مقدمـــة:

قامت شركة مايكروسوفت بتصميم مجموعة البرامج التي أطلقت عليها اسم (مجموعة برامج المكتب Office Programs Group) لتلبي حاجات المستخدم من الحاسوب ، ومن أبرز هذه البرامج برنامج تحرير النصوص ومعالجتها (Microsoft Word) ، وبرنامج معالجة الجداول الإلكترونية (Microsoft Excel).

ولإفساح المجال أمام المطورين ليدلوا بدلوهم في هذه المجموعة ، أتاحت مايكروسوفت البرمجة بلغة فجول بيسك ضمن مجموعة برامج المكتب ، بحيث ضمنتها بكل برنامج ، ووفرت الأوامر المتعلقة بإنجاز كل وظيفة لتلائم لغة فجول بيسك ، وهو ما يسمى بالتوافق بين البرامج.

وفي هذه المذكرة ، قمت بتسليط شيء من الضوء على بعض من الأوامر التي يمكن استخدامها في فجول بيسك للتعامل مع الوثيقة (الملف) المنشأ من قبل وورد ، بحيث نتحكم في خصائص الوثيقة وكافة محتوياتها من خلال برنامج في فجول بيسك دون تدخل برنامج وورد ، وكذلك التعامل مع برنامج أكسل بحيث نتحكم في الملف وأوراق عمله وخلاياه ونطاقاته ومخططاته.

وهذه المذكرة موجهة بالأساس إلى المحترفين في البرمجة بلغة فجول بيسك ، فهم وحدهم القادرون على فهم ما أقصده من عبارات وجمل قد تغيب عن المبتدئ والمتوسط.

والله أسأل أن تؤتي المذكرة أكلها ، وأن تنفع بها كل من بحث في دهاليز المنتديات بكلمة (وورد أو أكسل) ، إنه نعم المجيب.

أولاً: برنامج وورد Microsoft Word:

ببساطةٍ هو برنامج معالجة النصوص والرسوم والجداول وغيرها من مكونات الوثائق والرسائل والكتب والصحف والمجلات وغيرها.

وللتعامل معه برمجياً من خلال فجول بيسك ، يجب تعريفه للغة أولاً ، وذلك بإدراج مكتباته في اللغة ثم تعريف متغير على أنه يمثل برنامج وورد بكامل خصائصه ووظائفه.

إدراج مكتبات برنامج وورد:

لإدراج مكتبات البرنامج وورد في مشروع في فجول بيسك نتبع الخطوات التالية:

1. نفتح مشروعاً جديداً.
2. نفتح القائمة (Project) ثم نختار الأمر (References).
3. يظهر مربع الحوار الخاص بالمكتبات ، نحدد الخيار (Microsoft Word 10.0 Object Library) بوضع علامة (صح) أمامه.
4. ننقر فوق الزر (Ok).

تعريف متغير على أنه يمثل برنامج وورد:

نكتب الكود التالي في المكان المناسب:

كـــــــود برمجـــــــي
`Dim WordPro As Word.Application`

وفي هذا الكود ، تم تعريف المتغير Wrd_Pro على أنه برنامج وورد بالكامل ، ويمكننا إجراء ما يمكن إجراؤه في برنامج وورد على هذا المتغير ، فالبرنامج يتكون من مجموعة من الأوامر ، منها ما يتعلق به كبرنامج ، ومنها ما يتعلق بالوثائق والملفات المفتوحة منها والمغلقة ، ومنها ما يتعلق بمكونات الوثيقة من جداول وفقرات وكلمات وحدود وتظليل ورأس الصفحة وتذييلها وغير ذلك.

وتسمى الوثيقة برمجياً بالكائن Document Object ، وكل مكون من مكوناتها يعتبر كائناً بذاته ، وينتمي لتجمع لنفس النوع من الكائنات بمعنى أن مجموعة الوثائق تنتمي لكائن أوسع وأشمل يسمى (تجمع الوثائق Documents Collection) ، والجداول تنتمي لكائن أوسع وهو (تجمع الجداول Tables Collection) وهكذا.

الكائن Application:

ويمثل البرنامج وورد بكامل فعالياته ووظائفه وخصائصه.

بعض أهم خصائص الكائن Application:

1. **ActiveDocument**: وتمثل الوثيقة النشطة المفتوحة حالياً.
2. **ActivePrinter**: وتمثل الطابعة الافتراضية النشطة.
3. **DisplayRecentFiles**: قيمتها True إذا كانت هناك أسماء وثائق مفتوحة مسبقاً في قائمة (ملف File).
4. **Selection**: وتمثل النص المحدد في الوثيقة.

بعض أهم وظائف وطرق الكائن Application:

1. **Quit**: وتستخدم عند إغلاق برنامج وورد.
2. **SendFax**: وتستخدم لبدء تشغيل معالج الفاكس.
3. **ToggleKeyboard**: وتستخدم لتغيير اللغة من عربية إلى لاتينية والعكس.

بعض أهم أحداث الكائن Application:

1. **DocumentBeforeClose**: وينطلق قبل إغلاق الوثيقة تماماً.
2. **DocumentBeforePrint**: وينطلق قبل البدء في طباعة الوثيقة.
3. **DocumentBeforeSave**: وينطلق قبل حفظ التغييرات في الوثيقة.
4. **DocumentChange**: ينطلق عند حدوث تغيرات في محتويات الوثيقة.
5. **DocumentOpen**: ينطلق عند فتح الوثيقة.
6. **NewDocument**: ينطلق عند إنشاء وثيقة جديدة.
7. **Quit**: ينطلق عندما يغلق المستخدم برنامج وورد.
8. **WindowActive**: ينطلق عندما تصير نافذة الوثيقة نشطة.
9. **WindowDeactivate**: وينطلق عندما يغيب التركيز عن نافذة الوثيقة النشطة.

الكائن Document (الوثيقة):

ويمثل وثيقة في برنامج وورد ، وعند فتح وثيقة جديدة أو موجودة مسبقاً يتم إضافتها إلى ما يسمى (تجمع الوثائق Document Collection).

تجمع الوثائق Document Collection:

وهو عبارة عن كائن يضم مجموعة الوثائق المفتوحة حالياً. وتعرف كل وثيقة بدلالة اســمها أو رقم فهرسها.

أهم خصائص تجمع الوثائق:

1. **الخاصية Application:** ويمثل البرنامج Word.
2. **الخاصية Count:** ويمثل عدد الوثائق المفتوحة حالياً.
3. **الخاصية Creator:** وتحدد هل البرنامج الذي أنشأ الوثيقة هو Microsoft Word أم لا.

أهم وظائف تجمع الوثائق:

1. **الوظيفة Add:** وتستخدم في إنشاء وثيقة جديدة وضمها إلى التجمع.
2. **الوظيفة Open:** وتستخدم لفتح وثيقة موجودة مسبقاً وضمها للتجمع.
3. **الوظيفة Save:** وتستخدم لحفظ التغييرات الحاصلة على محتويات الوثيقة.
4. **الوظيفة Close:** وتستخدم لإغلاق وثيقة وإزالتها من التجمع.

أهم الأحداث التي تتعرض لها الوثيقة:

1. **الحدث New:** وينطلق عند إنشاء وثيقة جديدة.
2. **الحدث Open:** وينطلق عند فتح وثيقة موجودة مسبقاً.
3. **الحدث Close:** وينطلق عند إغلاق وثيقة.

وقبل إجراء أي عملية على الوثيقة أو تجمع الوثائق يجب أن نعلم طريقة تعريفهما فـــي المشـــروع. انظر للمثال التالي:

كـــــــود برمجـــــــي

```
Dim Doc As Word.Document
DimDocs As Word.Documents
```

تم في السطر الأول تعريف المتغير Doc بأنه يمثل وثيقة ، والمتغير Docs بأنه تجمع للوثائق فـــي برنامج وورد.

بعض العمليات التي تجرى على الوثيقة:

1. إنشاء وثيقة جديدة برمجياً:

يضم تجمع الوثائق Documents Collection كافة الوثائق المفتوحة. ولإنشاء وثيقة جديدة (Document Object) ، نستخدم الطريقة Add التابعة لهذا التجمع. والمثال التالي يوضـــح كيفيـــة إنشاء وثيقة جديدة:

كـــــــود برمجـــــــي

```
Documents.Add
```

والطريقة الأفضل لإنشاء وثيقة جديدة هي إسناد القيمة الراجعة إلى متغير كـــائن. فالطريقـــة Add أرجعت الكائن Document الذي يشير إلى الوثيقة الجديدة.

وفي المثال التالي يتم إسناد الكائن Document الراجع عن طريق استخدام Add إلى متغير كائن. وبعد ذلك يتم ضبط العديد من خصائص ووظائف هذا الكائن.

كـــــــود برمجـــــــي

```
Sub NewSampleDoc()
    Dim docNew As Document
    Set docNew = Documents.Add
    With docNew
        .Content.Font.Name = "Tahoma"
        .SaveAs FileName:="Sample.doc"
    End With
End Sub
```

2. فتح وثيقة:

لفتح وثيقة نستخدم الطريقة Open التابعة للتجمع Documents. التعليمات التالية تفتح وثيقة باسم (Sample.doc) موجودة في المجلد (MyFolder).

كـــــود برمجـــــي

```
Sub OpenDocument()
    Documents.Open FileName:="C:\MyFolder\Sample.doc"
End Sub
```

3. حفظ وثيقة:

لحفظ وثيقة نستخدم الطريقة Save التابعة للتجمع Documents. وفي المثال التالي يتم حفظ الوثيقة باسم (Sales.doc).

كـــــود برمجـــــي

```
Sub SaveDocument()
    Documents("Sales.doc").Save
End Sub
```

وبإمكاننا حفظ كافة الوثائق المفتوحة وذلك باستخدام الطريقة Save التابعة للتجمع Documents بالصورة التالية:

كـــــود برمجـــــي

```
Sub SaveAllOpenDocuments()
    Documents.Save
End Sub
```

4. حفظ نسخة أخرى من الوثيقة:

نستخدم لذلك الطريقة SaveAs التابعة للتجمع Documents. فالمثال التالي يحفظ الوثيقة النشطة باسم (Temp.doc) في المجلد الحالي.

كـــــود برمجـــــي

```
Sub SaveNewDocument()
    ActiveDocument.SaveAs FileName:="Temp.doc"
End Sub
```

5. إغلاق وثيقة:

نستخدم لذلك الطريقة Close التابعة لنفس التجمع ، والمثال التالي يحفظ ويغلق الوثيقة المسماة (Sales.doc):

كـــــود برمجـــــي

```
Sub CloseDocument()
    Documents("Sales.doc").Close SaveChanges:=wdSaveChanges
End Sub
```

ويمكننا إغلاق كافة الوثائق باستخدام نفس الطريقة ، والمثال التالي يغلق الوثائق دون حفظ التغييرات:

كـــــود برمجـــــي

```
Sub CloseAllDocuments()
    Documents.Close SaveChanges:=wdDoNotSaveChanges
End Sub
```

والمثال التالي يستشير المستخدم في حفظ التغييرات قبل الإغلاق:

كـــــود برمجـــــي

```
Sub PromptToSaveAndClose()
    Dim doc As Document
    For Each doc In Documents
        doc.Close SaveChanges:=wdPromptToSaveChanges
    Next
End Sub
```

6. تنشيط وثيقة:

لتنشيط وثيقة ما ، نستخدم الطريقة Activate التابعة للكائن Document. والمثال التالي ينشط الوثيقة المسماة (Sales.doc):

كـــــود برمجـــــي

```
Sub ActivateDocument()
    Documents("Sales.doc").Activate
End Sub
```

7. هل هناك وثيقة مفتوحة؟

لاختبار ما إذا كانت هناك وثيقة مفتوحة ، نقوم بالتجول برمجياً داخل التجمع Document عن طريق استعمال العبارة For Each… Next. والمثال التالي ينشط الوثيقة المسماة Sample.doc إذا كانت الوثيقة مفتوحة ، أو يفتحها إن كانت غير مفتوحة حالياً:

كـــــود برمجـــــي

```
Sub ActivateOrOpenDocument()
    Dim doc As Document
    Dim docFound As Boolean

    For Each doc In Documents
        If InStr(1, doc.Name, "sample.doc", 1) Then
            doc.Activate
            docFound = True
            Exit For
        Else
            docFound = False
        End If
```

```
    Next doc

    If docFound = False Then
        Documents.Open FileName:="Sample.doc"
    End If
End Sub
```

8. الإشارة إلى وثيقة:

عوضاً عن الإشارة إلى وثيقة ما عن طريق اسمها أو رقم فهرسها ، فإننا نستخدم الخاصية ActiveDocument التي ترجع كائناً من النوع Document الذي يشير بدوره إلى الوثيقة النشطة حالياً. المثال التالي يعرض اسم الوثيقة النشطة ، وإن لم تكن هناك وثيقة مفتوحة يعرض رسالة:

كـــــود برمجـــــي

```
    If Documents.Count >= 1 Then
        MsgBox ActiveDocument.Name
    Else
        MsgBox "No documents are open"
    End If
```

الكائن RecentFile:

ويمثل آخر الوثائق التي تم فتحها في برنامج وورد. فكلما تم فتح وثيقة يتم تسجيلها تلقائياً ضمن تجمع الـــRecentFiles.

فلفتح وعرض آخر وثائق تم فتحها وتعديلها نستخدم كائناً يسمى (RecentFile) كما بالمثال التالي:

كـــــود برمجـــــي

```
Private Sub cmdOpenAllDocs_Click()
Dim rFile As RecentFile
    For Each rFile In RecentFiles
        rFile.Open
    Next rFile
End Sub
```

المثال التالي يحدد عدد الوثائق في التجمع:

كـــــود برمجـــــي

```
RecentFiles.Maximum = 5
```

في حين أن المثال التالي يبطل عمل قائمة آخر الوثائق:

كـــــــود برمجـــــــي

```
RecentFiles.Maximum = 0
```

والمثال التالي يتأكد من حفظ الوثيقة الحالية ، فإن حفظت يقوم بتسجيلها في قائمة الـــRecentFiles:

كـــــــود برمجـــــــي

```
If ActiveDocument.Saved = True Then
    RecentFiles.Add Document:=ActiveDocument.FullName, _
        ReadOnly:=True
End If
```

والمثال التالي يفتح أول وثيقة في الـــRecentFiles:

كـــــــود برمجـــــــي

```
If RecentFiles.Count >= 1 Then RecentFiles(1).Open
```

والمثال التالي يزيد عدد الوثائق في الـــRecentFiles:

كـــــــود برمجـــــــي

```
num = RecentFiles.Maximum
If num <> 9 Then RecentFiles.Maximum = num + 1
```

التعامل مع النصوص في الوثيقة:

إدراج نص في وثيقة:

لإدراج نـــص معـــين فـــي الوثيقـــة نســـتخدم إمـــا الطريقـــة InsertAfter أو الطريقـــة InsertBefore.

والمثال التالي يدرج نصاً في نهاية الوثيقة النشطة:

كـــــــود برمجـــــــي

```
Sub InsertTextAtEndOfDocument()
    ActiveDocument.Content.InsertAfter Text:=" The end."
End Sub
```

والمثال التالي يدرج نصاً قبل التحديد:

كـــــــود برمجـــــــي

```
Sub AddTextBeforeSelection()
    Selection.InsertBefore Text:="new text "
End Sub
```

تحديد نص في وثيقة:

نستخدم الطريقة Select لتحديد عنصر ما في الوثيقة. وهناك العديد من الكائنات التي تـــدعم هذه الطريقة مثل: الكائن BookMark ، والكائن Field ، والكائن Range ، والكائن Table.

والمثال التالي يحدد أول جدول في الوثيقة النشطة:

كـــــــود برمجـــــــي

```
Sub SelectTable()
    ActiveDocument.Tables(1).Select
End Sub
```

والمثال التالي يحدد أول حقل في الوثيقة النشطة:

كـــــــود برمجـــــــي

```
Sub SelectField()
    ActiveDocument.Fields(1).Select
End Sub
```

المثال التالي يحدد أول أربع فقرات في الوثيقة النشطة. تم استخدام الطريقة Range لإنشـــاء الكـــائن Range الذي يضم الفقرات الأربع.

كـــــــود برمجـــــــي

```
Sub SelectRange()
    Dim rngParagraphs As Range
    Set rngParagraphs = ActiveDocument.Range( _
        Start:=ActiveDocument.Paragraphs(1).Range.Start, _
        End:=ActiveDocument.Paragraphs(4).Range.End)
    rngParagraphs.Select
End Sub
```

هل هناك نص محدد أم لا؟:

تقوم الخاصية Type التابعة للكائن Selection بإرجاع معلومات عن نوع التحديد. والمثال التالي يعرض رسالة إذا كان التحديد هو نقطة إدراج Insertion Point.

كـــــــود برمجـــــــي

```
Sub IsTextSelected()
    If Selection.Type = wdSelectionIP Then
       MsgBox "Nothing is selected"
    End if
End Sub
```

استرجاع نص من وثيقة:

نستخدم الخاصية Text التابعة للكائن Selection والتي تمثل الـنص المحـدد لاسـترجاع النصوص كما بالمثال:

كـــــود برمجـــــي

```
Sub ShowSelection()
    Dim strText As String
    strText = Selection.Text
    MsgBox strText
End Sub
```

والمثال التالي يقوم باسترجاع الكلمة الأولى في الوثيقة النشطة:

كـــــود برمجـــــي

```
Sub ShowFirstWord()
    Dim strFirstWord As String
    strFirstWord = ActiveDocument.Words(1).Text
    MsgBox strFirstWord
End Sub
```

التعامل مع الجداول Tables:

يمثل الكائن Table أي جدول في وثيقة وورد ، وبمجرد إنشاء جدول أو أكثر يـتم إنشـاء تجمع الجداول Tables Collection يضم كافة الجداول.

إنشاء جدول:

المثال التالي يقوم بإنشاء جدول في وثيقة:

كـــــود برمجـــــي

```
Sub CreateNewTable()
    Dim docActive As Document
    Dim tblNew As Table
    Dim celTable As Cell
    Dim intCount As Integer

    Set docActive = ActiveDocument
    Set tblNew = docActive.Tables.Add( _
        Range:=docActive.Range(Start:=0, End:=0),_
        NumRows:=3, NumColumns:=4)
    intCount = 1

    For Each celTable In tblNew.Range.Cells
        celTable.Range.InsertAfter "Cell " & intCount
        intCount = intCount + 1
    Next celTable
```

```
    tblNew.AutoFormat Format:=wdTableFormatColorful2, _
        ApplyBorders:=True, ApplyFont:=True, ApplyColor:=True
End Sub
```

إدراج نص في خلية بجدول:

المثال التالي يقوم بإدراج نص في أول خلية في أول جدول بالوثيقة النشطة. وتم استعمال الوظيفة Delete لحذف النص الموجود بها ، والوظيفة InsertAfter لإدراج النص "Cell 1,1":

كـــــــود برمجـــــــي

```
Sub InsertTextInCell()
    If ActiveDocument.Tables.Count >= 1 Then
        With ActiveDocument.Tables(1).Cell(Row:=1,
Column:=1).Range
            .Delete
            .InsertAfter Text:="Cell 1,1"
        End With
    End If
End Sub
```

استرجاع نص من خلية بجدول:

المثال التالي يقوم باسترجاع وعرض محتويات كل خلية في الصف الأول من أول جدول بالوثيقة النشطة:

كـــــــود برمجـــــــي

```
Sub ReturnTableText()
    Dim tblOne As Table
    Dim celTable As Cell
    Dim rngTable As Range

    Set tblOne = ActiveDocument.Tables(1)
    For Each celTable In tblOne.Rows(1).Cells
        Set rngTable =
ActiveDocument.Range(Start:=celTable.Range.Start, _
            End:=celTable.Range.End - 1)
        MsgBox rngTable.Text
    Next celTable
End Sub
Sub ReturnCellText()
    Dim tblOne As Table
    Dim celTable As Cell
    Dim rngTable As Range

    Set tblOne = ActiveDocument.Tables(1)
    For Each celTable In tblOne.Rows(1).Cells
        Set rngTable = celTable.Range
        rngTable.MoveEnd Unit:=wdCharacter, Count:=-1
        MsgBox rngTable.Text
    Next celTable
End Sub
```

تحويل نص إلى جدول:

المثال التالي يقوم بإدراج نصوص تفصل بينها علامة الجدولة (Tab) ومن ثم يحولها إلـــى جدول في الوثيقة:

كـــــــود برمجـــــــي

```
Sub ConvertExistingText()
    With Documents.Add.Content
        .InsertBefore "one" & vbTab & "two" & vbTab & "three"
& vbCr
        .ConvertToTable Separator:=Chr(9), NumRows:=1,
NumColumns:=3
    End With
End Sub
```

نسخ كافة الجداول بالوثيقة ولصقها في وثيقة جديدة:

المثال التالي يقوم بنسخ جميع الجداول في الوثيقة النشطة وفتح وثيقة جديدة ولصق الجـــداول المختارة فيها:

كـــــــود برمجـــــــي

```
Sub CopyTablesToNewDoc()
    Dim docOld As Document
    Dim rngDoc As Range
    Dim tblDoc As Table

    If ActiveDocument.Tables.Count >= 1 Then
        Set docOld = ActiveDocument
        Set rngDoc = Documents.Add.Range(Start:=0, End:=0)
        For Each tblDoc In docOld.Tables
            tblDoc.Range.Copy
            With rngDoc
                .Paste
                .Collapse Direction:=wdCollapseEnd
                .InsertParagraphAfter
                .Collapse Direction:=wdCollapseEnd
            End With
        Next
    End If
End Sub
```

الكائن PageSetup:

ونستفيد منه في إعداد صفحات الوثيقة. فالأمثلة التالية تعدل خصـــائص صـــفحات الوثيقـــة النشطة من حيث هوامشها ، وطريقة عرضها (طولياً – عرضياً) وغير ذلك من الخصائص الأخرى:

1. عرض الصفحة طولياً:

كـــــــود برمجـــــــي

```
ActiveDocument.PageSetup.Orientation = wdOrientPortrait
```

والخاصية OrientPortrait تحتمل القيمتين:

- ❖ `wdOrientPortrait` وتعنى عرض الصفحة طولياً.
- ❖ `wdOrientLandscape` وتعني عرض الصفحة عرضياً.

2. عرض الصفحة عرضياً:

كـــــــود برمجـــــــي

```
ActiveDocument.PageSetup.Orientation = wdOrientLandscape
```

3. تعديل هوامش الصفحة:

كـــــــود برمجـــــــي

```
With ActiveDocument.PageSetup
    .LeftMargin = InchesToPoints(0.75)
    .RightMargin = InchesToPoints(0.75)
    .TopMargin = InchesToPoints(1.5)
    .BottomMargin = InchesToPoints(1)
End With
```

4. تغيير شكل وأبعاد الصفحة:

كـــــــود برمجـــــــي

```
Private Sub cmdChange_Click()
Documents(1).PageSetup.PaperSize = wdPaperEnvelope10
End Sub
```

حيث أن الخاصية PaperSize تحتمل إحدى القيم التالية:

– wdPaperA4 – wdPaperA3 – dPaper11x17 – wdPaper10x14

wdPaperB4 – wdPaperA5 – wdPaperA4Small ، وغيرها كثير.

أمثلة متنوعة:

1. المثال التالي يوضح كيفية صنع (سمة) لخلفية الوثيقة:

كـــــــود برمجـــــــي

```
Sub CheckTheme()
    ActiveDocument.ApplyTheme "artsy 100"
    MsgBox ActiveDocument.ActiveTheme
End Sub
```

2. المثال التالي يعرض عنوان الوثيقة النشطة حالياً:

كـــــــود برمجـــــــي

```
Sub WindowCaption()
    MsgBox ActiveDocument.ActiveWindow.Caption
End Sub
```

طريقة أخرى:

كـــــــود برمجـــــــي

```
If Application.Documents.Count >= 1 Then
    MsgBox ActiveDocument.Name
Else
    MsgBox "No documents are open"
End If
```

3. المثال التالي يضبط لون خلفية الوثيقة باللون المطلوب ، شرط أن يكون العرض بتخطيط ويب:

كـــــــود برمجـــــــي

```
ActiveDocument.ActiveWindow.View.Type = wdWebView
With ActiveDocument.Background.Fill
    .Visible = True
    .ForeColor.RGB = RGB(192, 192, 192)
End With
```

4. المثال التالي يحدد صورة لتكون خلفية للوثيقة (في وضع العرض بتخطيط ويب):

كـــــــود برمجـــــــي

```
ActiveDocument.ActiveWindow.View.Type = wdWebView
ActiveDocument.Background.Fill.UserPicture _
    PictureFile:="C:\Windows\Bubbles.bmp"
```

5. المثال التالي يضبط نوع وحجم خط محتويات الوثيقة النشطة:

كـــــــود برمجـــــــي

```
Set myRange = ActiveDocument.Content
With myRange.Font
    .Name = "Arial"
    .Size = 10
End With
```

6. المثال التالي يفتح وثيقة ويضع لها كلمة سر ثم يغلقها:

كـــــــود برمجـــــــي

```
Set myDoc = Documents.Open(FileName:="C:\computer.doc")
myDoc.Password = "why"
myDoc.Close
```

7. المثال التالي يطبع محتويات وثيقة إلى الطباعة:

كـــــــود برمجـــــــي

```
Private Sub mnuprint40_Click()
Dim b As Word.Document

Set b = GetObject("c:\acc\new\reports\40.doc")
b.Activate
b.PrintOut FileName = "c:\acc\new\reports\40.doc"
End Sub
```

8. المثال التالي يقوم:

أ. بتلوين النص المحدد باللون الأزرق.

ب. بجعل الخط أسوداً عريضاً.

ج. طباعة الجملة This is some text بعد التحديد مباشرة.

كـــــــود برمجـــــــي

```
Private Sub Command1_Click()
With Selection
    .Font.Color = wdColorBlue
    .Font.Bold = True
End With
Selection.InsertAfter "This is some text."
End Sub
```

9. المثال التالي يكتب نصاً في بداية الوثيقة ومن ثم يطبعها:

كـــــــود برمجـــــــي

```
Dim rngTemp As Range

Set rngTemp = ActiveDocument.Range(Start:=0, End:=0)
With rngTemp
    .InsertBefore "Company Report"
    .Font.Name = "Arial"
    .Font.Size = 24
    .InsertParagraphAfter
End With
ActiveDocument.PrintOut
```

10. المثال التالي يعرض اسم الطابعة الافتراضية النشطة:

كـــــــود برمجـــــــي

```
MsgBox "اسم الطابعة النشطة لديك هو: " & ActivePrinter
```

11. المثال التالي يعين نوعاً معيناً من الطابعات كطابعة نشطة على المنفذ LPT1:

كـــــــود برمجـــــــي

```
Application.ActivePrinter = "HP LaserJet 4 local on LPT1:"
```

12. المثال التالي يطلب من المستخدم حفظ كافة الوثائق لأن برنامج وورد في طريقه للإغلاق:

كـــــــود برمجـــــــي

```
Dim intResponse As Integer

intResponse = MsgBox("هل تريد حفظ كافة الوثائق؟", vbYesNo)
If intResponse = vbYes Then Application.Quit _
    SaveChanges:=wdSaveChanges, OriginalFormat:=wdWordDocument
```

13. المثال التالي يبدل اللغة في الوثيقة:

كـــــــود برمجـــــــي

```
Private Sub cmdChangeLanguage_Click()
Dim x As Integer

x = MsgBox("Switch the keyboard language setting?", vbYesNo)
If x = vbYes Then wrddoc.Application.ToggleKeyboard
End Sub
```

ثانياً: برنامج أكسل Microsoft Excel:

ببساطة هو برنامج لمعالجة الجداول الإلكترونية ، ويستخدم بكثـــرة فـــي إنجـــاز العمليـــات الحسابية والإحصائية خصوصاً تلك التي تحتاج إلى تنسيق البيانات في جداول.

وللتعامل معه برمجياً من خلال فجول بيسك ، يجب تعريفه للغة أولاً ، وذلك بإدراج مكتباته في اللغة ثم تعريف متغير على أنه يمثل برنامج أكسل بكامل خصائصه ووظائفه.

إدراج مكتبات برنامج أكسل:

لإدراج مكتبات البرنامج أكسل في مشروع في فجول بيسك نتبع الخطوات التالية:

5. نفتح مشروعاً جديداً.
6. نفتح القائمة (Project) ثم نختار الأمر (References).
7. يظهر مربع الحوار الخاص بالمكتبات ، نحدد الخيار (Microsoft Excel 10.0 Object Library) بوضع علامة (صح) أمامه.
8. ننقر فوق الزر (Ok).

تعريف متغير على أنه يمثل برنامج أكسل:

نكتب الكود التالي في المكان المناسب:

كـــــــود برمجـــــــي
`Dim Excel_Pro As Excel.Application`

وفي هذا الكود ، تم تعريف المتغير Excel_Pro على أنه برنامج أكسل بالكامل ، ويمكننـــا إجراء ما يمكن إجراؤه في برنامج أكسل على هذا المتغير ، فالبرنامج يتكون مـــن مجموعـــة مـــن الأوامر ، منها ما يتعلق به كبرنامج ، ومنها ما يتعلق بالصفحات وأوراق العمل ، ومنها مـــا يتعلـــق بمكونات الورقة من خلايا ونطاقات ومخططات ورسومات وغيرها.

وكل ما ذكرته يسمى كائناً ، فالملف في أكسل كائن ، والورقة كـــائن ، والخليـــة والنطـــاق والمخططات كلها كائنات.

مصطلحات أساسية:

1. WorkBook: ويعني ملف يحتوي على ورقة عمل أو أكثر ، ويخزن في القرص باللاحقة (xls). وسنطلق عليه اسم (وورك بوك) في هذه المذكرة.
2. WorkSheet: ورقة العمل ، وهي عبارة عن مجموعة كبيرة من الصفوف والأعمدة ، تسمى نقطة تلاقي الصف بالعمود (بالخلية). وهناك عدة أنواع من أوراق العمل منها: worksheets, charts, modules, and dialog sheets.
3. Cell: الخلية ، وهي نقطة التقاء الصف بالعمود ، وتحدد الخلية من خلال رقم الصف والحرف الدال على العمود.
4. Chart: المخطط ، وهو رسم بياني يمثل قيم مجموعة من الخلايا.

ملف أكسل Excel WorkBook:

وهو ملف يمثل مجموعة من أوراق العمل WorkSheets التي تتكون من عدد كبير جداً من الصفوف Rows والأعمدة Columns.

بعض أهم خصائص الـWorkBook:

1. **ActiveSheet:** وتمثل الورقة النشطة حالياً.
2. **Password:** ويستفاد منها في وضع كلمة مرور للملف.
3. **Saved:** وتختبر هل الملف محفوظ أم لا؟
4. **Sheets:** وتمثل كافة الأوراق المفتوحة في الـــWorkBook الحالي.

بعض أهم وظائف الـWorkBook:

1. **Activate:** وتستخدم لتنشيط الملف أو ورقة العمل أو خلية أو نطاق من الخلايا...الخ.
2. **Close:** لإغلاق الـــWorkBook.
3. **Save:** لحفظ الـــWorkBook.
4. **SaveAs:** لحفظ الـــWorkBook باسم آخر.

العمليات على ملفات أكسل وأوراق عملها:

إنشاء WorkBook جديد:

لإنشاء ملف أكسل جديد برمجياً نستخدم الطريقة Add التابعة لتجمع الملفات WorkBooks، والمثال التالي يوضح ذلك:

كـــــود برمجـــــي

```
Sub AddOne()
    Workbooks.Add
End Sub
```

طريقة أخرى:

كـــــود برمجـــــي

```
Sub AddNew()
Set NewBook = Workbooks.Add
    With NewBook
        .Title = "All Sales"
        .Subject = "Sales"
        .SaveAs Filename:="Allsales.xls"
    End With
End Sub
```

وفي هذا المثال ، تم إنشاء كائن جديد ، وضبطت قيم خصائصه ، وخزن باسم (Allsales.xls).

فتح ملف أكسل برمجياً:

نستخدم الوظيفة Open التابعة لتجمع ملفات أكسل بالصورة التالية:

كـــــود برمجـــــي

```
Sub OpenUp()
    Workbooks.Open("C:\MyFolder\MyBook.xls")
End Sub
```

إغلاق ملف أكسل:

نستخدم الوظيفة Close لإغلاق الملف بالصورة التالية:

كـــــود برمجـــــي

```
Workbooks("BOOK1.XLS").Close SaveChanges:=False
```

في المثال السابق تم إغلاق الملف المشار إليه دون تخزين التغييرات.

الإشارة إلى ورقة عمل داخل الـــ(WorkBook) عن طريق رقم الفهرس:

ويتم بالصورة التالية:

كـــــــود برمجـــــــي

```
Sub FirstOne()
    Worksheets(1).Activate
End Sub
```

أما إذا أردنا الإشارة إلى أي نوع من أوراق العمل فنكتب:

كـــــــود برمجـــــــي

```
Sub FourthOne()
    Sheets(4).Activate
End Sub
```

ملاحظة / يتغير ترتيب رقم الفهرس إذا تم إضافة أو حذف ورقة عمل أو أكثر.

الإشارة إلى ورقة عمل داخل الـــ(WorkBook) عن طريق اسمها:

ويتم بالصورة التالية:

كـــــــود برمجـــــــي

```
Worksheets("Sheet1").Activate
Charts("Chart1").Activate
DialogSheets("Dialog1").Activate
```

أو بالصورة التالية:

كـــــــود برمجـــــــي

```
Sub ActivateChart()
    Sheets("Chart1").Activate
End Sub
```

حفظ العمل كصفحة ويب:

نستخدم الخاصية SaveAs لحفظ ملف الأكسل بصيغة HTML الخاصة بصفحات ويب كما بالصورة التالية:

كـــــــود برمجـــــــي

```
Sub ActivateChart()
    Sheets("Chart1").Activate
End Sub
```

الخلايا والنطاقات Cells & ranges:

1. كيفية الإشارة إلى الخلايا أو النطاقات:

إن الطريقة الشائعة لتحديد خلية أو نطاق خلايا ومن ثم إجراء بعض العمليات عليها كإدراج صيغة رياضية ما أو تنسيقها تنسيقاً معيناً ، هو استعمال عبارة برمجية تحدد النطاق ، وتغيـــر قيمـــة خاصية معينة أو تنفذ طريقة ما.

يمثل الكائن Range في لغة فجول بيسك خلية واحدة أو مجموعة من الخلايا. والمواضـــيع التالية ستوضح الطرق الشائعة في التعامل مع النطاقات Range.

أ. الإشارة إلى الخلايا والنطاقات بدلالة العمود A1:

يمكننا الإشارة إلى خلية ما أو مجموعة من الخلايا ضمن العمود A1 عن طريق الخاصـــية Range. المثال التالي يغمق لون كتابة خلايا النطاق (A1:D5):

كـــــــود برمجـــــــي

```
Sub FormatRange()
    Workbooks("Book1").Sheets("Sheet1").Range("A1:D5") _
        .Font.Bold = True
End Sub
```

والجدول التالي يوضح بعض الإشارات عن طريق A1 وباستخدام الخاصية Range:

العبـــــــارة	المعنى
Range("A1")	الخلية A1
Range("A1:B5")	النطاق من A1 وحتى B5
Range("C5:D9,G9:H16")	نطاقين مختلفين
Range("A:A")	العمود A بكامله
Range("1:1")	الصف 1 بأكمله
Range("A:C")	العمود A حتى العمود C بأكملهما
Range("1:5")	الصف 1 وحتى 5 بأكملهما
Range("1:1,3:3,8:8")	الصفوف 1 ، و3 ، و8
Range("A:A,C:C,F:F")	الأعمدة A ، C ، F

ب. الإشارة إلى الخلايا عن طريق رقم فهرسها:

نستخدم الخاصية Cells للإشارة إلى خلية ما عن طريق رقم فهرسها (عمود : صف). هـذه الخاصية ترجع الكائن Range الذي يمثل تلك الخلية. في المثال التالي الصيغة Cells(6, 1) ترجـع الخلية A6 وتضبط قيمتها بـ10:

كـــــود برمجـــــي

```
Sub EnterValue()
    Worksheets("Sheet1").Cells(6, 1).Value = 10
End Sub
```

تعمل الخاصية Cells بكفاءة عند التجول في نطاق معين للخلايا ، لأننـا نسـتطيع تعيـين متغيرات تمثل العدادات (عدادات الصفوف والأعمدة) كما هو موضح بالمثال التالي:

كـــــود برمجـــــي

```
Sub CycleThrough()
    Dim Counter As Integer
    For Counter = 1 To 20
        Worksheets("Sheet1").Cells(Counter, 3).Value = Counter
    Next Counter
End Sub
```

ج. الإشارة إلى الصفوف والأعمدة:

نستخدم الخاصية Rows التي تمثل الصفوف و/أو الخاصية Columns التي تمثل الأعمـدة للتعامل مع الصفوف أو الأعمدة المحددة. هاتان الخاصيتان ترجعان الكـائن Range الـذي يمثـل النطاق المحدد.

وفي المثال التالي ، ترجع العبارة Rows(1) الصف رقم (1) في الورقة النشطة ، بعد ذلك يتم تغميق لون كتابتها:

كـــــود برمجـــــي

```
Sub RowBold()
    Worksheets("Sheet1").Rows(1).Font.Bold = True
End Sub
```

والجدول التالي يبين طرق الإشارة إلى الصفوف والأعمدة برمجياً:

العبـــــارة	المعنى
Rows(1)	الصف الأول
Rows	كافة صفوف الورقة
Columns(1)	العمود الأول
Columns("A")	العمود الأول
Columns	كافة الأعمدة في الورقة

وللتعامل مع عدة صفوف وأعمدة في نفس الوقت ، نستخدم الطريقـــة Union التـــي تضـــم العديد من الخلايا لإحدى الخاصيتين Rows أو Columns. والمثال التالي يغمق لون كتابة الصفوف 1 ، 3 ، 5 من الورقة النشطة:

كـــــــود برمجـــــي

```
Sub SeveralRows()
    Worksheets("Sheet1").Activate
    Dim myUnion As Range
    Set myUnion = Union(Rows(1), Rows(3), Rows(5))
    myUnion.Font.Bold = True
End Sub
```

د. الإشارة إلى الخلايا المرتبطة بغيرها من الخلايا الأخرى:

إن الطريقة الشائعة للتعامل مع خلية مرتبطة بغيرها هي استعمال الخاصـــية Offset. ففـــي المثال التالي فإن محتويات الخلية التي احداثيها صفاً واحداً لأسفل ، وثلاثة أعمد لأعلى على الورقـــة النشطة مسطرة بخطين:

كـــــــود برمجـــــي

```
Sub Underline()
    ActiveCell.Offset(1, 3).Font.Underline = xlDouble
End Sub
```

هـــ. الإشارة إلى الخلايا عن طريق الكائن Range:

إذا تم إسناد متغير إلى الكائن Range ، فنه يمكننا وبسهولة التعامل مع النطاق باستخدام اسم ذلك المتغير.

فالإجراء التالي ينشيء كائناً متغيراً (myRange) ثـم يعطيـه النطـاق (A1:D5) علـى الـSheet1 في الورقة النشطة ن ويقوم ببعض المهام الأخرى:

كـــــود برمجـــــي

```
Sub Random()
    Dim myRange As Range
    Set myRange = Worksheets("Sheet1").Range("A1:D5")
    myRange.Formula = "=RAND()"
    myRange.Font.Bold = True
End Sub
```

و. الإشارة إلى كافة الخلايا على الـWorkSheet:

عندما نستخدم الخاصية Cells دون أية بارامترات ، فهذا يعني الإشارة إلى كافة الخلايا في الورقة النشطة.

والإجراء التالي يمسح بيانات كافة الخلايا:

كـــــود برمجـــــي

```
Sub ClearSheet()
    Worksheets("Sheet1").Cells.ClearContents
End Sub
```

ز. الإشارة إلى نطاقات متعددة:

نستخدم الطريقتين Range و Union للإشارة إلى مجموعة من النطاقـات ، ثـم نسـتخدم الخاصية Areas للإشارة إلى مجموعة النطاقات المختارة في الورقة النشطة.

*** استخدام الخاصية Range:**

نستطيع الإشارة إلى عدة نطاقات باستخدام الخاصية Range وذلك بوضع فواصل بين كـل إشارتين أو أكثر. والمثال التالي يمسح محتويات ثلاث نطاقات على الـSheet1:

كـــــود برمجـــــي

```
Sub ClearRanges()
    Worksheets("Sheet1").Range("C5:D9,G9:H16,B14:D18"). _
        ClearContents
End Sub
```

*** استعمال الطريقة Union:**

نستطيع ربط مجموعة من النطاقات بكائن Range واحد باستخدام الطريقة Union. والمثال التالي ينشيء الكائن Range ويسميه `myMultipleRange` ، ويعرفه على أنـه النطـاق (A1:B2) و(C3:D4) في نفس الوقت ويغمق محتوياتهما:

كـــــــود برمجـــــــي

```
Sub MultipleRange()
    Dim r1, r2, myMultipleRange As Range
    Set r1 = Sheets("Sheet1").Range("A1:B2")
    Set r2 = Sheets("Sheet1").Range("C3:D4")
    Set myMultipleRange = Union(r1, r2)
    myMultipleRange.Font.Bold = True
End Sub
```

*** استعمال الخاصية Areas:**

يمكننا استعمال الخاصية Areas للإشارة إلى النطاق المختار أو تجمع النطاقات في (اختيار متعدد المناطق). الإجراء الحالي يعد المناطق المختارة ، فإن كانت أكثر من منطقـــة واحـــدة تظهـــر رسالة تحذير:

كـــــــود برمجـــــــي

```
Sub FindMultiple()
    If Selection.Areas.Count > 1 Then
        MsgBox "Cannot do this to a multiple selection."
    End If
End Sub
```

2. التجول داخل الخلايا والنطاقات برمجياً:

عند استعمال لغة فجول بيسك ، فإننا نرغب دائماً في تنفيذ نفس الكود على خليـــة أو نطـــاق خلايا. ولعمل ذلك ، فإننا نقوم بالتجول Looping داخل هذا النطاق بحيـــث ننفـــذ كافـــة العمليـــات المطلوبة على خلايا النطاق المحدد الواحدة تلو الأخرى.

وهناك طريقة للتجول داخل نطاق خلايا معين باســـتعمال العبـــارة For… Next ، بحيـــث نستفيد من الخاصية cells.

وباستعمال الخاصية Cells ، نتجول بدلالة موقع الخلية ، أو بدلالة متغيرات نستخدمها لهذا الشأن. وفي المثال التالي ، يتم تعريف متغير كعداد ، يمثل أرقام الصفوف ، ومن ثـــم يتجـــول فـــي النطاق (C1:C20) ، ويضبط قسمة الخلية بـــ 0 إذا كانت قيمتها المطلقة أقل من 0.01:

كـــــــود برمجـــــــي

```
Sub RoundToZero1()
    For Counter = 1 To 20
        Set curCell = Worksheets("Sheet1").Cells(Counter, 3)
        If Abs(curCell.Value) < 0.01 Then curCell.Value = 0
    Next Counter
End Sub
```

وهناك طريقة أخرى للتجول داخل النطاق باستخدام العبارة For Each… Next بحيث تتعامل مع تجمع الخلايا المحددة في الخاصية range. يقوم فجول بيسك تلقائياً بإسناد متغير كائن يمثل الخلية التالية كلما نُفذ التجول (التكرار).

والإجراء التالي يتجول في النطاق (A1:D10) ويضع 0 في كل خلية تقل قيمتها عن 0.01:

كـــــود برمجـــــي

```
Sub RoundToZero2()
    For Each c In Worksheets("Sheet1").Range("A1:D10").Cells
        If Abs(c.Value) < 0.01 Then c.Value = 0
    Next
End Sub
```

ملاحظة مهمة:

إذا لم تعرف حدود النطاق الذي تريد التجول في خلاياه ، يمكنك استخدام الخاصية CurrentRegion التي ترجع لك النطاق الذي توجد به الخلية النشطة. فالإجراء التالي على سبيل المثال ، يتجول في النطاق الذي توجد به الخلية النشطة ، ويقوم بضبط قيم الخلايا التي قيمها أقل من 0.01 بقيمة الصفر:

كـــــود برمجـــــي

```
Sub RoundToZero3()
    For Each c In ActiveCell.CurrentRegion.Cells
        If Abs(c.Value) < 0.01 Then c.Value = 0
    Next
End Sub
```

3. اختيار وتنشيط الخلايا:

عندما نعمل في برنامج أكسل ، نقوم عادة باختيار خلية أو أكثر ثم نجري عليها ما نريد من عمليات. أما في حالة التعامل مع الخلايا عن طريق فجول بيسك فلا يجب تحديد الخلية قبل إجراء أي عملية عليها.

فعلى سبيل المثال ، إذا أردنا إدراج صيغة ما على الخلية D6 فلا يجب أن نحدد تلك الخلية. بل نحتاج فقط إلى إرجاع كائن Range يمثل الخلية المطلوبة ثم نعدل قيمة الخاصية Formula بحيث تحمل الصيغة المطلوب تنفيذها على محتوى تلك الخلية:

كـــــود برمجـــــي

```
Sub EnterFormula()
    Worksheets("Sheet1").Range("D6").Formula = "=SUM(D2:D5)"
End Sub
```

تحديد الخلايا على الورقة النشطة:

عند استخدام الطريقة Select لتحديد الخلايا ، يجب التنبه إلى أن التحديد يسري فقط على الورقة النشطة. فإن نفذنا الكود من الوحدة النمطية Module فستفشل الطريقة Select إلا إذا نشطنا الورقة برمجياً قبل استخدام الطريقة Select على نطاق الخلايا.

فعلى سبيل المثال ، فإن الإجراء التالي ينسخ صفاً من الـ Sheet1 إلى الـ Sheet2 في ملف أكسل النشط:

كـــــود برمجـــــي

```
Sub CopyRow()
    Worksheets("Sheet1").Rows(1).Copy
    Worksheets("Sheet2").Select
    Worksheets("Sheet2").Rows(1).Select
    Worksheets("Sheet2").Paste
End Sub
```

تنشيط خلية ضمن نطاق من الخلايا المحددة:

نستخدم الخاصية Activate لتنشيط خلية موجودة ضمن نطاق محدد من الخلايا. فهناك خلية واحدة فقط يمكن تنشيطها حتى وإن تم اختيار نطاق كامل من الخلايا. والإجراء التالي يحدد نطاقاً من الخلايا ثم ينشط خلية موجودة ضمنه دون تغيير التحديد:

كـــــود برمجـــــي

```
Sub MakeActive()
    Worksheets("Sheet1").Activate
    Range("A1:D4").Select
    Range("B2").Activate
End Sub
```

4. التعامل مع نطاقات ثلاثية الأبعاد:

إذا كنا نعمل مع نفس النطاق في أكثر من ورقة واحدة نستخدم الدالة Array لتحديد ورقتين أو أكثر. ادرس المثال التالي حيث يحدد نطاق ثلاثي من الخلايا:

كـــــود برمجـــــي

```
Sub FormatSheets()
    Sheets(Array("Sheet2", "Sheet3", "Sheet5")).Select
    Range("A1:H1").Select
    Selection.Borders(xlBottom).LineStyle = xlDouble
End Sub
```

5. التعامل مع الخلية النشطة:

تقوم الخاصية ActiveCell بإرجاع كائن Range يحدد الخلية النشطة في الورقة النشـــطة.

وبالتالي يمكن تعديل خصائص هذا الكائن الذي يمثل الخلية المطلوبة كما هو مبين بالمثال التالي:

كـــــــود برمجـــــــي

```
Sub SetValue()
    Worksheets("Sheet1").Activate
    ActiveCell.Value = 35
End Sub
```

تغيير الخلية النشطة:

نستطيع باستخدام الطريقة Activate أن نغير الخلية النشطة. ادرس المثال التالي:

كـــــــود برمجـــــــي

```
Sub SetActive()
    Worksheets("Sheet1").Activate
    Worksheets("Sheet1").Range("B5").Activate
    ActiveCell.Font.Bold = True
End Sub
```

استخدام بعض الدوال التابعة لـSheets في فجول بيسك:

يمكننا استعمال معظم الدوال التي يمكن استعمالها في أوراق عمل أكسل بسهولة في فجـــول بيسك ، وفيما يلي قائمة بأهم تلك الدوال:

الدالـــــــة	وظيفـــتها
ACOS	إرجاع قوس جيب التمام، أو معكوس جيب التمام لزاوية. قوس جيب التمام هـــو زاوية جيب تمامها عبارة عن رقم. وتكون قيمة الزاوية التـــي يـــتم إرجاعهـــا بالتقدير الدائري في النطاق 0 (صفر) إلى pi.
ACOSH	إرجاع معكوس جيب التمام لقطع زائد لرقم. يجب أن يكون الرقم أكبر مـــن أو يساوى 1.
AND	إرجاع TRUE إذا كانت كافة وسائطها TRUE ؛ وإرجاع FALSE إذا كانت هناك وسيطة واحدة أو أكثر FALSE.
ASIN	إرجاع قوس الجيب، أو معكوس الجيب لزاوية. قوس الجيب هو الزاويـــة التـــي يكون جيب زاويتها عبارة عن رقم. ويتم إرجاع الزاوية بالتقويم الـــدائري فـــي النطاق من pi/2- إلى pi/2.
ASINH	إرجاع معكوس الجيب لقطع زائد لرقم.
ATAN2	إرجاع قوس الظل أو معكوس الظل للإحداثين س وص المحددين. قوس الظـــل هو الزاوية من المحور س لخط يحتوي على نقطة الأصل (0 .0) ونقطـــة ذات الإحداثيات (س وص). يتم تقديم الزاوية بالتقدير الـــدائري بـــين pi- وpi، ولا تشمل pi-.
ATANH	إرجاع معكوس ظل الزاوية لقطع زائد. يجب أن يكون رقم الزاوية بين 1- و 1 (ليس فيه 1- و 1).
AVEDEV	إرجاع متوسط الانحرافات لنقاط البيانات عن وسطها. AVEDEV عبارة عـــن مقياس التباين في مجموعة البيانات.
AVERAGE	إرجاع متوسط الوسائط (الوسط الحسابي).
BETADIST	إرجاع دالة كثافة احتمالات بيتا التراكمية. يتم استخدام دالة كثافة احتمالات بيتـــا التراكمية لدراسة التباين النسب المئوية لشيء ما في مجموعة العينات، مثل جزء اليوم الذي يقضيه الأشخاص في مشاهدة التليفزيون.
BETAINV	إرجاع معكوس دالة كثافة احتمالات بيتا التراكمية.
BINOMDIST	إرجاع الحدود الجبرية الفردية لاحتمالية التوزيع ذات الحدين.

يتبع / استخدام بعض الدوال التابعة لـــSheets في فجول بيسك:

الدالـــــــة	وظيفـــتها
CHIINV	إرجاع معكوس الاحتمال وحيد الطرف لتوزيع كاي تربيع.
CHITEST	إرجاع اختبار الاستقلالية.
CHOOSE	لتحديد واحدة من القيم التي يصل عددها 29 كحد أقصـــى اســـتناداً إلـــى رقـــم الفهرس. فعلي سبيل المثال، إذا كانت تمثل القيمة 1 إلى القيمة 7 أيام الأسبوع، فترجع CHOOSE أحد الأيام عند استخدام رقم بـــين 1 و 7 كـــرقم فهـــرس (index_num).
CLEAN	إزالة كافة الأحرف غير القابلة للطباعة من النص. اســـتخدم CLEAN علـــى النص الذي تم استيراده من تطبيقات أخرى تحتوي على أحرف التي قد لا يـــتم طباعتها على نظام التشغيل لديك. فعلي سبيل المثال، يمكنك استخدام CLEAN لإزالة بعض تعليمات الكمبيوتر البرمجية ذات المستوى المنخفض التي تتكـــرر باستمرار في بداية ونهاية ملفات البيانات ولا يمكن طباعتها.
CONFIDENCE	إرجاع فترة الثقة لوسط مجموعة بيانات.
COUNT	حساب عدد الخلايا التي تحتوي على أرقام وأيضاً أرقام داخل قائمة الوســـائط، استخدم COUNT للحصول على عدد الإدخالات في حقـــل أرقـــام بنطـــاق أو مصفوفة الأرقام.
COUNTIF	حساب عدد الخلايا داخل نطاق يفي بالمعايير المعطاة.
DAYS360	إرجاع عدد الأيام بين تاريخين استناداً إلى سنة تتكون من 360 يوماً (اثنا عشر شهراً يتكون كلٌ منها من 30 يوم)، الذي يتم استخدامه فـــي بعـــض العمليـــات المحاسبية. استخدم هذه الدالة في حساب دفعات السداد إذا كان نظام المحاســـبة الذي تستخدمه مستنداً إلى اثنى عشر شهراً يتكون كلٌ منها من 30 يوم.
DEGREES	تحويل التقويم الدائري إلى درجات.
DMAX	إرجاع أكبر رقم في أحد أعمدة قائمة أو قاعدة بيانات تطـــابق الشـــروط التـــي تحددها.
DMIN	إرجاع أصغر رقم في أحد أعمدة قائمة أو قاعدة بيانات تطابق الشـــروط التـــي تضعها.
DSUM	لجمع الأرقام التي تطابق الشروط التي تحددها في عمود ضمن قائمة أو قاعـــدة بيانات .

يتبع / استخدام بعض الدوال التابعة لـــSheets في فجول بيسك:

الدالـــــــة	وظيفـــتها
EVEN	لإرجاع رقم بعد تقريبه لأقرب رقم صحيح زوجي.
FACT	إرجاع مضروب أحد الأرقام. مضروب الرقم يساوي 1*2*3*...* رقم.
FIND	FIND تقوم بالبحث عن سلسلة نصية (find_text) ضمن سلسلة نصية أخرى (within_text)، وإرجاع رقم موضع بدايـــة لسلســـلة المـــراد البحـــث عنهـــا (find_text)، من الحرف الأول للسلسلة المراد البحث فيهـــا (within_text). يمكنك أيضاً استخدام SEARCH للعثور على سلسلة نصية واحدة داخل سلسلة أخرى، ولكن بخلاف SEARCH، فإن FIND تكون حساســـة للأحـــرف ولا تسمح بأحرف البدل.
FORECAST	لحساب أو التنبؤ بقيمة مستقبلية باستخدام قيم موجودة. تكـــون القيمـــة المتوقعـــة عبارة عن قيمة حل لقيمة س المعطاة. القيم المعطـــاة هـــي قـــيم س وقـــيم ص الموجودة، وقيم التنبؤ بالقيمة الجديدة باستخدام الانحدار الخطي. يمكنك اســـتخدام هذه الدالة للتنبؤ بالمبيعات، ومتطلبات المخزون، واتجاهات السوق المستقبلية.
HARMEAN	لإرجاع الوسط التوافقي لمجموعة بيانات. الوسط التوافقي هو معكوس الوســـط الحسابي لمقلوب الأرقام.
ISNONTEXT	تشير إلى أي عنصر ليس نصاً. (لاحظ أن تلك الدالة تقوم بإرجـــاع TRUE إذا كانت value تشير إلى خلية فارغة.
ISNUMBER	تشير إلى رقم.
ISLOGICAL	تشير إلى أي قيمة منطقية.
ISTEXT	تشير إلى نص.
LN	لإرجاع اللوغاريتم الطبيعي لرقم. تستند اللوغاريتمات الطبيعيـــة إلـــى ثابـــت e (2.718281845904).
LOG	لإرجاع لوغاريتم رقم بالأساس الذي تحدده.
MAX	لإرجاع أكبر قيمة في مجموعة قيم.
MIN	لإرجاع أصغر رقم في مجموعة من القيم.
ODD	لإرجاع رقم يتم تقريبه إلى أقرب أعلى عدد صحيح فردي.
OR	لإرجاع TRUE إذا كانت أي من الوسائط تساوي TRUE؛ وإرجاع FALSE إذا كانت كافة الوسائط تساوي FALSE.

يتبع / استخدام بعض الدوال التابعة لـــSheets في فجول بيسك:

الدالـــــــة	وظيفـــتها
PI	إرجاع الرقم 3.14159265358979، الثابت الرياضي pi، بدقة تصل إلى 15 رقماً.
POWER	إرجاع النتيجة لرقم مرفوع إلى أس.
RADIANS	لتحويل الدرجات إلى تقدير دائري .
REPLACE	تقوم REPLACE باستبدال جزء من السلسلة النصـــية، بالاســـتناد إلـــى عـــدد الأحرف التي تحددها، بسلسلة نصية أخرى.
ROUND	لتقريب رقم بعدد محدد من الخانات.
SEARCH	تُرجع SEARCH عدد الأحرف التي عثر عندها على حرف معين أو سلســـلة نصية، بداية بـــ start_num(رقم_البدء). استخدم SEARCH لتحديـــد موقـــع الحرف أو السلسلة النصية داخل سلاسل نصية أخرى حيـــث يمكنـــك اســـتخدام الدالات MID أو REPLACE لتغيير النص.
SMALL	إرجاع ترتيب أصغر قيمة k في مجموعة بيانات. استخدم هذه الدالـــة لإرجـــاع القيم حسب ثابت نسبي محدد في مجموعة بيانات.
SUM	جمع كافة الأرقام الموجودة في نطاق من الخلايا.
SUMIF	جمع الخلايا المحددة حسب معايير معطاة.
TEXT	تحويل قيمة إلى نص في تنسيق رقمي محدد.

الخـــــاتمة

بعد هذا الجهد المضني ، والعمل الدؤوب ، وضعنا أقدامنا على أول خطوة في طريق تطوير برامج تستفيد من خاصية التكامل التي أوجدتها مايكروسوفت بين برامجها ، بحيث يمكننا الاســتفادة من إمكانيات أي برنامج من تصميم مايكروسوفت في فجول بيسك.

فقد سلطت الضوء على كيفية تعريف البرنامجين (وورد وأكسل) في لغــة فجــول بيســك ، وكيف يمكننا استعمال أهم خصائصهما ووظائفهما وأحداثهما في التعامل مع الوثائق وأوراق العمل.

وبذا ، أرجو أن أكون قد وفقت في إيصال الرسالة التي مفادها: لا صعب مع اجتهاد!!.

السلام عليكم

أبوبكر شرف الدين سويدان