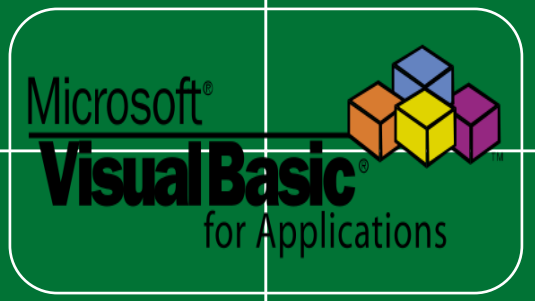
VBA NEDİR?

Visual Basic, Microsoft tarafından geliştirilen bir yazılım dilidir. VB.net ile benzer bir kodlama yapısı vardır.

Office programları ile entegre biçimde çalışan Visual Basic sistemine VBA denir. Hazır kod parçacıklarına ise Makro denir.



VBA ile çok küçük programlardan, çok büyük otomasyonlara kadar her türlü kodlamayı yapmak mümkündür. Veri analizi için de oldukça elverişlidir.

Kişisel olarak, yazılıma giriş anlamında en iyi seçeneğin VBA olduğunu düşünüyorum.

VBA KOD YAPISI

VBA'da prosedürler, SUB ile başlar ve End Sub ile biter.

```
Sub Deneme()
Dim ismim as String
ismim="Mustafa"
Msgbox ismim
End Sub
```

4 Temel Prosedür Yapısı Bulunur.

Public Sub

- Sub ile Public Sub arasında fark yoktur. Genelde Public yazmayız. Bu kodlar diğer kodlarla etkileşime girebilir.

Private Sub

- Sadece bulunduğu alanda çalışır. Diğer kodlarla etkileşime giremez.

Function

- Kullanıcı tanımlı fonksiyonlar oluşturmak için bu yapıyı kullanırız.

Option Explicit

- Module içinde birden fazla prosedür içinde aynı değişkeni kullanabilmek için global değişken oluşturulur.

VERİ TİPLERİ VE DEĞİŞKENLER

Değişken	Tip	Kapladığı Alan	Değer Aralığı
Byte	Tam Sayı	1 Byte	0-255
Integer	Tam Sayı	2 Byte	-32768 +32768
Long	Tam Sayı	4 Byte	-2,147,483,648 +2,147,483,647
Double	Rasyonel Sayı	8 Byte	16 Hane
Boolean	Mantıksal	2 Byte	True/False
String	Metin	10 Byte	Metnin Uzunluğu Kadar
Date	Tarih	8 Byte	01.01.100-31.12.9999
Object	Nesne	4 Byte	Nesne Boyutu Kadar
Variant	Herhangi	16 Byte	Değişken Boyutu Kadar

Değişken tanımlamaları tüm dillerde önemlidir. Çünkü gerek hafızayı doldurmamak gerekse daha hızlı kodlar yazmak için değişken tanımlamaları yapılmalıdır.

VERİ TİPLERİ VE DEĞİŞKENLER

Ayrıca farklı şekillerde de değişken tanımlamaları yapılabilir.

Kod	Veri Tipi
DefBool	Boolean
DefByte	Byte
DefCur	Currency
DefDate	Date
DefDbl	Double
DefDec	Decimal
DefInt	Integer
DefLng	Long
DefObj	Object
DefStr	String
DefVar	Variant

Tanım	Veri Tipi
Dim sayi%	Integer
Dim buyuksayi&	Long
Dim alttoplam#	Double
Dim para@	Currency
Dim isim$	String

Bu eğitim boyunca klasik yöntem olan deklarasyonu kullanacağız.
Dim isim as String

OPERATÖRLER

Aritmetik Operatörler

Operatör	İşlevi
^	Üs almak için
/	Bölme İşlemi için
MOD	Mod Almak için
*	Çarpma İşlemi İçin
+	Toplam İşlemi İçin
-	Çıkarma İşlemi İçin
&	Metinleri Birleştirmek İçin

Karşılaştırma Operatörleri

Operatör	İşlevi
>,>=	Büyüktür veya Büyük Eşittir
<,<=	Küçüktür veya Küçük Eşittir
=	Eşittir
<>	Eşit Değildir

Mantıksal Operatörler

Operatör	İşlevi
And	İki Mantıksal İfade Kurar
Or	İki Mantıksal İfadeden, birinin DOĞRU olmasını kontrol eder
NOT	Mantıksal Olmaması Durumunu Sorgular
LIKE	Değerin, Eşleşmesini Kontrol Eder

HÜCRE BAŞVURU YÖNTEMLERİ

Temel olarak 2 farklı hücre başvuru yöntemi bulunmaktadır.

CELLS

Cells(Satır numarası, Sütun numarası)

Cells(1,1)

A1 Hücresini belirtir.

RANGE

Range(”A1”)

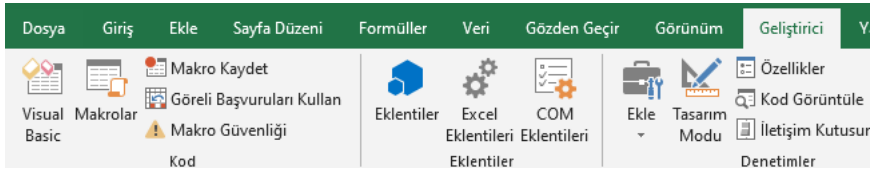
A1 hücresini belirtir. Tırnak işareti içinde sütun ismi ve satır numarası yazılır.

Range ile Cells iç içe kullanılabilir.
Range(Cells(1, 1), Cells(2, 5))
A1 ile E2 aralığını temsil eder.

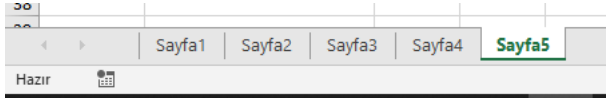
MAKRO KAYDETME

Excel’de kod yazmak ya da makro kaydetmek için geliştirici sekmesi aktif edilmelidir. Dosya/Seçenekler/Şeridi Özelleştir kısmından, Geliştirici sekmesi işaretlenerek; aktif edilebilir.

Makro kaydetme yöntemi ile hazır kodlar oluşturulabilir. Geliştirici sekmesinden Makro Kaydet tuşu kullanılabilir.

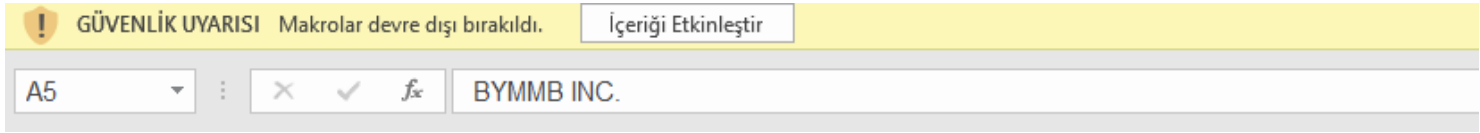


Ya da Excel sekmesinin sol altında yer alan Tuşa tıklanarak makro kaydedilebilir.



MAKRO GÜVENLİĞİ

Makro içeren bir Excel dosyası açıldığında, içeri etkinleştirme uyarısı verebilir. Bu özellikle dosya ilk açıldığında ya da içeriğinde çeşitli bağlantı kodları olduğunda çıkar. İçeriği etkinleştir diyerek kodların çalışmasına müsaade edebiliriz. Ancak bilmediğimiz birinden gelen bir dosyaysa, etkinleştirmeden önce mutlaka kodlara bakmak gerekir. VBA kodları ile bilgisayarı hacklemek mümkündür. Hacklenmemek için mutlaka buna dikkat edin.



Makrolar devre dışı bırakıldı yazısına tıklayın ve bilgilendirme sayfasından Güven Merkezi ayarlarına gidin. Buradan Güvenilen Konumlar'a tıklayın. Yeni konum ekle diyerek dosyanın olduğu konumu seçin. Artık dosya güvenilir olarak işaretlenmiş olacaktır.

NESNE MODELİ

Excel VBA nesne tabanlı bir dildir. Nesne tabanlı ne demektir? Baytlardan oluşan bilgisayar temelli objeler üzerinden kodlama yapılmasına OOP(Object Oriented Programming) denir.

Örneğin Excel programı bir nesnedir. Excel içindeki sekmeler de nesnedir. Grafikler, resimler, alanlar, hücreler… tamamı nesnelerdir. Nesnelerin belli özellikleri vardır. Bu özellikler doğrultusunda bazen sadece okuma bazen de okuma ve yazma işlemi yapılabilir.

Excel nesne hiyerarşisinde her zaman ana kod Application'dır. Bunu her yerde kullanmayız. Ancak kullanmasak bile VBA bunu arka planda default olarak tanımlar. Örneğin A1 hücresine tam referans vermek için yazılması gereken kod şöyledir;

```
Excel.Application.Workbooks("Kitap1").Worksheets("Sayfa1").Range("A1").Value = "VBA"
```

Buradaki hiyerarşiye dikkat etmek gerekiyor. Tam olarak Türkçesi şudur;

Excel uygulamasındaki Kitap1 dosyasının sayfa1 sekmesindeki A1 hücresine VBA yaz.

Eğer farklı bir sekmeye veri gönderilecekse `Worksheets("Sayfa2").Range("A1").Value = "VBA"` Ya da farklı bir kitaba gönderilecekse;
`Workbooks("Kitap2").Worksheets("Sayfa1").Range("A1").Value = "VBA"` şeklinde yazmamız gerekiyor.

TEMEL APPLICATION NESNELERİ

Activecell
Msgbox Application.Activecell.Adress -Seçili olan hücrenin adresini verir.

Activesheet
Application.Activesheet.Range(“A1”).Select -Aktif olan sekmedeki A1 hücresini seçer.

Calculate
Application.Calculate -Excel’deki hesaplamaları yaptırır. Excel seçeneklerinden formüller sekmesindeki hesaplama alanı elle olarak seçilmişse, bu kod sayesinde yeniden hesaplama yaptırılabilir.

Columns
Application.Columns(5).Select -5. Sütunu seçer.

Rows
Application.Rows(12).Select -12. Satırı seçer.

Quit
Application.Quit -Programı kapatır.

Selection
Msgbox Application.Selection.Adress -Seçili olan hücrelerin adresini verir.

Thisworkbook
Msgbox Application.Thisworkbook.Name -Çalışılan Excel dosyasının ismini verir.

Username
Msgbox Application.Username -Kullanıcı ismini verir.

TEMEL WINDOW NESNELERİ

Activate
Windows(“Kitap1”).Activate -Kitap 1 isimli dosyayı aktif eder.

ActivateNext
ActiveWindow.ActivateNext -Bulunulan kitaptan bir sonraki kitabı aktif eder.

ActivatePrevious
ActiveWindow.ActivatePrevious -Bulunulan kitaptan bir önceki kitabı aktif eder.

Activecell
Msgbox Windows(“Kitap1”).Activecell.Adress -Bulunulan hücrenin adresini verir.

Activesheet
Msgbox Windows(“Kitap1”).Activesheet.Name -Bulunulan sekmenin ismini verir.

Close
ActiveWindow.Close -Bulunulan pencereyi kapatır.

Displayworkbooktabs
Activewindow.Displayworkbooktabs=false -Sekmeleri gizler.

Displaygridlines
Activewindow.displaygridlines=false -Çizgileri kaldırır.

Displayheadings
Activewindow.displayheadings=false -Başlıkları kaldırır.

WORKSHEET/ WORKBOOK

Activate, çalışma kitabı aktif edildiğinde kodlar çalışır.
Addininstall, eklenti yüklendiğinde kodlar çalışır.
Beforeclose, çalışma kitabı kapatılmak istendiğinde kodlar çalışır.
Beforesave, çalışma kitabı kaydedildiğinde kodlar çalışır.
Open, çalışma kitabı açıldığında kodlar çalışır.

Excel açıldığında otomatik tam ekran olması için bir kod yazalım.

```
Workbook_open()
range(“a1:k20”).select
Activewindow.Zoom=True
End Sub
```

Worksheet komutlarından en çok kullandığımız yapılar Change ve Selection Change yapılarıdır.

IF THEN ELSE MANTIKSAL SINAMALAR

Tüm yazılım dillerinde olduğu gibi VBA'da da en önemli yapı taşı if bloklarıdır. Çünkü çeşitli kriterler doğrultusunda işlem yapabilmek için karar mekanizmalarını çalıştırmak gerekir. VBA'da kodlar alt alta ve satır satır yazılır. Dolayısıyla kodlar sırayla çalışır. Ancak bazı durumlarda kodları atlayabilmek mümkündür. Bu gibi işlemleri yapabilmek için neredeyse her zaman IF yapısını kullanırız.

Aslında buna ilkel tip bir yapay zeka diyebiliriz. Öğrenme kabiliyeti yoktur ancak programcısının verdiği komutları işleterek, karar mekanizmalarını oluşturabilir.

```
If Koşul Then
Eylemler
ElseIf Koşul Then
Eylemler
ElseIf Koşul Then
Eylemler
Else
End If
```

```
If Range("B2") > 1 And Range("B2") <= 100 Then
Range("C2") = Range("B2") * 1.05
ElseIf Range("B2") > 100 And Range("B2") <= 500 Then
Range("C2") = Range("B2") * 1.08
ElseIf Range("B2") < 1 Then
Range("C2") = "Prim Yok"
Else
Range("C2") = Range("B2") * 1.1
End If
```

SELECT CASE MANTIKSAL SINAMALAR

Select Case yapısı özü itibariyle If Then Else yapısına benzerdir. Kullanımı bence daha kolaydır. Ancak her durumda If yapısını ikame edemez.

SCALA			
PUAN	STATÜ		**55**
0-30	FF		**CC**
31-50	DD		
51-70	CC		
71-90	BB		
91-100	AA		

Böyle bir tablodan, nota denk gelen harfi bulmak için yazılacak kod;

```
a = Range("D2")
Select Case a
Case 0 To 30
Range("D3") = "FF"
Case 31 To 50
Range("D3") = "DD"
Case 51 To 70
Range("D3") = "CC"
Case 71 To 90
Range("D3") = "BB"
Case 91 To 100
Range("D3") = "AA"
Case Else
Range("D3") = "Hatalı Puan"
End Select
```

MESSAGE VE INPUTBOX

Mesaj kutuları interaktif programlama için çok önemlidir. Bazen kullanıcıya bilgi vermek için bazen kullanıcıya seçim yaptırmak için bazen de kodun akıbetini belirlemek için mesaj kutuları kullanılır. VBE içinde msgbox yazıp boşluk tuşuna bastığımızda, karşımıza mesaj kutusu söz dizimi çıkar.

Prompt yazan kısım kullanıcıya metin olarak gösterilecek şeydir. 4 farklı temel mesaj kutusu vardır. Uyarı şeklinde verilen mesaj;

```
MsgBox "Bu işlemi yapamazsınız.", vbCritical
MsgBox "Bu işlemi yapamazsınız.", vbInformation
MsgBox "Bu işlemi yapamazsınız.", vbQuestion
MsgBox "Bu işlemi yapamazsınız.", vbYesNo
```

Aynı zamanda birden fazla tipi aynı anda da gösterebiliyoruz. Bunun için araya + işareti koymamız gerekiyor.

MsgBox "Bu işlemi yapamazsınız.", vbCritical + vbYesNo

```
cevap = MsgBox("Program çalışmaya devam etsin mi?", vbYesNo)
If cevap = vbYes Then
MsgBox "Program devam ediyor"
Else
MsgBox "Program durduruldu"
End If
```

MESSAGE VE INPUTBOX

Inputboxlar kullanıcından doğrudan bilgi istediğimiz araçlardır. Buraya girilen veriler, yine kodu gidişatını belirler.

```
girdi = InputBox("Bir sayı girin", "Sayı Gir")
MsgBox girdi
```

Basit bir kod ile inputbox değerini işleme alalım.

```
girdi = InputBox("Bir sayı girin", "Kare Al")
MsgBox girdi ^ 2
```

FOR NEXT

For Next döngüsü, en basit ve en sık kullanılan döngü tipidir. Herhangi bir koleksiyon ihtiyacı yoksa ben genel olarak bu döngü tipini kullanırım. Tüm yazılım dillerinde benzer bir döngü tipi bulunmaktadır. Bu nedenle, bu konuyu net olarak anladığımızda genel olarak yazılım mantığını kavramış oluruz.

Döngüler belli bir değerle başlayıp yine belli bir değerle sonlanır. Örneğin;

For i= 1 to 10 dediğimiz zaman; i isimli bir değişkeni 1 ile başlatıp 10 ile sonlandırıyoruz.

Tam olarak syntax şöyledir.

For i= 1 to 10

‘yapılacak işlemler

Next i

Döngüyü geriye doğru çalıştırmak için;

```
For i= 10 To 1 Step -1
Cells(i,1)=i*10
Next i
```

DO WHILE LOOP

Aslında tüm döngü yöntemlerinin çalışma mantığı aynıdır. Belli bir koşul içinde hareket ederler. Do while loop döngüsü de for next döngüsü gibi çalışır ancak tabiki syntax farklıdır.

```
Dim i As Integer
i = 1
Do While i < 25
Cells(i, 1) = i
i = i + 1
Loop
```

DO LOOP WHILE

Do while loop ile birebir çalışma mantığı aynıdır. Sadece kod yapısı farklıdır.

Excel kitabımıza 5 yeni sekmeyi döngü ile ekleyelim.

```
Dim i As Integer
i = 1
Do
Sheets.Add
i = i + 1
Loop While i < 6
```

WHILE WEND

Do while loop ile benzerdir. Örnek bir kod yazarak bakalım. Ekrana 7 tane mesaj kutusu çıkaralım.

```
Dim i As Integer
i = 1
While i < 8
MsgBox i
i = i + 1
Wend
```

FOR EACH NEXT

For Next döngüsünden sonra en sık kullanılan 2. Döngü tipidir. Bu döngünün yaygın olarak kullanım alanı; koleksiyonlar arasında işlem yapmasıdır. Koleksiyonlar nelerdi? Aynı özellikler sahip nesnelerdi.

Excel sayfasına birkaç tane şekil ekleyelim. Makro kaydetme yöntemini açalım ve şekillerin rengini değiştirelim.

```
Dim shp As Shape
For Each shp In ActiveSheet.Shapes
shp.Fill.ForeColor.RGB = RGB(255, 255, 0)
Next
```

HATA AYIKLAMA

Kullanıcı kaynaklı ya da sistemsel hataları kodlardan kurtarmak için goto deyimi çok önemlidir.

Kod içinde bir çıkış yolu açarız. Kod hata ile karşılaştığında bu çıkış yolunu kullanır. Bazen kod yazarken tüm olasılıkları düşünemeyiz. Bu gibi durumlar için goto yapısı bir Else ifadesi gibidir. Aslında temel düzeyde ilkel bir yapay zekadır.

```
Sub hata_bul()
For i = 1 To 25
If IsNumeric(Cells(i, 1)) Then
Cells(i, 2) = Cells(i, 1) / 2
Else
GoTo kurtul
End If
kurtul:
End Sub
```

HATA AYIKLAMA

Bazen hata ile karşılaşılsa bile kodun devam etmesi istenebilir. Bunun için On Error Resume Next ifadesi kullanılır.

```
Sub hata_say()
On Error Resume Next
For i = 1 To 25
Cells(i, 2) = Cells(i, 1) / 2
If Err.Number <> 0 Then
hatasayisi = hatasayisi + 1
Err.Clear
End If
Next i
MsgBox hatasayisi
End Sub
```

HATA AYIKLAMA

Bazen kodların tamamının çalışmasını beklemek istemeyiz. Özellikle döngüler çalışırken belli bir noktada kodu sonlandırmak isteyebiliriz. For Next yapısında Exit For, Do yapılarında ise Exit Do komutunu kullanırız. Eğer iç içe döngüler varsa her biri için Exit yapısını kullanmamız gerekiyor. Ayrıca prosedürü tamamen bitirmek içinse Exit Sub kullanıyoruz.

```
For i = 1 To 8
If IsNumeric(Cells(i, 1)) Then
Cells(i, 2) = Cells(i, 1) * 5
Else
MsgBox i & " Satırında Metinsel ifade bulundu"
Exit For
End If
Next i
```

ONTIME METODU

On time aslında döngü değildir. Herhangi bir nesne ya da koleksiyon içinde işlem yapmaz. Daha karmaşık olarak, kodlar arasında döngü oluşturur. Mesela 2 tane kod prosedürü oluşturduğumuzda, kodlar arası çağırma işlemi yaparak sonsuz döngü oluşturabiliriz. Buna en güzel örnek, saat oluşturmaktır. Saatin saniyelerinin sürekli artması ancak böyle bir yöntemle mümkün olur.

Bir hatırlatma mesajı uyarısı oluşturalım.

```
Sub hatirlatma()
MsgBox "Vakit Geldi."
End Sub

Sub zamanlama()
Application.OnTime TimeValue("00:00:00"), "hatirlatma"
End Sub
```

DİZİLER

Tüm yazılım dillerinde bulunan bir kavramdır. Litaratürdeki ismi Array'dir.

Dizim=Array("elma","armut","kiraz","karpuz") şeklinde en basit tanımlama yapılabilir. Bu manuel tanımlama için yapılan yöntemdir. Bir diğer manuel tanımlama yöntemi ise doğrudan index içine verileri aktarmaktır.

```
Dim dizim(3) As String
dizim(0) = "ali"
dizim(1) = "mehmet"
dizim(2) = "mert"
dizim("3") = "ata"
```

Locals windowdan baktığımızda dizi elemanlarının tanımlandığını görebiliriz.

Dim dizim(1 to 10) as String

Bu tanımlar tek boyutlu diziler için geçerlidir. Excel üzerinden anlatacak olursak, tek sütun ya da tek satırdan oluşan bir yapıdır diyebiliriz.

Çok boyutlu dizilerde ise tanımlama farklıdır.

Dim dizim(2,5) as String

Bu tanımlama, dizi'nin 2 satır ve 5 sütundan oluştuğunu söylemektedir.

Benzer şekilde 3 boyutlu dizi tanımlamak içinse;

Dim dizim(2,5,4) as String diyerek 3 boyutlu dizi tanımlayabiliriz.