكنك التعامل مع نافذة معاينة قبل الطباعة قبل اغلاق النموذج وبذلك تكون قد دخلت بداخل حلقة مفرغة. لذلك يجب ادخال Unload Me في بداية الإجرائية.

لكتابة الكود الخاص بزر الأمر Landscape انقر عليه نقراً مزدوجاً ثم أدخل الكود التالي:

```
Private Sub cmd_portrait_Click()
  Unload Me
  With ActiveSheet
        .PageSetup.Orientation = xlPortrait
        .PrintPreview
  End With
End Sub
```

أصبح لدينا الآن الكود جاهز لجميع العناصر. ولتشغيل البرنامج يمكن استدعاء الإجرائية print_form التي تم انشائها سابقاً وهي كالتالي:

```
Sub print_from()
  UserForm1.Show
End Sub
```

من الممكن إضافة اختصار للبرنامج إلى شريط الأدوات Ribbon من خلال تخصيص الشريط كما يلي:

1- انقر بزر الماوس الأيمن فوق الشريط Ribbon ثم اختر Customize the ribbon.
2- أضف مجموعة جديدة لأحد ألسنة التبويب الموجودة عن طريق اختيار لسان التبويب المطلوب ثم الضغط على زر New Group. مثلاً إذا أردت إضافة مجموعة جديدة لتبويب "الصفحة الرئيسية Home" فحدد Home ثم انقر على New Group. بعد الإضافة يمكن اعادة تسمية المجموعة عن طريق تحديدها ثم اختيار Rename.

3- أضف البرنامج المطلوب للشريط عن طريق اختيار Macros من القائمة المنسدلة Choose commands from ثم تحديد اسم الإجرائية المطلوبة واضافتها للمجموعة كما في الشكل 12-8. وبالطبع يمكنك إعادة تسمية الأيقونة على شريط الأدوات واعطائها شكل مخصص.

للمزيد حول تخصيص شريط الأدوات راجع الفصل الأول من كتاب "اكسيل 2019 الدليل السهل"

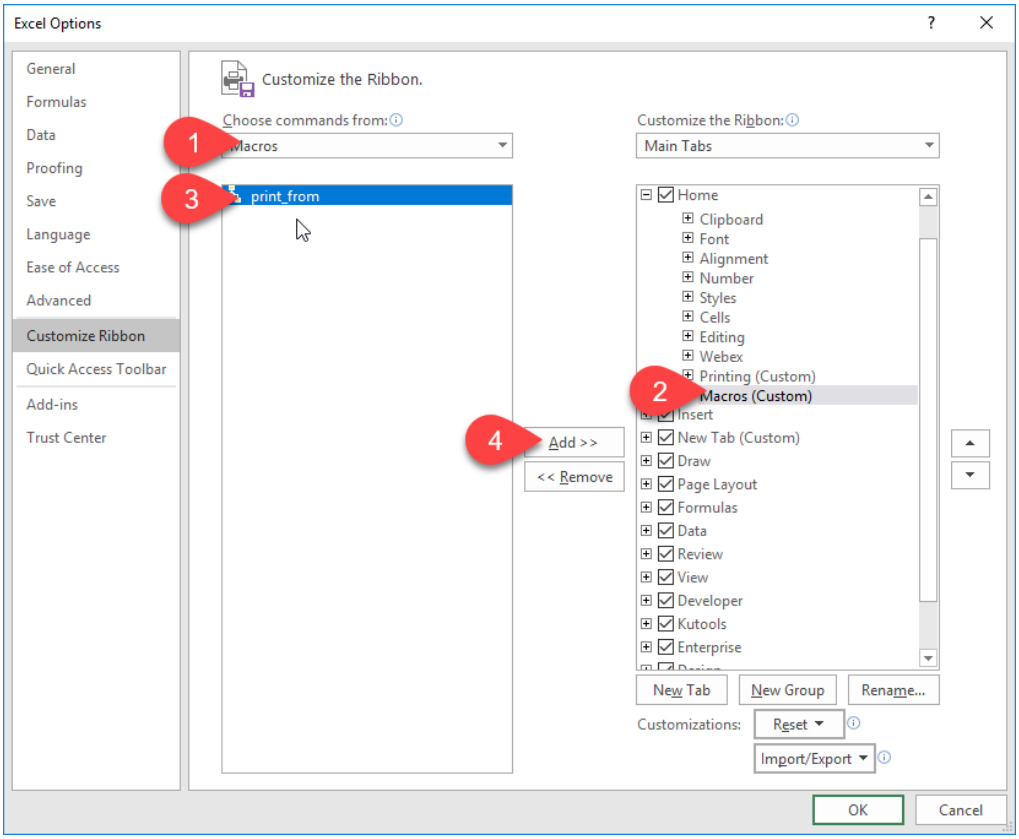
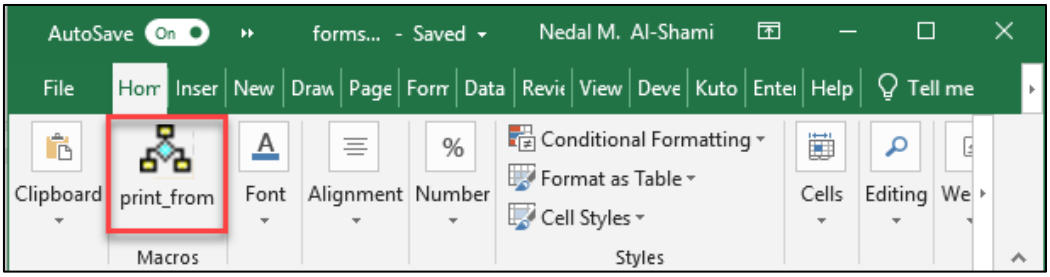


Figure12-8

Figure12-9



عنصر التحكم "التسمية التوضيحية" Label

من الممكن استخدام عنصر التسمية التوضيحية لإدراج نص سواء بشكل ثابت أو متغير. عادةً ما يتم استخدام التسمية التوضيحية مع عناصر التحكم الأخرى مثل مربع النص لتوضيح طبيعة البيانات الموجودة بذلك العنصر. لإدراج تسمية توضيحية نختار Label من صندوق الأدوات Toolbox ثم ندرج التسمية في المكان المطلوب، بالطبع يمكن التحكم بحجم التسمية من خلال السحب والافلات.

من الممكن عرض قيمة ثابتة في التسمية التوضيحية من خلال كتابة تلك القيمة في بند Caption الموجود ضمن خصائص التسمية التوضيحية. أو عرض قيمة متغيرة من خلال كتابة كود برمجي كما في المثال التالي.

مثال: في هذا المثال سوف نعدل على المثال السابق بحيث يتم اظهار اسم ورقة العمل التي سوف طباعتها مع المصنف الذي يحتويها في أعلى النموذج كما في الشكل 11-12

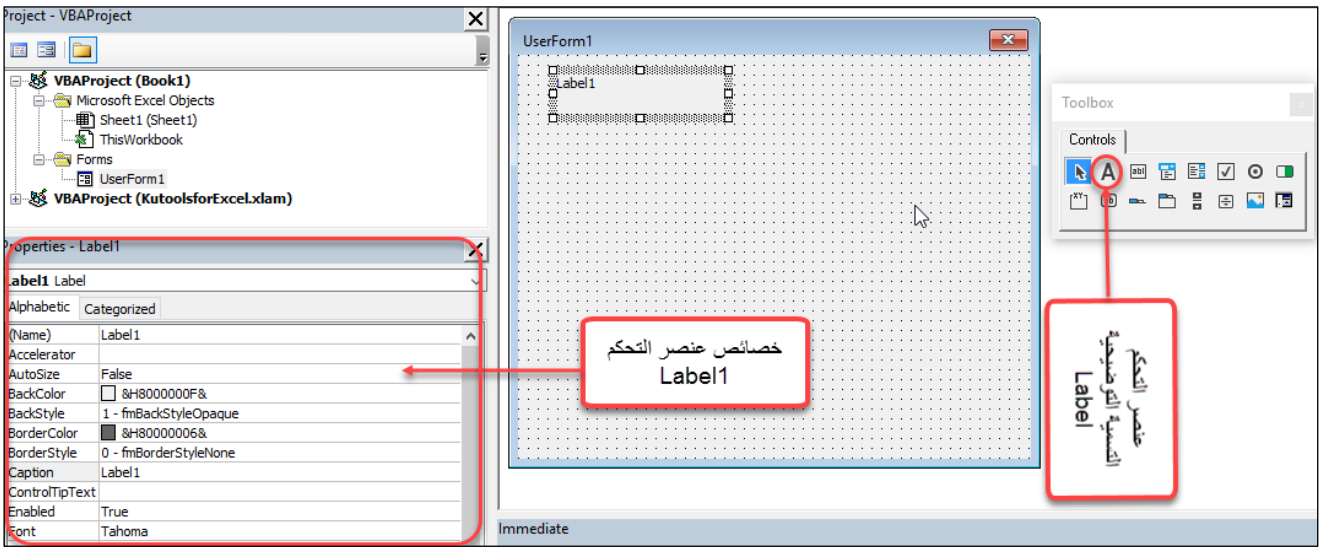
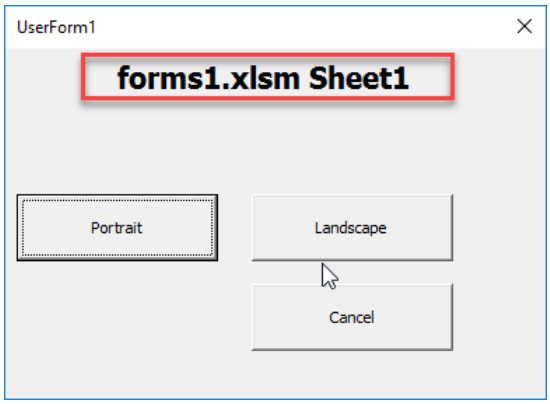


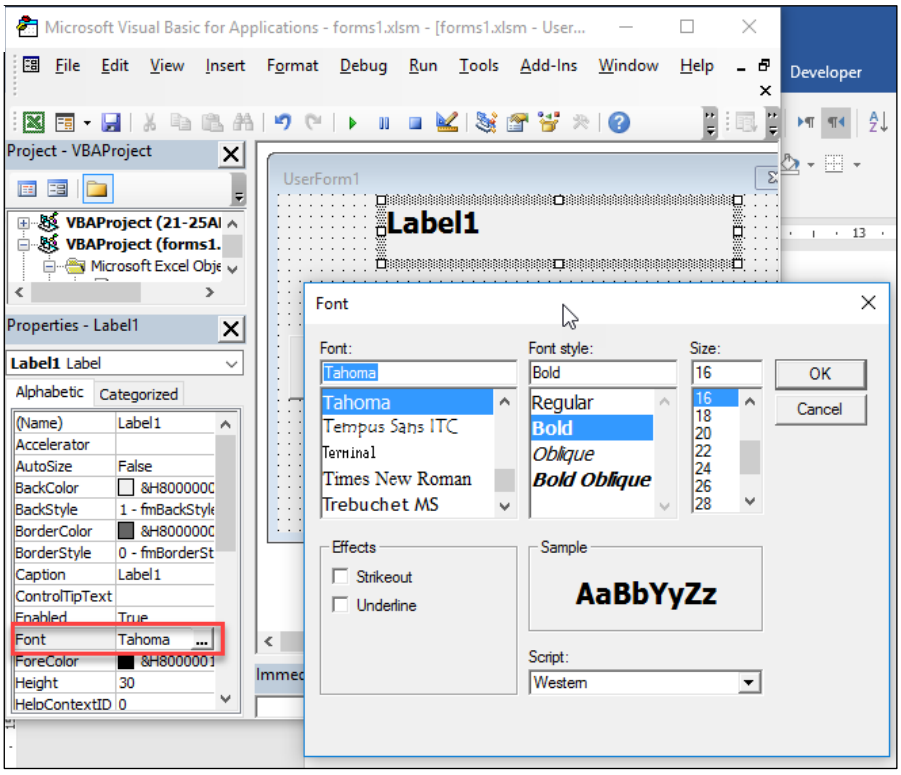
Figure12-10

Figure12-11



لعمل ذلك نقوم بإدراج عنصر Label إلى النموذج، نعدل نوع الخط في التسمية التوضيحية من خلال تعديل نوع الخط من خصائص التسمية التوضيحية كما في الشكل 12-12.

Figure12-12



ولجعل التسمية التوضيحية تظهر اسم المصنف الحالي وورقة العمل النشطة عند تشغيل النموذج، نظهر الكود الخاص بالنموذج عن طريق النقر بزر الماوس الأيمن فوق أي مكان على النموذج ثم اختيار View Code ثم من القائمة المنسدلة الخاصة بالأحداث نختار الحدث initialize ثم ندخل الكود التالي في إجرائية معالج الحدث Event-Handler procedure (انظر الشكل 12-13)

```
Private Sub UserForm_Initialize()

   Label1.Caption = ActiveWorkbook.Name & " " & ActiveSheet.Name

End Sub
```

والآن بمجرد تشغيل النموذج سوف يتم عرض اسم المصنف الحالي وورقة العمل النشطة في التسمية التوضيحية Label1.

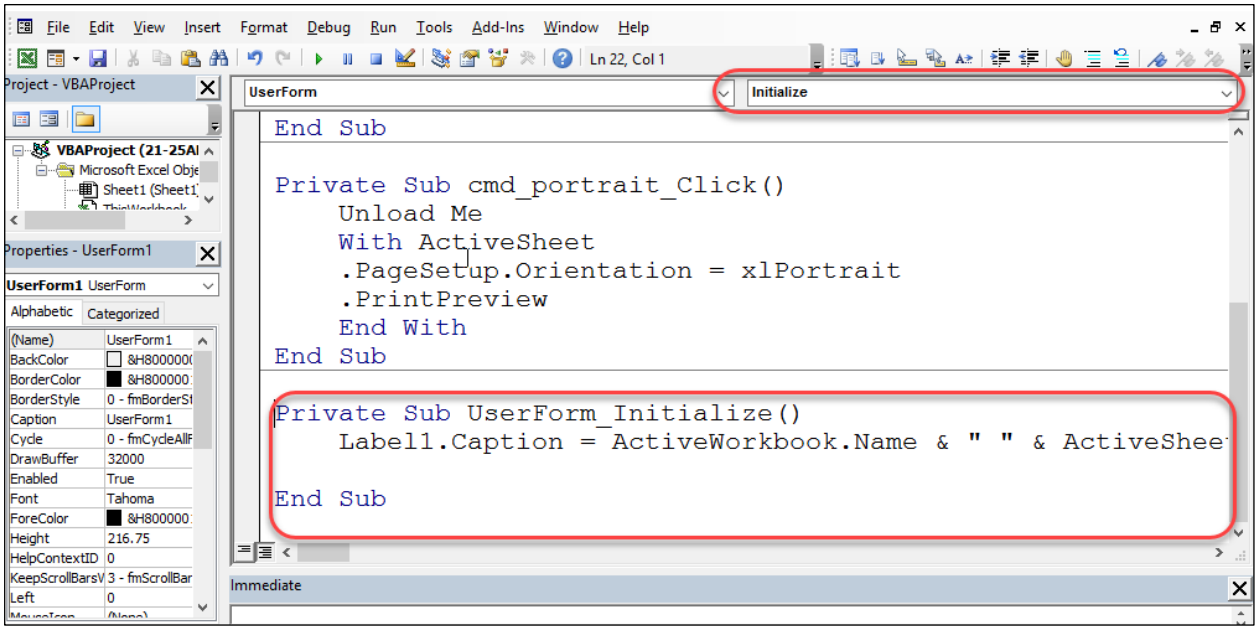


Figure12-13

أزرار الاختيار Option Buttons

تكون أزرار الاختيار مفيدة عندما يحتاج المستخدم إلى اختيار قيمة ضمن عدد صغير من القيم. عادة ما تستخدم أزرار الاختيار ضمن مجموعات تتكون من زرين على الأقل وعادةً ما توضع المجموعة بداخل إطار Frame. ويستطيع المستخدم اختيار قيمة واحدة فقط من المجموعة. في الشكل 12-14 تظهر لنا مجموعتين من الأزرار ضمن اطارين مختلفين ويستطيع المستخدم أن يختار عنصر واحد فقط من كل مجموعة.

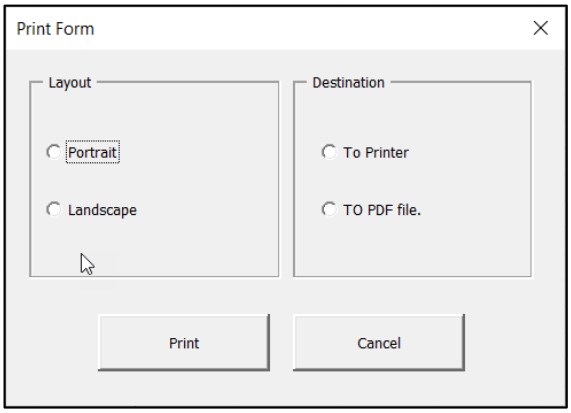


Figure12-14

لتحديد المجموعة التي ينتمي إليها أحد أزرار الاختيار يمكنك استخدام احدى الطريقتين الآتيتين:

- أضف إطار Frame للنموذج ومن ثم ضع أزرار الاختيار التي تنتمي لنفس المجموعة لذلك الإطار. هذه الطريقة هي المستخدمة في النموذج الظاهر في الشكل 12-15 .
- أعطِ لجميع أزرار المجموعة نفس اسم المجموعة Group Name من خلال الخاصية GroupName. من الممكن أن تحدد جميع أزرار الاختيار التي تريد ضمها لنفس المجموعة ومن ثم تغير اسم المجموعة لجميع الأزرار مرة واحدة من خلال تغيير GroupName كما في الشكل 12-15.

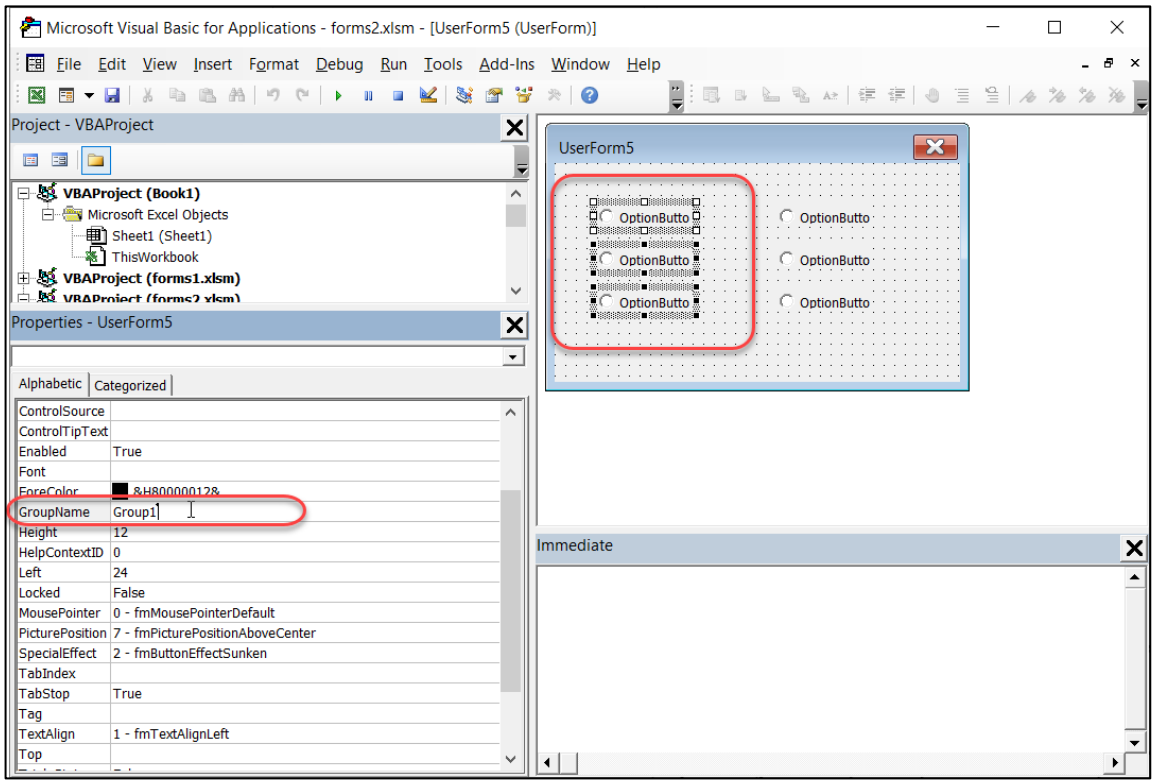


Figure12-15

أهم الخصائص الخاصة بزر الاختيار Option Button

- **GroupName**: من خلال هذه الخاصية يمكن تعيين اسم مجموعة لزر الأمر.
- **ControlSource**: خلية ورقة العمل المرتبطة بزر الاختيار. عند اختيار الزر سوف تكون قيمة هذه الخلية True وعند عدم الاختيار سوف تكون False.
- **Value**: عند اختيار الزر ستأخذ هذه الخاصية القيمة True وعند عدم الاختيار ستأخذ القيمة False.

مثال

في هذا المثال المطلوب التعديل على نموذج الطباعة الذي تعرضنا له مسبقاً بحيث نستخدم أزرار الاختيار لتحديد هل الطباعة طولية أم عرضية بدلاً من أزرار الأوامر.

1- أولاً قم بإدراج نموذج فارغ ثم أدرج إطار Frame ثم أدرج زري اختيار OptionButtons بداخل الاطار.
2- غير خصائص الزرين بحيث يأخذ زر الطباعة الطولية الاسم Name: optPortrait و العنوان Caption: Portrait وزر الطباعة العرضية الاسم optLandscape والعنوان Landscape.
3- أدرج زري أمر Cancel و Print. بالنسبة لزر Cancel فأعطه نفس الكود الموجود في المثال السابق. وبالنسبة للزر Print فأعطه الكود التالي:

```
Private Sub cmdPrint_Click()
  Unload Me
  If optPortrait.Value = True Then
    With ActiveSheet
      .PageSetup.Orientation = xlPortrait
      .PrintPreview
    End With
  ElseIf optLandscape.Value = True Then
    With ActiveSheet
      .PageSetup.Orientation = xlPortrait
      .PrintPreview
    End With
```

```
  Else
     MsgBox "Please choose Portrait or Landscape"
     frmPrint.Show
  End If
End Sub
```

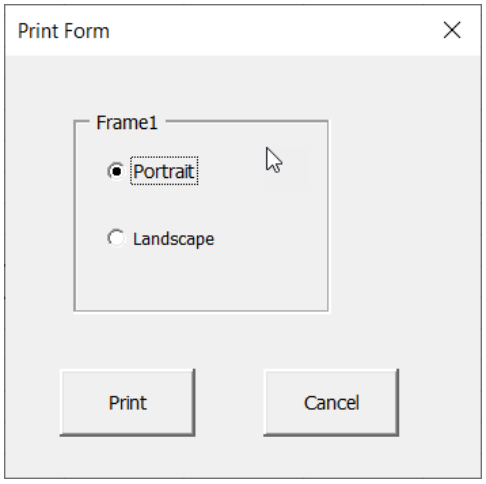


Figure12-16

يبدأ الكود بفحص قيمة زر الاختيار optPortrait فإذا كانت قيمته True (أي أنه تم اختيار هذا الخيار) فسيتم تنفيذ الكود الخاص بالطباعة الطولية كما ورد سابقاً. أما إذا كانت قيمة الاختيار optLandscape هي True فسيتم تنفيذ كود الطباعة العرضية. أما إذا لم يتم اختيار أي من الخيارين فسوف تظهر رسالة تنبيه للمستخدم تطلب منه اختيار أحد الخيارين، ومن ثم يتم اظهار النموذج مرة أخرى من خلال السطر البرمجي frmPrint.Show (حيث أن النموذج تم اخفاؤه في السطر الأول من البرنامج للسبب الذي تم ذكره في المثال في الصفحة 168).

من الممكن أن تجعل البرنامج يقوم بتحديد أحد الأزرار بشكل افتراضي Default selection من خلال تعديل الكود الخاص ببدء النموذج، حسب الخطوات التالية:

- اعرض الكود الخاص بالنموذج عن طريق النقر بزر الماوس الأيمن فوق أي مكان فارغ على النموذج واختيار View Code.
- من القائمة المنسدلة الخاصة بالأحداث
- من القائمة المنسدلة الخاصة بالأحداث (أعلى يمين النافذة) اختر الحدث Initialize ثم أدرج الكود التالية في إجرائية معالجة الحدث UserForm_Initializ():

```
optPortrait.Value = True
```

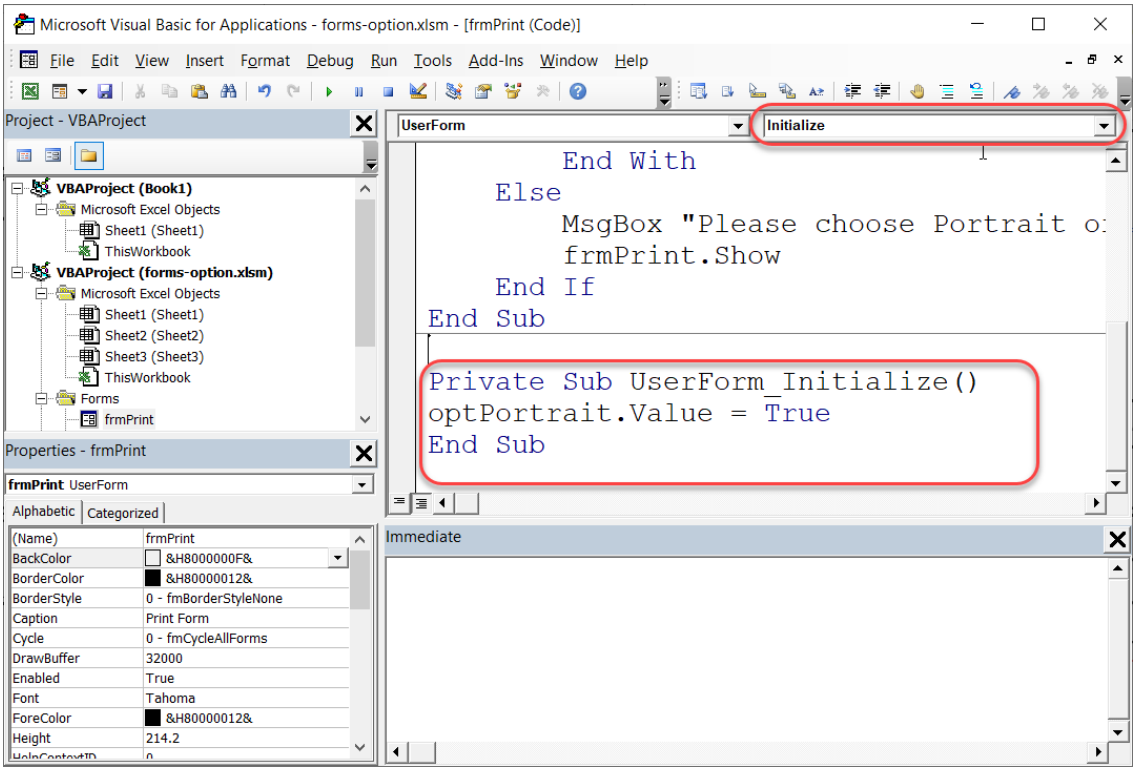


Figure 12-17

صندوق الاختيار Check Box

يكون مربع الاختيار مفيداً عندما يكون لديك الاختيار بين نعم ولا أو True/False أو On/Off وهكذا.
فيما يلي أهم الخصائص الخاصة بمربع الاختيار:

- **ControlSource**: خلية ورقة العمل المرتبطة بصندوق الاختيار. عند اختيار الزر سوف تكون قيمة هذه الخلية True وعند عدم الاختيار سوف تكون False.
- **Value**: عند اختيار الزر ستأخذ هذه الخاصية القيمة True وعند عدم الاختيار ستأخذ القيمة False.

مثال

في النموذج الظاهر في الشكل 18-12 تم إضافة زر صندوق اختيار لنموذج الطباعة الذي ورد في المثال السابق، بحيث ان اختيار هذا الزر يقوم بعرض خطوط الشبكة Gridlines عند الطباعة.

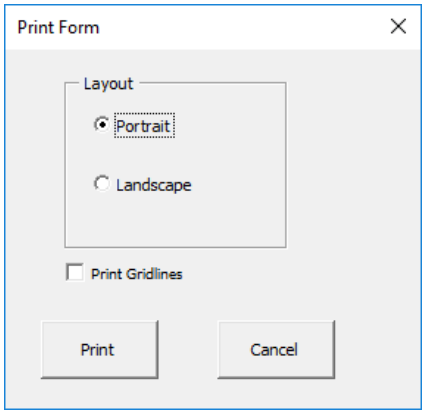


Figure12-18

لعمل هذا النموذج، قم بإدراج صندوق اختبار للنموذج الذي استخدمناه في المثال السابق (الخاص بأزرار الاختيار) ثم قم بتعديل خصائص صندوق الاختيار كالتالي:

- Name :chkGrid
- Caption :Print Gridlines

بعد أن يتم ادراج صندوق الاختيار وتغيير خصائصه كما سبق، قم بالتعديل على الكود المرتبط بزر الأمر Print عن طريق النقر المزدوج عليه ومن ثم ادراج جملة If التالية في بداية إجرائية معالجة الحدث كما هو ظاهر في الشكل 19-12 :

```
If chkGrid.Value = True Then
    ActiveSheet.PageSetup.PrintGridlines = True
Else
    ActiveSheet.PageSetup.PrintGridlines = False
End If
```

تقوم هذه الجملة بفحص ما إذا كانت قيمة صندوق الاختيار chkGrid هي True فإذا كانت كذلك تنفذ السطر التالي:

```
ActiveSheet.PageSetup.PrintGridlines = True
```

والذي يقوم بعرض خطوط الشبكة عند الطباعة وإلا سوف يتم تنفيذ السطر:

```
ActiveSheet.PageSetup.PrintGridlines = False
```

وبالتالي لن يتم عرض خطوط الشبكة عند الطباعة.

```
Private Sub cmdPrint_Click()
    Unload Me
    If chkGrid Then
        ActiveSheet.PageSetup.PrintGridlines = True
    Else
        ActiveSheet.PageSetup.PrintGridlines = False
    End If
    If optPortrait = True Then
        With ActiveSheet
            .PageSetup.Orientation = xlPortrait
            .PrintPreview
        End With
    ElseIf optLandscape = True Then
        With ActiveSheet
            .PageSetup.Orientation = xlPortrait
            .PrintPreview
        End With
    Else
        MsgBox "Please choose Portrait or Landscape"
        frmPrint.Show
    End If

End Sub
```

Figure12-19

القائمة List Box

يتم من خلال عنصر التحكم هذا عرض مجموعة من العناصر ليتم اختيار عنصر أو أكثر من قبل المستخدم. يُظهر لنا الشكل مثالاً لقائمة ListBox، من الممكن تهيئة القائمة للسماح للمستخدم باختيار عنصر واحد من القائمة فقط أو السماح بالاختيار المتعدد.

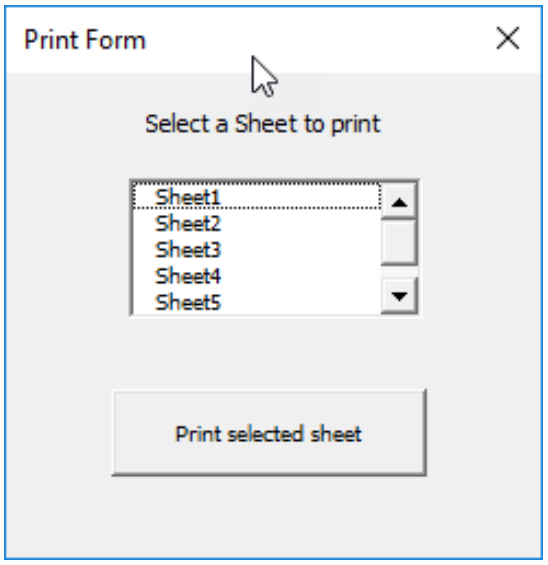


Figure12-20

أهم خصائص عنصر التحكم List Box

- **ControlSource**: من الممكن استخدام هذه الخاصية لتحديد إحدى خلايا ورقة العمل بحيث يتم تخزين القيمة المحددة من القائمة بداخلها.
- **ListStyle**: من خلال هذه الخاصية يتم التحكم بمظهر القائمة.
- **MultiSelect**: من هنا يتم تحديد هل من المسموح بالاختيار المتعدد أم لا.
- **RowSource**: من المكن استخدام هذه الخاصية لتحديد نطاق من خلايا ورقة العمل يحتوى على عناصر القائمة.
- **Value**: العنصر المُحدد بالقائمة.

ملاحظة

إذا تم تهيئة القائمة بحيث تقبل الاختيار المتعدد (من خلال الخاصية MultiSelect) فلا يمكن في هذه الحالة تحديد خلية لتعمل كـ ControlSource حيث أن نتيجة التحديد أكثر من قيمة وبالتالي لا يمكن

وضع جميع تلك القيم بداخل خلية واحدة. وفي هذه الحالة يجب كتابة ماكرو يقوم بتخزين القيم المُختارة بداخل نطاق ما.

تعبئة القائمة Populating List

هناك طريقتين لتعبئة القائمة: اما من خلال الخاصية RowSource وإما برمجياً من خلال AddItem method. لتعبئة القائمة من خلال الخاصية RowSource، حدد القائمة المطلوبة ثم من جزء الخصائص الخاص بها أدخل النطاق الذي يحتوي على عناصر القائمة في البند RowSource كما يظهر في الشكل 12-21.

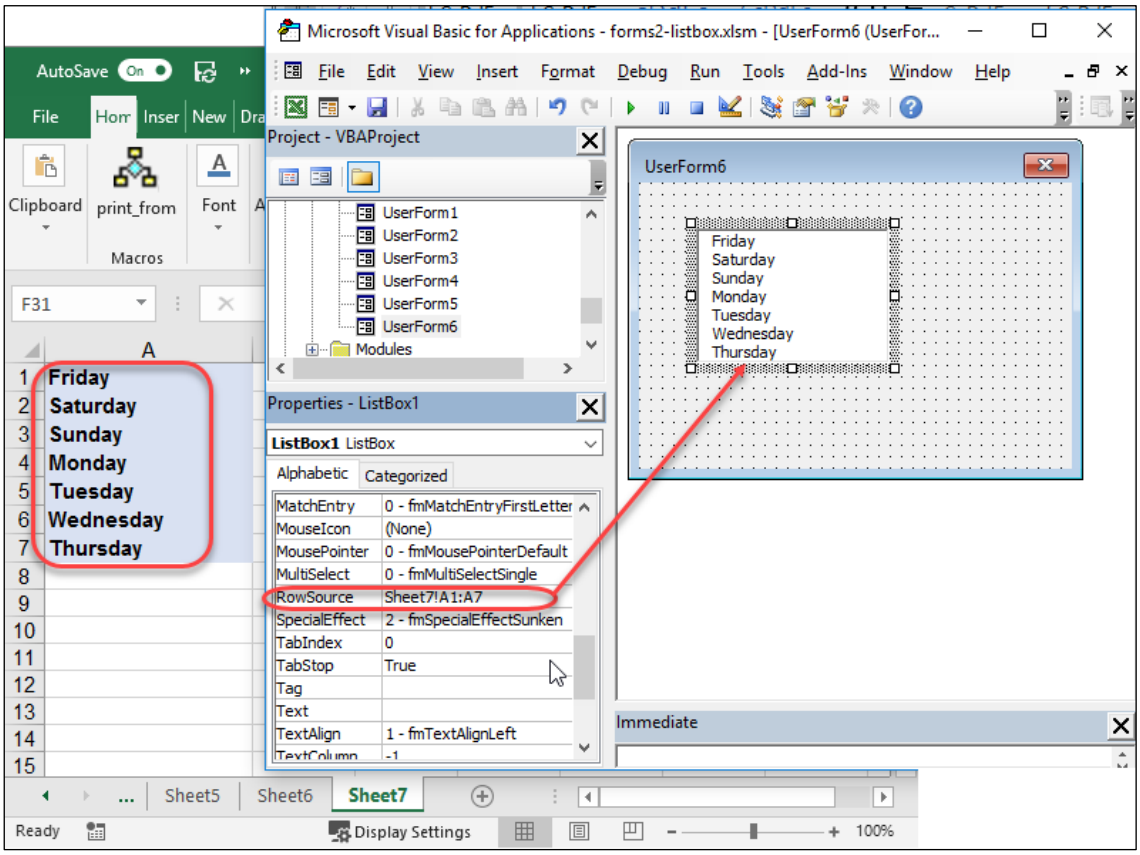


Figure12-21

وعلى الرغم من أن الطريقة السابقة قد تكون مفيدة في بعض الأحيان، إلا أنه هناك العديد من الحالات التي قد ترغب فيها بأن يتم تعبئة القائمة برمجياً وفي هذه الحالة يمكن استخدام الحدث الخاص ببدء النموذج UserForm Initialize event لتعبئة القائمة. وذلك باستخدام AddItem method.

كمثال على ذلك لنفترض أنك تريد أن تعبئ القائمة بأسماء أشهر الربع الأول من السنة بدون أن يتم تخزين تلك القيم بداخل خلايا ورقة العمل. لعمل ذلك من الممكن كتابة الكود التالي في إجرائية معالجة الحدث الخاصة ببدء النموذج UserForm Initialize event-handler procedure:

```
Private Sub UserForm_Initialize()
  With ListBox1
    .RowSource = ""
    .AddItem "January"
    .AddItem "Febreuary"
    .AddItem "March"
  End With
End Sub
```

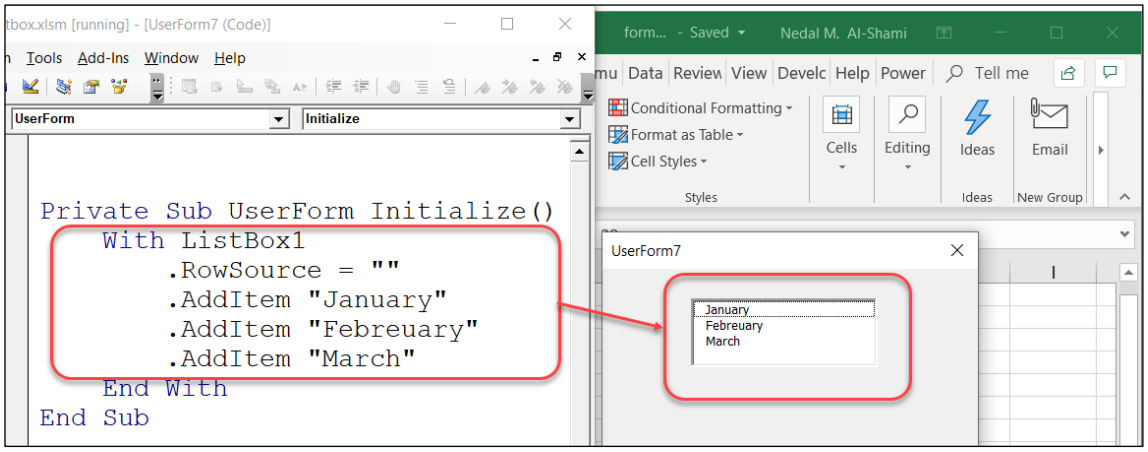


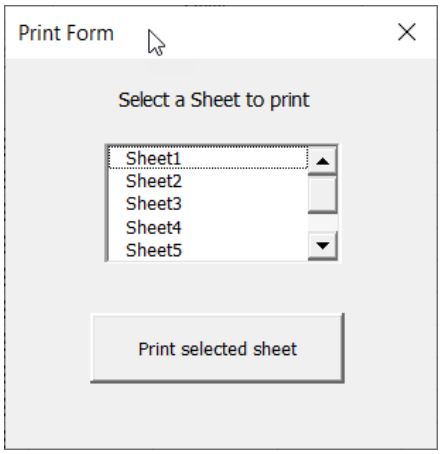
Figure 12-22

لاحظ أننا استخدمنا السطر البرمجي "" = RowSource. في بداية الكود حتى نكون متأكدين من حذف أي قيم سابقة قد تكون موجودة في القائمة.

مثال على استخدام List Box في النماذج

يقوم النموذج في الشكل 12-23 بعرض جميع أوراق العمل للمصنف الحالي في قائمة List Box ولطباعة محتويات أي ورقة يتم تحديد الورقة من القائمة ثم الضغط على زر Print selected sheet.

Figure12-23



لعمل ذلك النموذج ندرج قائمة ListBox من صندوق الأدوات (في هذا المثال أبقينا على اسم القائمة الافتراضي ListBox1)، ثم نعبئ القائمة بأسماء أوراق العمل للمصنف الحالي من خلال ادخال الكود التالي في إجرائية معالجة الحدث الخاصة ببدء النموذج:

```
Private Sub UserForm_Initialize()
Dim oSheet As Worksheet
    For Each oSheet In ActiveWorkbook.Worksheets
        ListBox1.AddItem oSheet.Name
    Next oSheet
ListBox1.ListIndex = 0
End Sub
```

يعمل هذا الكود كالتالي:

1- في البداية تم تعريف متغير باسم oSheet من نوع Worksheet
2- تم استخدام جملة التكرار For….Next للتكرار عبر أوراق العمل بداخل المصنف ثم ادراج اسم كل ورقة عمل بداخل القائمة من خلال الجملة البرمجية:

```
ListBox1.AddItem oSheet.Name
```

3- ثم استخدمنا ListIndex method لاختيار أحد العناصر بداخل القائمة بشكل تلقائي من خلال السطر البرمجي ListBox1.ListIndex = 0 الذي يقوم بتحديد العنصر الأول في القائمة بشكل افتراضي Default value.

لإدراج الكود الخاص بزر الطباعة، انقر نقراً مزدوجاً فوق الزر لفتح إجرائية معالجة الحدث الخاص بالزر، ثم أدرج الكود التالي:

```
Private Sub CommandButton1_Click()
  Selected_Sheet = ListBox1.Value
  Unload Me
  Worksheets(Selected_Sheet).PrintPreview
End Sub
```

حيث يقوم الكود بعمل معاينة قبل الطباعة لورقة العمل المُختارة. ولطباعة ورقة العمل مباشرة بدون عمل معاينة يمكنك استخدام الكود التالي:

```
Private Sub CommandButton1_Click()
  Unload Me
Worksheets(ListBox1.Value).PrintOut
End Sub
```

الآن أصبح الكود الكامل كما في الشكل 12-24

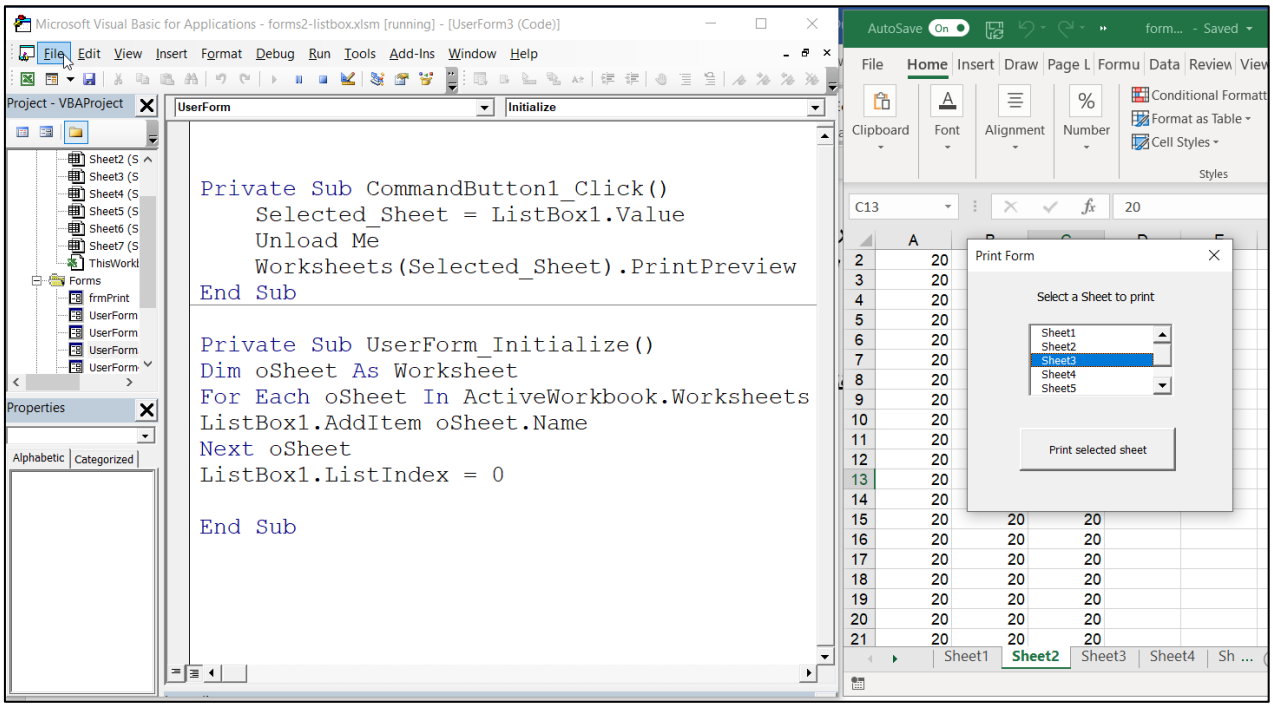


Figure12-24

ملاحظة

لاحظ هنا التالي: قائمة List Box وبخلاف عناصر التحكم الأخرى (مثل ComboBox و CheckBox) لا تحتفظ بقيمتها بعد تطبيق Unload Me ولذلك خزنّا القيمة المحددة بداخل القائمة بداخل المتغير Selected_Sheet قبل تطبيق Unload Me.

استخدام الاختيار المتعدد مع القائمة Multi-Select ListBox

تعطيك قائمة List Box إمكانية اختيار أكثر من عنصر في نفس الوقت من خلال تعديل الخاصية Mutiselect كما في الشكل 12-25. لاحظ أن هذه القائمة تحتوي على ثلاث خيارات:

1- fmMultiSelectSingle: لاختيار عنصر واحد فقط
2- fmMultiSelectMulti: لاختيار أكثر من عنصر باستخدام النقر بالماوس.
3- fmMultiSelectExtended: لاختيار أكثر من عنصر باستخدام النقر بالماوس مع زر CTRL.

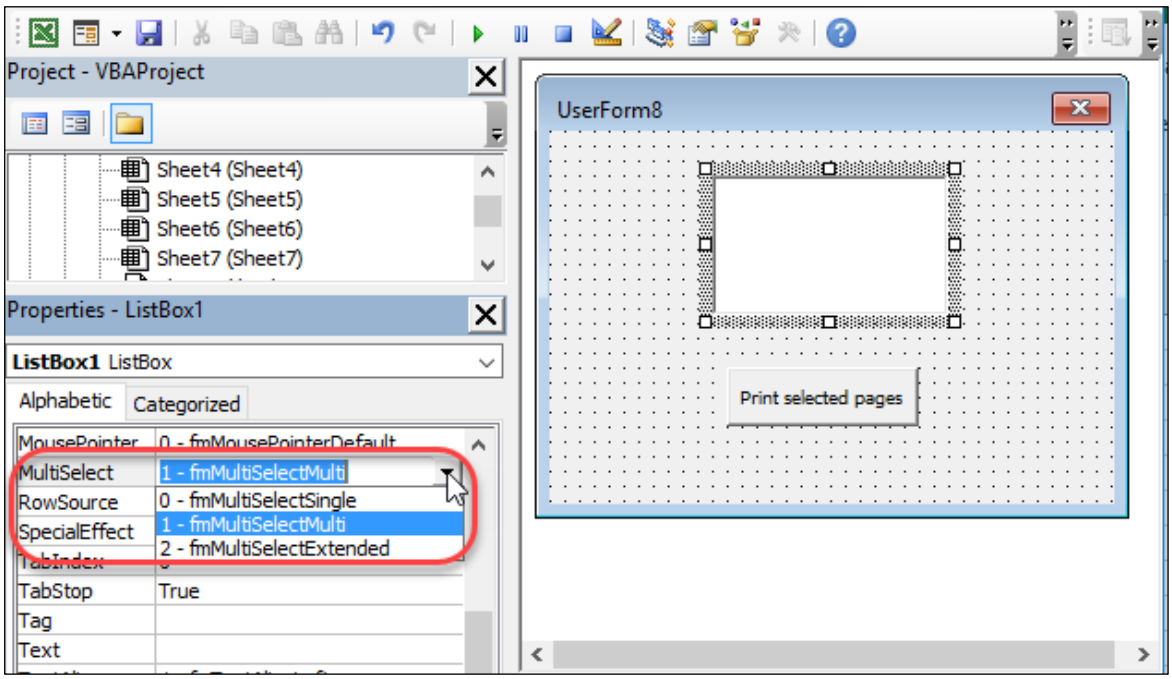


Figure12-25

من الممكن تعديل هذه الخاصية من خلال جزء الخصائص الخاص بالقائمة كما بالشكل أعلاه. أو برمجياً من خلال الـ MultiSelect property. فمثلاً لجعل القائمة ذات اختيار متعدد من خلال البرمجة بالإمكان ادخال الكود التالي في إجرائية الحدث الخاصة ببدء النموذج:

```
ListBox1.MultiSelect = fmMultiSelectMulti
```

يساعدك محرر الأكواد في ادخال الخيار المناسب لك من خلال ميزة الاختيار التلقائي كما في الشكل 12-26.

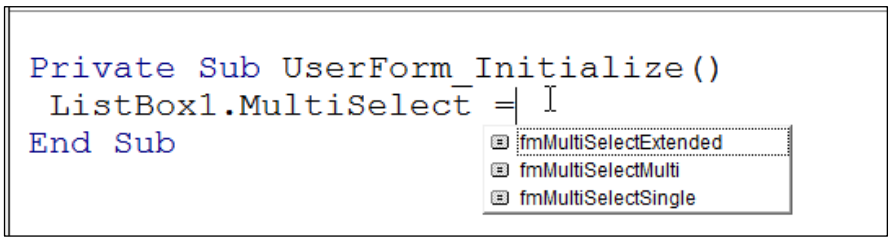
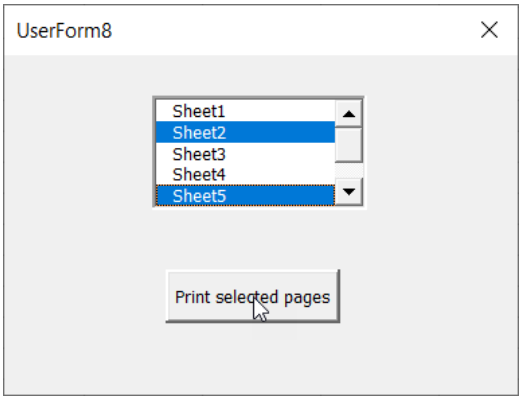


Figure 12-26

مثال على استخدام القائمة مع الاختيار المتعدد

في الشكل 12-27 مثال على استخدام الاختيار المتعدد مع قائمة List Box حيث أنك من خلال هذا النموذج تستطيع تحدد أكثر من ورقة عمل ثم عرض معاينة قبل الطباعة لأوراق العمل المحددة.

Figure12-27



لعمل هذا النموذج، استخدم المثال السابق؛ من محرر الأكواد VBE افتح النموذج، حدد القائمة ثم عدل خصائصها من جزء الخصائص واختر الخيار fmMultiSelectMulti أو fmMultiSelectExtended .

أدخل الكود التالي والخاص بزر الطباعة عن طريق النقر المزدوج فوق زر الطباعة ومن ثم كتابة الكود في إجرائية معالجة الحدث التي سوف تفتح:

```
Private Sub CommandButton1_Click()
    Unload Me
    For i = 0 To ListBox1.ListCount - 1
        If ListBox1.Selected(i) Then
        Sheets(ListBox1.List(i)).PrintPreview
        End If
    Next i
End Sub
```

بما أننا سمحنا بالاختيار المتعدد في القائمة ListBox1 فلا يعد بالإمكان استخدام الخاصية Value حيث أن الاختيار قد يكون أكثر من قيمة وبالتالي فإننا استخدمنا جملة التكرار For……Next للتكرار عبر عناصر القائمة وفحص كل عنصر، هل هو مُحدد أم لا؟ من خلال الخاصية Selected، فإذا كان مُحدد يتم عرض المعاينة قبل الطباعة لورقة العمل التي تحمل نفس اسم العنصر المحدد.

القائمة المنسدلة Combo Box

القائمة المنسدلة Combo Box تجمع ما بين خصائص القائمة List Box و مربع ادخال النص Input Box (سيتم التطرق له لاحقاً)، حيث أنك من خلال الـ Combo Box تستطيع إعطاء المستخدم إمكانية ادخال قيمة جديدة بخلاف العناصر الموجودة في القائمة بالإضافة إلى إمكانية اختيار عنصر من عناصر القائمة. وبإمكانك أن تلغي إمكانية ادخال القيم الجديدة والاقتصار على الاختيار من القائمة فقط.

مثال على استخدام القائمة المنسدلة

تم تعديل المثال السابق بحيث نستخدم القائمة المنسدلة بدلاً من قائمة List Box لاختيار ورقة العمل التي نريد طباعتها. لعمل هذا النموذج قم بإدراج Combo Box على النموذج (في هذا المثال أبقينا على الاسم الافتراضي للقائمة المنسدلة وهو ComboBox1). ثم قم بتعبئة القائمة المنسدلة بأسماء أوراق العمل بنفس الطريقة التي استخدمناها في المثال السابق، من خلال إدخال الكود التالي في إجرائية الحدث الخاصة ببدء النموذج:

```
Private Sub UserForm_Initialize()
Dim oSheet As Worksheet
For Each oSheet In ActiveWorkbook.Worksheets
        ComboBox1.AddItem oSheet.Name
Next oSheet
ComboBox1.ListIndex = 0
End Sub
```

لاحظ هنا أننا حددنا القيمة الافتراضية للقائمة المنسدلة من خلال السطر البرمجي:

```
ComboBox1.ListIndex = 0
```

وبالنسبة للكود الخاص بزر الأمر فسوف يكون كالتالي:

```
Private Sub CommandButton1_Click()
   Unload Me
   Sheets(ComboBox1.Value).PrintPreview
End Sub
```

تجدر الملاحظة أنه لا توجد إمكانية للاختيار المتعدد في القائمة المنسدلة.

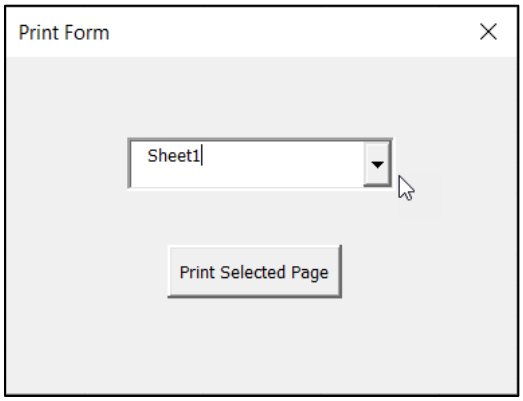


Figure12-28

لإلغاء إمكانية إدخال قيم جديدة (غير موجودة ضمن القائمة) عدل الخاصية Style للقيمة fmStyleDropDownList.

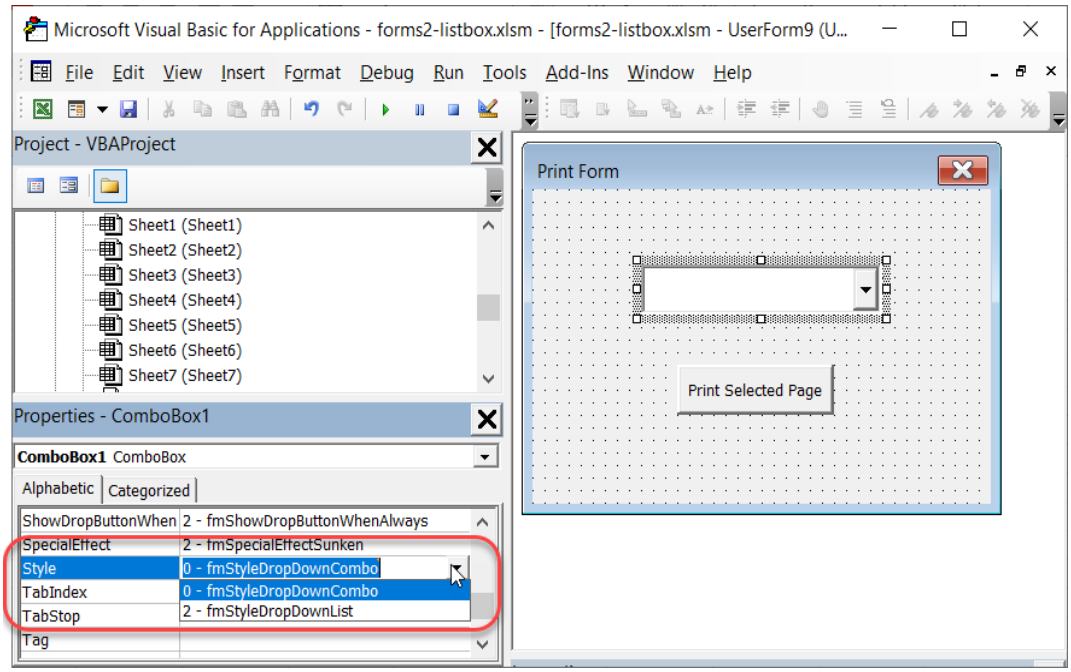


Figure12-29

مربع النص Text Box

يعتبر مربع النص من الوسائل المثالية لإدخال البيانات حيث يمكنك من خلاله ادخال أي نوع من البيانات مثل النصوص أو الأرقام أو غيرها. الشكل 12-30 يظهر مثالاً على استخدام مربع النص في النماذج. فيما يلي أهم الخصائص الخاصة بمربع النص:

- ControlSource: من الممكن استخدام هذه الخاصية لإدخال عنوان الخلية التي تحتوي على النص الذي نرغب بإظهاره في مربع النص.
- MaxLength: الحد الأقصى للحروف التي يمكن إدخالها في مربع النص.
- MultiLine: اذا كانت قيمة هذه الخاصية True فيمكن ادخال أو عرض أكثر من سطر في مربع النص.
- TextAlligne: للتحكم بمحاذاة النص.

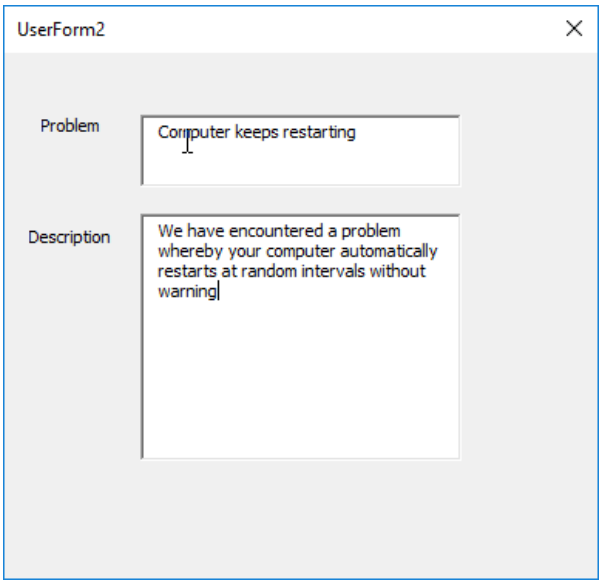
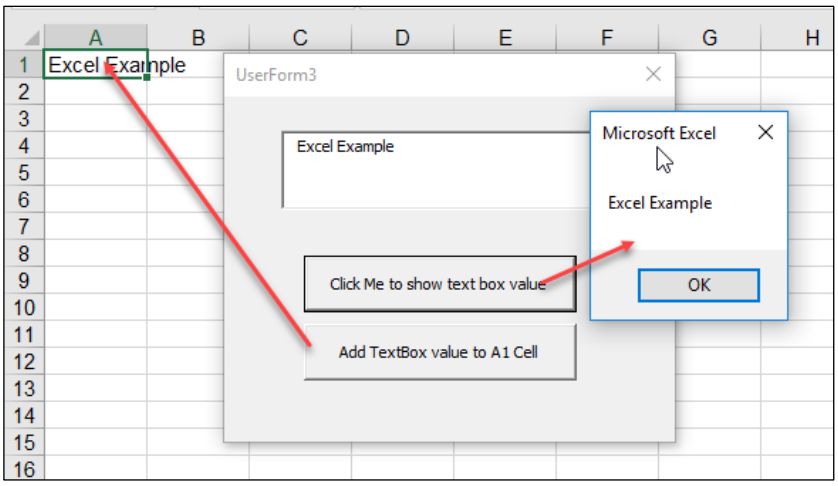


Figure12-30

مثال على استخدام مربع النص

في الشكل 12-31 تم ادراج مربع نص وزري أمر في النموذج، والمطلوب برمجة الزر العلوي بحيث يعرض القيمة التي تم إدخالها في مربع النص أما الزر السفلي فيقوم بتخزين النص الموجود في مربع النص في الخلية A1 لورقة العمل النشطة.

Figure12-31



لعمل ذلك انقر فوق الزر العلوي لفتح إجرائية معالجة الحدث الخاصة به ثم أدخل الكود التالي:

```
Private Sub CommandButton1_Click()
    MsgBox TextBox1.Value
End Sub
```

اما بالنسبة للزر السفلي فأدخل الكود التالي في إجرائية معالجة الحدث الخاصة به:

```
Private Sub CommandButton2_Click()
    Range("A1").Value = TextBox1.Value
End Sub
```

مثال. بناء نموذج لترحيل البيانات

يقوم النموذج في الشكل 32-12 بتخزين البيانات الموجودة في مربعي النص على النموذج إلى ورقة العمل Sheet1 كما هو ظاهر في الشكل.

لعمل هذا النموذج أدرج عناصر التحكم كما هو ظاهر في الشكل. أعطِ مربع الاسم الخاص باسم المنتج Product Name الاسم: Name_box ومربع الاسم الخاص بوصف المنتج Product Description الاسم: Desc_box.

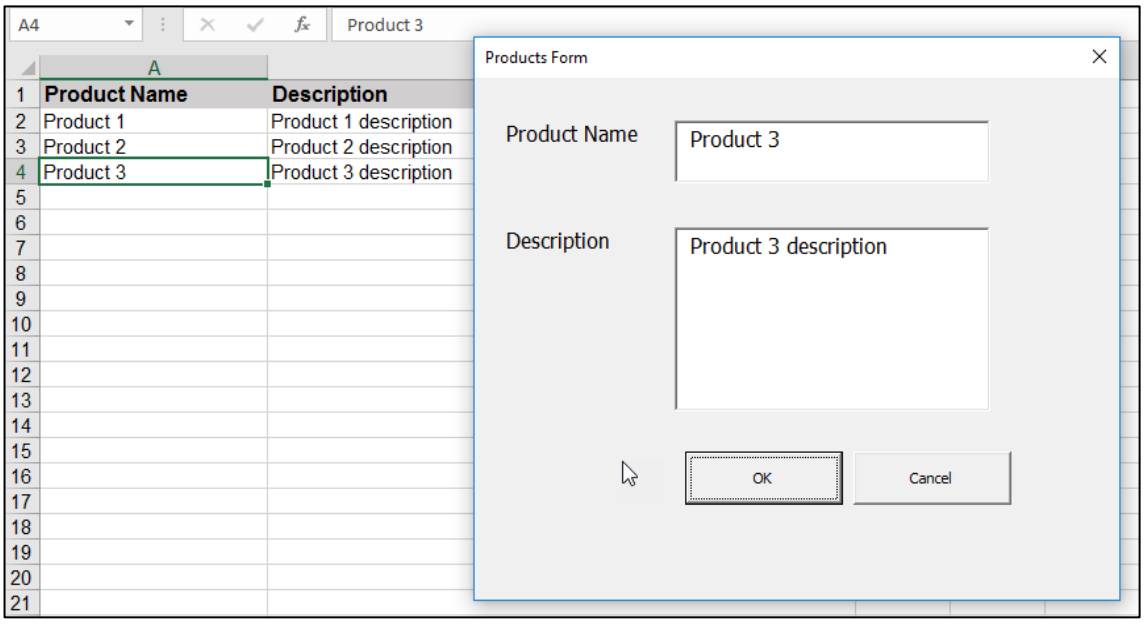


Figure12-32

انقر فوق زر OK نقراً مزدوجاً لفتح إجرائية معالجة الحدث الخاصة بالنقر فوق الزر ثم أدخل التالي:

```
Private Sub CommandButton1_Click()
last_row = Cells(Rows.Count, 1).End(xlUp).Row
With Sheets("sheet1")
   .Cells(last_row + 1, 1) = Name_box.Value
   .Cells(last_row + 1, 2) = desc_box.Value
End With
End Sub
```

يعمل هذا الكود كالتالي:

- أولا تم تحديد آخر صف يحتوي على بيانات في العمود الأول من خلال السطر البرمجي:

```
last_row = Cells(Rows.Count, 1).End(xlUp).Row
```

حيث يعمل هذا السطر كالتالي: أولاً يقوم بتحديد آخر خلية في العمود الأول من خلال Cells(Rows.Count, 1). ثم يقوم بالسير للأعلى حتى يصل لأول خلية بها بيانات تقابله من خلال End(xlUp). ثم يرجع رقم الصف من خلال Row.

- بعد أن عرفنا آخر صف يحتوي على بيانات في العمود الأول المطلوب الآن أن نرحل البيانات الموجودة في مربع النص الخاص باسم المنتج إلى الخلية التي تلي آخر خلية بها بيانات في العمود الأول، وأن نرحل البيانات الموجودة في مربع النص الخاص بوصف المنتج إلى الخلية التي تلي آخر خلية بها بيانات في العمود الثاني. وذلك تم من خلال الكود التالي:

```
With Sheets("sheet1")
   .Cells(last_row + 1, 1) = Name_box.Value
   .Cells(last_row + 1, 2) = desc_box.Value
End With
```

أولاً حددنا الورقة التي نريد أن نرحل البيانات لها من خلال With Sheets("sheet1") ثم من خلال السطرين التاليين حددنا الخليتين اللتين تليان آخر صف في العمود الأول والثاني وتم تعيين قيمة مربعي النص الخاصين بالاسم والوصف إليهما.

إضافة زر لمحو بيانات نموذج الترحيل

من المفضل إضافة زر للنموذج يقوم بمحو البيانات من مربعات النص كما في الشكل 33-12 .

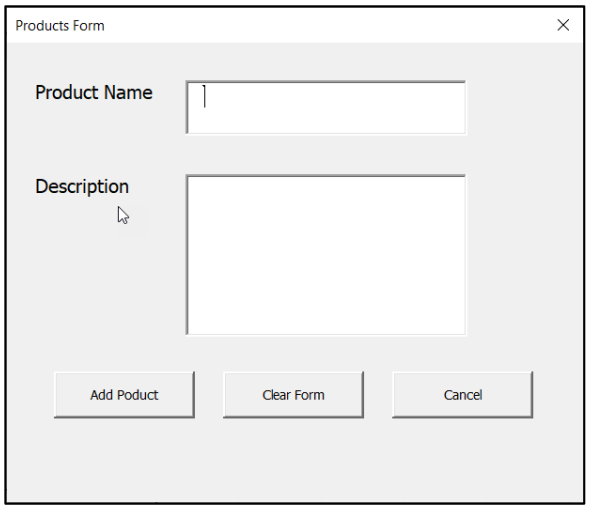


Figure12-33

أدرج زر أمر جديد وأعطه الاسم Clr_Button والتسمية Clear Form ثم انقر عليه نقراً مزدوجاً وأدخل الكود التالي في إجرائية معالجة الحدث:

```
Private Sub Clr_Button_Click()
        Name_box.Value = ""
        desc_box.Value = ""
End Sub
```

إضافة حقل خاص برقم المنتج إلى نموذج الترحيل مع عدم السماح بتكرار رقم المنتج

للسماح للمستخدم بإدخال رقم المنتج بالإضافة إلى اسمه ووصفه، عدل ورقة العمل كما في الشكل 12-34 بإضافة عمود جديد باسم Product ID قبل عمود اسم المنتج. ثم أضف مربع نص جديد للنموذج السابق وأعطه الاسم id_box والتسمية Product ID ثم عدل الكود الخاص بزر Add Product ليصبح كالتالي:

```
Private Sub Add_Button_Click()
If WorksheetFunction.CountIf(Range("A:A"), id_box.Value) > 0 Then
   MsgBox ("Duplicate ID!! please use non duplicate ID ...")
   Exit Sub
End If
last_row = Cells(Rows.Count, 1).End(xlUp).Row
With Sheets("sheet1")
   .Cells(last_row + 1, 1) = id_box.Value
   .Cells(last_row + 1, 2) = Name_box.Value
   .Cells(last_row + 1, 3) = desc_box.Value
End With
End Sub
```

من المنطقي أن يكون لكل منتج رقم غير متكرر (فريد) وللتحقق من أن رقم المنتج غير مستخدم من قبل استخدمنا جملة IF كالتالي:

```
If WorksheetFunction.CountIf(Range("A:A"), id_box.Value) > 0 Then
   MsgBox ("Duplicate ID!! please use non duplicate ID ...")
   Exit Sub
End If
```

يقوم السطر الأول في جملة IF باستخدام دالة ورقة العمل CountIf لفحص ما إذا كانت القيمة الموجودة في مربع النص الخاص برقم المنتج id_box موجودة من قبل في العمود A فإذا كانت موجودة فستظهر رسالة خطأ تفيد بأن القيمة المدخلة متكررة ثم تخرج من الإجرائية من خلال Exit Sub.

في النهاية عدل الكود الخاص بزر محو بيانات النموذج ليصبح كالتالي:

```
Private Sub Clr_Button_Click()
   id_box.Value = ""
   Name_box.Value = ""
   desc_box.Value = ""
End Sub
```

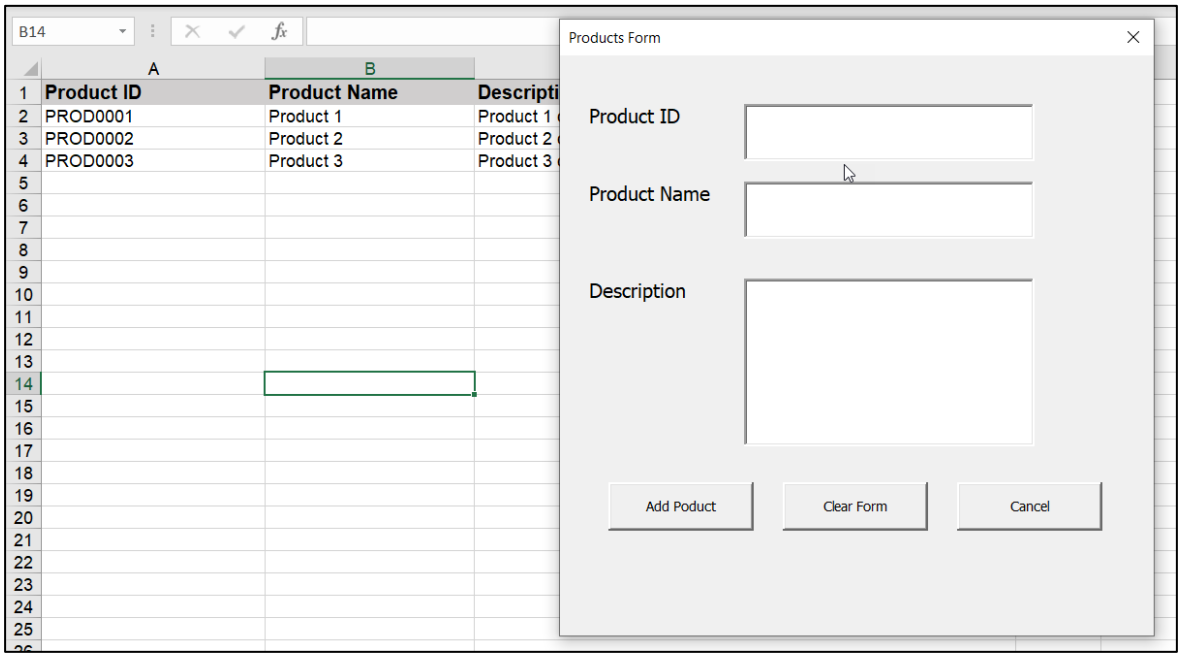


Figure 12-34