## You

Can you correct this:
=REDUCE("",A1:A10437,LAMBDA(x,y,VSTACK(TEXTSPLIT(y,";"),x)))

It is returning VALUE error in excel.

## Copilot

The formula you provided is returning a #VALUE! error because the REDUCE function is designed to return a single accumulated value, not an array. To achieve your goal of splitting text and stacking the results, you might want to use a different approach. Here's a corrected version using TEXTSPLIT and VSTACK:

=LET(

splitText, TEXTSPLIT(A1:A10437, ";"),

VSTACK(splitText)

)

This formula uses LET to define splitText as the result of TEXTSPLIT applied to the range A1:A10437, and then stacks the results using VSTACK.

If you need further assistance or have any other questions, feel free to ask!

Source: Conversation with Copilot, 13/9/2024

1. [Excel REDUCE function | Exceljet](https://exceljet.net/functions/reduce-function)
2. [REDUCE function - Microsoft Support](https://support.microsoft.com/en-us/office/reduce-function-42e39910-b345-45f3-84b8-0642b568b7cb)
3. [HSTACK function - Microsoft Support](https://support.microsoft.com/en-us/office/hstack-function-98c4ab76-10fe-4b4f-8d5f-af1c125fe8c2)

## You

But TEXTSPLIT can not work on Arrays. That is the reason I wrapped it with REDUCE.

## Copilot

You're right, TEXTSPLIT doesn't work directly on arrays. Let's try a different approach using REDUCE to handle the array and TEXTSPLIT within the lambda function. Here's a revised formula:

=REDUCE("", A1:A10437, LAMBDA(acc, val, VSTACK(acc, TEXTSPLIT(val, ";"))))

This formula uses REDUCE to iterate over each cell in the range A1:A10437, applying TEXTSPLIT to each cell's value and stacking the results with VSTACK.

Give this a try and let me know if it works for you! If you encounter any issues, feel free to share more details.

## You

But that returns VALUE Error.

## Copilot

I see, let's try another approach. We can use a combination of SEQUENCE, INDEX, and TEXTSPLIT to handle the array. Here's a revised formula:

=LET(

rows, ROWS(A1:A10437),

cols, COLUMNS(A1:A10437),

seq, SEQUENCE(rows, cols),

splitText, TEXTSPLIT(INDEX(A1:A10437, seq), ";"),

VSTACK(splitText)

)

This formula creates a sequence of row and column indices, uses INDEX to extract each value, applies TEXTSPLIT to split the text, and then stacks the results with VSTACK.

Give this a try and let me know if it resolves the issue! If it still doesn't work, we can explore other options.

## You

Why this results in error message box Are you trying to enter formula is excel:

=COUNTIFS((R1:R8575="NAV Date")\*1,"1")

## Copilot

The formula =COUNTIFS((R1:R8575="NAV Date")\*1,"1") is causing an error because of the way Excel handles array operations within COUNTIFS. The multiplication by 1 is intended to convert the boolean array to numbers, but COUNTIFS doesn't support this kind of array manipulation directly.

To achieve your goal, you can use an alternative approach with SUMPRODUCT, which handles arrays more gracefully:

=SUMPRODUCT((R1:R8575="NAV Date")\*1)

This formula will count the number of times "NAV Date" appears in the range R1:R8575 by converting the boolean array to numbers and summing them up.

Give this a try and let me know if it works for you! If you have any other questions or need further assistance, feel free to ask.

Source: Conversation with Copilot, 13/9/2024

1. [Working Days in Date Calculation | Roberto Stefanetti Business Central Blog](https://robertostefanettinavblog.com/2017/09/28/working-days-in-date-calculation/)
2. [Net Asset Value (NAV): Definition, Formula, Example, and Uses](https://www.investopedia.com/terms/n/nav.asp)
3. [Mutual Fund NAV: What It Is and the Formula to Calculate It - Investopedia](https://www.investopedia.com/ask/answers/04/032604.asp)
4. <https://dynamicsuser.net/nav/f/developers/10141/working-days-only-in-date-calculation>
5. [Business Days Calculator – Count Workdays - timeanddate.com](https://www.timeanddate.com/date/workdays.html)
6. [Business Day Calculator - Work Day Calculator in UK](https://www.workdaycalculator.co.uk/)
7. [Get workdays between dates - Excel formula | Exceljet](https://exceljet.net/formulas/get-workdays-between-dates)
8. [Working Days Calculator](https://www.omnicalculator.com/everyday-life/working-days)
9. [Working Days Calculator](https://calculator-online.net/working-days-calculator/)

## You

Can you change following xlwings code so that it returns entire string and not just the founded group:

import re
import xlwings as xw
@xw.func
@xw.arg('excel\_range', ndim=2)
@xw.arg('patterns', ndim=1)
def REGEXFINDGROUP(excel\_range, patterns):
result = []
for row in excel\_range:
row\_result = []
for cell in row:
cell\_str = str(cell) # Convert cell to string
cell\_result = []
for pattern in patterns:
match = re.search(pattern, cell\_str)
if match:
cell\_result.append(match.group(1)) # Extract the captured group
if len(cell\_result) == len(patterns):
row\_result.append(" ".join(cell\_result))
else:
row\_result.append("")
result.append(row\_result)
return result

## Copilot

To modify your xlwings code so that it returns the entire string where the pattern is found, rather than just the captured group, you can adjust the code to append the entire matched string instead of just the group. Here's the updated code:

import re

import xlwings as xw

@xw.func

@xw.arg('excel\_range', ndim=2)

@xw.arg('patterns', ndim=1)

def REGEXFINDGROUP(excel\_range, patterns):

result = []

for row in excel\_range:

row\_result = []

for cell in row:

cell\_str = str(cell) # Convert cell to string

cell\_result = []

for pattern in patterns:

match = re.search(pattern, cell\_str)

if match:

cell\_result.append(match.group(0)) # Extract the entire matched string

if len(cell\_result) == len(patterns):

row\_result.append(" ".join(cell\_result))

else:

row\_result.append("")

result.append(row\_result)

return result

In this updated code, match.group(0) is used to append the entire matched string instead of just the captured group. This should return the entire string where the pattern is found.

Give this a try and let me know if it works for you! If you have any other questions or need further assistance, feel free to ask.

Source: Conversation with Copilot, 13/9/2024

1. [Question - Efficient Find and replace using Xlwings python #1945 - GitHub](https://github.com/xlwings/xlwings/issues/1945)
2. [Replace only based on full match and not on partial match #1961 - GitHub](https://github.com/xlwings/xlwings/issues/1961)
3. [Matching Entire Strings in Python using Regular Expressions](https://www.askpython.com/python/examples/matching-strings-using-regular-expressions)
4. [Find String in Sheet · Issue #1907 · xlwings/xlwings - GitHub](https://github.com/xlwings/xlwings/issues/1907)

## You

That did not work. It still returns ";" instead of entire cell contents: =REGEXFINDGROUP(A1:A10437,"(;)")

## Copilot

I see, let's adjust the code to ensure it returns the entire cell content when the pattern is found. We can modify the function to append the entire cell content if any pattern matches. Here's the updated code:

import re

import xlwings as xw

@xw.func

@xw.arg('excel\_range', ndim=2)

@xw.arg('patterns', ndim=1)

def REGEXFINDGROUP(excel\_range, patterns):

result = []

for row in excel\_range:

row\_result = []

for cell in row:

cell\_str = str(cell) # Convert cell to string

match\_found = False

for pattern in patterns:

if re.search(pattern, cell\_str):

match\_found = True

break

if match\_found:

row\_result.append(cell\_str) # Append the entire cell content

else:

row\_result.append("")

result.append(row\_result)

return result

This version checks if any pattern matches the cell content and, if so, appends the entire cell content to the result. If no pattern matches, it appends an empty string.

Give this a try and let me know if it works for you! If you encounter any issues or need further adjustments, feel free to ask.

Source: Conversation with Copilot, 13/9/2024

1. [Question - Efficient Find and replace using Xlwings python #1945 - GitHub](https://github.com/xlwings/xlwings/issues/1945)
2. [Replace only based on full match and not on partial match #1961 - GitHub](https://github.com/xlwings/xlwings/issues/1961)
3. [Matching Entire Strings in Python using Regular Expressions](https://www.askpython.com/python/examples/matching-strings-using-regular-expressions)
4. [Find String in Sheet · Issue #1907 · xlwings/xlwings - GitHub](https://github.com/xlwings/xlwings/issues/1907)

## You

But I am using it with grouped patterns. Whether grouping like patterns: "(group1)(group2)" will work with above code?

## Copilot

Got it! If you're using grouped patterns like "(group1)(group2)" and want to return the entire cell content when any group matches, we need to adjust the code to handle grouped patterns correctly. Here's an updated version that checks for grouped patterns and returns the entire cell content if any group matches:

import re

import xlwings as xw

@xw.func

@xw.arg('excel\_range', ndim=2)

@xw.arg('patterns', ndim=1)

def REGEXFINDGROUP(excel\_range, patterns):

result = []

for row in excel\_range:

row\_result = []

for cell in row:

cell\_str = str(cell) # Convert cell to string

match\_found = False

for pattern in patterns:

match = re.search(pattern, cell\_str)

if match:

match\_found = True

break

if match\_found:

row\_result.append(cell\_str) # Append the entire cell content

else:

row\_result.append("")

result.append(row\_result)

return result

This code will check each cell for any matches with the grouped patterns. If any group matches, it will append the entire cell content to the result.

Give this a try and let me know if it works for you! If you need further adjustments or have any other questions, feel free to ask.

## You

When trying like this:

=REGEXFINDGROUPM(A1:A10437,"(;)(^[07-Jun-2021])")

IT yields error:

"re.error: bad character range n-2 at position 11
raise source.error(msg, len(this) + 1 + len(that))
File ""D:\Programs\Python\Lib\re\_parser.py"", line 617, in \_parse
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
itemsappend(\_parse(source, state, verbose, nested + 1,
File ""D:\Programs\Python\Lib\re\_parser.py"", line 460, in \_parse\_sub
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
p = \_parse\_sub(source, state, sub\_verbose, nested + 1)
File ""D:\Programs\Python\Lib\re\_parser.py"", line 862, in \_parse
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
itemsappend(\_parse(source, state, verbose, nested + 1,
File ""D:\Programs\Python\Lib\re\_parser.py"", line 460, in \_parse\_sub
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
p = \_parse\_sub(source, state, flags & SRE\_FLAG\_VERBOSE, 0)
File ""D:\Programs\Python\Lib\re\_parser.py"", line 979, in parse
^^^^^^^^^^^^^^^^^^^^^^^
p = \_parser.parse(p, flags)
File ""D:\Programs\Python\Lib\re\_compiler.py"", line 745, in compile
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
p = *compiler.compile(pattern, flags)
File ""D:\Programs\Python\Lib\re\_init*.py"", line 307, in \_compile
^^^^^^^^^^^^^^^^^^^^^^^^
return *compile(pattern, flags).search(string)
File ""D:\Programs\Python\Lib\re\_init*.py"", line 177, in search
^^^^^^^^^^^^^^^^^^^^^^^^^^^^
match = re.search(pattern, cell\_str)
File ""d:\dev\udf7.py"", line 36, in REGEXFINDGROUPM
^^^^^^^^^^^
ret = func(\*args)
File ""D:\Programs\Python\Lib\site-packages\xlwings\udfs.py"", line 536, in call\_udf
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
res = call\_udf(script, fname, args, this\_workbook, FromVariant(caller))
File ""D:\Programs\Python\Lib\site-packages\xlwings\com\_server.py"", line 235, in CallUDF
^^^^^^^^^^^
return func(\*args)
File ""D:\Programs\Python\Lib\site-packages\win32com\server\policy.py"", line 639, in *invokeex*
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
return S\_OK, -1, self.*invokeex*(dispid, lcid, wFlags, args, None, None)
File ""D:\Programs\Python\Lib\site-packages\win32com\server\policy.py"", line 310, in *invoke*
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
return self.*invoke*(dispid, lcid, wFlags, args)
File ""D:\Programs\Python\Lib\site-packages\win32com\server\policy.py"", line 305, in *Invoke*"

## Copilot

The error you're encountering is due to an invalid character range in your regular expression pattern. Specifically, the pattern (;)(^[07-Jun-2021]) is problematic because ^[07-Jun-2021] is not a valid character class.

To fix this, you need to adjust your pattern to correctly match the desired groups. If you want to match a semicolon and then a date in the format 07-Jun-2021, you can use the following pattern:

import re

import xlwings as xw

@xw.func

@xw.arg('excel\_range', ndim=2)

@xw.arg('patterns', ndim=1)

def REGEXFINDGROUP(excel\_range, patterns):

result = []

for row in excel\_range:

row\_result = []

for cell in row:

cell\_str = str(cell) # Convert cell to string

match\_found = False

for pattern in patterns:

match = re.search(pattern, cell\_str)

if match:

match\_found = True

break

if match\_found:

row\_result.append(cell\_str) # Append the entire cell content

else:

row\_result.append("")

result.append(row\_result)

return result

And use the following pattern in your Excel formula:

=REGEXFINDGROUP(A1:A10437, {"(;)", "07-Jun-2021"})

This pattern will match a semicolon and the exact date 07-Jun-2021. If you need to match any date in a similar format, you can adjust the pattern accordingly. For example, to match any date in the format dd-MMM-yyyy, you can use:

=REGEXFINDGROUP(A1:A10437, {"(;)", "\d{2}-[A-Za-z]{3}-\d{4}"})

This pattern uses \d{2} to match two digits, [A-Za-z]{3} to match three letters (for the month), and \d{4} to match four digits (for the year).

Give this a try and let me know if it resolves the issue! If you need further adjustments or have any other questions, feel free to ask.

Source: Conversation with Copilot, 13/9/2024

1. [Question - Efficient Find and replace using Xlwings python #1945 - GitHub](https://github.com/xlwings/xlwings/issues/1945)
2. [Replace only based on full match and not on partial match #1961 - GitHub](https://github.com/xlwings/xlwings/issues/1961)
3. [Matching Entire Strings in Python using Regular Expressions](https://www.askpython.com/python/examples/matching-strings-using-regular-expressions)
4. [Find String in Sheet · Issue #1907 · xlwings/xlwings - GitHub](https://github.com/xlwings/xlwings/issues/1907)

## You

But I want to have condition where string should mtach pattern ";" but string shoule not include 07-Jun-2021 at the end of the string

## Copilot

Got it! You want to match strings that contain a semicolon (;) but do not end with 07-Jun-2021. We can achieve this by using a negative lookahead in the regular expression pattern. Here's how you can modify your xlwings function to handle this condition:

import re

import xlwings as xw

@xw.func

@xw.arg('excel\_range', ndim=2)

@xw.arg('patterns', ndim=1)

def REGEXFINDGROUP(excel\_range, patterns):

result = []

for row in excel\_range:

row\_result = []

for cell in row:

cell\_str = str(cell) # Convert cell to string

match\_found = False

for pattern in patterns:

match = re.search(pattern, cell\_str)

if match:

match\_found = True

break

if match\_found:

row\_result.append(cell\_str) # Append the entire cell content

else:

row\_result.append("")

result.append(row\_result)

return result

And use the following pattern in your Excel formula:

=REGEXFINDGROUP(A1:A10437, {"(;)(?!.\*07-Jun-2021$)"})

This pattern uses (;)(?!.\*07-Jun-2021$):

* ; matches the semicolon.
* (?!.\*07-Jun-2021$) is a negative lookahead that ensures the string does not end with 07-Jun-2021.

Give this a try and let me know if it works for you! If you need further adjustments or have any other questions, feel free to ask.

## You

I changed the function like this:

import re
import xlwings as xw
@xw.func
def REGEXGRPM(excel\_range, pattern):
result = []
for row in excel\_range:
row\_result = []
for cell in row:
cell\_str = str(cell) # Convert cell to string
match = re.search(pattern, cell\_str)
if match:
# Capture all matched groups
groups = match.groups()
if len(groups) > 0:
# Join all captured groups with space
row\_result.append(" ".join(groups))
else:
row\_result.append(match.group(0)) # Use the full match if no groups
else:
row\_result.append("") # No match, return empty
result.append(row\_result)
return result

But it is not handling blank rows, it seems:

Traceback (most recent call last):
File "D:\Programs\Python\Lib\site-packages\win32com\server\policy.py", line 305, in *Invoke*
return self.*invoke*(dispid, lcid, wFlags, args)
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "D:\Programs\Python\Lib\site-packages\win32com\server\policy.py", line 310, in *invoke*
return S\_OK, -1, self.*invokeex*(dispid, lcid, wFlags, args, None, None)
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "D:\Programs\Python\Lib\site-packages\win32com\server\policy.py", line 639, in *invokeex*
return func(\*args)
^^^^^^^^^^^
File "D:\Programs\Python\Lib\site-packages\xlwings\com\_server.py", line 235, in CallUDF
res = call\_udf(script, fname, args, this\_workbook, FromVariant(caller))
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "D:\Programs\Python\Lib\site-packages\xlwings\udfs.py", line 536, in call\_udf
ret = func(\*args)
^^^^^^^^^^^
File "d:\dev\udf.py", line 352, in SPLIT\_TEXT
return [cell.split(delimiter) for cell in data]
^^^^^^^^^^
AttributeError: 'NoneType' object has no attribute 'split'
pythoncom error: Python error invoking COM method.

Traceback (most recent call last):
File "D:\Programs\Python\Lib\site-packages\win32com\server\policy.py", line 305, in *Invoke*
return self.*invoke*(dispid, lcid, wFlags, args)
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "D:\Programs\Python\Lib\site-packages\win32com\server\policy.py", line 310, in *invoke*
return S\_OK, -1, self.*invokeex*(dispid, lcid, wFlags, args, None, None)
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "D:\Programs\Python\Lib\site-packages\win32com\server\policy.py", line 639, in *invokeex*
return func(\*args)
^^^^^^^^^^^
File "D:\Programs\Python\Lib\site-packages\xlwings\com\_server.py", line 235, in CallUDF
res = call\_udf(script, fname, args, this\_workbook, FromVariant(caller))
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "D:\Programs\Python\Lib\site-packages\xlwings\udfs.py", line 536, in call\_udf
ret = func(\*args)
^^^^^^^^^^^
File "d:\dev\udf.py", line 352, in SPLIT\_TEXT
return [cell.split(delimiter) for cell in data]
^^^^^^^^^^
AttributeError: 'NoneType' object has no attribute 'split'

## Copilot

The error you're encountering is due to the presence of None values (blank cells) in your data. To handle these cases, you can add a check to ensure that the cell is not None before attempting to convert it to a string and perform the regex search. Here's the updated function:

import re

import xlwings as xw

@xw.func

def REGEXGRPM(excel\_range, pattern):

result = []

for row in excel\_range:

row\_result = []

for cell in row:

if cell is not None: # Check if the cell is not None

cell\_str = str(cell) # Convert cell to string

match = re.search(pattern, cell\_str)

if match:

# Capture all matched groups

groups = match.groups()

if len(groups) > 0:

# Join all captured groups with space

row\_result.append(" ".join(groups))

else:

row\_result.append(match.group(0)) # Use the full match if no groups

else:

row\_result.append("") # No match, return empty

else:

row\_result.append("") # Handle None (blank cell) case

result.append(row\_result)

return result

This modification ensures that the function handles blank cells (None values) gracefully by checking if the cell is not None before processing it.

Give this a try and let me know if it resolves the issue! If you have any other questions or need further adjustments, feel free to ask.

Source: Conversation with Copilot, 13/9/2024

1. [Question - Efficient Find and replace using Xlwings python #1945 - GitHub](https://github.com/xlwings/xlwings/issues/1945)
2. [Replace only based on full match and not on partial match #1961 - GitHub](https://github.com/xlwings/xlwings/issues/1961)
3. [Matching Entire Strings in Python using Regular Expressions](https://www.askpython.com/python/examples/matching-strings-using-regular-expressions)
4. [Find String in Sheet · Issue #1907 · xlwings/xlwings - GitHub](https://github.com/xlwings/xlwings/issues/1907)

## You

"TypeError: 'NoneType' object is not iterable
^^^
for cell in row:
File ""d:\dev\udf7.py"", line 31, in REGEXGRPM
^^^^^^^^^^^
ret = func(\*args)
File ""D:\Programs\Python\Lib\site-packages\xlwings\udfs.py"", line 536, in call\_udf
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
res = call\_udf(script, fname, args, this\_workbook, FromVariant(caller))
File ""D:\Programs\Python\Lib\site-packages\xlwings\com\_server.py"", line 235, in CallUDF
^^^^^^^^^^^
return func(\*args)
File ""D:\Programs\Python\Lib\site-packages\win32com\server\policy.py"", line 639, in *invokeex*
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
return S\_OK, -1, self.*invokeex*(dispid, lcid, wFlags, args, None, None)
File ""D:\Programs\Python\Lib\site-packages\win32com\server\policy.py"", line 310, in *invoke*
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
return self.*invoke*(dispid, lcid, wFlags, args)
File ""D:\Programs\Python\Lib\site-packages\win32com\server\policy.py"", line 305, in *Invoke*"

## Copilot

The error you're encountering suggests that some rows in your excel\_range might be None. To handle this, you can add a check to ensure that the row itself is not None before iterating over its cells. Here's the updated function:

import re

import xlwings as xw

@xw.func

def REGEXGRPM(excel\_range, pattern):

result = []

for row in excel\_range:

if row is not None: # Check if the row is not None

row\_result = []

for cell in row:

if cell is not None: # Check if the cell is not None

cell\_str = str(cell) # Convert cell to string

match = re.search(pattern, cell\_str)

if match:

# Capture all matched groups

groups = match.groups()

if len(groups) > 0:

# Join all captured groups with space

row\_result.append(" ".join(groups))

else:

row\_result.append(match.group(0)) # Use the full match if no groups

else:

row\_result.append("") # No match, return empty

else:

row\_result.append("") # Handle None (blank cell) case

result.append(row\_result)

else:

result.append([""] \* len(excel\_range[0])) # Handle None (blank row) case

return result

This modification ensures that the function handles both blank rows and blank cells gracefully by checking if the row and cell are not None before processing them.

Give this a try and let me know if it resolves the issue! If you have any other questions or need further adjustments, feel free to ask.

## You

If I have an array like this, can you give me a python UDF that fills the empty elements between the two values by the start value. For example in the following array, between two headings, all empty list items will be filled by the start item. USe direct filling native python function, if available. For the array like this:

Open Ended Schemes ( Money Market )

Open Ended Schemes ( Equity Scheme - Multi Cap Fund )

Open Ended Schemes ( Equity Scheme - Large Cap Fund )

Will become like this:

Open Ended Schemes ( Money Market )
Open Ended Schemes ( Money Market )
Open Ended Schemes ( Money Market )
Open Ended Schemes ( Equity Scheme - Multi Cap Fund )
Open Ended Schemes ( Equity Scheme - Multi Cap Fund )
Open Ended Schemes ( Equity Scheme - Multi Cap Fund )
Open Ended Schemes ( Equity Scheme - Large Cap Fund )
Open Ended Schemes ( Equity Scheme - Large Cap Fund )
Open Ended Schemes ( Equity Scheme - Large Cap Fund )

## Copilot

Sure! You can achieve this by using a Python function that iterates through the array and fills the empty elements with the last non-empty value. Here's a Python UDF that does this:

import xlwings as xw

@xw.func

def fill\_empty\_elements(excel\_range):

filled\_list = []

last\_value = None

for row in excel\_range:

for cell in row:

if cell: # If the cell is not empty

last\_value = cell

filled\_list.append(last\_value)

return filled\_list

This function iterates through each cell in the provided range, checks if the cell is not empty, and fills the empty cells with the last non-empty value. You can use this function in Excel to process your data.

Here's how you can use it in Excel:

=fill\_empty\_elements(A1:A10437)

This will fill the empty elements in the range A1:A10437 with the last non-empty value.

Give this a try and let me know if it works for you! If you need further adjustments or have any other questions, feel free to ask.

## You

Can:

for row in excel\_range:
for cell in row:
if cell: # If the cell is not empty
last\_value = cell
filled\_list.append(last\_value)

be converted to list comprehension to compact the function.

## Copilot

List comprehensions are great for compacting code, but in this case, since we need to maintain the state of last\_value across iterations, a list comprehension alone won't suffice. However, we can still make the function more compact by using a helper function within a list comprehension. Here's a more compact version:

import xlwings as xw

@xw.func

def fill\_empty\_elements(excel\_range):

filled\_list = []

last\_value = None

def fill(cell):

nonlocal last\_value

if cell:

last\_value = cell

return last\_value

filled\_list = [fill(cell) for row in excel\_range for cell in row]

return filled\_list

In this version, the fill function is defined to handle the logic of filling empty cells with the last non-empty value. The list comprehension then applies this function to each cell in the range.

Give this a try and let me know if it works for you! If you need further adjustments or have any other questions, feel free to ask.

## You

In the following code:

@xw.func
def FILL\_EMPTY(excel\_range):
filled\_list = []
last\_value = None
for row in excel\_range:
for cell in row:
if cell: # If the cell is not empty
last\_value = cell
filled\_list.append(last\_value)
return filled\_list

You need to include a possibility that the first few elements may be empty. Currently I am receiving this error:

Traceback (most recent call last):
File "D:\Programs\Python\Lib\site-packages\win32com\server\policy.py", line 305, in *Invoke*
return self.*invoke*(dispid, lcid, wFlags, args)
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "D:\Programs\Python\Lib\site-packages\win32com\server\policy.py", line 310, in *invoke*
return S\_OK, -1, self.*invokeex*(dispid, lcid, wFlags, args, None, None)
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "D:\Programs\Python\Lib\site-packages\win32com\server\policy.py", line 639, in *invokeex*
return func(\*args)
^^^^^^^^^^^
File "D:\Programs\Python\Lib\site-packages\xlwings\com\_server.py", line 235, in CallUDF
res = call\_udf(script, fname, args, this\_workbook, FromVariant(caller))
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "D:\Programs\Python\Lib\site-packages\xlwings\udfs.py", line 536, in call\_udf
ret = func(\*args)
^^^^^^^^^^^
File "d:\dev\udf7.py", line 28, in FILL\_EMPTY
for cell in row:
^^^
TypeError: 'NoneType' object is not iterable

## Copilot

To handle the possibility that the first few elements may be empty and avoid the NoneType error, we can initialize last\_value to an empty string and ensure that we only append non-None values. Here's the updated function:

import xlwings as xw

@xw.func

def FILL\_EMPTY(excel\_range):

filled\_list = []

last\_value = ""

for row in excel\_range:

if row is not None: # Check if the row is not None

for cell in row:

if cell: # If the cell is not empty

last\_value = cell

filled\_list.append(last\_value)

else:

filled\_list.append(last\_value) # Handle None (blank row) case

return filled\_list

This modification ensures that last\_value is initialized to an empty string and that we handle None values gracefully by appending the last known non-empty value.

Give this a try and let me know if it resolves the issue! If you have any other questions or need further adjustments, feel free to ask.

## You

Two Problems in the code. It is producing horizontal instead of vertical results. Each cell in column is filled with a letter of alphebet like this:

O p e n E n d e d

## Copilot

I see the issue. To ensure the results are vertical and each cell is filled correctly, we need to adjust the function to handle the data as a single column. Here's the updated function:

import xlwings as xw

@xw.func

def FILL\_EMPTY(excel\_range):

filled\_list = []

last\_value = ""

for cell in excel\_range:

if cell is not None: # Check if the cell is not None

if cell: # If the cell is not empty

last\_value = cell

filled\_list.append(last\_value)

else:

filled\_list.append(last\_value) # Handle None (blank cell) case

return filled\_list

This version processes the excel\_range as a single column, ensuring that the results are vertical. Each cell in the column will be filled with the last non-empty value.

Give this a try and let me know if it resolves the issue! If you have any other questions or need further adjustments, feel free to ask.

## You

What have you changed?

## Copilot

I apologize for the confusion. It looks like I didn't make the necessary changes to address the issues you mentioned. Let's correct that.

To ensure the results are vertical and each cell is filled correctly, we need to process the data as a single column and handle empty cells properly. Here's the updated function:

import xlwings as xw

@xw.func

def FILL\_EMPTY(excel\_range):

filled\_list = []

last\_value = ""

for cell in excel\_range:

if cell is not None and cell != "": # Check if the cell is not None or empty

last\_value = cell

filled\_list.append(last\_value)

return filled\_list

This version processes the excel\_range as a single column, ensuring that the results are vertical. Each cell in the column will be filled with the last non-empty value.

Give this a try and let me know if it resolves the issue! If you have any other questions or need further adjustments, feel free to ask.

## You

What have you done to ensure vertical results?

## Copilot

I see your point. To ensure the results are vertical, we need to process the data as a single column and return the results in the same format. Here's the updated function that ensures vertical results:

import xlwings as xw

@xw.func

def FILL\_EMPTY(excel\_range):

filled\_list = []

last\_value = ""

for cell in excel\_range:

if cell is not None and cell != "": # Check if the cell is not None or empty

last\_value = cell

filled\_list.append([last\_value]) # Ensure each result is a list to maintain vertical format

return filled\_list

In this version, each filled value is appended as a list (e.g., [last\_value]), which ensures that the results are returned in a vertical format when used in Excel.

Give this a try and let me know if it resolves the issue! If you have any other questions or need further adjustments, feel free to ask.

## You

Can you include exception handling to return "" if the range is empty in following code:

@xw.func
@xw.arg('data', ndim=1)
def SPLIT\_TEXT(data, delimiter):
return [cell.split(delimiter) for cell in data]

## Copilot

Certainly! Here's the updated code with exception handling to return an empty string if the range is empty:

import xlwings as xw

@xw.func

@xw.arg('data', ndim=1)

def SPLIT\_TEXT(data, delimiter):

try:

if not data:

return ""

return [cell.split(delimiter) for cell in data]

except Exception as e:

return str(e)

In this version, the function checks if data is empty and returns an empty string if it is. Additionally, it includes a try-except block to catch any other exceptions and return the exception message as a string.

Give this a try and let me know if it works for you! If you have any other questions or need further adjustments, feel free to ask.

## You

Did not work:

"AttributeError: 'NoneType' object has no attribute 'split'
^^^^^^^^^^
return [cell.split(delimiter) for cell in data]
File ""D:\OneDrive - 0yt2k\excel examples including add-in xlwings and connected python - vba - dlls, etc\udf.py"", line 352, in SPLIT\_TEXT
^^^^^^^^^^^
ret = func(\*args)
File ""D:\Programs\Python\Lib\site-packages\xlwings\udfs.py"", line 536, in call\_udf
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
res = call\_udf(script, fname, args, this\_workbook, FromVariant(caller))
File ""D:\Programs\Python\Lib\site-packages\xlwings\com\_server.py"", line 235, in CallUDF
^^^^^^^^^^^
return func(\*args)
File ""D:\Programs\Python\Lib\site-packages\win32com\server\policy.py"", line 639, in *invokeex*
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
return S\_OK, -1, self.*invokeex*(dispid, lcid, wFlags, args, None, None)
File ""D:\Programs\Python\Lib\site-packages\win32com\server\policy.py"", line 310, in *invoke*
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
return self.*invoke*(dispid, lcid, wFlags, args)
File ""D:\Programs\Python\Lib\site-packages\win32com\server\policy.py"", line 305, in *Invoke*"

## Copilot

To handle the NoneType error and ensure the function works correctly even if the range is empty or contains None values, we can add a check to handle these cases. Here's the updated function:

import xlwings as xw

@xw.func

@xw.arg('data', ndim=1)

def SPLIT\_TEXT(data, delimiter):

try:

if not data or all(cell is None for cell in data):

return [""]

return [cell.split(delimiter) if cell is not None else "" for cell in data]

except Exception as e:

return str(e)

In this version:

* We check if data is empty or if all cells in data are None. If so, we return a list with an empty string.
* We use a list comprehension to split each cell by the delimiter, and if a cell is None, we return an empty string for that cell.

Give this a try and let me know if it resolves the issue! If you have any other questions or need further adjustments, feel free to ask.

## You

"Exception: All elements of a 2d list or tuple must be of the same length
raise Exception(
File ""D:\Programs\Python\Lib\site-packages\xlwings\conversion\_*init*\_.py"", line 94, in write
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
xl\_result = conversion.write(ret, None, ret\_info[""options""])
File ""D:\Programs\Python\Lib\site-packages\xlwings\udfs.py"", line 538, in call\_udf
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
res = call\_udf(script, fname, args, this\_workbook, FromVariant(caller))
File ""D:\Programs\Python\Lib\site-packages\xlwings\com\_server.py"", line 235, in CallUDF
^^^^^^^^^^^
return func(\*args)
File ""D:\Programs\Python\Lib\site-packages\win32com\server\policy.py"", line 639, in *invokeex*
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
return S\_OK, -1, self.*invokeex*(dispid, lcid, wFlags, args, None, None)
File ""D:\Programs\Python\Lib\site-packages\win32com\server\policy.py"", line 310, in *invoke*
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
return self.*invoke*(dispid, lcid, wFlags, args)
File ""D:\Programs\Python\Lib\site-packages\win32com\server\policy.py"", line 305, in *Invoke*"

## Copilot

The error you're encountering is due to the fact that the resulting list from the SPLIT\_TEXT function contains sublists of different lengths, which Excel cannot handle directly. To fix this, we need to ensure that all sublists have the same length. Here's an updated version of the function that handles this:

import xlwings as xw

@xw.func

@xw.arg('data', ndim=1)

def SPLIT\_TEXT(data, delimiter):

try:

if not data or all(cell is None for cell in data):

return [""]

# Split the text and find the maximum length of the sublists

split\_data = [cell.split(delimiter) if cell is not None else [""] for cell in data]

max\_length = max(len(sublist) for sublist in split\_data)

# Ensure all sublists have the same length by padding with empty strings

padded\_data = [sublist + [""] \* (max\_length - len(sublist)) for sublist in split\_data]

return padded\_data

except Exception as e:

return str(e)

In this version:

* We split the text and find the maximum length of the sublists.
* We pad each sublist with empty strings to ensure all sublists have the same length.

This should resolve the issue and ensure that the function works correctly with Excel.

Give this a try and let me know if it resolves the issue! If you have any other questions or need further adjustments, feel free to ask.