## You

Why this pattern: (?<=^()(.\*?)(?=\sNSE) fails to return only "BAJAJ-AUTO" from "(BAJAJ-AUTO NSE) BAJAJ AUTO LTD." using following xlwings python UDF:
import re
@xw.func
@xw.arg('excel\_range', ndim=2)
@xw.arg('patterns', ndim=1)
def REGEXFINDM(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(cell\_str) # Return the entire string
if len(cell\_result) == len(patterns):
row\_result.append(" ".join(cell\_result))
else:
row\_result.append("")
result.append(row\_result)
return result

## Copilot

The issue you're encountering with your xlwings Python UDF seems to be related to the regular expression pattern not returning the expected result. Let's break down the problem and find a solution.

1. **Pattern Explanation:**
   The pattern (?<=^\()(.\*?)(?=\sNSE) is designed to match the text between parentheses at the beginning of a string (e.g., "(BAJAJ-AUTO NSE) BAJAJ AUTO LTD."). Specifically:
   * (?<=^\(): This part of the pattern uses a positive lookbehind to match the position right after an opening parenthesis at the start of the string.
   * (.\*?): This captures any characters (non-greedy) between the opening and closing parentheses.
   * (?=\sNSE): This uses a positive lookahead to ensure that the closing parenthesis is followed by a space and the letters "NSE."
2. **Issue:**
   The pattern is not correctly extracting only "BAJAJ-AUTO" from the given string. It seems that the entire string within the parentheses is being returned.
3. **Possible Solution:**
   To extract only "BAJAJ-AUTO," you can modify the pattern as follows:
4. Here's the breakdown:
   * \(: Matches an opening parenthesis.
   * (.\*?): Captures any characters (non-greedy) inside the parentheses.
   * \sNSE: Matches a space followed by "NSE."
   * \): Matches a closing parenthesis.
5. You can use this modified pattern in your REGEXFINDM function. Make sure to update the patterns argument accordingly.
6. **Updated UDF:**
7. Now your UDF should correctly extract "BAJAJ-AUTO" from the given string. Remember to adjust the patterns argument when calling the function.

Feel free to test this updated UDF, and let me know if you encounter any further issues! 😊🚀

Source: Conversation with Copilot, 8/7/2024

1. [Excel paste formula not working for regex pattern matching? #1950 - GitHub](https://github.com/xlwings/xlwings/issues/1950)
2. [Troubleshooting - xlwings Documentation](https://docs.xlwings.org/en/stable/troubleshooting.html)
3. [Troubleshooting - xlwings Documentation](https://docs.xlwings.org/en/0.27.1/troubleshooting.html)
4. [HTML Pattern - regex not working - Stack Overflow](https://stackoverflow.com/questions/16542665/html-pattern-regex-not-working)

## You

Let me test this in python itself. Give me a small function to test this pattern in python.

## Copilot

Certainly! Here's a small Python function that uses the modified pattern to extract the desired substring from a given string:

import re

def extract\_bajaj\_auto(text):

pattern = r'\((.\*?)\sNSE\)' # Modified pattern

match = re.search(pattern, text)

if match:

return match.group(1) # Extract the captured group

else:

return None # Return None if no match found

# Example usage:

input\_string = "(BAJAJ-AUTO NSE) BAJAJ AUTO LTD."

result = extract\_bajaj\_auto(input\_string)

print(f"Extracted substring: {result}")

Replace input\_string with your desired input, and the function will return the extracted substring (e.g., "BAJAJ-AUTO"). Let me know if you need further assistance! 😊🚀

## You

That succeeded in extracting exactly BAJAJ-AUTO. Why following UDF fails?

import re
@xw.func
@xw.arg('excel\_range', ndim=2)
@xw.arg('patterns', ndim=1)
def REGEXFINDM(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(cell\_str) # Return the entire string
if len(cell\_result) == len(patterns):
row\_result.append(" ".join(cell\_result))
else:
row\_result.append("")
result.append(row\_result)
return result

## Copilot

The issue with your current REGEXFINDM UDF lies in how it processes the matches. Let's break down the problem and provide a solution.

1. **Issue:**
   In your existing UDF, you're appending the entire cell string (cell\_str) to cell\_result when a match is found. This means that even if the pattern extracts only a portion of the string (e.g., "BAJAJ-AUTO"), you're still returning the entire cell content.
2. **Solution:**
   To fix this, modify your UDF to extract the captured group from the match (i.e., the part of the string that matches the pattern). Specifically, use match.group(1) to get the desired substring.
3. Here's the updated UDF:
4. Now your UDF should correctly return only the extracted substring (e.g., "BAJAJ-AUTO") instead of the entire cell content.

Feel free to test this updated UDF, and let me know if you encounter any further issues! 😊🚀

## You

Okay. But will this be successful as a generic function in other cases also - supppose there is no group then?

## Copilot

Great question! Let's consider the generic behavior of the updated REGEXFINDM function when there is no capturing group in the pattern.

1. **Capturing Group in Pattern:**
   * When your pattern contains a capturing group (e.g., r'\((.\*?)\sNSE\)'), the function extracts the content within the parentheses.
   * For example, given the input string "(BAJAJ-AUTO NSE) BAJAJ AUTO LTD.," it correctly returns "BAJAJ-AUTO."
2. **No Capturing Group in Pattern:**
   * If your pattern doesn't contain any capturing groups, the function will still work but will return the entire matched substring.
   * For instance, if you use a pattern like r'\bBAJAJ-AUTO\b', it will match the whole string "(BAJAJ-AUTO NSE) BAJAJ AUTO LTD." and return the entire content.
3. **Handling No Capturing Group:**
   * To handle cases where there is no capturing group, you can modify the function to return the entire matched substring (as it currently does) when no capturing group is present.
   * If you specifically want to return an empty string when there's no match, you can adjust the logic accordingly.
4. **Generic Behavior:**
   * The updated function is more generic because it can handle both cases (with and without capturing groups).
   * If you encounter scenarios where you need to extract specific parts of the matched string, ensure that your pattern includes capturing groups.

Remember that the behavior of the function depends on the regular expression pattern you provide. Adjust the pattern based on your specific requirements, and the function should work as expected! If you have any other questions or need further assistance, feel free to ask. 😊🚀

## You

I renamed above function as REGEXFINDGROUP and kept original function (not having selection of a group from a pattern) as REGEXFINDM.

I want to succeed with the following pattern for this new function but not able to do so:

(?i)^\d{6}((?=.\*direct)|(?=.\*growth)|(?=.\*gold)|(?=.*silver))((?!(equity|hybrid|Solution Oriented|FOF|elss|regular|idcw|dividend|div|hybrid|balanced advantage|index|nifty)).)*$

If I change pattern like this:

(?i)^\d{6}((?=.\*direct)|(?=.\*growth)|(?=.\*gold)|(?=.*silver))((?!(equity|hybrid|Solution Oriented|FOF|elss|regular|idcw|dividend|div|hybrid|balanced advantage|index|nifty)).)*$

then it hits the error that global flags are not at the beginning:

"re.error: global flags not at the start of the expression at position 1
raise source.error('global flags not at the start '
File ""D:\Programs\Python\Lib\re\_parser.py"", line 840, 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\regexp.py"", line 80, in REGEXFINDGROUP
^^^^^^^^^^^
ret = func(\*args)
File ""D:\Programs\Python\Lib\site-packages\xlwings\udfs.py"", line 502, 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 with the global flags not being at the beginning of your regular expression pattern is due to the way Python's re module handles flags. Let's address this issue and find a solution.

1. **Global Flags:**
   * In Python, global flags (such as (?i) for case-insensitive matching) must appear at the beginning of the regular expression pattern.
   * When you place the global flags after the start of the pattern, it results in an error.
2. **Solution:**
   * To fix this, ensure that the global flags are placed at the beginning of your pattern.
   * Here's the corrected pattern with the global flags at the start:
3. **Updated UDF:**
   * Use this corrected pattern in your REGEXFINDGROUP function to ensure that the global flags are in the correct position.
4. **Example Usage:**
5. Replace input\_text with your desired input, and the function will return the entire matched substring based on your pattern.

Feel free to test this corrected pattern, and let me know if you encounter any further issues! 😊🚀

Source: Conversation with Copilot, 8/7/2024

1. [Excel paste formula not working for regex pattern matching? #1950 - GitHub](https://github.com/xlwings/xlwings/issues/1950)
2. [Troubleshooting - xlwings Documentation](https://docs.xlwings.org/en/stable/troubleshooting.html)
3. [Troubleshooting - xlwings Documentation](https://docs.xlwings.org/en/0.27.1/troubleshooting.html)
4. [HTML Pattern - regex not working - Stack Overflow](https://stackoverflow.com/questions/16542665/html-pattern-regex-not-working)
5. [Replace only based on full match and not on partial match #1961 - GitHub](https://github.com/xlwings/xlwings/issues/1961)

## You

((?i)^\d{6}((?=.\*direct)|(?=.\*growth)|(?=.\*gold)|(?=.*silver))((?!(equity|hybrid|Solution Oriented|FOF|elss|regular|idcw|dividend|div|hybrid|balanced advantage|index|nifty)).)*$)

See I have following dll (source which succeeds in getting data) with the above pattern:

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <glib.h>

// Function to match pattern in an array of strings
int match\_pattern\_in\_array(char \*\*input\_array, int array\_length, const char \*pattern, char \*\*\*output\_array) {
GError \*error = NULL;
GRegex \*regex = g\_regex\_new(pattern, 0, 0, &error);

if (error) {

fprintf(stderr, "Error compiling regex: %s\n", error->message);

g\_error\_free(error);

return -1;

}

// Initial allocation for the matches array

size\_t allocated\_size = 100;

char \*\*matches = malloc(allocated\_size \* sizeof(char \*));

if (!matches) {

g\_regex\_unref(regex);

return -1;

}

int match\_count = 0;

for (int i = 0; i < array\_length; i++) {

if (g\_regex\_match(regex, input\_array[i], 0, NULL)) {

if (match\_count >= allocated\_size) {

// Reallocate memory if needed

allocated\_size \*= 2;

char \*\*new\_matches = realloc(matches, allocated\_size \* sizeof(char \*));

if (!new\_matches) {

g\_regex\_unref(regex);

for (int j = 0; j < match\_count; j++) {

free(matches[j]);

}

free(matches);

return -1;

}

matches = new\_matches;

}

matches[match\_count] = strdup(input\_array[i]);

if (!matches[match\_count]) {

g\_regex\_unref(regex);

for (int j = 0; j < match\_count; j++) {

free(matches[j]);

}

free(matches);

return -1;

}

match\_count++;

}

}

g\_regex\_unref(regex);

\*output\_array = matches;

return match\_count;

}

// Function to free the array of matched strings
void free\_matches(char \*\*matches, int match\_count) {
for (int i = 0; i < match\_count; i++) {
free(matches[i]);
}
free(matches);
}
//Compilation: gcc regex-C-xlwings-2.c -shared $(pkg-config --cflags --libs glib-2.0) -o regex-C-xlwings-2.dll

But following python UDF fails:

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

Python error is:

"re.error: global flags not at the start of the expression at position 1
raise source.error('global flags not at the start '
File ""D:\Programs\Python\Lib\re\_parser.py"", line 840, in \_parse
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
itemsappend(\_parse(source, state, verbose, nested + 1,

Can you look in to?

## Copilot

Certainly! Let's address the issue you're encountering with the Python UDF and the global flags not being at the start of the expression. We'll find a solution to make it work correctly.

1. **Global Flags in Python Regular Expressions:**
   * In Python, global flags (such as (?i) for case-insensitive matching) must appear at the beginning of the regular expression pattern.
   * When you place the global flags after the start of the pattern, it results in an error.
2. **Solution:**
   * To fix this, ensure that the global flags are placed at the beginning of your pattern.
   * Here's the corrected pattern with the global flags at the start:
3. **Updated UDF:**
   * Use this corrected pattern in your REGEXFINDGROUP function to ensure that the global flags are in the correct position.
4. **Example Usage:**
5. **Usage in Excel:**
   * In your Excel sheet, use this UDF with the desired range and patterns (e.g., =REGEXFINDGROUP(A1:A10, {"pattern1", "pattern2"})).

Feel free to test this updated UDF, and let me know if you encounter any further issues! 😊🚀

Source: Conversation with Copilot, 8/7/2024

1. [Excel paste formula not working for regex pattern matching? #1950 - GitHub](https://github.com/xlwings/xlwings/issues/1950)
2. [Troubleshooting - xlwings Documentation](https://docs.xlwings.org/en/stable/troubleshooting.html)
3. [Troubleshooting - xlwings Documentation](https://docs.xlwings.org/en/0.27.1/troubleshooting.html)
4. [HTML Pattern - regex not working - Stack Overflow](https://stackoverflow.com/questions/16542665/html-pattern-regex-not-working)
5. [Replace only based on full match and not on partial match #1961 - GitHub](https://github.com/xlwings/xlwings/issues/1961)

## You

What to do if my pattern is in cell K1 and I am using function like this:

=REGEXFINDGROUP(A2:A12022,CONCAT(K1))

## Copilot

If your pattern is stored in cell K1, you can dynamically reference it in your REGEXFINDGROUP function using the INDIRECT function. Here's how you can modify your formula:

=REGEXFINDGROUP(A2:A12022, INDIRECT(K1))

This will use the value from cell K1 as the pattern in your REGEXFINDGROUP function. Make sure that the value in K1 is a valid regular expression pattern.

Feel free to try this updated formula, and let me know if you have any further questions! 😊🚀

## You

Problem - Just for testing I used:
=INDIRECT(K1)

REF# error

## Copilot

I apologize for the confusion. It appears that the INDIRECT function is returning a #REF! error. Let's troubleshoot this issue.

1. **Check Cell Reference:**
   * Ensure that the value in cell K1 contains a valid cell reference (e.g., "A1" or "Sheet1!A1").
   * If the value in K1 is not a valid cell reference, the INDIRECT function will result in an error.
2. **Verify Worksheet Name:**
   * If you're using a sheet name in the cell reference (e.g., "Sheet1!A1"), make sure that the sheet name exists in your workbook.
   * Double-check the spelling and capitalization of the sheet name.
3. **Evaluate Formula:**
   * To debug further, you can evaluate the formula step by step:
     1. Select the cell containing the formula.
     2. Press Ctrl + ~ (tilde) to display the formula view.
     3. Check the intermediate results to identify any issues.
4. **Cell Permissions:**
   * Ensure that the cell referenced by the INDIRECT function is not locked or protected.
   * If the cell is protected, you may need to adjust the worksheet protection settings.
5. **Circular References:**
   * Be cautious of circular references. If the INDIRECT function indirectly refers back to itself, it can cause errors.
6. **Error Handling:**
   * Wrap the INDIRECT function with an IFERROR function to handle errors gracefully:
   * This will display "Invalid reference" if the INDIRECT function encounters an error.

Please review these points, and let me know if you need further assistance or if there are any specific details about your workbook setup that I should consider. 😊🚀

## You

Instead of using INDIRECT what is the issue with following:

=REGEXFINDGROUP(A2:A12022,CONCAT(K1))

## Copilot

I apologize for any confusion. Let's address the issue with your formula:

The problem lies in the use of CONCAT(K1) within the REGEXFINDGROUP function. The CONCAT function concatenates the values in the specified range or array, but it doesn't directly reference a cell. Therefore, it won't work as expected when used with INDIRECT.

To dynamically reference the pattern stored in cell K1, you can directly use the value from K1 as the pattern in your REGEXFINDGROUP function. Here's the corrected formula:

=REGEXFINDGROUP(A2:A12022, K1)

Make sure that the value in cell K1 is a valid regular expression pattern. If you encounter any further issues, feel free to ask! 😊🚀

## You

Problem is that the pattern is not considered as group in Python and REGEXGROUP returns Group 1. How to change the pattern so that function thinks it is group 1?

## Copilot

I understand the issue you're facing. To ensure that your pattern is treated as Group 1 in Python's regular expressions, you can explicitly create a capturing group around the entire pattern. Let's modify your pattern accordingly:

1. **Original Pattern:**
2. **Modified Pattern (with Capturing Group):**

In the modified pattern, I've added an additional pair of parentheses around the entire expression. This ensures that the entire pattern is treated as Group 1. Now when you use match.group(1), it will correctly extract the content based on this capturing group.

Feel free to test this updated pattern, and let me know if you encounter any further issues! 😊🚀

## You

Slightly changed the pattern to make worksround work:
(?i)(^\d{6}((?=.\*direct)|(?=.\*growth)|(?=.\*gold)|(?=.*silver))((?!(equity|hybrid|Solution Oriented|FOF|elss|regular|idcw|dividend|div|hybrid|balanced advantage|index|nifty)).)*)

I have put bracket "(" before ^\d{6} instead of after that other wise first 6 digits were not returned.

## Copilot

Thank you for sharing the modified pattern! I appreciate your attention to detail. It's great to hear that the adjustment resolved the issue, and now the first 6 digits are correctly returned.

If you have any more questions or need further assistance, feel free to ask. I'm here to help! 😊🚀