RegEx (Regular Expression)

Shubham Verma

Linkedin https://www.linkedin.com/in/shubham-verma-3968a5119

In [1]:
```
import re
```

In [44]:
```
test_string = '123abc456789abc123ABC'
```

In [45]:
```
# 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 [46]:
```
# 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 [47]:
```
# findall() method gives all matching strings

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

```
abc
abc
```

In [48]:
```
# match() method gives first match at the beginning of string.

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

```
None
```

In [49]:
```
match = re.match(r"123", test_string)
print(match)
```

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

In [50]:
```
# search() method gives first matching string at any location

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

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

Methods on a Match object

1. group(): Return the string matched by the RE
2. start(): Return 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 [51]:
```
# 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 [52]:
```
pattern = re.compile(r'abc')
matches = pattern.finditer(test_string)

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

```
abc
abc
```

All meta characters `. ^ $ * + ? { } \ | ( )`

Note: Meta characters need to be escaped (with) if we actually want to search for the char.

1. `.` any character (except new line character)
2. `^` startswith "^hello"
3. `$` endswith "world$"
4. `*` zero or more occurances
5. `+` one or more occurances
6. `{}` exactly specified no of occurances "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` Returns a match where the specified characters are at the beginning or at the end of a word r"\bain" r"ain\b"
8. `\B` Returns 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` Returns a match if the specified characters are at the beginning of the string "\AThe"
10. `\Z` Returns a match if the specified characters are at the end of the string "Spain\Z"

In [53]:
```
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 [54]:
```
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 [55]:
```
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 [56]:
```
pattern = re.compile(r"^abc")
matches = pattern.finditer(test_string)
```

```
for match in matches:
    print(match)
# no match
```

In [57]:
```
pattern = re.compile(r"abc$")
matches = pattern.finditer(test_string)

for match in matches:
    print(match)

# no match
```

In [58]:
```
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 [59]:
```
test_string2 = 'hello 123_ heyho hohey'
```

In [60]:
```
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 [61]:
```
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 [62]:
```
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 [63]:
```
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 [64]:
```
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 [65]:
```
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 [66]:
```
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 [67]:
```
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 [68]:
```
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 [69]:
```
test_string3 = 'hello 123_'
```

In [70]:
```
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 [71]:
```
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 [72]:
```
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 [73]:
```
pattern = re.compile(r"[0-9]")
matches = pattern.finditer(test_string3)

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 [74]:
```
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 [75]:
```
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 [76]:
```
test_string4 = 'hello_123'
```

In [77]:
```
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 [78]:
```
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 [79]:
```
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 [80]:
```
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 [81]:
```
pattern = re.compile(r"_\d")
matches = pattern.finditer('hello123')

for match in matches:
    print(match)

# no match
```

In [82]:
```
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 [83]:
```
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 [84]:
```
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 [85]:
```
pattern = re.compile(r"\d{4}")
matches = pattern.finditer('hello123')

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

In [86]:
```
pattern = re.compile(r"\d{1,3}")
matches = pattern.finditer('hello123')

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

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

In [87]:
```
# 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 [88]:
```
#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 [89]:
```
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 [90]:
```
#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 [91]:
```
#3. find all date in may, june, july
pattern = re.compile(r"\d\d\d\d[-/]0[5-7][-/]\d\d")
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'>
```

In [92]:
```
pattern = re.compile(r"\d{4}[-/]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 [2]:
```
my_str = """
hello world
1223
2022-05-20
Mr Curry
Mrs Curry
Mr. Thompson
Mrs Green
Mr. T
"""
# return all names
```

In [3]:
```
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 [4]:
```
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 [5]:
```
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 [6]:
```
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 [7]:
```
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 [8]:
```
pattern = re.compile(r"[a-zA-z0-9-]+@[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 [9]:
```
pattern = re.compile(r"[a-zA-z0-9-]+@[a-zA-Z-]+\.[a-zA-Z]*")
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 [10]:
```
pattern = re.compile(r"([a-zA-z0-9-]+)@([a-zA-Z-]+)\.([a-zA-Z]+)")
matches = pattern.finditer(emails)

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

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

Modifying strings

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 [11]:
```
my_string = '123abc456789abc123ABC'
```

In [12]:
```
pattern = re.compile(r"abc")
splitted = pattern.split(my_string)
print(splitted)
```

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

In [13]:
```
test_str = 'hello world, you are the best world'
```

In [14]:
```
pattern = re.compile(r"world")
subbed_string = pattern.sub("planet", test_str)
print(subbed_string)
```

```
hello planet, you are the best planet
```

In [16]:
```
# Task 2
urls = """
http://zara-fashion.com
https://www.world-healthorganisation.org
http://www.iNeuron.ai
"""

# return url
```

In [17]:
```
pattern = re.compile(r"(https?://)(www\.)?([a-zA-Z-]+)\.[a-zA-Z]+")
matches = pattern.finditer(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, 87), match='http://www.iNeuron.ai'>
```

In [18]:
```
pattern = re.compile(r"(https?://)(www\.)?([a-zA-Z-]+)\.([a-zA-Z]+)")
matches = pattern.finditer(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, 87), match='http://www.iNeuron.ai'>
```

In [19]:
```
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 [20]:
```
subbed_url = pattern.sub("hello", urls)
print(subbed_url)
```

```
hello
hello
hello
```

In [21]:
```
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 [22]:
```
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 [23]:

```
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'>
```