# You

I have following xlwings python script which is actually python UDF called from excel using 2 arguments. Basically, the script checks whether in the excel range given as input has the required regular expression pattern as per python. There are 2 inputs to UDF: excel range and the pattern to be searched:

@xw.func
@xw.arg('excel\_range', ndim=2)
@xw.arg('patterns', ndim=1)
def REGEXFIND(excel\_range, patterns):
result = []
for row in excel\_range:
row\_result = []
for cell in row:
cell\_result = []
for pattern in patterns:
match = regex.search(pattern, cell)
if match:
cell\_result.append(match.group())
if len(cell\_result) == len(patterns):
row\_result.append(" ".join(cell\_result))
else:
row\_result.append("Pattern Not Found")
result.append(row\_result)
return result

I want you to teak the code so that there are 3 arguments: excel range, pattern to be found, replacement string. So basically I want you to define a new udf REGEXREPLACE.

# Copilot

Sure, I can help with that. Here's a modified version of your function that takes an additional argument for the replacement string and uses the regex.sub function to replace the matched patterns:

import regex

@xw.func

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

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

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

def REGEXREPLACE(excel\_range, pattern, replacement):

result = []

for row in excel\_range:

row\_result = []

for cell in row:

cell\_result = regex.sub(pattern, replacement, cell)

row\_result.append(cell\_result)

result.append(row\_result)

return result

This function will replace all occurrences of the pattern in each cell of the given Excel range with the provided replacement string. If the pattern is not found in a cell, the cell's content will remain unchanged. Please note that this function uses Python's regex module, which supports a more extensive set of regular expression syntax than the built-in re module. If you prefer to use re, you can replace regex with re. Also, please ensure that the xlwings and regex packages are installed in your Python environment. You can install them using pip:

pip install xlwings regex

# You

But check whether your output is a replaced string (list of lists) so that it becomes dynamic array in excel.

# Copilot

Yes, the output of the REGEXREPLACE function is a list of lists, where each inner list represents a row in the Excel range. Each cell in the range is replaced with the result of the regex substitution, which is a string. This structure should be compatible with Excel's dynamic array feature.

Here's an example of how the output might look for a 2x2 range, where the pattern "abc" is replaced with "def" in each cell:

[

['def', 'Original String'],

['Another def', 'Yet Another String']

]

In this example, "abc" in the first and third cells of the range has been replaced with "def". The other cells remain unchanged because they did not contain the pattern "abc".

Please note that the actual output will depend on the contents of your Excel range and the pattern and replacement string you provide. If a cell does not contain the pattern, the cell's content will remain unchanged. If a cell is empty or contains a non-string value, it will also remain unchanged.

Remember to call this function from Excel after defining it in your Python script. You can do this by using the function name REGEXREPLACE directly in your Excel formulas. For example:

=REGEXREPLACE(A1:B2, "abc", "def")

This will replace all occurrences of "abc" with "def" in the range A1:B2 and return the result as a dynamic array. If you're using an older version of Excel that doesn't support dynamic arrays, you may need to enter this formula as an array formula by pressing Ctrl+Shift+Enter.

# You

Did not work. Traceback:

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 502, in call\_udf
ret = func(\*args)
^^^^^^^^^^^
File "D:\OneDrive - 0yt2k\excel examples including add-in xlwings and connected python - vba - dlls, etc\xlwings\_rdrand\_regex\_test.py", line 70, in REGEXREPLACE
cell\_result = regex.sub(pattern, replacement, cell)
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "D:\Programs\Python\Lib\site-packages\regex\regex.py", line 277, in sub
pat = \_compile(pattern, flags, ignore\_unused, kwargs, True)
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "D:\Programs\Python\Lib\site-packages\regex\regex.py", line 464, in \_compile
if \_locale\_sensitive.get(locale\_key, True) or (flags & LOCALE) != 0:
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
TypeError: unhashable type: 'list'

# Copilot

I apologize for the confusion. The error message indicates that a list is being used where a string is expected. In the REGEXREPLACE function, the pattern and replacement arguments are expected to be strings, not lists.

Here's the corrected version of the function:

import regex

@xw.func

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

def REGEXREPLACE(excel\_range, pattern, replacement):

result = []

for row in excel\_range:

row\_result = []

for cell in row:

cell\_result = regex.sub(pattern, replacement, cell)

row\_result.append(cell\_result)

result.append(row\_result)

return result

In this version, pattern and replacement are regular arguments, not xw.arg decorated arguments. This means they will be treated as strings, not lists. You can call this function from Excel with a range and two strings as arguments, like this:

=REGEXREPLACE(A1:B2, "abc", "def")

This will replace all occurrences of "abc" with "def" in the range A1:B2 and return the result as a dynamic array. If you're using an older version of Excel that doesn't support dynamic arrays, you may need to enter this formula as an array formula by pressing Ctrl+Shift+Enter. Please try this and let me know if it works for you. If you still encounter issues, please provide the details and I'll be glad to assist further.

# You

You have not considered a scenario which was ther in my original function that I gave you. "Pattern Not Found" See this part of my original code:

for pattern in patterns:

match = regex.search(pattern, cell)

if match:

cell\_result.append(match.group())

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

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

else:

row\_result.append("Pattern Not Found")

# Copilot

I apologize for the oversight. You're right, the original function was designed to handle multiple patterns and return a "Pattern Not Found" message when none of the patterns are found in a cell. Here's a revised version of the REGEXREPLACE function that incorporates this feature:

import regex

@xw.func

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

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

def REGEXREPLACE(excel\_range, patterns, replacement):

result = []

for row in excel\_range:

row\_result = []

for cell in row:

cell\_result = []

for pattern in patterns:

if regex.search(pattern, cell):

cell\_result.append(regex.sub(pattern, replacement, cell))

if cell\_result:

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

else:

row\_result.append("Pattern Not Found")

result.append(row\_result)

return result

In this version, the function iterates over each pattern for each cell in the range. If a pattern is found in a cell, it replaces the pattern with the replacement string. If none of the patterns are found in a cell, it appends "Pattern Not Found" to the result for that cell. The function returns a list of lists, where each inner list represents a row in the Excel range. This should be compatible with Excel's dynamic array feature.

Please try this and let me know if it works for you. If you still encounter issues, please provide the details and I'll be glad to assist further. Remember to call this function from Excel after defining it in your Python script. You can do this by using the function name REGEXREPLACE directly in your Excel formulas. For example:

=REGEXREPLACE(A1:B2, {"abc", "def"}, "ghi")

This will replace all occurrences of "abc" and "def" with "ghi" in the range A1:B2 and return the result as a dynamic array. If you're using an older version of Excel that doesn't support dynamic arrays, you may need to enter this formula as an array formula by pressing Ctrl+Shift+Enter. If a cell does not contain either "abc" or "def", the cell's content will be replaced with "Pattern Not Found".

# You

There are 3 input arguments - you have used only two in the following:

@xw.arg('excel\_range', ndim=2)
@xw.arg('patterns', ndim=1)

Where is 3rd @xw.arg?

# Copilot

I apologize for the oversight. You're correct, the replacement argument should also be decorated with @xw.arg. Here's the corrected version of the function:

import regex

@xw.func

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

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

@xw.arg('replacement')

def REGEXREPLACE(excel\_range, patterns, replacement):

result = []

for row in excel\_range:

row\_result = []

for cell in row:

cell\_result = []

for pattern in patterns:

if regex.search(pattern, cell):

cell\_result.append(regex.sub(pattern, replacement, cell))

if cell\_result:

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

else:

row\_result.append("Pattern Not Found")

result.append(row\_result)

return result

In this version, replacement is a regular argument, not an xw.arg decorated argument. This means it will be treated as a string, not a list. You can call this function from Excel with a range, a list of patterns, and a replacement string as arguments, like this:

=REGEXREPLACE(A1:B2, {"abc", "def"}, "ghi")

This will replace all occurrences of "abc" and "def" with "ghi" in the range A1:B2 and return the result as a dynamic array. If you're using an older version of Excel that doesn't support dynamic arrays, you may need to enter this formula as an array formula by pressing Ctrl+Shift+Enter. If a cell does not contain either "abc" or "def", the cell's content will be replaced with "Pattern Not Found". Please try this and let me know if it works for you. If you still encounter issues, please provide the details and I'll be glad to assist further. Remember to call this function from Excel after defining it in your Python script. You can do this by using the function name REGEXREPLACE directly in your Excel formulas. For example:

=REGEXREPLACE(A1:B2, {"abc", "def"}, "ghi")

This will replace all occurrences of "abc" and "def" with "ghi" in the range A1:B2 and return the result as a dynamic array. If you're using an older version of Excel that doesn't support dynamic arrays, you may need to enter this formula as an array formula by pressing Ctrl+Shift+Enter. If a cell does not contain either "abc" or "def", the cell's content will be replaced with "Pattern Not Found".

# You

See I want output as range so why not: @xw.arg('replacement', ndim=2)?

# Copilot

I see, you want to use a range of replacement strings instead of a single string. In that case, you can indeed use @xw.arg('replacement', ndim=2) to accept a range of replacement strings. However, you would need to adjust the function to handle this correctly. Here's how you could modify the function:

import regex

@xw.func

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

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

@xw.arg('replacement', ndim=2)

def REGEXREPLACE(excel\_range, patterns, replacement):

result = []

for i, row in enumerate(excel\_range):

row\_result = []

for j, cell in enumerate(row):

cell\_result = []

for pattern, replace in zip(patterns, replacement[i]):

if regex.search(pattern, cell):

cell\_result.append(regex.sub(pattern, replace, cell))

if cell\_result:

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

else:

row\_result.append("Pattern Not Found")

result.append(row\_result)

return result

In this version, the function iterates over each pattern and corresponding replacement string for each cell in the range. If a pattern is found in a cell, it replaces the pattern with the corresponding replacement string. If none of the patterns are found in a cell, it appends "Pattern Not Found" to the result for that cell. The function returns a list of lists, where each inner list represents a row in the Excel range. This should be compatible with Excel's dynamic array feature.

Please note that this function assumes that the replacement range has the same shape as the excel\_range and that each pattern in patterns has a corresponding replacement string in each row of replacement. If this is not the case, you may need to adjust the function or your input data accordingly.

You can call this function from Excel with a range, a list of patterns, and a range of replacement strings as arguments, like this:

=REGEXREPLACE(A1:B2, {"abc", "def"}, C1:D2)

This will replace all occurrences of "abc" with the value in C1 and "def" with the value in D1 in the range A1:B2, and so on for each cell in the range. The result is returned as a dynamic array. If you're using an older version of Excel that doesn't support dynamic arrays, you may need to enter this formula as an array formula by pressing Ctrl+Shift+Enter. If a cell does not contain either "abc" or "def", the cell's content will be replaced with "Pattern Not Found". Please try this and let me know if it works for you. If you still encounter issues, please provide the details and I'll be glad to assist further. Remember to call this function from Excel after defining it in your Python script. You can do this by using the function name REGEXREPLACE directly in your Excel formulas. For example:

=REGEXREPLACE(A1:B2, {"abc", "def"}, C1:D2)

This will replace all occurrences of "abc" with the value in C1 and "def" with the value in D1 in the range A1:B2, and so on for each cell in the range. The result is returned as a dynamic array. If you're using an older version of Excel that doesn't support dynamic arrays, you may need to enter this formula as an array formula by pressing Ctrl+Shift+Enter. If a cell does not contain either "abc" or "def", the cell's content will be replaced with "Pattern Not Found".

# You

Same 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 502, in call\_udf
ret = func(\*args)
^^^^^^^^^^^
File "D:\OneDrive - 0yt2k\excel examples including add-in xlwings and connected python - vba - dlls, etc\xlwings\_rdrand\_regex\_test.py", line 70, in REGEXREPLACE
cell\_result = regex.sub(pattern, replacement, cell)
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "D:\Programs\Python\Lib\site-packages\regex\regex.py", line 277, in sub
pat = \_compile(pattern, flags, ignore\_unused, kwargs, True)
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "D:\Programs\Python\Lib\site-packages\regex\regex.py", line 464, in \_compile
if \_locale\_sensitive.get(locale\_key, True) or (flags & LOCALE) != 0:
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
TypeError: unhashable type: 'list'

I used code:

@xw.func
@xw.arg('excel\_range', ndim=2)
@xw.arg('patterns', ndim=1)
@xw.arg('replacement')
def REGEXREPLACE(excel\_range, patterns, replacement):
result = []
for row in excel\_range:
row\_result = []
for cell in row:
cell\_result = []
for pattern in patterns:
if regex.search(pattern, cell):
cell\_result.append(regex.sub(pattern, replacement, cell))
if cell\_result:
row\_result.append(" ".join(cell\_result))
else:
row\_result.append("Pattern Not Found")
result.append(row\_result)
return result

Return replacement as dynamic array.

# Copilot

I apologize for the confusion. The error message indicates that a list is being used where a string is expected. In the REGEXREPLACE function, the pattern and replacement arguments are expected to be strings, not lists.

Here's the corrected version of the function:

import regex

@xw.func

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

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

@xw.arg('replacement', ndim=2)

def REGEXREPLACE(excel\_range, patterns, replacement):

result = []

for i, row in enumerate(excel\_range):

row\_result = []

for j, cell in enumerate(row):

cell\_result = cell

for k, pattern in enumerate(patterns):

if regex.search(pattern, cell):

cell\_result = regex.sub(pattern, replacement[i][k], cell\_result)

row\_result.append(cell\_result)

result.append(row\_result)

return result

In this version, the function iterates over each pattern for each cell in the range. If a pattern is found in a cell, it replaces the pattern with the corresponding replacement string. If none of the patterns are found in a cell, the cell's content will remain unchanged. The function returns a list of lists, where each inner list represents a row in the Excel range. This should be compatible with Excel's dynamic array feature.

Please note that this function assumes that the replacement range has the same shape as the excel\_range and that each pattern in patterns has a corresponding replacement string in each row of replacement. If this is not the case, you may need to adjust the function or your input data accordingly.

You can call this function from Excel with a range, a list of patterns, and a range of replacement strings as arguments, like this:

=REGEXREPLACE(A1:B2, {"abc", "def"}, C1:D2)

This will replace all occurrences of "abc" with the value in C1 and "def" with the value in D1 in the range A1:B2, and so on for each cell in the range. The result is returned as a dynamic array. If you're using an older version of Excel that doesn't support dynamic arrays, you may need to enter this formula as an array formula by pressing Ctrl+Shift+Enter. If a cell does not contain either "abc" or "def", the cell's content will be replaced with "Pattern Not Found". Please try this and let me know if it works for you. If you still encounter issues, please provide the details and I'll be glad to assist further. Remember to call this function from Excel after defining it in your Python script. You can do this by using the function name REGEXREPLACE directly in your Excel formulas. For example:

=REGEXREPLACE(A1:B2, {"abc", "def"}, C1:D2)

This will replace all occurrences of "abc" with the value in C1 and "def" with the value in D1 in the range A1:B2, and so on for each cell in the range. The result is returned as a dynamic array. If you're using an older version of Excel that doesn't support dynamic arrays, you may need to enter this formula as an array formula by pressing Ctrl+Shift+Enter. If a cell does not contain either "abc" or "def", the cell's content will be replaced with "Pattern Not Found".

# You

Don't return multiple replacements like: C1:D2

Just change code below to return replacement as dynamic array:

@xw.func
@xw.arg('excel\_range', ndim=2)
@xw.arg('patterns', ndim=1)
@xw.arg('replacement')
def REGEXREPLACE(excel\_range, patterns, replacement):
result = []
for row in excel\_range:
row\_result = []
for cell in row:
cell\_result = []
for pattern in patterns:
if regex.search(pattern, cell):
cell\_result.append(regex.sub(pattern, replacement, cell))
if cell\_result:
row\_result.append(" ".join(cell\_result))
else:
row\_result.append("Pattern Not Found")
result.append(row\_result)
return result

# Copilot

I apologize for misunderstanding your requirement earlier. If you want to use a single replacement string for all patterns, you can indeed use @xw.arg('replacement') to accept a single replacement string. Here's the corrected version of the function:

import regex

@xw.func

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

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

@xw.arg('replacement')

def REGEXREPLACE(excel\_range, patterns, replacement):

result = []

for row in excel\_range:

row\_result = []

for cell in row:

cell\_result = cell

for pattern in patterns:

if regex.search(pattern, cell\_result):

cell\_result = regex.sub(pattern, replacement, cell\_result)

row\_result.append(cell\_result)

result.append(row\_result)

return result

In this version, the function iterates over each pattern for each cell in the range. If a pattern is found in a cell, it replaces the pattern with the replacement string. If none of the patterns are found in a cell, the cell's content will remain unchanged. The function returns a list of lists, where each inner list represents a row in the Excel range. This should be compatible with Excel's dynamic array feature.

You can call this function from Excel with a range, a list of patterns, and a replacement string as arguments, like this:

=REGEXREPLACE(A1:B2, {"abc", "def"}, "ghi")

This will replace all occurrences of "abc" and "def" with "ghi" in the range A1:B2 and return the result as a dynamic array. If you're using an older version of Excel that doesn't support dynamic arrays, you may need to enter this formula as an array formula by pressing Ctrl+Shift+Enter. If a cell does not contain either "abc" or "def", the cell's content will remain unchanged. Please try this and let me know if it works for you. If you still encounter issues, please provide the details and I'll be glad to assist further. Remember to call this function from Excel after defining it in your Python script. You can do this by using the function name REGEXREPLACE directly in your Excel formulas. For example:

=REGEXREPLACE(A1:B2, {"abc", "def"}, "ghi")

This will replace all occurrences of "abc" and "def" with "ghi" in the range A1:B2 and return the result as a dynamic array. If you're using an older version of Excel that doesn't support dynamic arrays, you may need to enter this formula as an array formula by pressing Ctrl+Shift+Enter. If a cell does not contain either "abc" or "def", the cell's content will remain unchanged.

# You

For more one or more spaces whether: \s+ is correct?

# Copilot

Yes, you're correct. In regular expressions, \s matches any whitespace character (spaces, tabs, line breaks), and + means "one or more". So, \s+ will match one or more consecutive whitespace characters. This can be useful for matching and replacing multiple spaces with a single space, for example. Please note that the exact syntax and behavior can vary between different regular expression engines and programming languages. In Python's re and regex modules, \s+ works as described. If you're using a different language or library, please check its documentation to confirm.

# You

Why:

=REGEXREPLACE(A7:A15998,"\s+"," ")

While:

=REGEXREPLACE(A7:A15998,"\s+","")

results in exception:

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 502, in call\_udf
ret = func(\*args)
^^^^^^^^^^^
File "D:\OneDrive - 0yt2k\excel examples including add-in xlwings and connected python - vba - dlls, etc\xlwings\_rdrand\_regex\_test.py", line 76, in REGEXREPLACE
cell\_result = regex.sub(pattern, replacement, cell\_result)
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "D:\Programs\Python\Lib\site-packages\regex\regex.py", line 278, in sub
return pat.sub(repl, string, count, pos, endpos, concurrent, timeout)
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
SystemError: <built-in function \_\_import\_\_> returned a result with an exception set

I want to replace spaces with empty string.

# Copilot

I apologize for the confusion. The error message indicates that a list is being used where a string is expected. In the REGEXREPLACE function, the pattern and replacement arguments are expected to be strings, not lists.

Here's the corrected version of the function:

import regex

@xw.func

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

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

@xw.arg('replacement')

def REGEXREPLACE(excel\_range, patterns, replacement):

result = []

for row in excel\_range:

row\_result = []

for cell in row:

cell\_result = cell

for pattern in patterns:

if regex.search(pattern, cell\_result):

cell\_result = regex.sub(pattern, replacement, cell\_result)

row\_result.append(cell\_result)

result.append(row\_result)

return result

In this version, the function iterates over each pattern for each cell in the range. If a pattern is found in a cell, it replaces the pattern with the replacement string. If none of the patterns are found in a cell, the cell's content will remain unchanged. The function returns a list of lists, where each inner list represents a row in the Excel range. This should be compatible with Excel's dynamic array feature.

You can call this function from Excel with a range, a list of patterns, and a replacement string as arguments, like this:

=REGEXREPLACE(A1:B2, {"abc", "def"}, "ghi")

This will replace all occurrences of "abc" and "def" with "ghi" in the range A1:B2 and return the result as a dynamic array. If you're using an older version of Excel that doesn't support dynamic arrays, you may need to enter this formula as an array formula by pressing Ctrl+Shift+Enter. If a cell does not contain either "abc" or "def", the cell's content will remain unchanged. Please try this and let me know if it works for you. If you still encounter issues, please provide the details and I'll be glad to assist further. Remember to call this function from Excel after defining it in your Python script. You can do this by using the function name REGEXREPLACE directly in your Excel formulas. For example:

=REGEXREPLACE(A1:B2, {"abc", "def"}, "ghi")

This will replace all occurrences of "abc" and "def" with "ghi" in the range A1:B2 and return the result as a dynamic array. If you're using an older version of Excel that doesn't support dynamic arrays, you may need to enter this formula as an array formula by pressing Ctrl+Shift+Enter. If a cell does not contain either "abc" or "def", the cell's content will remain unchanged.

# You

You have not understood:

=REGEXREPLACE(A7:A15998,"\s+"," ") Works perfactly with above code. But when I want to replace spaces with null like this in excel:

=REGEXREPLACE(A7:A15998,"\s+","") - See the last argument double quotes without anything in between, then there is a traceback:

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 502, in call\_udf
ret = func(\*args)
^^^^^^^^^^^
File "D:\OneDrive - 0yt2k\excel examples including add-in xlwings and connected python - vba - dlls, etc\xlwings\_rdrand\_regex\_test.py", line 75, in REGEXREPLACE
cell\_result = regex.sub(pattern, replacement, cell\_result)
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "D:\Programs\Python\Lib\site-packages\regex\regex.py", line 278, in sub
return pat.sub(repl, string, count, pos, endpos, concurrent, timeout)
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
SystemError: <built-in function \_\_import\_\_> returned a result with an exception set

# Copilot

I apologize for the confusion. The error you're encountering is unusual. The Python regex.sub function should be able to handle an empty string as the replacement argument. It's possible that this issue is specific to the interaction between xlwings and Excel.

As a workaround, you could modify the function to check if the replacement string is empty and handle this case separately:

import regex

@xw.func

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

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

@xw.arg('replacement')

def REGEXREPLACE(excel\_range, patterns, replacement):

result = []

for row in excel\_range:

row\_result = []

for cell in row:

cell\_result = cell

for pattern in patterns:

if regex.search(pattern, cell\_result):

if replacement == "":

cell\_result = regex.sub(pattern, "", cell\_result)

else:

cell\_result = regex.sub(pattern, replacement, cell\_result)

row\_result.append(cell\_result)

result.append(row\_result)

return result

In this version of the function, if the replacement string is empty, the function uses an empty string directly as the replacement argument to regex.sub. This should have the same effect as using the replacement variable when it's an empty string, but it might avoid the error you're encountering.

Please try this version of the function and let me know if it works for you. If you still encounter issues, please provide the details and I'll be glad to assist further. Remember to call this function from Excel after defining it in your Python script. You can do this by using the function name REGEXREPLACE directly in your Excel formulas. For example:

=REGEXREPLACE(A1:B2, {"abc", "def"}, "ghi")

This will replace all occurrences of "abc" and "def" with "ghi" in the range A1:B2 and return the result as a dynamic array. If you're using an older version of Excel that doesn't support dynamic arrays, you may need to enter this formula as an array formula by pressing Ctrl+Shift+Enter. If a cell does not contain either "abc" or "def", the cell's content will remain unchanged. If you want to replace spaces with an empty string, you can use the formula:

=REGEXREPLACE(A7:A15998, "\s+", "")

This will replace all occurrences of one or more spaces with an empty string in the range A7:A15998. If a cell does not contain any spaces, the cell's content will remain unchanged. If you're using an older version of Excel that doesn't support dynamic arrays, you may need to enter this formula as an array formula by pressing Ctrl+Shift+Enter.

# You

Some issue here in your code:

if replacement == "":

cell\_result = regex.sub(pattern, "", cell\_result)

Same traceback:

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 502, in call\_udf
ret = func(\*args)
^^^^^^^^^^^
File "D:\OneDrive - 0yt2k\excel examples including add-in xlwings and connected python - vba - dlls, etc\xlwings\_rdrand\_regex\_test.py", line 78, in REGEXREPLACE
cell\_result = regex.sub(pattern, replacement, cell\_result)
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "D:\Programs\Python\Lib\site-packages\regex\regex.py", line 278, in sub
return pat.sub(repl, string, count, pos, endpos, concurrent, timeout)
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
SystemError: <built-in function \_\_import\_\_> returned a result with an exception set

Check whatis there is python if I want to return empty string.

# Copilot

I apologize for the confusion. The error you're encountering is unusual. The Python regex.sub function should be able to handle an empty string as the replacement argument. It's possible that this issue is specific to the interaction between xlwings and Excel.

As a workaround, you could modify the function to check if the replacement string is empty and handle this case separately:

import regex

@xw.func

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

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

@xw.arg('replacement')

def REGEXREPLACE(excel\_range, patterns, replacement):

result = []

for row in excel\_range:

row\_result = []

for cell in row:

cell\_result = cell

for pattern in patterns:

if regex.search(pattern, cell\_result):

if replacement == "":

cell\_result = regex.sub(pattern, "", cell\_result)

else:

cell\_result = regex.sub(pattern, replacement, cell\_result)

row\_result.append(cell\_result)

result.append(row\_result)

return result

In this version, the function iterates over each pattern for each cell in the range. If a pattern is found in a cell, it replaces the pattern with the replacement string. If none of the patterns are found in a cell, the cell's content will remain unchanged. The function returns a list of lists, where each inner list represents a row in the Excel range. This should be compatible with Excel's dynamic array feature.

You can call this function from Excel with a range, a list of patterns, and a replacement string as arguments, like this:

=REGEXREPLACE(A1:B2, {"abc", "def"}, "ghi")

This will replace all occurrences of "abc" and "def" with "ghi" in the range A1:B2 and return the result as a dynamic array. If you're using an older version of Excel that doesn't support dynamic arrays, you may need to enter this formula as an array formula by pressing Ctrl+Shift+Enter. If a cell does not contain either "abc" or "def", the cell's content will remain unchanged. Please try this and let me know if it works for you. If you still encounter issues, please provide the details and I'll be glad to assist further. Remember to call this function from Excel after defining it in your Python script. You can do this by using the function name REGEXREPLACE directly in your Excel formulas. For example:

=REGEXREPLACE(A1:B2, {"abc", "def"}, "ghi")

This will replace all occurrences of "abc" and "def" with "ghi" in the range A1:B2 and return the result as a dynamic array. If you're using an older version of Excel that doesn't support dynamic arrays, you may need to enter this formula as an array formula by pressing Ctrl+Shift+Enter. If a cell does not contain either "abc" or "def", the cell's content will remain unchanged. If you want to replace spaces with an empty string, you can use the formula:

=REGEXREPLACE(A7:A15998, "\s+", "")

This will replace all occurrences of one or more spaces with an empty string in the range A7:A15998. If a cell does not contain any spaces, the cell's content will remain unchanged. If you're using an older version of Excel that doesn't support dynamic arrays, you may need to enter this formula as an array formula by pressing Ctrl+Shift+Enter.

# You

See how I tested regex in python itself:

>>> str=""

>>> import re

>>> import regex

>>> x = "This is some string"

>>> regex.sub("s+","",x)

'Thi i ome tring'

>>> regex.sub("\s+","",x)

<stdin>:1: SyntaxWarning: invalid escape sequence '\s'

'Thisissomestring'

IT works perfactly as you can see, except that warning.

# Copilot

Yes, you're correct. The Python regex.sub function works as expected in your test. The warning you're seeing is because the Python interpreter is expecting an escape sequence after the backslash (\), but \s is not a valid escape sequence in a regular Python string.

In Python, you can use raw strings to avoid this issue. Raw strings treat backslashes as literal characters and do not interpret them as the start of escape sequences. You can create a raw string by prefixing the string literal with an r or R, like this: r"\s+".

Here's how you can use a raw string in your test:

x = "This is some string"

regex.sub(r"\s+", "", x)

This should work without any warnings and produce the expected result: 'Thisissomestring'.

In the context of your Excel formula, you can try entering the pattern as a raw string like this:

=REGEXREPLACE(A7:A15998, r"\s+", "")

However, please note that Excel does not support Python's raw string syntax. If entering the pattern as a raw string in Excel does not work, you may need to double the backslashes in the pattern, like this:

=REGEXREPLACE(A7:A15998, "\\s+", "")

This tells Excel to treat the backslash as a literal character, which should allow the pattern to be passed correctly to the Python function. Please try this and let me know if it works for you. If you still encounter issues, please provide the details and I'll be glad to assist further.

# You

Can we not use built in re module for this function code as a workaround solution since regex import creates some error when replacement string is empty?

# Copilot

Yes, you can certainly use Python's built-in re module as a workaround. Here's how you can modify the function to use re instead of regex:

import re

@xw.func

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

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

@xw.arg('replacement')

def REGEXREPLACE(excel\_range, patterns, replacement):

result = []

for row in excel\_range:

row\_result = []

for cell in row:

cell\_result = cell

for pattern in patterns:

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

cell\_result = re.sub(pattern, replacement, cell\_result)

row\_result.append(cell\_result)

result.append(row\_result)

return result

This function should work the same way as the previous version, but it uses re instead of regex. Please try this and let me know if it works for you. If you still encounter issues, please provide the details and I'll be glad to assist further. Remember to call this function from Excel after defining it in your Python script. You can do this by using the function name REGEXREPLACE directly in your Excel formulas. For example:

=REGEXREPLACE(A1:B2, {"abc", "def"}, "ghi")

This will replace all occurrences of "abc" and "def" with "ghi" in the range A1:B2 and return the result as a dynamic array. If you're using an older version of Excel that doesn't support dynamic arrays, you may need to enter this formula as an array formula by pressing Ctrl+Shift+Enter. If a cell does not contain either "abc" or "def", the cell's content will remain unchanged. If you want to replace spaces with an empty string, you can use the formula:

=REGEXREPLACE(A7:A15998, "\s+", "")

This will replace all occurrences of one or more spaces with an empty string in the range A7:A15998. If a cell does not contain any spaces, the cell's content will remain unchanged. If you're using an older version of Excel that doesn't support dynamic arrays, you may need to enter this formula as an array formula by pressing Ctrl+Shift+Enter. If a cell does not contain either "abc" or "def", the cell's content will remain unchanged.

# You

Since my previous UDF above this one uses regex, how I can unload regex before using re for this udf?

# Copilot

In Python, once a module is imported, it stays in memory and there's no built-in way to unload it. However, the re and regex modules can coexist without any issues because they operate independently of each other.

If you're encountering a specific issue when using both modules in the same script, it might be due to a conflict between the functions or variables defined in these modules. In that case, you could consider using import aliases to avoid naming conflicts. Here's an example:

import re as re\_module

import regex as regex\_module

Now you can use re\_module and regex\_module in your code to refer to the re and regex modules, respectively. This way, even if both modules define a function with the same name (like sub), there won't be any conflict because you can specify which module's function you want to use:

re\_module.sub(pattern, replacement, string) # uses re.sub

regex\_module.sub(pattern, replacement, string) # uses regex.sub

Please note that this doesn't unload the regex module; it just provides a way to use both re and regex in the same script without naming conflicts. If you're encountering a different issue, please provide more details and I'll be glad to assist further.

# You

There is still traceback When I use re instead of regex but it is diffrent.

When I use your above code and render function like this: =REGEXREPLACE(A7:A15998,"\s+","")

Traceback:

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 502, in call\_udf
ret = func(\*args)
^^^^^^^^^^^
File "D:\OneDrive - 0yt2k\excel examples including add-in xlwings and connected python - vba - dlls, etc\xlwings\_rdrand\_regex\_test.py", line 75, in REGEXREPLACE
cell\_result = re.sub(pattern, replacement, cell\_result)
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "D:\Programs\Python\Lib\re\_*init*\_.py", line 186, in sub
return *compile(pattern, flags).sub(repl, string, count)
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "D:\Programs\Python\Lib\re\_\_init\_*.py", line 334, in \_compile\_template
return \_sre.template(pattern, \_parser.parse\_template(repl, pattern))
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "D:\Programs\Python\Lib\re\_parser.py", line 999, in parse\_template
s = Tokenizer(source)
^^^^^^^^^^^^^^^^^
File "D:\Programs\Python\Lib\re\_parser.py", line 235, in **init**
string = str(string, 'latin1')
^^^^^^^^^^^^^^^^^^^^^
TypeError: decoding to str: need a bytes-like object, NoneType found

Just put a check that is NoneType is found than pythin should return empty string (converted to excel "")

# Copilot

I apologize for the confusion. The error message indicates that the replacement argument is None when it should be a string. This could happen if Excel is passing an empty cell as the replacement argument.

You can modify the function to check if the replacement argument is None and replace it with an empty string in that case:

import re

@xw.func

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

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

@xw.arg('replacement')

def REGEXREPLACE(excel\_range, patterns, replacement):

result = []

if replacement is None:

replacement = ""

for row in excel\_range:

row\_result = []

for cell in row:

cell\_result = cell

for pattern in patterns:

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

cell\_result = re.sub(pattern, replacement, cell\_result)

row\_result.append(cell\_result)

result.append(row\_result)

return result

In this version of the function, if the replacement argument is None, it is replaced with an empty string before the function starts processing the Excel range. This should prevent the TypeError you're seeing when the replacement argument is None.

Please try this version of the function and let me know if it works for you. If you still encounter issues, please provide the details and I'll be glad to assist further. Remember to call this function from Excel after defining it in your Python script. You can do this by using the function name REGEXREPLACE directly in your Excel formulas. For example:

=REGEXREPLACE(A1:B2, {"abc", "def"}, "ghi")

This will replace all occurrences of "abc" and "def" with "ghi" in the range A1:B2 and return the result as a dynamic array. If you're using an older version of Excel that doesn't support dynamic arrays, you may need to enter this formula as an array formula by pressing Ctrl+Shift+Enter. If a cell does not contain either "abc" or "def", the cell's content will remain unchanged. If you want to replace spaces with an empty string, you can use the formula:

=REGEXREPLACE(A7:A15998, "\s+", "")

This will replace all occurrences of one or more spaces with an empty string in the range A7:A15998. If a cell does not contain any spaces, the cell's content will remain unchanged. If you're using an older version of Excel that doesn't support dynamic arrays, you may need to enter this formula as an array formula by pressing Ctrl+Shift+Enter. If a cell does not contain either "abc" or "def", the cell's content will remain unchanged.

# You

That finally worked!

# Copilot

I'm glad to hear that it worked! If you have any more questions or need further assistance, feel free to ask. Happy coding! 😊