

RegEx (Regular Expression)

Arasalan Shaikh

Linkedin : <https://www.linkedin.com/in/arosalanshaikh/>

Credits: W3School for questions

```
In [1]: import re

In [3]: test_string = "123abc456789abc123ABC"

In [4]: # finditer() method gives match object
pattern = re.compile(r"abc")
matches = pattern.finditer(test_string)

for match in matches:
    print(match)

<re.Match object; span=(3, 6), match='abc'>
<re.Match object; span=(12, 15), match='abc'>

In [5]: # or can also be done in following way

matches = re.finditer(r"abc", test_string)
for match in matches:
    print(match)

<re.Match object; span=(3, 6), match='abc'>
<re.Match object; span=(12, 15), match='abc'>

In [6]: #.findall() method gives all matching strings

matches = re.findall(r"abc", test_string)
for match in matches:
    print(match)

abc
abc

In [8]: # match() method gives first match at the begining of string

match = re.match(r"abc", test_string)
print(match)

None

In [9]: match = re.match(r"123", test_string)
print(match)

<re.Match object; span=(0, 3), match='123'>
```

Method on a Match object

1. group() : Returns the string matched by the RE
2. start() : Returns the starting position of the match
3. end() : Return the ending position of the match
4. span() : Return a tuple containing the (start, end) positions of the match

```
In [10]: # moving forward with finditer() method
pattern = re.compile(r"abc")
matches = pattern.finditer(test_string)

for match in matches:
    print(match.span(), match.start(), match.end())

# group, start, end, span method

(3, 6) 3 6
(12, 15) 12 15

In [11]: pattern = re.compile(r"abc")
matches = pattern.finditer(test_string)

for match in matches:
    print(match.group())

abc
abc
```

All meta characters . ^

*+? | () Note : Metacharacters need to be escaped (with \) if we actually want to search for the char.

1.. any character (except newline character)

2.* starts with "ello" 3

endswith "world\$" 4. * zero or more occurrences 5. + one or more occurrences 6. {} exactly specified no. of occurrences "al{2}" 7. [] A set of characters "[a-m]" 8. \ Signals a special sequence (can also be used to escape special characters) "\d" 9. | Either or "falls|stays" 10. () Capture and group More Metacharacters / Special Sequences A special sequence is a \ followed by one of the characters in the list below and has a special meaning: 1. \d : Matches any decimal digit; this is equivalent to the class [0-9] 2. \D : Matches any non-digit character, this is equivalent to the class [^0-9] 3. \s : Matches any whitespace character 4. \S : Matches any non-whitespace character 5. \w : Matches any alphanumeric (word) character; this is equivalent to the class [a-zA-Z0-9_] 6. \W : Matches any non-alphanumeric character; this is equivalent to the class [^a-zA-Z0-9_] 7. \b : Return a match where the specified character are at the beginning or at the end of a word r"\bain"\r'ain\b" 8. \B : Return a match where the specified characters are present, but NOT at the beginning (or at the end) of a word r"\Bain"\r'ain\b" 9. \A : Return a match if the specified characters are at the beginning of the string "\AThe" 10. \Z : Return a match if the specified characters are at the end of the string "Spain\Z"

```
In [14]: pattern = re.compile(r".")
matches = pattern.finditer(test_string)

for match in matches:
    print(match.group(), end = " ")

1 2 3 a b c 4 5 6 7 8 9 a b c 1 2 3 A B C
```

```
In [15]: test_string1 = "123abc456789abc123ABC."

pattern = re.compile(r"\.")
matches = pattern.finditer(test_string1)

for match in matches:
    print(match)

<re.Match object; span=(21, 22), match='.'>
```

```
In [17]: test_string = "123abc456789abc123ABC"
pattern = re.compile(r"^123")
matches = pattern.finditer(test_string)

for match in matches:
    print(match)

<re.Match object; span=(0, 3), match='123'>
```

```
In [19]: pattern = re.compile(r"^\abc")
matches = pattern.finditer(test_string)

for match in matches:
    print(match)
# No Match
```

```
In [20]: pattern = re.compile(r"abc$")
matches = pattern.finditer(test_string)

for match in matches:
    print(match)

# No Match
```

```
In [21]: pattern = re.compile(r"ABC$")
matches = pattern.finditer(test_string)

for match in matches:
    print(match)

<re.Match object; span=(18, 21), match='ABC'>
```

```
In [22]: test_string2 = "hello 123_ heyho hohey"
```

```
In [24]: pattern = re.compile(r"\d")
matches = pattern.finditer(test_string2)

for match in matches:
    print(match)

<re.Match object; span=(6, 7), match='1'>
<re.Match object; span=(7, 8), match='2'>
<re.Match object; span=(8, 9), match='3'>
```

```
In [25]: pattern = re.compile(r"\D")
matches = pattern.finditer(test_string2)

for match in matches:
    print(match)
```

```
<re.Match object; span=(0, 1), match='h'>
<re.Match object; span=(1, 2), match='e'>
<re.Match object; span=(2, 3), match='l'>
<re.Match object; span=(3, 4), match='l'>
<re.Match object; span=(4, 5), match='o'>
<re.Match object; span=(5, 6), match=' '>
<re.Match object; span=(9, 10), match='_ '>
<re.Match object; span=(10, 11), match='-'>
<re.Match object; span=(11, 12), match='h'>
<re.Match object; span=(12, 13), match='e'>
<re.Match object; span=(13, 14), match='y'>
<re.Match object; span=(14, 15), match='h'>
<re.Match object; span=(15, 16), match='o'>
<re.Match object; span=(16, 17), match=' '>
<re.Match object; span=(17, 18), match='h'>
<re.Match object; span=(18, 19), match='o'>
<re.Match object; span=(19, 20), match='h'>
<re.Match object; span=(20, 21), match='e'>
<re.Match object; span=(21, 22), match='y'>
```

```
In [26]: pattern = re.compile(r"\s")
matches = pattern.finditer(test_string2)

for match in matches:
    print(match)

<re.Match object; span=(5, 6), match=' '>
<re.Match object; span=(10, 11), match=' '>
<re.Match object; span=(16, 17), match=' '>
```

```
In [28]: pattern = re.compile(r"\S")
matches = pattern.finditer(test_string2)

for match in matches:
    print(match)

<re.Match object; span=(0, 1), match='h'>
<re.Match object; span=(1, 2), match='e'>
<re.Match object; span=(2, 3), match='l'>
<re.Match object; span=(3, 4), match='l'>
<re.Match object; span=(4, 5), match='o'>
<re.Match object; span=(6, 7), match='1'>
<re.Match object; span=(7, 8), match='2'>
<re.Match object; span=(8, 9), match='3'>
<re.Match object; span=(9, 10), match='_ '>
<re.Match object; span=(11, 12), match='h'>
<re.Match object; span=(12, 13), match='e'>
<re.Match object; span=(13, 14), match='y'>
<re.Match object; span=(14, 15), match='h'>
<re.Match object; span=(15, 16), match='o'>
<re.Match object; span=(17, 18), match='h'>
<re.Match object; span=(18, 19), match='o'>
<re.Match object; span=(19, 20), match='h'>
<re.Match object; span=(20, 21), match='e'>
<re.Match object; span=(21, 22), match='y'>
```

```
In [29]: pattern = re.compile(r"\w")
matches = pattern.finditer(test_string2)

for match in matches:
    print(match)

<re.Match object; span=(0, 1), match='h'>
<re.Match object; span=(1, 2), match='e'>
<re.Match object; span=(2, 3), match='l'>
<re.Match object; span=(3, 4), match='l'>
<re.Match object; span=(4, 5), match='o'>
<re.Match object; span=(6, 7), match='1'>
<re.Match object; span=(7, 8), match='2'>
<re.Match object; span=(8, 9), match='3'>
<re.Match object; span=(9, 10), match='_ '>
<re.Match object; span=(11, 12), match='h'>
<re.Match object; span=(12, 13), match='e'>
<re.Match object; span=(13, 14), match='y'>
<re.Match object; span=(14, 15), match='h'>
<re.Match object; span=(15, 16), match='o'>
<re.Match object; span=(17, 18), match='h'>
<re.Match object; span=(18, 19), match='o'>
<re.Match object; span=(19, 20), match='h'>
<re.Match object; span=(20, 21), match='e'>
<re.Match object; span=(21, 22), match='y'>
```

```
In [30]: pattern = re.compile(r"\W")
matches = pattern.finditer(test_string2)

for match in matches:
    print(match)

<re.Match object; span=(5, 6), match=' '>
<re.Match object; span=(10, 11), match=' '>
<re.Match object; span=(16, 17), match=' '>
```

```
In [31]: pattern = re.compile(r"\bhello")
matches = pattern.finditer(test_string2)

for match in matches:
    print(match)

<re.Match object; span=(0, 5), match='hello'>
```

```
In [32]: pattern = re.compile(r"\bhey")
matches = pattern.finditer(test_string2)

for match in matches:
    print(match)

<re.Match object; span=(11, 14), match='hey'>
```

```
In [33]: pattern = re.compile(r"\Bhey")
matches = pattern.finditer(test_string2)

for match in matches:
    print(match)

<re.Match object; span=(19, 22), match='hey'>
```

Sets

1. A set is a set of characters inside a pair of square brackets [] with a special meaning. Append multiple conditions back-to-back, e.g. [aA-Z]. 2. A ^ (caret) inside a set negates the expression. 3. A - (dash) in a set specifies a range if it is in between, otherwise the dash itself. Examples: 1. [arn] Returns a match where one of the specified characters (a, r, or n) are present 2. [a-n] Returns a match for any lower case character, alphabetically between a and n 3. [^arn] Returns a match for any character EXCEPT a, r, and n 4. [0123] Returns a match where any of the specified digits (0, 1, 2, or 3) are present 5. [0-9] Returns a match for any digit between 0 and 9 6. 0-5 Returns a match for any two-digit numbers from 00 and 59 7. [a-zA-Z] Returns a match for any character alphabetically between a and z, lower case OR upper case

```
In [34]: test_string3 = "hello 123_"
```

```
In [35]: pattern = re.compile(r"[lo]")
matches = pattern.finditer(test_string3)

for match in matches:
    print(match)

<re.Match object; span=(2, 3), match='l'>
<re.Match object; span=(3, 4), match='l'>
<re.Match object; span=(4, 5), match='o'>
```

```
In [36]: pattern = re.compile(r"[a-z]")
matches = pattern.finditer(test_string3)

for match in matches:
    print(match)

<re.Match object; span=(0, 1), match='h'>
<re.Match object; span=(1, 2), match='e'>
<re.Match object; span=(2, 3), match='l'>
<re.Match object; span=(3, 4), match='l'>
<re.Match object; span=(4, 5), match='o'>
```

```
In [37]: pattern = re.compile(r"[23]")
matches = pattern.finditer(test_string3)

for match in matches:
    print(match)

<re.Match object; span=(7, 8), match='2'>
<re.Match object; span=(8, 9), match='3'>
```

```
In [38]: pattern = re.compile(r"[0-p]")
matches = pattern.finditer(test_string3)

for match in matches:
    print(match)

<re.Match object; span=(0, 1), match='h'>
<re.Match object; span=(1, 2), match='e'>
<re.Match object; span=(2, 3), match='l'>
<re.Match object; span=(3, 4), match='l'>
<re.Match object; span=(4, 5), match='o'>
<re.Match object; span=(6, 7), match='1'>
<re.Match object; span=(7, 8), match='2'>
<re.Match object; span=(8, 9), match='3'>
<re.Match object; span=(9, 10), match='_'>
```

```
In [39]: pattern = re.compile(r"[0-9-]")
matches = pattern.finditer("hello 123-")

for match in matches:
    print(match)
```

```
<re.Match object; span=(6, 7), match='1'>
<re.Match object; span=(7, 8), match='2'>
<re.Match object; span=(8, 9), match='3'>
<re.Match object; span=(9, 10), match='-'>
```

```
In [40]: pattern = re.compile(r"[a-zA-Z0-9]")
matches = pattern.finditer("helloHELLO 123_")

for match in matches:
    print(match)

<re.Match object; span=(0, 1), match='h'>
<re.Match object; span=(1, 2), match='e'>
<re.Match object; span=(2, 3), match='l'>
<re.Match object; span=(3, 4), match='l'>
<re.Match object; span=(4, 5), match='o'>
<re.Match object; span=(5, 6), match='H'>
<re.Match object; span=(6, 7), match='E'>
<re.Match object; span=(7, 8), match='L'>
<re.Match object; span=(8, 9), match='L'>
<re.Match object; span=(9, 10), match='O'>
<re.Match object; span=(11, 12), match='1'>
<re.Match object; span=(12, 13), match='2'>
<re.Match object; span=(13, 14), match='3'>
```

Quantifier

1. * : 0 or more
2. + : 1 or more
3. ?: 0 or 1, used when a character can be optional
4. {4} : exact number
5. {4, 6} : range numbers (min, max)

```
In [43]: test_string4 = "hello_123"
```

```
In [45]: pattern = re.compile(r"\d*")
matches = pattern.finditer(test_string4)

for match in matches:
    print(match)

<re.Match object; span=(0, 0), match=' '>
<re.Match object; span=(1, 1), match=' '>
<re.Match object; span=(2, 2), match=' '>
<re.Match object; span=(3, 3), match=' '>
<re.Match object; span=(4, 4), match=' '>
<re.Match object; span=(5, 5), match=' '>
<re.Match object; span=(6, 9), match='123'>
<re.Match object; span=(9, 9), match=' '>
```

```
In [46]: pattern = re.compile(r"\d")
matches = pattern.finditer(test_string4)

for match in matches:
    print(match)

<re.Match object; span=(6, 7), match='1'>
<re.Match object; span=(7, 8), match='2'>
<re.Match object; span=(8, 9), match='3'>
```

```
In [55]: pattern = re.compile(r"\d+")
matches = pattern.finditer(test_string4)

for match in matches:
    print(match)

<re.Match object; span=(6, 9), match='123'>
```

```
In [56]: pattern = re.compile(r"_\d+")
matches = pattern.finditer(test_string4)

for match in matches:
    print(match)

<re.Match object; span=(5, 9), match='_123'>
```

```
In [57]: pattern = re.compile(r"_\d")
matches = pattern.finditer("hello123")

for match in matches:
    print(match)
```

```
In [58]: pattern = re.compile(r"?_\d")
matches = pattern.finditer("hello123")

for match in matches:
    print(match)

<re.Match object; span=(5, 6), match='1'>
<re.Match object; span=(6, 7), match='2'>
<re.Match object; span=(7, 8), match='3'>
```

```
In [59]: pattern = re.compile(r"\d{3}")
matches = pattern.finditer("hello123")
```

```

for match in matches:
    print(match)

<re.Match object; span=(5, 8), match='123'>

```

```

In [60]: pattern = re.compile(r"\d{2}")
matches = pattern.finditer("hello123")

for match in matches:
    print(match)

<re.Match object; span=(5, 7), match='12'>

```

```

In [61]: pattern = re.compile(r"\d{4}")
matches = pattern.finditer("hello123")

for match in matches:
    print(match)

```

```

In [62]: pattern = re.compile(r"\d{1, 3}")
matches = pattern.finditer("hello123")

for match in matches:
    print(match)

```

```
In [63]: # task 1
```

```

dates = """
hello
11.04.2022

2022.04.21

2022-04-30
2022-05-23
2022-06-12
2022-07-15
2022-08-19

2022/04/22

2022_04_04
"""

```

```
In [64]: #1. find all date in this 2020-04-01 format
```

```

pattern = re.compile(r"\d.-\d.-\d.")
matches = pattern.finditer(dates)

for match in matches:
    print(match)

<re.Match object; span=(33, 41), match='22-04-30'>
<re.Match object; span=(44, 52), match='22-05-23'>
<re.Match object; span=(55, 63), match='22-06-12'>
<re.Match object; span=(66, 74), match='22-07-15'>
<re.Match object; span=(77, 85), match='22-08-19'>

```

```
In [65]: pattern = re.compile(r"\d\d\d\d-\d\d\d\d")
matches = pattern.finditer(dates)
```

```

for match in matches:
    print(match)

<re.Match object; span=(31, 41), match='2022-04-30'>
<re.Match object; span=(42, 52), match='2022-05-23'>
<re.Match object; span=(53, 63), match='2022-06-12'>
<re.Match object; span=(64, 74), match='2022-07-15'>
<re.Match object; span=(75, 85), match='2022-08-19'>

```

```
In [66]: #2. find all date in this 2020-04-01 and 2020/04/02 format
```

```

pattern = re.compile(r"\d\d\d\d[-/]\d\d[-/]\d\d")
matches = pattern.finditer(dates)

for match in matches:
    print(match)

<re.Match object; span=(31, 41), match='2022-04-30'>
<re.Match object; span=(42, 52), match='2022-05-23'>
<re.Match object; span=(53, 63), match='2022-06-12'>
<re.Match object; span=(64, 74), match='2022-07-15'>
<re.Match object; span=(75, 85), match='2022-08-19'>
<re.Match object; span=(87, 97), match='2022/04/22'>

```

```
In [67]: #3. find all date in may, june, july
```

```
pattern = re.compile(r"\d\d\d\d[-/]0[5-7][-/]\d{2}")
matches = pattern.finditer(dates)
```

```

for match in matches:
    print(match)

<re.Match object; span=(42, 52), match='2022-05-23'>
<re.Match object; span=(53, 63), match='2022-06-12'>
<re.Match object; span=(64, 74), match='2022-07-15'>

```

Conditions

Use the | for either or condition

```

In [73]: my_str = """
hello world
1223
2022-05-20
Mr Curry
Mrs Curry
Mr. Thompson
Mrs Green
Mr. T
"""

# Return all names

```

```

In [69]: pattern = re.compile(r"Mr\s\w+")
matches = pattern.finditer(my_str)

for match in matches:
    print(match)

<re.Match object; span=(29, 37), match='Mr Curry'>

```

```

In [70]: pattern = re.compile(r"Mr\.\s\w+") # '\.' is used to treat . as dot
matches = pattern.finditer(my_str)

for match in matches:
    print(match)

<re.Match object; span=(48, 60), match='Mr. Thompson'>
<re.Match object; span=(71, 76), match='Mr. T'>

```

```

In [71]: pattern = re.compile(r"Mr\.\?\s\w+")
matches = pattern.finditer(my_str)

for match in matches:
    print(match)

<re.Match object; span=(29, 37), match='Mr Curry'>
<re.Match object; span=(48, 60), match='Mr. Thompson'>
<re.Match object; span=(71, 76), match='Mr. T'>

```

```

In [72]: pattern = re.compile(r"(Mr|Ms|Mrs)\.\?\s\w+")
matches = pattern.finditer(my_str)

for match in matches:
    print(match)

<re.Match object; span=(29, 37), match='Mr Curry'>
<re.Match object; span=(38, 47), match='Mrs Curry'>
<re.Match object; span=(48, 60), match='Mr. Thompson'>
<re.Match object; span=(61, 70), match='Mrs Green'>
<re.Match object; span=(71, 76), match='Mr. T'>

```

Grouping

() is used to group substrings in the matches.

```

In [74]: emails = """
hello world
1223
2022-05-20
Mr Curry
Mrs Curry
Mr. Thompson
Mrs Green
Mr. T
stephencurry@gmail.com
Stephen-curry@gmx.de
stephen-curry123@my-domain.org
"""

# return all emails

```

```

In [78]: pattern = re.compile(r"[a-zA-Z]-\d-]+@[a-zA-Z-]+\.(com|de|org)")
matches = pattern.finditer(emails)

```

```

for match in matches:
    print(match)

<re.Match object; span=(77, 99), match='stephencurry@gmail.com'>
<re.Match object; span=(100, 120), match='Stephen-curry@gmx.de'>
<re.Match object; span=(121, 151), match='stephen-curry123@my-domain.org'>

In [79]: pattern = re.compile(r"[a-zA-Z]-[0-9-]+@[a-zA-Z-]+\.[a-zA-Z]*")
matches = pattern.findall(emails)

for match in matches:
    print(match)

<re.Match object; span=(77, 99), match='stephencurry@gmail.com'>
<re.Match object; span=(100, 120), match='Stephen-curry@gmx.de'>
<re.Match object; span=(121, 151), match='stephen-curry123@my-domain.org'>

In [81]: pattern = re.compile(r"([a-zA-Z0-9-]+)@([a-zA-Z-]+)\.([a-zA-Z]+)")
matches = pattern.findall(emails)

for match in matches:
    print("all groups : ", match.group(0)) # this means all groups
    print("first group : ", match.group(1)) # this means first group
    print("second group : ", match.group(2)) # this means second group
    print("third group : ", match.group(3)) # this means third group

all groups : stephencurry@gmail.com
first group : stephencurry
second group : gmail
third group : com
all groups : Stephen-curry@gmx.de
first group : Stephen-curry
second group : gmx
third group : de
all groups : stephen-curry123@my-domain.org
first group : stephen-curry123
second group : my-domain
third group : org

```

Modifying String

1. `split()` : Split the string into a list, splitting it wherever the RE matches
2. `sub()` : Find all substrings where the RE matches, and replace them with a different string

```

In [83]: my_string = "123abc456789abc123ABC"

In [84]: pattern = re.compile(r"abc")
splitted = pattern.split(my_string)
print(splitted)

['123', '456789', '123ABC']

In [85]: test_str = "hello world, you are the best world"

In [86]: pattern = re.compile(r"world")
subbed_string = pattern.sub("planet", test_str)
print(subbed_string)

hello planet, you are the best planet

In [87]: # Task 2
urls = """
http://zara-fashion.com
https://www.world-healthorganisation.org
https://www.iNeuron.ai
"""

# return url

In [88]: pattern = re.compile(r"(https?://)(www\.)?([a-zA-Z-]+)\.[a-zA-Z]+")
matches = pattern.findall(urls)

for match in matches:
    print(match)

<re.Match object; span=(1, 24), match='http://zara-fashion.com'>
<re.Match object; span=(25, 65), match='https://www.world-healthorganisation.org'>
<re.Match object; span=(66, 88), match='https://www.iNeuron.ai'>

In [89]: pattern = re.compile(r"(https?://)(www\.)?([a-zA-Z-]+)\.([a-zA-Z]+)")
matches = pattern.findall(urls)

for match in matches:
    print(match)

<re.Match object; span=(1, 24), match='http://zara-fashion.com'>
<re.Match object; span=(25, 65), match='https://www.world-healthorganisation.org'>
<re.Match object; span=(66, 88), match='https://www.iNeuron.ai'>

```

```
In [90]: pattern = re.compile(r"https://(www\.)?([a-zA-Z]+)\.([a-zA-Z]+)")
matches = pattern.finditer(urls)
for match in matches:
    print(match.group(2))
```

zara-fashion
world-healthorganisation
iNeuron

```
In [91]: subbed_url = pattern.sub("hello", urls)
print(subbed_url)
```

hello
hello
hello

```
In [92]: subbed_url = pattern.sub(r"\2\3", urls) #\2 and \3 are used for specifying groups
print(subbed_url)
```

zara-fashioncom
world-healthorganisationorg
iNeuronai

Compilation Flags

1. ASCII, A : Makes several escapes like \w, \b, \s and \d match only on ASCII characters with the respective property.
2. DOTALL, S : Make . match any character, including newlines.
3. IGNORECASE, I : Do case-insensitive matches.
4. LOCALE, L : Do a locale-aware match.
5. MULTILINE, M : Multi-line matching, affecting ^ and \$.
6. VERBOSE, X (for 'extended') : Enable verbose REs, which can be organized more cleanly and understandably.

```
In [93]: my_string = "Hello World"

pattern = re.compile(r"world")
matches = pattern.finditer(my_string)

for match in matches:
    print(match)

# no match # case sensitive
```

```
In [94]: my_string = "Hello World"

pattern = re.compile(r"world", re.IGNORECASE) #in place of re.IGNORECASE we can use re.I
matches = pattern.finditer(my_string)

for match in matches:
    print(match)

<re.Match object; span=(6, 11), match='World'>
```