



Sorry, we no longer support Internet Explorer

Please upgrade to [Google Chrome](#) or [Firefox](#). Learn more about our [browser support](#).

Search less. Build more. Use Stack Overflow for Teams at work to share knowledge with your colleagues. Free 30 day trial. [Start your trial](#).



How to use Regular Expressions (Regex) in Microsoft Excel both in-cell and loops

Asked 5 years, 11 months ago Active today Viewed 848k times



566



407



How can I use regular expressions in Excel and take advantage of Excel's powerful grid-like setup for data manipulation?

- In-cell function to return a matched pattern or replaced value in a string.
- Sub to loop through a column of data and extract matches to adjacent cells.
- What setup is necessary?
- What are Excel's special characters for Regular expressions?

I understand Regex is not ideal for many situations ([To use or not to use regular expressions?](#)) since excel can use Left , Mid , Right , Instr type commands for similar manipulations.

regex

excel

vba

edited May 24 '19 at 15:01



Andrei Konstantinov

4,233 3 26 49

asked Mar 20 '14 at 19:09



Portland Runner

25.3k 11 48 75

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

9

I high recommend [this VB/VBA Regexp article](#) by Patrick Matthews – [brettdj](#) Apr 17 '14 at 13:26



Try this free add-in: seotoolsforexcel.com/regexpfind – [Niels Bosma](#) Sep 23 '15 at 6:28

- 1 Let's not forget the [Like operator](#), which provides a sort of light version of regex-style functionality. It's typically much faster than regex, even if wrapped in a sub or function procedure. – [Egalth](#) Nov 1 '18 at 10:09

9 Answers



By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).





[Regular expressions](#) are used for Pattern Matching.

To use in Excel follow these steps:

Step 1: Add VBA reference to "Microsoft VBScript Regular Expressions 5.5"

- Select "Developer" tab ([I don't have this tab what do I do?](#))
- Select "Visual Basic" icon from 'Code' ribbon section
- In "Microsoft Visual Basic for Applications" window select "Tools" from the top menu.
- Select "References"
- Check the box next to "Microsoft VBScript Regular Expressions 5.5" to include in your workbook.
- Click "OK"

Step 2: Define your pattern

Basic definitions:

- Range.

- E.g. `a-z` matches an lower case letters from a to z
- E.g. `0-5` matches any number from 0 to 5

[] Match exactly one of the objects inside these brackets.

- E.g. `[a]` matches the letter a
- E.g. `[abc]` matches a single letter which can be a, b or c
- E.g. `[a-z]` matches any single lower case letter of the alphabet.

() Groups different matches for return purposes. See examples below.

{ } Multiplier for repeated copies of pattern defined before it.

- E.g. `[a]{2}` matches two consecutive lower case letter a: aa
- E.g. `[a]{1,3}` matches at least one and up to three lower case letter a , aa , aaa

+ Match at least one, or more, of the pattern defined before it.

- E.g. `a+` will match consecutive a's a , aa , aaa , and so on

? Match zero or one of the pattern defined before it.

By using our product you acknowledge that you understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

→ E.g. `[a-z]?` matches empty string or any single lower case letter.



* Match zero or more of the pattern defined before it. - E.g. Wildcard for pattern that may or may not be present. - E.g. `[a-z]*` matches empty string or string of lower case letters.

. Matches any character except newline `\n`

- E.g. `a.` Matches a two character string starting with a and ending with anything except `\n`

| OR operator

- E.g. `a|b` means either `a` or `b` can be matched.
- E.g. `red|white|orange` matches exactly one of the colors.

^ NOT operator

- E.g. `[^0-9]` character can not contain a number
- E.g. `[^aA]` character can not be lower case `a` or upper case `A`

\ Escapes special character that follows (overrides above behavior)

- E.g. `\.`, `\\`, `\(`, `\?`, `\$`, `\^`

Anchoring Patterns:

^ Match must occur at start of string

- E.g. `^a` First character must be lower case letter `a`
- E.g. `^[0-9]` First character must be a number.

\$ Match must occur at end of string

- E.g. `a$` Last character must be lower case letter `a`

Precedence table:

Order	Name	Representation
1	Parentheses	<code>()</code>
2	Multipliers	<code>? + * {m,n} {m, n}?</code>
3	Sequence & Anchors	<code>abc ^ \$</code>
4	Alternation	<code> </code>

Predefined Character Abbreviations:

abr	same as	meaning
<code>\d</code>	<code>[0-9]</code>	Any single digit
<code>\D</code>	<code>[^0-9]</code>	Any single character that's <i>not a digit</i>
<code>\w</code>	<code>[a-zA-Z0-9_]</code>	Any word character
<code>\W</code>	<code>[^a-zA-Z0-9_]</code>	Any non-word character
<code>\s</code>	<code>[\r\t\n\f]</code>	Any space character
<code>\S</code>	<code>[^\r\t\n\f]</code>	Any non-space character
<code>\n</code>	<code>[\n]</code>	New line

By using our site you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).



Example 1: Run as macro

The following example macro looks at the value in cell A1 to see if the first 1 or 2 characters are digits. If so, they are removed and the rest of the string is displayed. If not, then a box appears telling you that no match is found. Cell A1 values of 12abc will return abc, value of 1abc will return abc, value of abc123 will return "Not Matched" because the digits were not at the start of the string.

```
Private Sub simpleRegex()
    Dim strPattern As String: strPattern = "[0-9]{1,2}"
    Dim strReplace As String: strReplace = ""
    Dim regEx As New RegExp
    Dim strInput As String
    Dim Myrange As Range

    Set Myrange = ActiveSheet.Range("A1")

    If strPattern <> "" Then
        strInput = Myrange.Value

        With regEx
            .Global = True
            .Multiline = True
            .IgnoreCase = False
            .Pattern = strPattern
        End With

        If regEx.Test(strInput) Then
            MsgBox (regEx.Replace(strInput, strReplace))
        Else
            MsgBox ("Not matched")
        End If
    End If
End Sub
```

Example 2: Run as an in-cell function

This example is the same as example 1 but is setup to run as an in-cell function. To use, change the code to this:

```
Function simpleCellRegex(Myrange As Range) As String
    Dim regEx As New RegExp
    Dim strPattern As String
    Dim strInput As String
    Dim strReplace As String
    Dim strOutput As String

    strPattern = "[0-9]{1,3}"

    If strPattern <> "" Then
        strInput = Myrange.Value
        strReplace = ""
```

```

.IgnoreCase = False
.Pattern = strPattern
End With

If regEx.test(strInput) Then
    simpleCellRegex = regEx.Replace(strInput, strReplace)
Else
    simpleCellRegex = "Not matched"
End If
End If
End Function

```

Place your strings ("12abc") in cell A1 . Enter this formula =simpleCellRegex(A1) in cell B1 and the result will be "abc".

	A	B
1	12abc	abc
2		
3		

Example 3: Loop Through Range

This example is the same as example 1 but loops through a range of cells.

```

Private Sub simpleRegex()
    Dim strPattern As String: strPattern = "^[0-9]{1,2}"
    Dim strReplace As String: strReplace = ""
    Dim regEx As New RegExp
    Dim strInput As String
    Dim Myrange As Range

    Set Myrange = ActiveSheet.Range("A1:A5")

    For Each cell In Myrange
        If strPattern <> "" Then
            strInput = cell.Value

            With regEx
                .Global = True
                .Multiline = True
                .IgnoreCase = False
                .Pattern = strPattern
            End With

            If regEx.Test(strInput) Then
                MsgBox (regEx.Replace(strInput, strReplace))
            Else
                MsgBox ("Not matched")
            End If
        End If
    Next
End Sub

```

Example 4: Splitting apart different patterns

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).



This example loops through a range (A1 , A2 & A3) and looks for a string starting with three digits followed by a single alpha character and then 4 numeric digits. The output splits apart the pattern matches into adjacent cells by using the () . \$1 represents the first pattern matched within the first set of () .

```
Private Sub splitUpRegexPattern()
    Dim regEx As New RegExp
    Dim strPattern As String
    Dim strInput As String
    Dim Myrange As Range

    Set Myrange = ActiveSheet.Range("A1:A3")

    For Each C In Myrange
        strPattern = "^([0-9]{3})([a-zA-Z])([0-9]{4})"

        If strPattern <> "" Then
            strInput = C.Value

            With regEx
                .Global = True
                .Multiline = True
                .IgnoreCase = False
                .Pattern = strPattern
            End With

            If regEx.test(strInput) Then
                C.Offset(0, 1) = regEx.Replace(strInput, "$1")
                C.Offset(0, 2) = regEx.Replace(strInput, "$2")
                C.Offset(0, 3) = regEx.Replace(strInput, "$3")
            Else
                C.Offset(0, 1) = "(Not matched)"
            End If
        End If
    Next
End Sub
```

Results:

	A	B	C	D
1	123A4567	123	A	4567
2	321A7654	321	A	7654
3	A1234567	(Not matched)		

Additional Pattern Examples

String	Regex Pattern	Explanation
a1aaa	[a-zA-Z][0-9][a-zA-Z]{3}	Single alpha, single digit, three alpha characters
a1aaa	[a-zA-Z]?[0-9][a-zA-Z]{3}	May or may not have preceding alpha character
a1aaa	[a-zA-Z][0-9][a-zA-Z]{0,3}	Single alpha, single digit, 0 to 3 alpha characters
a1aaa	[a-zA-Z][0-9][a-zA-Z]*	Single alpha, single digit, followed by any number of alpha characters

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#)



edited 18 hours ago

answered Mar 20 '14 at 19:09



TylerH

17.4k 11 58 74



Portland Runner

25.3k 11 48 75

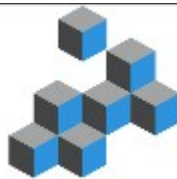
1 You should not forget to `Set regex = Nothing`. You will get Out Of Memory exceptions, when that Sub is executed frequently enough. – Kiril Mar 13 '15 at 10:28

I adapted example 4 with [SubMatches](#) for treating more complex regex, basically I don't use replace when splitting, if anyone is interested: stackoverflow.com/questions/30218413/... – Armfoot May 13 '15 at 14:58

) Late binding line: `Set regex = CreateObject("VBScript.RegExp")` – ZygD Dec 5 '15 at 11:23

! Okay, I'm pretty sure it's because the code is in `ThisWorkbook`. Try moving the code to a separate Module. – Portland Runner May 9 '19 at 4:00 ✎

! @PortlandRunner in the "project explorer" (?) this excel file lacked a "Modules" subfolder, although another file showed one. Right-clicked the file and chose 'insert module', then double-clicked "Module 1" and pasted the code. Saved. Back to workbook and keyed in the function again - it worked. Might be noteworthy in the answer, for the sake of the inexperienced like me? Thanks for the help. – youcanttryreachingme May 10 '19 at 5:52



Build and develop apps with Azure.
Free until you say otherwise.

Try Azure f

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).





To make use of regular expressions directly in Excel formulas the following UDF (user defined function) can be of help. It more or less directly exposes regular expression functionality as an excel function.

How it works

It takes 2-3 parameters.

1. A text to use the regular expression on.
2. A regular expression.
3. A format string specifying how the result should look. It can contain \$0 , \$1 , \$2 , and so on. \$0 is the entire match, \$1 and up correspond to the respective match groups in the regular expression. Defaults to \$0 .

Some examples

Extracting an email address:

```
=regex("Peter Gordon: some@email.com, 47", "\w+@\w+\.\w+")
=regex("Peter Gordon: some@email.com, 47", "\w+@\w+\.\w+", "$0")
```

Results in: some@email.com

Extracting several substrings:

```
=regex("Peter Gordon: some@email.com, 47", "^(.+): (.+), (\d+)$", "E-Mail: $2, Name: $1")
```

Results in: E-Mail: some@email.com, Name: Peter Gordon

To take apart a combined string in a single cell into its components in multiple cells:

```
=regex("Peter Gordon: some@email.com, 47", "^(.+): (.+), (\d+)$", "$" & 1)
=regex("Peter Gordon: some@email.com, 47", "^(.+): (.+), (\d+)$", "$" & 2)
```

Results in: Peter Gordon some@email.com ...

How to use

To use this UDF do the following (roughly based on [this Microsoft page](#). They have some good additional info there!):

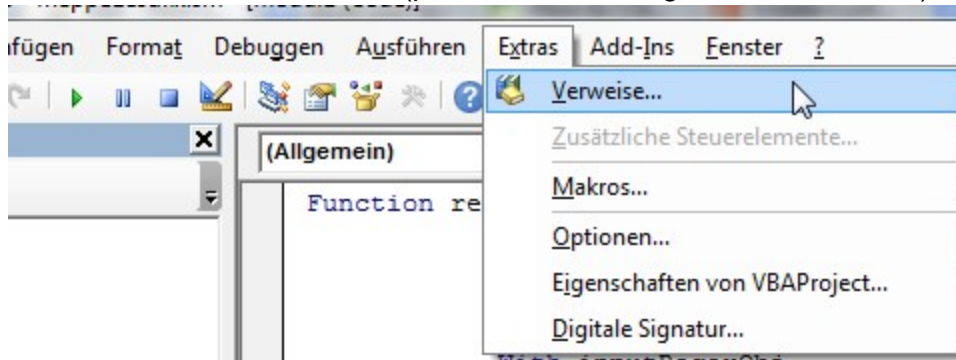
1. In Excel in a Macro enabled file ('.xlsm') push ALT+F11 to open the *Microsoft Visual Basic for Applications* Editor.

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).



2. Add VBA reference to the Regular Expressions library (shamelessly copied from [Portland Runners++ answer](#)):

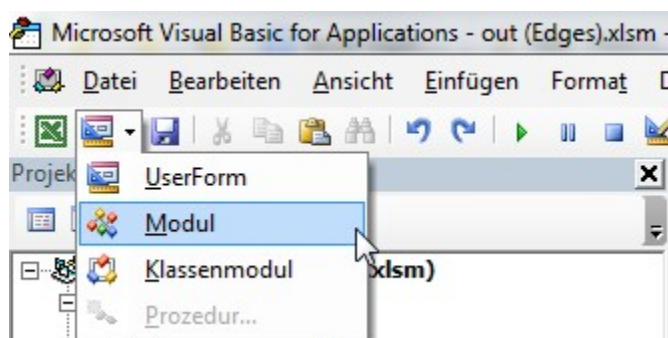
1. Click on *Tools -> References* (please excuse the german screenshot)



2. Find *Microsoft VBScript Regular Expressions 5.5* in the list and tick the checkbox next to it.

3. Click OK.

3. Click on *Insert Module*. If you give your module a different name make sure the Module does *not* have the same name as the UDF below (e.g. naming the Module `Regex` and the function `regex` causes `#NAME!` errors).



4. In the big text window in the middle insert the following:

```
Function regex(strInput As String, matchPattern As String, Optional ByVal outputPattern
As String = "$0") As Variant
    Dim inputRegexObj As New VBScript_RegExp_55.RegExp, outputRegexObj As New
VBScript_RegExp_55.RegExp, outReplaceRegexObj As New VBScript_RegExp_55.RegExp
    Dim inputMatches As Object, replaceMatches As Object, replaceMatch As Object
    Dim replaceNumber As Integer

    With inputRegexObj
        .Global = True
        .MultiLine = True
        .IgnoreCase = False
        .Pattern = matchPattern
    End With
    With outputRegexObj
        .Global = True
        .MultiLine = True
        .IgnoreCase = False
        .Pattern = "$(\d+)"
    End With
    With outReplaceRegexObj
        .Global = True
        .MultiLine = True
        .IgnoreCase = False
```

```

End With

Set inputMatches = inputRegexObj.Execute(strInput)
If inputMatches.Count = 0 Then
    regex = False
Else
    Set replaceMatches = outputRegexObj.Execute(outputPattern)
    For Each replaceMatch In replaceMatches
        replaceNumber = replaceMatch.SubMatches(0)
        outReplaceRegexObj.Pattern = "$" & replaceNumber

        If replaceNumber = 0 Then
            outputPattern = outReplaceRegexObj.Replace(outputPattern, inputMatches
(0).Value)
        Else
            If replaceNumber > inputMatches(0).SubMatches.Count Then
                'regex = "A to high $ tag found. Largest allowed is $" &
inputMatches(0).SubMatches.Count & "."
                regex = CVErr(xlErrValue)
                Exit Function
            Else
                outputPattern = outReplaceRegexObj.Replace(outputPattern,
inputMatches(0).SubMatches(replaceNumber - 1))
            End If
        End If
    Next
    regex = outputPattern
End If
End Function

```

5. Save and close the *Microsoft Visual Basic for Applications* Editor window.

edited Sep 28 '15 at 15:17

answered Jan 27 '15 at 17:47



Patrick Böker

2,612 1 17 20

- ⌋ This answer combined with the steps [here](#) to create an Add-In, has been very helpful. Thank you. Make sure you don't give your module and function the same name! – [Chris Hunt](#) Feb 24 '15 at 19:03
- ⌋ Just reiterating the comment above from Chris Hunt. Don't call your Module 'Regex' as well. Thought I was going mad for a while as the function wouldn't work due to a #NAME error – [Chris](#) Sep 28 '15 at 14:57
- Well, I'm gone nuts as I tried everything (including changing modules/names) and still getting the #NAME error >_> i.imgur.com/UUQ6eCi.png – [Enissay](#) Aug 15 '16 at 20:46
- @Enissay: Try creating a minimal Function foo() As Variant \n foo="Hello World" \n End Function UDF to see if that works. If yes, work your way up to the full thing above, if no something basic is broken (macros disabled?). – [Patrick Böker](#) Aug 16 '16 at 7:27
- I would recommend [This tool](#) it works better than the above version, there are flaws in the above version, doesn't work well with multiline matching. And it is an addin so easier to use. – [Vijay](#) Apr 27 '17 at 9:24



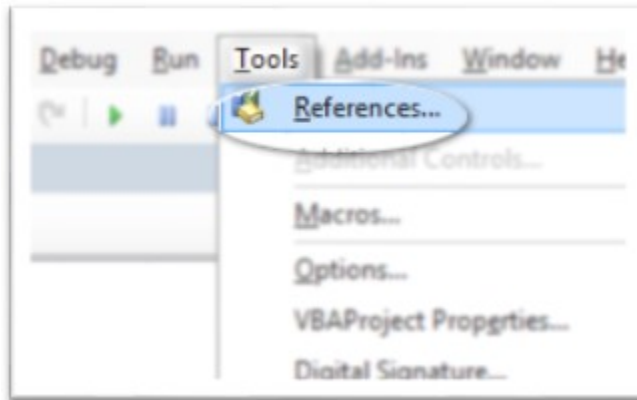
By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).



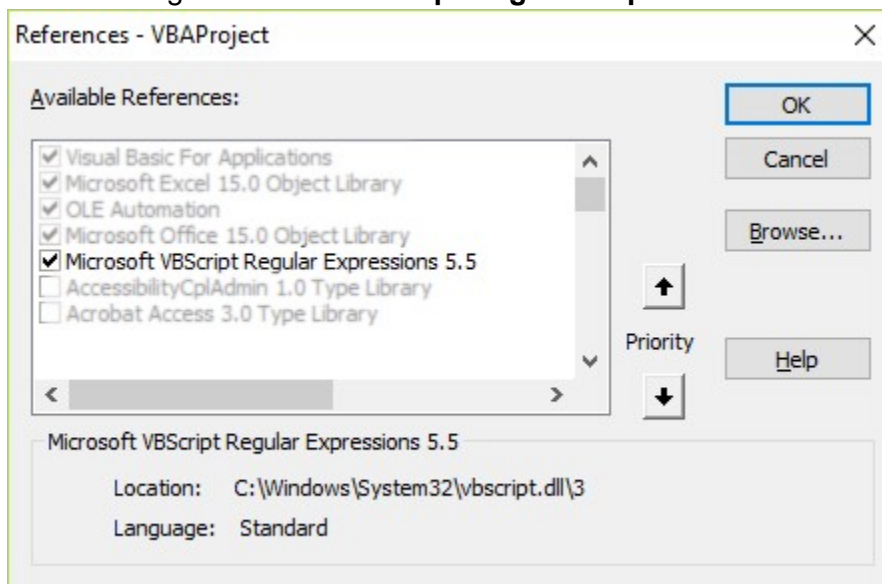


Expanding on [patszim's answer](#) for those in a rush.

1. Open Excel workbook.
2. **Alt** + **F11** to open VBA/Macros window.
3. Add reference to regex under **Tools** then **References**



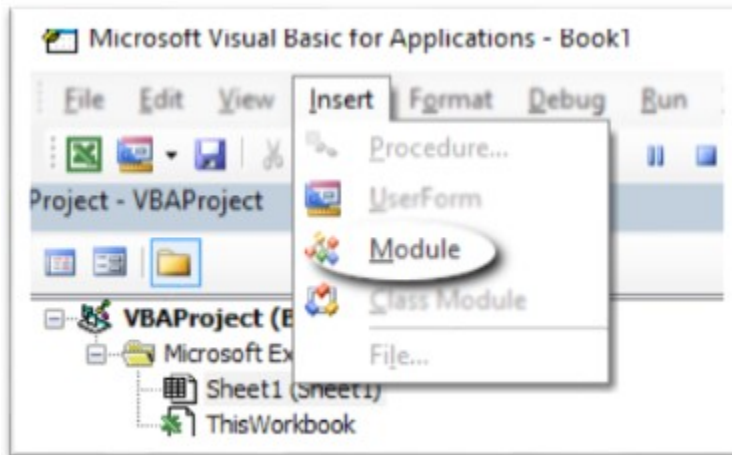
4. and selecting **Microsoft VBScript Regular Expression 5.5**



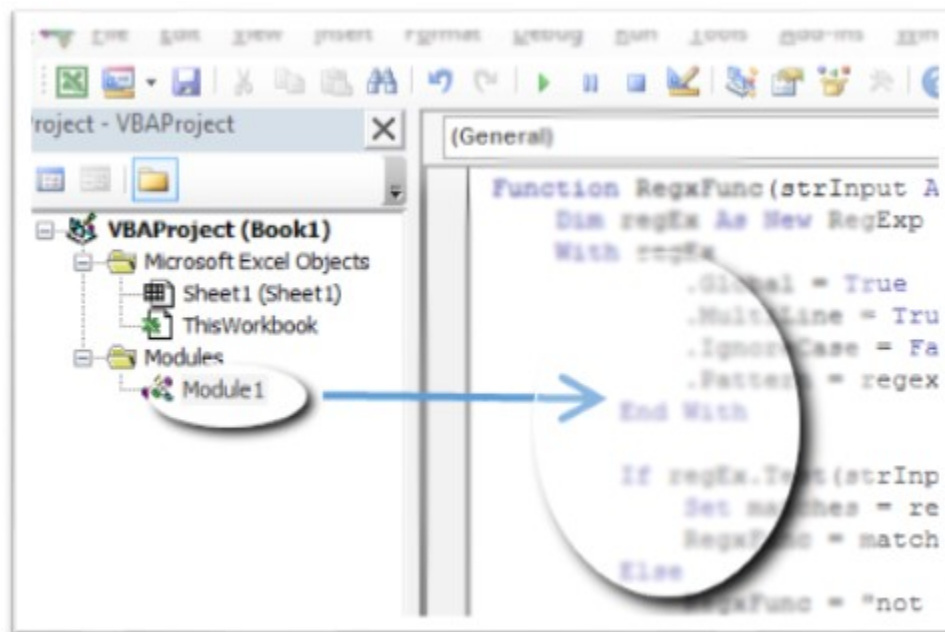
By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).



5. Insert a new module (code needs to reside in the module otherwise it doesn't work).



6. In the newly inserted module,



7. add the following code:

```
Function RegxFunc(strInput As String, regexPattern As String) As String
    Dim regEx As New RegExp
    With regEx
        .Global = True
        .MultiLine = True
        .IgnoreCase = False
        .Pattern = regexPattern
    End With

    If regEx.Test(strInput) Then
        Set matches = regEx.Execute(strInput)
        RegxFunc = matches(0).Value
    Else
        RegxFunc = "not matched"
    End If
End Function
```

8. The regex pattern is placed in one of the cells and **absolute referencing** is used on it.

The screenshot shows an Excel spreadsheet with columns A, B, C, and D, and rows 1 through 8. In cell B1, the text 'Regex pattern for numeric postal code' is displayed. In cell C1, the regex pattern '\d{3,10}' is entered. Below this, a table is shown with two columns: 'Input' and 'Extracted'. The 'Input' column contains three rows: 'Postal code 10022', 'ZIP code 37188', and 'Code: 89109'. The 'Extracted' column contains the results of the regex function: '=RegxFunc(B5,\$C\$2)' in the first row, '37188' in the second row, and '89109' in the third row. The formula bar on the right shows the formula '=RegxFunc(B5,\$C\$2)'.

Input	Extracted
Postal code 10022	=RegxFunc(B5,\$C\$2)
ZIP code 37188	37188
Code: 89109	89109

Function will be tied to workbook that its created in.

If there's a need for it to be used in different workbooks, store the function in

Personal.XLSB

edited May 23 '17 at 12:26



Community ♦

1 1

answered Mar 30 '17 at 21:28



SAm

1,588 24 25

Thanks for mentioning it needs to be in Personal.xlsb to be available in all Excel documents you work on. Most (?) other answers don't make that clear. Personal.XLSB would go in the folder (might need to create the folder) C:\Users\user name\AppData\Local\Microsoft\Excel\XLStart folder

– Mark Stewart Jun 7 '19 at 14:29



Here is my attempt:

```
Function RegParse(ByVal pattern As String, ByVal html As String)
    Dim regex As RegExp
    Set regex = New RegExp

    With regex
        .IgnoreCase = True 'ignoring cases while regex engine performs the search.
        .pattern = pattern 'declaring regex pattern.
        .Global = False 'restricting regex to find only first match.

        If .Test(html) Then 'Testing if the pattern matches or not
            mStr = .Execute(html)(0) 'Execute(html)(0) will provide the String
            which matches with Regex
            RegParse = .Replace(mStr, "$1") 'Replace function will replace the String
            with whatever is in the first set of braces - $1.
        Else
            RegParse = "#N/A"
        End If
    End With
End Function
```

edited Jun 9 '18 at 16:10



Neuron

3,298 3 20 37

answered Aug 17 '15 at 23:03



Vikas Gautam

1,201 14 20



By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).





I needed to use this as a cell function (like `SUM` or `VLOOKUP`) and found that it was easy to:

1. Make sure you are in a Macro Enabled Excel File (save as xlsx).
2. Open developer tools `Alt` + `F11`
3. Add *Microsoft VBScript Regular Expressions 5.5* as in other answers
4. Create the following function either in workbook or in its own module:

```
Function REGPLACE(myRange As Range, matchPattern As String, outputPattern As String) As Variant
    Dim regex As New VBScript_RegExp_55.RegExp
    Dim strInput As String

    strInput = myRange.Value

    With regex
        .Global = True
        .MultiLine = True
        .IgnoreCase = False
        .Pattern = matchPattern
    End With

    REGPLACE = regex.Replace(strInput, outputPattern)

End Function
```

5. Then you can use in cell with `=REGPLACE(B1, "(\w) (\d+)", "$1$2")` (ex: "A 243" to "A243")

edited Jun 9 '18 at 16:10



Neuron

3,298 3 20 37

answered Mar 18 '17 at 2:41



DeezCashews

1,767 17 20

This naming of outputPattern threw me off. It's the replacement value. – Thor Jun 1 '17 at 17:47

Yes. I suppose I left it named pattern so it was clear it wasn't just string substitution and you could use regex matching groups like \$1 \$2 etc. – DeezCashews Jun 1 '17 at 17:50



By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).





This isn't a direct answer but may provide a more efficient alternative for your consideration. Which is that Google Sheets has several built in [Regex Functions](#) these can be very convenient and help circumvent some of the technical procedures in Excel. Obviously there are some advantages to using Excel on your PC but for the large majority of users Google Sheets will offer an identical experience and may offer some benefits in portability and sharing of documents.

They offer

REGEXEXTRACT: Extracts matching substrings according to a regular expression.

REGEXREPLACE: Replaces part of a text string with a different text string using regular expressions.

SUBSTITUTE: Replaces existing text with new text in a string.

REPLACE: Replaces part of a text string with a different text string.

You can type these directly into a cell like so and will produce whatever you'd like

```
=REGEXMATCH(A2, "[0-9]+")
```

They also work quite well in combinations with other functions such as **IF** statements like so:

```
=IF(REGEXMATCH(E8, "MiB"), REGEXEXTRACT(E8, "\d*\.\d*|\d*")/1000, IF(REGEXMATCH(E8, "GiB"), REGEXEXTRACT(E8, "\d*\.\d*|\d*"), ""))
```

REGEXEXTRACT : General Usage

text	regular expression	Result	Formula
Google Doc 101	[0-9]+	101	=REGEXEXTRACT(A2, "[0-9]+")
The price today is \$826.25	[0-9]*\.[0-9]+[0-9]+	826.25	=REGEXEXTRACT(A8, "[0-9]*\.[0-9]+[0-9]+")
(Content) between brackets	\([A-Za-z]+\)	Content	=REGEXEXTRACT(A4, "\([A-Za-z]+\)")

Hopefully this provides a simple workaround for users who feel taunted by the VBS component of Excel.

answered Nov 26 '19 at 20:20



Alex Roseland

89 1 8

Thanks for sharing Alex. This is useful for those looking for Google version. You might consider writing & answering another question specific to Google Sheets Regex as it has it's own nomenclature and would be very useful to others. Regardless, you have my upvote!

– [Portland Runner](#) Nov 27 '19 at 0:08





By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).





Here is a `regex_subst()` function. Examples:

```
=regex_subst("watermellon", "[aeiou]", "")
---> wtrmlln
=regex_subst("watermellon", "[^aeiou]", "")
---> aeoo
```

Here is the simplified code (simpler for me, anyway). I couldn't figure out how to build a suitable output pattern using the above to work like my examples:

```
Function regex_subst( _
    strInput As String _
    , matchPattern As String _
    , Optional ByVal replacePattern As String = "" _
) As Variant
    Dim inputRegexObj As New VBScript_RegExp_55.RegExp

    With inputRegexObj
        .Global = True
        .MultiLine = True
        .IgnoreCase = False
        .Pattern = matchPattern
    End With

    regex_subst = inputRegexObj.Replace(strInput, replacePattern)
End Function
```

edited Jun 9 '18 at 16:09



Neuron

3,298 3 20 37

answered Jul 28 '17 at 21:19



jgreve

1,045 7 14





I don't want to have to enable a reference library as I need my scripts to be portable. The `Dim foo As New VBScript_RegExp_55.RegExp` line caused `User Defined Type Not Defined` errors, but I found a solution that worked for me.

What you'll want to do is put an example string in cell `A1` , then test your `strPattern` . Once that's working adjust then `rng` as desired.

```
Public Sub RegExSearch()  
    'https://stackoverflow.com/questions/22542834/how-to-use-regular-expressions-regex-in-microsoft-excel-both-in-cell-and-loops  
    'https://wellsr.com/vba/2018/excel/vba-regex-regular-expressions-guide/  
    'https://www.vitoshacademy.com/vba-regex-in-excel/  
    Dim regexp As Object  
    'Dim regex As New VBScript_RegExp_55.RegExp 'Caused "User Defined Type Not Defined" Error  
    Dim rng As Range, rcell As Range  
    Dim strInput As String, strPattern As String  
  
    Set regexp = CreateObject("vbscript.regexp")  
    Set rng = ActiveSheet.Range("A1:A1")  
  
    strPattern = "([a-z]{2})([0-9]{8})"  
    'Search for 2 Letters then 8 Digits Eg: XY12345678 = Matched  
  
    With regexp  
        .Global = False  
        .Multiline = False  
        .IgnoreCase = True  
        .Pattern = strPattern  
    End With  
  
    For Each rcell In rng.Cells  
  
        If strPattern <> "" Then  
            strInput = rcell.Value  
  
            If regexp.test(strInput) Then  
                MsgBox rcell & " Matched in Cell " & rcell.Address  
            Else  
                MsgBox "No Matches!"  
            End If  
        End If  
    Next  
End Sub
```

edited Oct 29 '19 at 0:44

answered Mar 22 '19 at 2:57



FreeSoftwareServers

1,261 10 24

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).





By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).





To add to the valuable content, I would like to create this reminder on why sometimes RegEx within VBA is not ideal. Not all expressions are supported, but instead may throw an `Error 5017` and may leave the author guessing (which I am a victim of myself).

Whilst we can find some [sources](#) on what **is** supported, it would be helpful to know which metacharacters etc. are **not** supported. A more in-depth explanation can be found [here](#). Mentioned in this source:

"Although "VBScript's regular expression ... version 5.5 implements quite a few essential regex features that were missing in previous versions of VBScript. ... JavaScript and VBScript implement Perl-style regular expressions. However, they lack quite a number of advanced features available in Perl and other modern regular expression flavors:"

So, **not** supported are:

- Start of String ancor `\A` , alternatively use the `^` caret to match postion before 1st char string
- End of String ancor `\Z` , alternatively use the `$` dollar sign to match postion after last ch in string
- Positive LookBehind, e.g.: `(?<=a)b` (whilst postive LookAhead **is** supported)
- Negative LookBehind, e.g.: `(?<!a)b` (whilst negative LookAhead **is** supported)
- [Atomic Grouping](#)
- [Possessive Quantifiers](#)
- Unicode e.g.: `\{uFFFF}`
- [Named Capturing Groups](#). Alternatively use [Numbered Capturing Groups](#)
- Inline modifiers, e.g.: `/i` (case sensitivity) or `/g` (global) etc. Set these through the `RegExp` object properties > `RegExp.Global = True` and `RegExp.IgnoreCase = True` if available.
- [Conditionals](#)
- [Regular Expression Comments](#). Add these with regular `'` comments in script

I'm a RegEx novice, but already hit a wall more than once using regular expressions within VBA. Usually with `LookBehind` but sometimes I even forget the modifiers. I have not experienced all these above mentioned backdrops myself but thought I would try to be extensive referring to some more in-depth information. Feel free to comment/correct/add. Big shout out to [regular-expressions.info](#) for a wealth of information.

P.S. You have mentioned regular VBA methods and functions, and I can confirm they (at least to myself) have been helpful in their own ways where RegEx would fail.

edited 18 hours ago



TylerH


17.4k 11 58 74

answered yesterday



JvdV

18.9k 3 12 27

 **Highly active question.** Earn 10 reputation in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).



By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

