How to extract a floating number from a string [duplicate]

This question already has an answer here:
Extract float/double value *4 answers*

I have a number of strings similar to `Current Level: 13.4 db.` and I would like to extract just the floating point number. I say floating and not decimal as it's sometimes whole. Can RegEx do this or is there a better way?

python regex floating-point data-extraction

edited Jan 24 at 21:55

martineau
49.4k 6 69 119

asked Jan 16 '11 at 2:11

Flowpoke
3,966 7 24 35

marked as duplicate by J.F. Sebastian python Oct 19 '14 at 5:06

This question has been asked before and already has an answer. If those answers do not fully address your question, please ask a new question.

Will it always have an integer portion? Even if it's 0? Do you need to match 0.4 or .4? – Falmarri Jan 16 '11 at 2:21

I would say yes. Input is manually entered so there is chance for inconsistency. – Flowpoke Jan 16 '11 at 2:22

8 Answers

If your float is always expressed in decimal notation something like

```
>>> import re
>>> re.findall("\d+\.\d+", "Current Level: 13.4 db.")
['13.4']
```

may suffice.

A more robust version would be:

```
>>> re.findall(r"[-+]?\d*\.\d+|\d+", "Current Level: -13.2 db or 14.2 or 3")
['-13.2', '14.2', '3']
```

If you want to validate user input, you could alternatively also check for a float by stepping to it directly:

```
user_input = "Current Level: 1e100 db"
for token in user_input.split():
    try:
        # if this succeeds, you have your (first) float
        print float(token), "is a float"
    except ValueError:
        print token, "is something else"

# => Would print ...
#
# Current is something else
# Level: is something else
# 1e+100 is a float
# db is something else
```

edited Dec 6 '15 at 1:51

answered Jan 16 '11 at 2:16

miku
107k 23 213 254

it is not always a decimal number. – Flowpoke Jan 16 '11 at 2:17

2 `re.findall(r"[-+]?\d*\.*\d+", "Current Level: -13.2 db or 14.2 or 3") ['-13.2', '14.2', '3']` – JuanPablo Aug 25 '14 at 22:27

1 I think you meant "\d+\.\d+" instead of "\d+.\d+" in your first code block. Right now it would extract

something like '13a4'. – abw333 Dec 6 '15 at 1:00

@abw333, spot on, thanks for taking a close look. – miku Dec 6 '15 at 1:52

how would you handle something like "level:12,25;" ? – maazza Jan 14 '16 at 15:39

|

You may like to try something like this which covers all the bases, including not relying on whitespace after the number:

```
>>> import re
>>> numeric_const_pattern = r"""
...     [-+]? # optional sign
...     (?:
...         (?: \d* \. \d+ ) # .1 .12 .123 etc 9.1 etc 98.1 etc
...         |
...         (?: \d+ \.? ) # 1. 12. 123. etc 1 12 123 etc
...     )
...     # followed by optional exponent part if desired
...     (?: [Ee] [+-]? \d+ ) ?
...     """
>>> rx = re.compile(numeric_const_pattern, re.VERBOSE)
>>> rx.findall(".1 .12 9.1 98.1 1. 12. 1 12")
['.1', '.12', '9.1', '98.1', '1.', '12.', '1', '12']
>>> rx.findall("-1 +1 2e9 +2E+09 -2e-9")
['-1', '+1', '2e9', '+2E+09', '-2e-9']
>>> rx.findall("current level: -2.03e+99db")
['-2.03e+99']
>>>
```

For easy copy-pasting:

```
numeric_const_pattern = '[-+]? (?: (?: \d* \. \d+ ) | (?: \d+ \.? ) )(?: [Ee] [+-]? \d+ )
?'
rx = re.compile(numeric_const_pattern, re.VERBOSE)
rx.findall("Some example: Jr. it. was .23 between 2.3 and 42.31 seconds")
```

edited Jun 22 at 17:40
Joop
1,252 13 33

answered Jan 16 '11 at 2:46
John Machin
56.6k 5 79 143

1 Very good! Finally I've found a really good pattern! – S.H Feb 13 '15 at 9:31

Python docs has an answer that covers +/-, and exponent notation
http://docs.python.org/2/library/re.html#simulating-scanf

```
scanf() Token      Regular Expression
%e, %E, %f, %g     [-+]?(\d+(\.\d*)?|\.\d+)([eE][-+]?\d+)?
%i                 [-+]?(0[xX][\dA-Fa-f]+|0[0-7]*|\d+)
```

This regular expression does not support international formats where a comma is used as the separator character between the whole and fractional part (3,14159). In that case, replace all `\.` with `[.,]` in the above float regex.

```
                   Regular Expression
International float [-+]?(\d+([.,]\d*)?|[.,]\d+)([eE][-+]?\d+)?
```

edited Aug 15 '16 at 2:50

answered Aug 14 '13 at 17:07
IceArdor
1,217 10 17

```
re.findall(r"[-+]?\d*\.\d+|\d+", "Current Level: -13.2 db or 14.2 or 3")
```

as described above, works really well! One suggestion though:

```
re.findall(r"[-+]?\d*\.\d+|[-+]?\d+", "Current Level: -13.2 db or 14.2 or 3 or -3")
```

will also return negative int values (like -3 in the end of this string)

answered Nov 25 '11 at 10:24
Martin
51 1 1

I think that you'll find interesting stuff in the following answer of mine that I did for a previous similar question:

https://stackoverflow.com/q/5929469/551449

In this answer, I proposed a pattern that allows a regex to catch any kind of number and since I have nothing else to add to it, I think it is fairly complete

edited May 23 at 12:10 Community ♦ 1 1

answered Nov 25 '11 at 11:08 eyquem 16.6k 4 23 35

You can use the following regex to get integer and floating values from a string:

```
re.findall(r'[\d\.\d]+', 'hello -34 42 +34.478m 88 cricket -44.3')
```

```
['34', '42', '34.478', '88', '44.3']
```

Thanks Rex

edited May 21 '14 at 15:37 Steven Westbrook 1,414 2 14 21

answered May 21 '14 at 15:14 user3613331 21 3

2 This regex will also find non-numeric combinations of periods and digits: `'.... 1.2.3.4 ..56..'` yields: `['....', '1.2.3.4', '..56..']` – scottbb Nov 6 '15 at 15:40

Another approach that may be more readable is simple type conversion. I've added a replacement function to cover instances where people may enter European decimals:

```
>>> for possibility in "Current Level: -13.2 db or 14,2 or 3".split():
...     try:
...         str(float(possibility.replace(',', '.')))
...     except ValueError:
...         pass
'-13.2'
'14.2'
'3.0'
```

This has disadvantages too however. If someone types in "1,000", this will be converted to 1. Also, it assumes that people will be inputting with whitespace between words. This is not the case with other languages, such as Chinese.

answered Jan 16 '11 at 2:40 Tim McNamara 11.7k 2 29 68

"4x size AAA 1.5V batteries included" :-) – John Machin Jan 16 '11 at 2:48

Those terrible users! Always entering in silly data. TBH, I've intentionally kept this example demonstrative rather than robust. When I begun writing this response, @The MYYN only provided regular expressions in the accepted answer. I wanted to provide an example of another way to go about things. – Tim McNamara Jan 16 '11 at 2:51

Something quick and dirty that should work

```
[0-9]*\.?[0-9]*
```

Edit:

This actually doesn't work =\

deleted by Bill the Lizard Feb 5 '15 at 14:57 edited Jan 16 '11 at 2:20

answered Jan 16 '11 at 2:15 Falmarri 29.3k 29 104 166