VBA voor Doe het Zelvers Deel 8

Handleiding van Helpmij.nl

Auteur: leofact

Juli 2014

Vorige aflevering

In de vorige aflevering werd het maken van keuzes besproken door middel van het gebruik van If Then Else en ElseIf. Ook **Select Case** en **Choose** werden besproken. Als laatste werd het maken van sprongen binnen de code behandeld met de mogelijke gevaren daarvan.

In deze aflevering

Dit keer draait het om het opzetten van lussen of Loops. Onder andere For Next en Do While worden behandeld. Daarbij worden tips gegeven om de Loops te versnellen. De code is te vinden in deze bijlage. Hierin zijn twee demomacro's opgenomen.

Loops

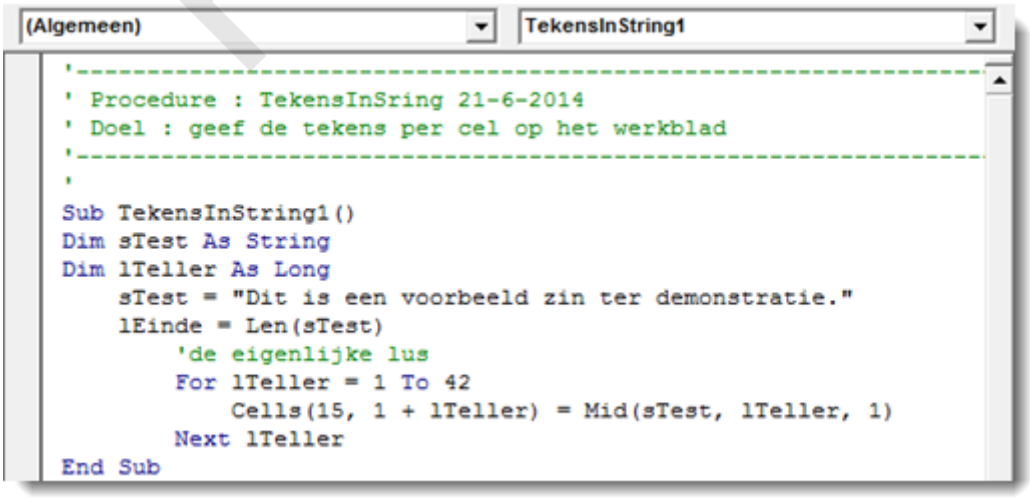
Een Loop (spreek uit loep) is een zich herhalend stukje code. Dit om een bepaalde handelingsreeks een aantal keer achterelkaar uit te voeren. In grote lijnen zijn er twee soorten Loops. Bij de eerste soort, de For Next wordt het aantal herhalingen gespecificeerd. Dat kan d.m.v. een getal of met gebruik van een variabele. Een afgeleide hiervan is de For Each. Hierbij worden alle objecten binnen een bepaald gebied of bereik langs gelopen. Dat kunnen alle cellen binnen een bepaald bereik zijn of alle werkbladen in een werkboek. Bij de tweede soort wordt de Loop herhaald tot er aan een specifieke voorwaarde wordt voldaan. Bijvoorbeeld door Do While Loop. Dit wil zeggen; herhaal zolang er aan de voorwaarde wordt voldaan. VBA doorloopt de lussen relatief traag. Door goed te programmeren en overbodige handelingen uit te sluiten kan er een flinke tijdwinst worden behaald. Hierover meer aan het einde van dit artikel.

Handig

Met een druk op CTRL + Break wordt een Loop onderbroken. Dat kan heel prettig zijn wanneer VBA in een oneindige Loop is terecht gekomen door een programmeerfout.

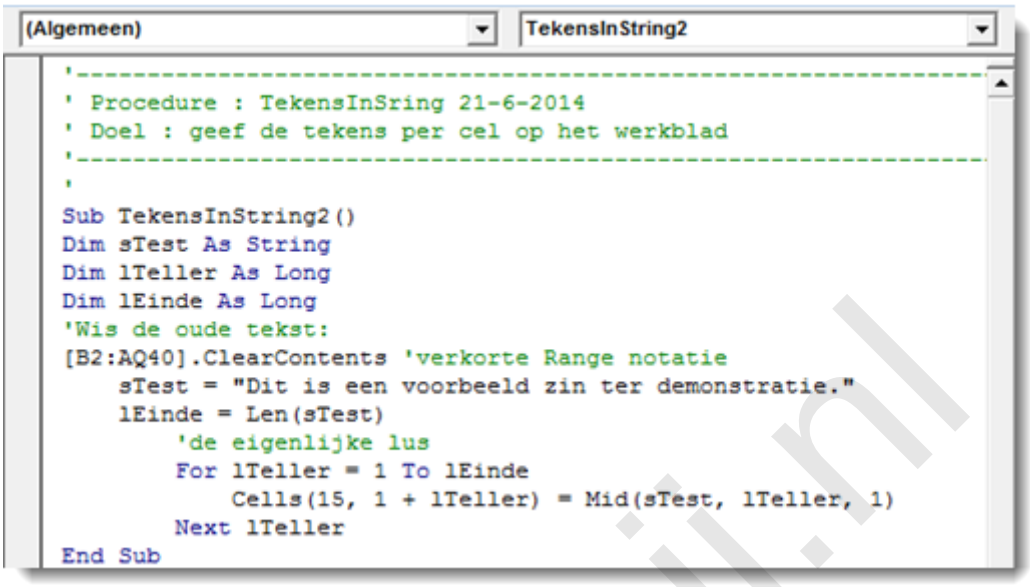
For Next

For Next wordt gebruikt als het aantal herhalingen (iteraties) van te voren vaststaat. De teller wordt hierbij automatisch opgehoogd. In het onderstaande voorbeeld wordt een string doorlopen en de letters worden per cel op het werkblad gezet:



Met variabele

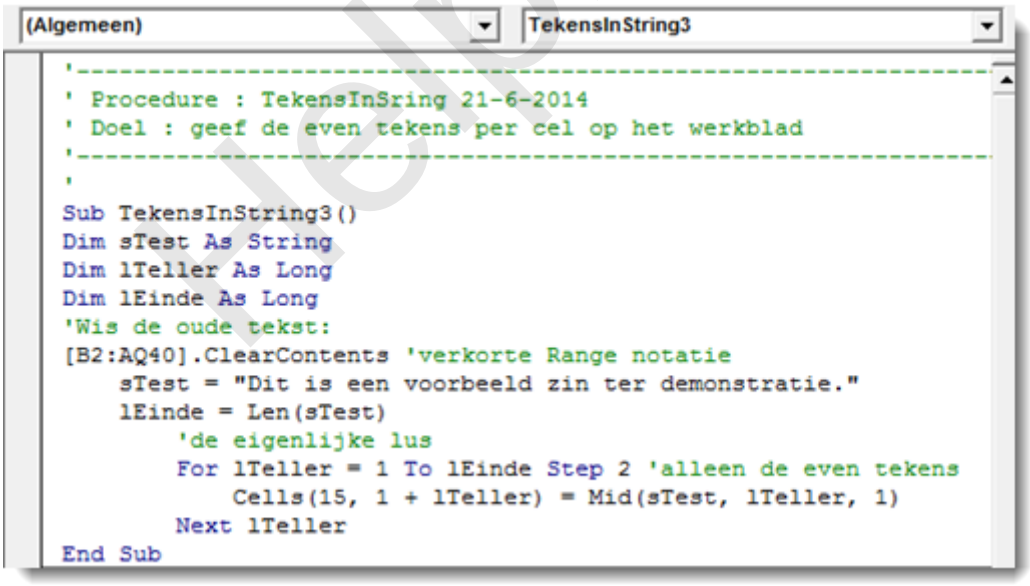
Het vorige voorbeeld werkt, maar het is natuurlijk niet handig wanneer je precies moet tellen hoeveel tekens de string bevat. Het is daarom handiger om een variabele in te zetten:




```
'---------------------------------------------------------------------
' Procedure : TekensInSring 21-6-2014
' Doel : geef de tekens per cel op het werkblad
'---------------------------------------------------------------------
'
Sub TekensInString2()
Dim sTest As String
Dim lTeller As Long
Dim lEinde As Long
'Wis de oude tekst:
[B2:AQ40].ClearContents 'verkorte Range notatie
    sTest = "Dit is een voorbeeld zin ter demonstratie."
    lEinde = Len(sTest)
        'de eigenlijke lus
        For lTeller = 1 To lEinde
            Cells(15, 1 + lTeller) = Mid(sTest, lTeller, 1)
        Next lTeller
End Sub
```

Step

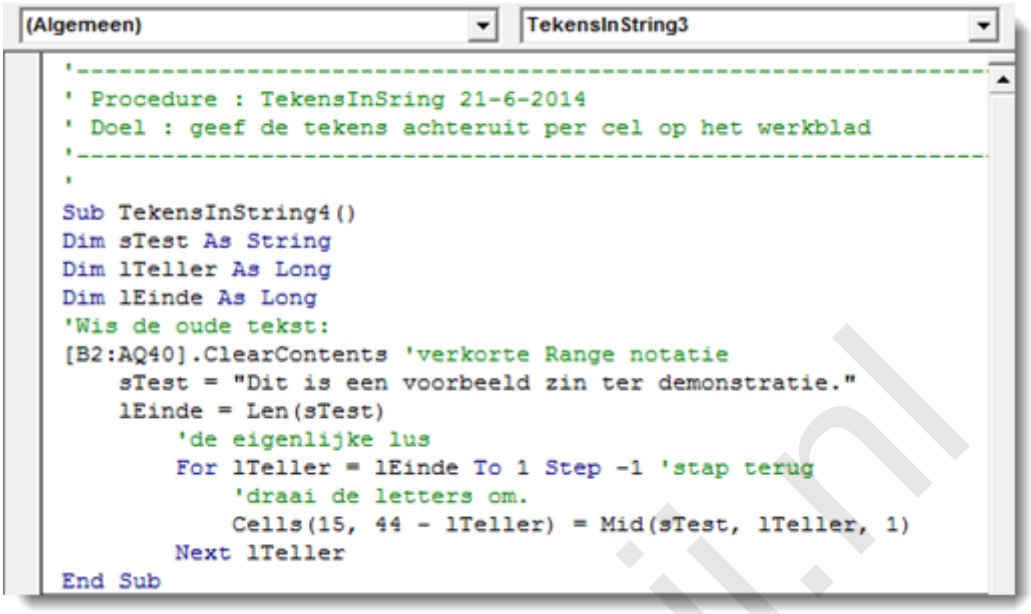
De stapgrootte kan worden ingesteld met behulp van Step. In het volgende voorbeeld worden alleen de even tekens weergegeven:




```
'---------------------------------------------------------------------
' Procedure : TekensInSring 21-6-2014
' Doel : geef de even tekens per cel op het werkblad
'---------------------------------------------------------------------
'
Sub TekensInString3()
Dim sTest As String
Dim lTeller As Long
Dim lEinde As Long
'Wis de oude tekst:
[B2:AQ40].ClearContents 'verkorte Range notatie
    sTest = "Dit is een voorbeeld zin ter demonstratie."
    lEinde = Len(sTest)
        'de eigenlijke lus
        For lTeller = 1 To lEinde Step 2 'alleen de even tekens
            Cells(15, 1 + lTeller) = Mid(sTest, lTeller, 1)
        Next lTeller
End Sub
```

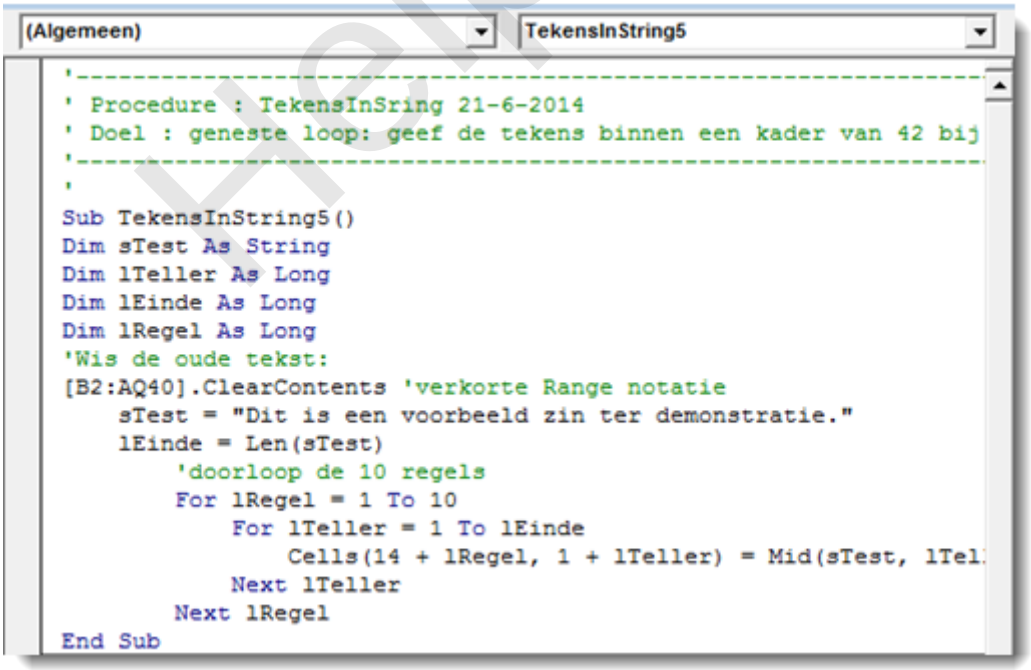
Step -1

Ook een negatieve stap (achteruit) is mogelijk:



Geneste Loop

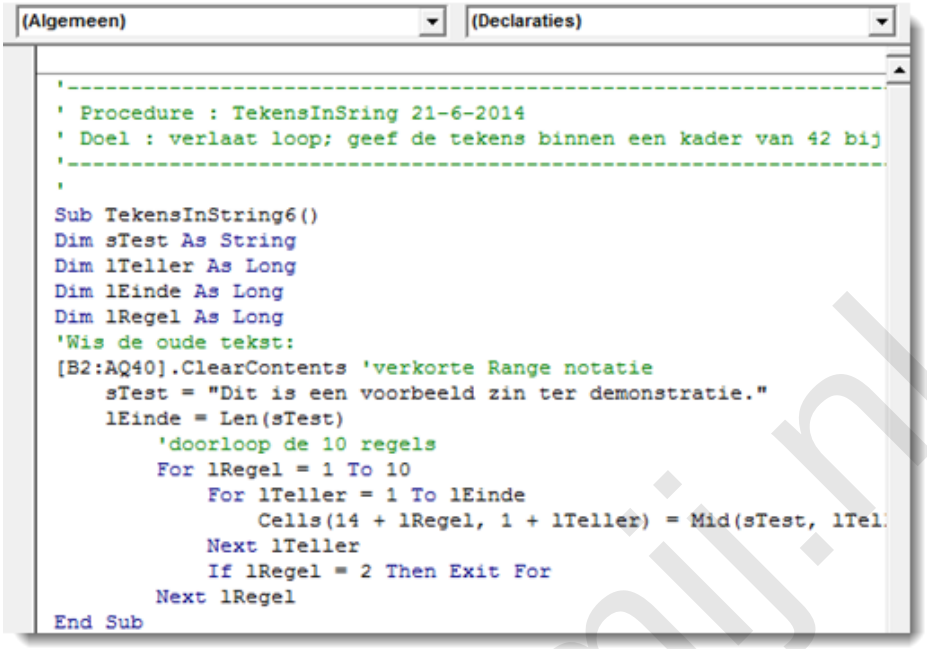
Met een variabele is het mogelijk om de string zo lang te maken als je wilt. Deze gaat dan natuurlijk wel buiten het aangegeven bereik lopen en dat willen we niet. Het is mogelijk om binnen de eerste Loop een tweede Loop te starten. Dit wordt een geneste Loop genoemd en kan verschillende niveaus diep gaan. In het volgende voorbeeld wordt de eerder gebruikte string 10 keer herhaald:



De procedure omvat maar 420 stapjes. Toch gaat deze al merkbaar trager lopen. Ook op een moderne snelle PC zal dat merkbaar zijn. Later volgen er tips om dit te versnellen.

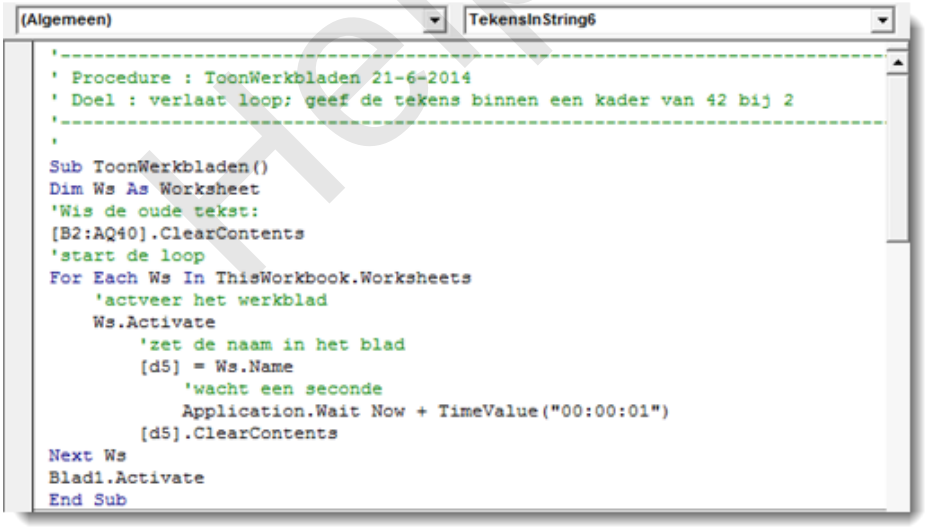
Exit For

Het is mogelijk om de Loop voortijdig te verlaten wanneer er aan een bepaalde voorwaarde wordt voldaan:



For Each

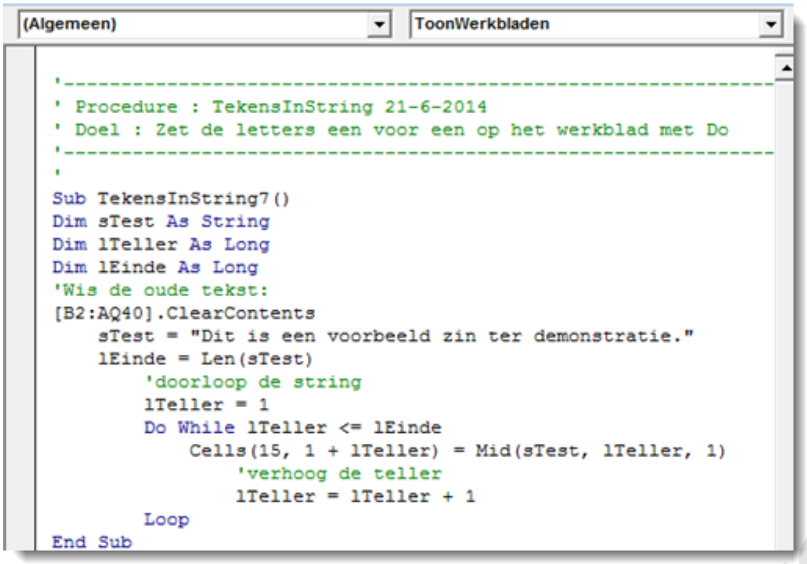
Bij de For Each Loop wordt er een aantal objecten doorlopen. In het volgende voorbeeld zijn dat alle werkbladen van dit werkboek:



In deze procedure is een wachttijd ingebouwd van telkens 1 seconde om het activeren zichtbaar te maken. Hiervoor is het Wait statement gebruikt.

Do While

De Do While Loop Loopt zolang er aan een bepaalde voorwaarde wordt voldaan. Bij deze Loops wordt er geen teller opgehoogd. Indien nodig moet dat dus “handmatig” worden ingebouwd:



Deze Loop blijft lopen zolang de teller kleiner of gelijk is aan de lengte van de string. Dit wordt gecheckt met de operator kleiner of gelijk aan: **<=**
Deze check wordt aan het begin van de Loop gedaan, dus voor de verschillende opdrachten.

Loop While

De Loop While gaat verder zolang er aan de voorwaarde wordt voldaan. Dit wordt aan het eind van de loop, dus na de opdrachten gecheckt:

In de bovenstaande macro gaat de Loop door tot het teken "z" aan de beurt is geweest. Daarna stopt de Loop. De "z" wordt dus nog wel weergegeven. Door gebruik te maken van de specifieke eigenschappen van de verschillende Loops kan er flexibel worden geprogrammeerd. Hierbij is het mogelijk om verschillende Loops te nesten en de code toch goed leesbaar te houden. In de volgende procedure worden een paar Loops gecombineerd; de tekst van de ondertussen overbekende string wordt in een cirkel van buiten naar binnen letter voor letter geplaatst. Hierna worden alle e-tjes in de tekst rood gekleurd. Om dit te tonen worden alle stappen met Select zichtbaar gemaakt. Dat zorgt voor een langzaam lopende routine. Mogelijk geeft Excel er zelfs de brui na een aantal rondjes. Het

pauzeert dan en geeft vervolgens de rest in een keer weer. Zie hiervoor de demo in de bijlage. De code van deze routine staat hieronder:

(Algemeen) TekensInStringDemo

```
'---------------------------------------------------------------------
' Procedure : TekensInString 21-6-2014
' Doel : doorloop de range in een cirvel van buiten naar binnen
'---------------------------------------------------------------------
'
Sub TekensInStringDemo()
Dim sTest As String
Dim X As Long
Dim Y As Long
Dim lHorLi As Long
Dim lHorRe As Long
Dim lVertBo As Long
Dim lVertBe As Long
Dim r As Range

'maak de rang leeg en kleur de tekens zwart
[B2:AQ40].ClearContents
[B2:AQ40].Font.ColorIndex = 1
    sTest = "Dit is een voorbeeld zin ter demonstratie."
' zet de begin positie in de variabelen
lHorLi = 2
lHorRe = 43
lVertBo = 15
lVertBe = 40
'Start de centrale loop
Do
    ' Ga naar rechts langs de X as
    For X = lHorLi To lHorRe
    'doorloop de string met X-1
        Cells(lVertBo, X).Select
        Cells(lVertBo, X) = Mid(sTest, X - 1, 1)
    Next X
    'Ga verticaal verder langs de Y as
    lVertBo = lVertBo + 1
    For Y = lVertBo To lVertBe
       Cells(Y, X - 1).Select
        Cells(Y, X - 1) = Mid(sTest, X - 2, 1)
    Next Y
    'Ga Horzontaal verder van Re naar Li
    lHorRe = lHorRe - 1
    For X = lHorRe To lHorLi Step -1
        'doorloop de string met X-1
        Cells(lVertBe, X).Select
        Cells(lVertBe, X) = Mid(sTest, X - 1, 1)
    Next X
    'ga verder naar boven
    lVertBe = lVertBe - 1
    For Y = lVertBe To lVertBo Step -1
        Cells(Y, X + 1).Select
        Cells(Y, X + 1) = Mid(sTest, X, 1)
    Next Y
    'maak een stapje naar links en ga verder
    lHorLi = lHorLi + 1
    'stop wanneer het midden gevuld is
Loop While lHorLi < 15
'kleur vervolgens alle e-tjes in het bereik rood
For Each r In Range("B15:AQ40")
    If r.Value = "e" Then r.Font.ColorIndex = 3
Next r

End Sub
```

Do's en dont's

- Zorg voor duidelijke namen voor de variabelen en structureer de Loops op een manier die voor jezelf goed is te begrijpen.
- Vermijd het gebruik van Select of Activate. Gebruik van deze statements is vrijwel nooit nuttig en het vertraagt alleen maar de uitvoering. Dit geldt zeker als er ook van de Eventmacro's (o.a. SelectionChange en/of WorksheetChange) gebruik gemaakt wordt. De Events worden dan iedere keer afgevuurd. Behalve dat dit vertraagt, kan dit allerlei onvoorspelbare neveneffecten veroorzaken. Gebruikte variabelen kunnen hierdoor bijvoorbeeld onverwachte waarden krijgen.
- Wanneer er in een Loop de inhoud van cellen wordt aangepast kan de Loop worden versneld door de schermverversing uit te zetten met:
  Application.ScreenUpdating = False
  De schermverversing wordt bij het beëindigen van de Macro automatisch weer aangezet door Excel.
- Het eerder genoemde probleem van onverwacht springen naar Eventmacro's kan worden ondervangen door de Events uit te zetten met:
  Application.EnableEvents = True
  De Events moeten altijd weer worden aangezet voordat de controle weer teruggeven wordt aan de gebruiker. Anders blijven de Events uitgeschakeld. Doe dit door eigenschap weer op True te zetten.
- Wanneer er veel gerekend wordt kan het helpen om automatisch herberekenen tijdelijk uit te zetten. Dat gaat met de volgende code:
  Application.Calculation = xlCalculationManual
  Herberekenen moet altijd weer worden aangezet met:
  Application.Calculation = xlCalculationAutomatic

Met deze laatste tips komt deze aflevering tot een einde.

Samenvatting:

Het doel van Loops en de verschillende manieren om deze op te zetten werd besproken aan de hand van verschillende voorbeelden. Daarnaast werden tips gegeven om de Loops te versnellen en ongewenste fouten te voorkomen.

Volgende aflevering

De volgende keer wordt het nut van foutafhandeling besproken en hoe je dit het beste kunt opzetten. Daarbij wordt gebruikt gemaakt van MZTools.