Regular Expression

Basic Regex

Character types

- `\w` matches with any alphanumeric character, including underline
- `.` matches to all characters, including symbols (Wildcard)
- `\d` matches to all single digits, ? `[0-9]`
- `\D` ???????? `[^0-9]`
- `\s` matches to all single space, tab and new line
- `\.` matches to the dot(period) character
- `[a-z]` ?????? a-z
- `[A-Z]` ?????? A-Z
- `[^a-z]` ??????? a-z
- `[0-9]` ?? 0-9
- `[^0-9]` ??????????
- `|` ?????????
- `p?each` ?? each ???? 0 ?? 1 ? p ???*each* ? *peach*

```
import re
re.findall("\w", "h32rb17")


import re
re.findall("\d", "h32rb17")
```

????

- `^` ??
- `$` ??
- `\b` ????? (????????) ??????
- `\B` ????? (????????) ??????

Quantify occurrences

?????????

- `+` : ?? 1 ????? `{1,}`

- `*` : ?? 0 ????? `{0,}`
- `?` : ?? 0 ? 1 ???? `p?each` ?? each ???? 0 ?? 1 ? p ????? each ? peach
- `{n}` : ?? n ?
- `{n,}` : ?? n ???
- `{0,n}` : ?? 0 - n ?
- `{n,m}` : ?? n - m ?
- `\d{2}` 2 ?????
- `\d{1,3}` ?? 1 - 3 ??
- `\d+` ?????????

Functions

.findall()

`.findall(<regex>, <string>)`

- ?????????
- ???? List
- ??????? `None`

```
import re
re.findall("\d+", "h32rb17")


import re
re.findall("\d*", "h32rb17")

import re
re.findall("\d{2}", "h32rb17 k825t0m c2994eh")

import re
re.findall("\d{1,3}", "h32rb17 k825t0m c2994eh")
```

```
import re
pattern = "\w+:\s\d+"
employee_logins_string = "1001 bmoreno: 12 Marketing 1002 tshah: 7 Human Resources 1003 sgilmore: 5
Finance"
print(re.findall(pattern, employee_logins_string))
```

```
['bmoreno: 12', 'tshah: 7', 'sgilmore: 5']
```

.search()

`.search(<regex>, <string>, re.IGNORECASE)`

- r"regex" : `r` ?? raw string?Python ??????????????????
- ???????????
- ???? Match Class
- ??????? `None`

```
import re
log = "July 31 07:51:48 mycomputer bad_process[12345]: ERROR Performing package upgrade"
regex = r"\[(\d+)\]"
result = re.search(regex, log)

print(result)     # Output: <_sre.SRE_Match object; span=(39, 46), match='[12345]'>
print(result[1])  # Output: 12345
```

```
import re
print(re.search(r"[Pp]ython", "Python"))

# Output: <_sre.SRE_Match object; span=(0, 6), match='Python'>
```

```
import re
print(re.search(r"Py.*n", "Pygmalion"))
print(re.search(r"Py.*n", "Python Programming"))
print(re.search(r"Py[a-z]*n", "Python Programming"))
print(re.search(r"Py[a-z]*n", "Pyn"))

# Output:
# <_sre.SRE_Match object; span=(0, 9), match='Pygmalion'>
# <_sre.SRE_Match object; span=(0, 17), match='Python Programmin'>
# <_sre.SRE_Match object; span=(0, 6), match='Python'>
# <_sre.SRE_Match object; span=(0, 3), match='Pyn'>
```

```
import re
print(re.search(r"o+l+", "goldfish"))
print(re.search(r"o+l+", "woolly"))
print(re.search(r"o+l+", "boil"))

# Output:
# <_sre.SRE_Match object; span=(1, 3), match='ol'>
# <_sre.SRE_Match object; span=(1, 5), match='ooll'>
```

```
# None
```

.split()

- ???? Regex ?????
- `.split(<regex>, <string>)` : ?? List ????
- `r"[.?!]"` ??????????????????????????????

```
import re
re.split(r"[.?!]", "One sentence. Another one? And the last one!")

# Output: ['One sentence', ' Another one', ' And the last one', '']
```

- `r"the|a"` : ???????????????

```
re.split(r"the|a", "One sentence. Another one? And the last one!")

# Output: ['One sentence. Ano', 'r one? And ', ' l', 'st one!']
```

- `r"([.?!])"` ??????????????????????????????????

```
import re
re.split(r"([.?!])", "One sentence. Another one? And the last one!")

# Output: ['One sentence', '.', ' Another one', '?', ' And the last one', '!', '']
```

.sub()

- ??????????
- `.sub(<regex>, <new-string>, <strings>)` : ?? <regex> ????? <new-string>??

```
import re
re.sub(r"[\w.%+-]+@[\w.-]+", "[REDACTED]", "Received an email for go_nuts95@my.example.com")

# Output: Received an email for [REDACTED]
```

```
re.sub(r"([A-Z])\.\s+(\w+)", r"Ms. \2", "A. Weber and B. Bellmas have joined the team.")

# Output: Ms. Weber and Ms. Bellmas have joined the team
```

- ????????? Regex

- ?? regex : `r"^([\w .-]*), ([\w .-]*)$"` ?(??1), (??2)
- ?? regex: `r"\2 \1"` ?\2 ???\1 ??1

```
import re
re.sub(r"^([\w .-]*), ([\w .-]*)$", r"\2 \1", "Lovelace, Ada")


# Output: Ada Lovelace
```

Advanced Regex

????

> **Alteration**: RegEx that matches any one of the alternatives separated by the pipe symbol

- `r"location.*(London|Berlin|Madrid)"` : location is London, location is Berlin, or location is Madrid.

????

- `r"[0-9$-,.]"` : This will match any of the digits zero through nine, or the dollar sign, hyphen, comma, or period

????

- `r"\d{3}-\d{3}-\d{4}"` This line of code matches U.S. phone numbers in the format 111-222-3333.
- `r"^-?\d*(\.\d+)?$"` ??????????????????
- `r"^(.+)\/([^\/]+)\/"` ??????

IP addr.

```
# Assign `log_file` to a string containing username, date, login time, and IP address for a series of login attempts
log_file = "eraab 2022-05-10 6:03:41 192.168.152.148 \niuduike 2022-05-09 6:46:40 192.168.22.115 \nsmartell 2022-05-09 19:30:32 192.168.190.178 \narutley 2022-05-12 17:00:59 1923.1689.3.24 \nrjensen 2022-05-11
```

```
0:59:26 192.168.213.128 \naestrada 2022-05-09 19:28:12 1924.1680.27.57 \nasundara 2022-05-11 18:38:07 192.168.96.200 \ndkot 2022-05-12 10:52:00 1921.168.1283.75 \nabernard 2022-05-12 23:38:46 19245.168.2345.49 \ncjackson 2022-05-12 19:36:42 192.168.247.153 \njclark 2022-05-10 10:48:02 192.168.174.117 \nalevitsk 2022-05-08 12:09:10 192.16874.1390.176 \njrafael 2022-05-10 22:40:01 192.168.148.115 \nyappiah 2022-05-12 10:37:22 192.168.103.10654 \ndaquino 2022-05-08 7:02:35 192.168.168.144"

# Assign `pattern` to a regular expression that matches with all valid IP addresses and only those
pattern = "\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}"

# Use `re.findall()` on `pattern` and `log_file` and assign `valid_ip_addresses` to the output
valid_ip_addresses = re.findall(pattern, log_file)

# Assign `flagged_addresses` to a list of IP addresses that have been previously flagged for unusual activity
flagged_addresses = ["192.168.190.178", "192.168.96.200", "192.168.174.117", "192.168.168.144"]

# Iterative statement begins here
# Loop through `valid_ip_addresses` with `address` as the loop variable
for address in valid_ip_addresses:

    # Conditional begins here
    # If `address` belongs to `flagged_addresses`, display "The IP address _____ has been flagged for further analysis."
    if address in flagged_addresses:
        print("The IP address", address, "has been flagged for further analysis.")

    # Otherwise, display "The IP address _____ does not require further analysis."
    else:
        print("The IP address", address, "does not require further analysis.")
```

??????

???? True ? False

```
import re
def check_aei (text):
  result = re.search(r".*a.+e.+i.*", text)
  return result != None

print(check_aei("academia")) # True
```

```
print(check_aei("aerial")) # False
print(check_aei("paramedic")) # True
```

??: ???????????????

```
import re
def check_punctuation (text):
  result = re.search(r"[^a-zA-Z ]", text)
  return result != None

print(check_punctuation("This is a sentence that ends with a period.")) # True
print(check_punctuation("This is a sentence fragment without a period")) # False
print(check_punctuation("Aren't regular expressions awesome?")) # True
```

```
import re
def compare_strings(string1, string2):
  # Convert both strings to lowercase
  # and remove leading and trailing blanks
  string1 = string1.lower().strip()
  string2 = string2.lower().strip()

  # Removed punctuation
  punctuation = r"[.?!,;:\-']"

  string1 = re.sub(punctuation, r"", string1)
  string2 = re.sub(punctuation, r"", string2)

  # DEBUG CODE GOES HERE
  #print(string1 == string2)
  return string1 == string2

print(compare_strings("Have a Great Day!", "Have a great day?")) # True
print(compare_strings("It's raining again.", "its raining, again")) # True
print(compare_strings("Learn to count: 1, 2, 3.", "Learn to count: one, two, three.")) # False
print(compare_strings("They found some body.", "They found somebody.")) # False
```

???check web address

```
import re
def check_web_address(text):
```

```
  pattern = r"[\w-]*\.[a-zA-Z]*$"
  result = re.search(pattern, text)
  return result != None

print(check_web_address("gmail.com")) # True
print(check_web_address("www@google")) # False
print(check_web_address("www.Coursera.org")) # True
print(check_web_address("web-address.com/homepage")) # False
print(check_web_address("My_Favorite-Blog.US")) # True
```

???check time

```
import re
def check_time(text):
  pattern = r"[1-9|10|11|12]:[0-5][0-9] *[AaPp][mM]$"
  result = re.search(pattern, text)
  return result != None

print(check_time("12:45pm")) # True
print(check_time("9:59 AM")) # True
print(check_time("6:60am")) # False
print(check_time("five o'clock")) # False
print(check_time("6:02 am")) # True
print(check_time("6:02km")) # False
```

??????????????????

```
import re
def contains_acronym(text):
  pattern = r"\([0-9A-Z][a-zA-z]*\)"
  result = re.search(pattern, text)
  return result != None

print(contains_acronym("Instant messaging (IM) is a set of communication technologies used for text-based communication")) # True
print(contains_acronym("American Standard Code for Information Interchange (ASCII) is a character encoding standard for electronic communication")) # True
print(contains_acronym("Please do NOT enter without permission!")) # False
print(contains_acronym("PostScript is a fourth-generation programming language (4GL)")) # True
print(contains_acronym("Have fun using a self-contained underwater breathing apparatus (Scuba)!")) # True
```

???Log ?? PID ? Message

```
import re
def extract_pid(log_line):
    regex = r"\[(\d+)\]: ([A-Z]*) "
    result = re.search(regex, log_line)
    if result is None:
        return None
    return "{} ({})".format(result[1], result[2])

print(extract_pid("July 31 07:51:48 mycomputer bad_process[12345]: ERROR Performing package upgrade")) # 12345 (ERROR)
print(extract_pid("99 elephants in a [cage]")) # None
print(extract_pid("A string that also has numbers [34567] but no uppercase message")) # None
print(extract_pid("July 31 08:08:08 mycomputer new_process[67890]: RUNNING Performing backup")) # 67890 (RUNNING)
```

?????????

```
import re
def transform_record(record):
  new_record = re.sub(r"(.*,)(\d{3}-[\d-]+)(,.*)", r"\1+1-\2\3", record)
  return new_record

print(transform_record("Sabrina Green,802-867-5309,System Administrator"))
# Sabrina Green,+1-802-867-5309,System Administrator

print(transform_record("Eli Jones,684-3481127,IT specialist"))
# Eli Jones,+1-684-3481127,IT specialist

print(transform_record("Melody Daniels,846-687-7436,Programmer"))
# Melody Daniels,+1-846-687-7436,Programmer

print(transform_record("Charlie Rivera,698-746-3357,Web Developer"))
# Charlie Rivera,+1-698-746-3357,Web Developer
```

```
import re
def convert_phone_number(phone):
  result = re.sub(r"([\w ]+)(\d{3})-(\d{3}-\d{4}.*)$", r"\1(\2) \3", phone)
  return result
```

```
print(convert_phone_number("My number is 212-345-9999.")) # My number is (212) 345-9999.
print(convert_phone_number("Please call 888-555-1234")) # Please call (888) 555-1234
print(convert_phone_number("123-123-12345")) # 123-123-12345
print(convert_phone_number("Phone number of Buckingham Palace is +44 303 123 7300")) # Phone number of Buckingham Palace is +44 303 123 7300
```

```
# phone.csv:
#123-456-7890
#(123) 456-7890
#1234567890
#

import re

with open("data/phones.csv", "r") as phones:
  for phone in phones:
  new_phone = re.sub(r"^\D*(\d{3})\D*(\d{3})\D*(\d{4})$", r"(\1) \2-\3", phone)
  print(new_phone)

# Output
#(123) 456-7890
#(123) 456-7890
#(123) 456-7890
```

????? a, e, i, o, u ???? 3 ??????

```
import re
def multi_vowel_words(text):
  pattern = r"\w+[aeiou]{3,}\w+"
  result = re.findall(pattern, text)
  return result

print(multi_vowel_words("Life is beautiful"))
# ['beautiful']

print(multi_vowel_words("Obviously, the queen is courageous and gracious."))
# ['Obviously', 'queen', 'courageous', 'gracious']

print(multi_vowel_words("The rambunctious children had to sit quietly and await their delicious dinner."))
```

```
# ['rambunctious', 'quietly', 'delicious']

print(multi_vowel_words("The order of a data queue is First In First Out (FIFO)"))
# ['queue']

print(multi_vowel_words("Hello world!"))
# []
```

\b ???

`\b` ????? (?????????) ??????

```
import re
print(re.findall(r"[a-zA-Z]{5}", "a scary ghost appeared"))

# Output: ['scary', 'ghost', 'appea']

import re
re.findall(r"\b[a-zA-Z]{5}\b", "A scary ghost appeared")

# Output: ['scary', 'ghost']
```

- ?????? `\b` ????????? eid

```
def find_eid(report):
  pattern = r"[A-Z]-[\d]{7,8}\b" #enter the regex pattern here
  result = re.findall(pattern, report) #enter the re method  here
  return result


print(find_eid("Employees B-1234567 and C-12345678 worked with products X-123456 and Z-123456789"))
# Should return ['B-1234567', 'C-12345678']
print(find_eid("Employees B-1234567 and C-12345678, not employees b-1234567 and c-12345678"))
#Should return ['B-1234567', 'C-12345678']
```

Capturing Groups

- ?????????? Regex ?????????
- ??????? Regex ?????????
- ????????????1???2
- `.groups()` method : ?? tuple ??????? (group1, group2, group3)
- result[0]: ???? ,result[1]: ??1, result[2]: ??2

```
import re
result = re.search(r"^(\w*), (\w*)$", "Lovelace, Ada")
print(result)
print(result.groups())
print(result[0])
print(result[1])
print(result[2])
"{} {}".format(result[2], result[1])

# Output
# <_sre.SRE_Match object; span=(0, 13), match='Lovelace, Ada'>
# ('Lovelace', 'Ada')
# Lovelace, Ada
# Lovelace
# Ada
# Ada Lovelace
```

Resources

- regex101: build, test, and debug regex
- ????? (Regular Expression) ???? | Vixual

Revision #79
Created 22 September 2024 11:44:16 by Admin
Updated 11 December 2024 14:56:48 by Admin