1. Munkalapok neveinek listázása

Ez a feladat egy olyan makró létrehozását kérte, amely listázza az összes munkalap nevét egy adott munkafüzetben. Az alábbi makró bemutatja, hogyan lehet végrehajtani ezt a feladatot.

```
Sub MunkalapokListazasa()
    Dim ws As Worksheet
    Dim lista As String
    lista = "Munkalapok listája:" & vbCrLf
    For Each ws In ThisWorkbook.Worksheets
        lista = lista & ws.Name & vbCrLf
    Next ws
    MsgBox lista, vbInformation, "Munkalapok"
End Sub
```

Hogyan Működik?

- A `Sub MunkalapokListazasa()` egy eljárás, amely nem vár visszatérési értéket.
- `Dim ws As Worksheet` - Egy változót deklarálunk, amely a munkafüzet minden egyes munkalapjára hivatkozik a ciklus során.
- `Dim lista As String` - Egy szöveges változót hozunk létre, amelyben összegyűjtjük a munkalapok neveit.
- A `For Each` ciklus végigmegy a jelenlegi munkafüzet összes munkalapján.
- A `lista` változóhoz hozzáadjuk a munkalap nevét, majd egy új sort, hogy elkülönítsük a neveket.
- Végül, a `MsgBox` függvény segítségével megjelenítjük az összegyűjtött munkalapneveket egy üzenetablakban.

2. Duplikált értékek keresése és kiemelése makróval

A feladat egy olyan makró létrehozása, amely megtalálja és kiemeli a duplikált értékeket egy adott tartományban az Excel munkalapon.

Sajnos függvénnyel nem lehet olyan cella paramétereit módosítani, ami más mint a függvény meghívásának cellája. Ezért a használat előtt ki kell jelölni azt a tartományt amit vizsgálni szeretnénk.

```
Sub DuplikaltakKiemelése()
    Dim tartomány As Range
    Set tartomány = Selection
    Dim cell As Range
```

```
    Dim ellenőrző As Object
    Set ellenőrző = CreateObject("Scripting.Dictionary")
    ' A tartomány celláinak bejárása
    For Each cell In tartomány
        If Not cell.Value = "" Then
            If ellenőrző.Exists(cell.Value) Then
                ' Duplikált érték kiemelése piros háttérszínnel
                cell.Interior.Color = RGB(255, 0, 0)
            Else
                ellenőrző.Add cell.Value, Nothing
            End If
        End If
    Next cell
End Sub
```

Hogyan Működik?

- A makró először meghatározza a felhasználó által kijelölt tartományt a `Set tartomány = Selection` utasítással.
- Egy `Scripting.Dictionary` objektumot hoz létre, amelyben csak különböző értékek tárolhatók.
- Végigiterál a tartomány összes celláján.
- Ellenőrzi, hogy a cella értéke üres-e. Ha nem, megvizsgálja, hogy az érték szerepel-e már a Dictionary-ben.
    - Ha az érték már szerepel (azaz duplikált), a cella hátterét pirosra színezi.
    - Ha az érték még nem szerepel, hozzáadja a Dictionary-hoz.

3. Dinamikus Tartományok Kezelése

Ez a feladat egy makró létrehozását valósítja meg, amely képes dinamikusan kezelni és frissíteni egy tartomány méretét az aktív munkalapon.

```
Sub DinamikusTartomanyFrissites()
    Dim tartomany As Range
    Dim utolsoSor As Long, utolsoOszlop As Long
    Dim munkalap As Worksheet
    Set munkalap = ActiveSheet
    With munkalap
        utolsoSor = .Cells(.Rows.Count, "A").End(xlUp).Row
        utolsoOszlop = .Cells(1, .Columns.Count).End(xlToLeft).Column
        Set tartomany = .Range(.Cells(1, 1), .Cells(utolsoSor,
utolsoOszlop))
    End With
    tartomany.Select
```

```
    MsgBox "A dinamikus tartomány kijelölve: " & tartomany.Address,
vbInformation, "Tartomány Frissítve"
End Sub
```

Hogyan Működik?

1. A makró először beállítja a munkalapot az aktív munkalapra (`ActiveSheet`).
2. Az `utolsoSor` és `utolsoOszlop` változókat használva megkeresi az utolsó nem üres cellát a “A” oszlopban és az 1. sorban, hogy meghatározza a tartomány határait.
3. Az `.End(xlUp).Row` és `.End(xlToLeft).Column` metódusok segítségével határozza meg az utolsó sor és oszlop számát.
4. Ezután létrehoz egy `Range` objektumot, amely az első cellától (A1) az utolsó nem üres celláig tart.
5. Végül kijelöli ezt a tartományt és megjelenít egy üzenetablakot, amely tájékoztatja a felhasználót a frissített tartomány címéről.

4. Munkalapok közötti adatátvitel

Ez a feladat egy makrót hoz létre, amely adatokat másol egyik munkalapról a másikra. A példa egyszerűsített, amely egy előre meghatározott tartomány adatait másolja át.

Ellenőrizzük, hogy van-e Munka1 és Munka2 elnevezésű munkalapunk. ha nincs hozzuk létre őket.

```
Sub AdatatvitelMunkalapokKozott()
    Dim forrasMunkalap As Worksheet
    Dim celMunkalap As Worksheet
    Dim masolandoTartomany As Range
    Dim celTartomany As Range
    ' Munkalapok és tartományok beállítása
    Set forrasMunkalap = ThisWorkbook.Sheets("Munka1")
    Set celMunkalap = ThisWorkbook.Sheets("Munka2")
    ' Másolandó tartomány beállítása a Forrás munkalapon
    Set masolandoTartomany = forrasMunkalap.Range("A1:A10")
    ' A cél tartomány beállítása a Cél munkalapon
    ' Itt feltételezzük, hogy a cél tartomány azonos méretű és helyen van,
mint a forrás
    Set celTartomany = celMunkalap.Range("A1:A10")
    ' Adatok másolása
    masolandoTartomany.Copy Destination:=celTartomany
    MsgBox "Adatok átmásolva a '" & forrasMunkalap.Name & "' munkalapról a
'" & celMunkalap.Name & "' munkalapra.", vbInformation, "Adatátvitel Kész"
End Sub
```

Hogyan Működik?

1. Először is, beállítjuk a forrás és cél munkalapokat, ahol a forrás és cél munkalapok neveit közvetlenül a kódban adjuk meg.
2. A másolandó tartományt beállítjuk a forrás munkalapon, itt az “A1:A10” tartományt használjuk példaként.
3. Hasonlóképpen, megadjuk a cél tartományt a cél munkalapon, ami ebben az esetben azonos méretű és azonos helyen van, mint a forrás tartomány.
4. A `.Copy` metódust használva másoljuk a forrás tartomány adatait a cél tartományba.
5. Egy üzenetablak tájékoztatja a felhasználót az adatátvitel sikerességéről.

A VBA-ban a zárójelek használata opcionális, amikor szubrutinokat (Sub) hívunk meg, és nincsenek visszatérési értékek. Ha a szubrutin vagy függvény paramétereit közvetlenül adjuk meg anélkül, hogy változókat vagy kifejezéseket használnánk, akkor a zárójelek elhagyhatók. Például:

```
Sub ExampleSub(parameter As String)
    MsgBox parameter
End Sub

' Hívás zárójel nélkül
ExampleSub "Hello World"

' Hívás zárójellel - általában akkor szükséges, ha az eredményt változóba
kell menteni vagy kifejezés részeként használjuk
Call ExampleSub("Hello World")
```

A := Jelölés a VBA-ban

A := jelölés a VBA-ban nevesített argumentumok (paraméterek) használatát teszi lehetővé függvények és eljárások hívásakor. Ez a szintaxis megengedi, hogy explicit módon megadjuk a paraméterek nevét és azokhoz tartozó értékeket, ami növeli a kód olvashatóságát és egyértelműségét, különösen akkor, ha a függvény vagy eljárás több paramétert is fogad.

Például:

```
Sub ExampleSubWithParams(firstParam As Integer, secondParam As String)
    MsgBox "Érték: " & firstParam & ", Szöveg: " & secondParam
End Sub

' Nevesített argumentumok használata
Call ExampleSubWithParams(firstParam:=10, secondParam:="Szöveg")

' vagy
ExampleSubWithParams firstParam:=10, secondParam:="Szöveg"
```

5. Hibaellenőrzés és hibakezelés makrókban

Ebben a feladatban egy makró létrehozására mutatunk példát, amely megfelelő hibakezelést implementál a futásidő hibák láthatóvá tétele miatt.

```
Sub HibakezelesMakroban()
    On Error GoTo Hibakezelo
    ' Itt történik a kód, ami hibát generálhat
    Dim osztando As Double
    Dim oszto As Double
    osztando = 10
    oszto = 0 ' Ezzel a sorral osztásnál hibát generálunk
    Dim eredmeny As Double
    eredmeny = osztando / oszto ' Ez hibát okoz, ha az oszto 0
    MsgBox "Az eredmény: " & eredmeny
    Exit Sub

Hibakezelo:
    MsgBox "Hiba történt: " & Err.Description, vbCritical, "Hiba"
    ' Itt történhetnek további hibakezelési lépések, pl. naplózás
End Sub
```

Hogyan Működik?

- A makró a `On Error GoTo Hibakezelo` utasítással kezdődik, ami azt jelenti, hogy ha a makró futtatása során hiba történik, a vezérlés átugrik a `Hibakezelo` címkéhez.
- A példában szándékosan generálunk egy hibát az osztásnál, ahol az osztó értéke 0.
- Ha ez a hiba megtörténik, a makró a `Hibakezelo` részre ugrik, ahol egy üzenetablakban megjelenik a hiba leírása.
- Az `Exit Sub` utasítás biztosítja, hogy ha hiba nélkül eljutottunk a makró végére, ne ugorjunk a hibakezelő részre.
- A hibakezelő részben lehetőség van további tevékenységekre, mint például a hiba naplózása vagy speciális hibakezelési rutinok végrehajtása.

Hogyan Működik?

- A makró a `On Error GoTo Hibakezelo` utasítással kezdődik, ami azt jelenti, hogy ha a makró futtatása során hiba történik, a vezérlés átugrik a `Hibakezelo` címkéhez.
- A példában szándékosan generálunk egy hibát az osztásnál, ahol az osztó értéke 0.
- Ha ez a hiba megtörténik, a makró a `Hibakezelo` részre ugrik, ahol egy üzenetablakban megjelenik a hiba leírása.
- Az `Exit Sub` utasítás biztosítja, hogy ha hiba nélkül eljutottunk a makró végére, ne ugorjunk a hibakezelő részre.
- A hibakezelő részben lehetőség van további tevékenységekre, mint például a hiba naplózása vagy speciális hibakezelési rutinok végrehajtása.

Ez a makró alapvető hibakezelési technikát mutat be, amely elengedhetetlen bármilyen robosztus VBA alkalmazás vagy makró fejlesztésekor, hogy a végfelhasználók számára kezelhető hibaüzeneteket biztosíthassunk.