sandeepsuryaprasad / **python_tutorials** Private

Code Issues Pull requests Actions Projects Wiki Securi

master python_tutorials / 13_regex / **_regular_expressions.py** / Jump to Go to file ...

Sandeep Suryaprasad cleanup Latest commit b4042cb 5 days ago History

0 contributors

605 lines (495 sloc) 24.7 KB Raw Blame

```python
import re
# ============ Characters ===================
# . - Matches any character except new line
# \. - Mathes a dot.
# \\ - Matches backslash
# \* - Matches astrick
# ============ Character set ===================
# [abcd] - any character which matches either 'a' or 'b' or 'c' or 'd'
# [^abcd] - any character but not 'a' or 'b' or 'c' or 'd'
# [a-z] - any character between 'a' through 'z'
# =========== Special Sequences ================
# \w - Word character. Same as [a-zA-Z0-9_]. Matches alphanumeric and undersc
# \W - Non-Word Character. Same as [^a-zA-Z0-9_]. Matches anything but word c
# \d - Matches a digit. Same as [0-9]
# \D - Matches a Non-Digit. Same as [^0-9]
# \s - Matches only whitespace.
# \S - Matches only Non-Whitespace.
# ========== Anchors =====================
# ^ - Start of String
# $ - End of String
# \b - Word boudary [a-zA-Z0-9_]
# \B - Not a word Boundry
# [ ] - Matches characters in square brackets
# [^ ] - Matches characters Not in square brackets

# Meta Characters that needs to be Escaped
# . ^ $ * + ? { } [ ] \ | ( )

# Quantifiers
"""
1. The standard quantifiers (?, +, *, and {min,max}) are greedy.
2. When one of these governs a subexpression, such as a?, (expr )+, [0-9]+,
   there is a minimum number of matches that are required before it can be co
   attempt it will ever attempt to match,
3. They always attempt to match as many times as they can, up to that maximum
4. The only time they settle for anything less than their maximum allowed is
   ends up causing some later part of the regex to fail.
5. plus, question mark and star are called quantifiers, because they influenc
"""
# * - Match expression 0 or more times
# + - Match expression 1 or more times
# ? - Match expression 0 or 1 times
# {min,max} - Matches expression exactly 3 times

# =========== Grouping ====================
# ("A"| "B" | "C") - Either "A" or "B" or "C"

# re.findall()   # returns a list of all the matches
# re.sub()  # replaces one pattern with other
# re.finditer()  # returns an iterator object
# re.search()  # stops at the first match

# Word Boundary (\b)
# "start of word" boudary is simply the position where a sequence of alphanum
# "end of word" is the position where a sequence of alphanumeric characters e
# -------------------------------------------------------------------------
# Rule: The Match That Begins Earliest (from left to right) Wins
# -------------------------------------------------------------------------
re.findall(r"the", "the theory of relativity")

re.findall(r"cat", "The dragging belly indicates your cat is too fat")

re.findall(r'python', 'python and java are object oriented')

re.findall(r'aeiou', 'hello how are you doing anna')

re.findall(r'aeiou', 'hello how are you doing anna, aeiou')
# -------------------------------------------------------------------------
# Character class or set
# -------------------------------------------------------------------------
# Matches with both "Smith" and "smith"
re.findall(r'[sS]mith', 'smith')
```

```python
re.findall(r'[sS]mith', 'Smith')

# Matches separate or saperate
re.findall(r's[ea]p[ae]rate', 'seperate')
re.findall(r's[ea]p[ae]rate', 'saparate')

# Match any one character in the character set (either a, e, i, o, u)
re.findall(r'[aeiou]', 'hello how are you doing anna')

# Match either a, b, c, d
re.findall(r'[abcd]', 'hello world')
re.findall(r'[abcd]', 'abcdefghijk')

# Matching any number between 0-9
re.findall(r'[0123456789]', 'The cost of the book is Rs.100')

# Matching HTML headers
re.findall(r'<h[123456]>', "<h1>")
re.findall(r'<h[123456]>', "<h2>")
re.findall(r'<h[123456]>', "<h3>")
re.findall(r'<h[123456]>', "<h4>")
re.findall(r'<h[123456]>', "<h5>")
re.findall(r'<h[123456]>', "<h6>")
# ---------------------------------------------------------------------------
# Range "-"
# ---------------------------------------------------------------------------
# Matches any number between 0-9
re.findall(r'[0-9]', 'The cost of the book is Rs.100')

# Matches only lower case letters
re.findall(r"[a-z]", 'hello HOW ARE YOU')

# Matches only upper case letters
re.findall(r"[A-Z]", 'hello HOW ARE YOU')

# Matches all upper case and lower case characters
re.findall(r"[a-zA-Z]", 'hello HOW ARE YOU')

# Matches any number between 1-6
re.findall(r"<h[1-6]>", "<h1>")
re.findall(r"<h[1-6]>", "<h2>")
re.findall(r"<h[1-6]>", "<h3>")
re.findall(r"<h[1-6]>", "<h4>")
re.findall(r"<h[1-6]>", "<h5>")
re.findall(r"<h[1-6]>", "<h6>")
```

```python

# Count total number of Upper case and Lower case letters
sentence = "Hello World Welcome To Python"
upper_case = re.findall(r'[A-Z]', sentence)
lower_case = re.findall(r'[a-z]', sentence)

print(f'Total No of upper case letters {len(upper_case)}')
print(f'Total No of lower case letters {len(lower_case)}')

# Write a program to count the number of white spaces in a given string
sentence = "Hello world welcome to Python Hi  How are you. Hi how are you"
spaces = re.findall(r' ', sentence)

# Write a program to count the number of occurrences of each lower case and u
sentence = 'hello@world! welcome!!! Python$ hi how are you & where are you?'
chrs = re.findall(r'[a-zA-Z]', sentence)
d = {chr: chrs.count(chr) for chr in chrs}

# ---------------------------------------------------------------------------
# Meta Character "+" (matches 1 or more occurances of previous expression)
# ---------------------------------------------------------------------------
re.findall(r'[0-9]+', 'The cost of the book is Rs.100')

re.findall(r'[abcd]+', 'abcdefg hijkab')

re.findall(r'an+a', 'annnnnnnnnnna')

# Matches each word in the string
re.findall(r"[a-zA-Z]+", "Hello World Welcome To Python Programming Pyt123on"

# Count the characters in each word. Please ignore special characters if ther
sentence = "Hi there! How are you:) How are you doing today!"
words = re.findall(r'[a-zA-Z]+', sentence)
word_len = { word: len(word) for word in words}

# Sum all the numbers in the below string.
word = "Pytho12nReg567exp2" # 1 + 2 + 5 + 6 + 7 + 2
total = 0
numbers = re.findall(r'[0-9]', word)
for number in numbers:
    total += int(number)

# Adding 12 + 567 + 2
word = "Pytho12nReg567exp2"
total = 0
```

```python
numbers = re.findall(r'[0-9]+', word)
for number in numbers:
    total += int(number)

# Match file names and extensions
message = "Downloading file archive.zip to downloads folder..."
# image.jpeg
# index.xhtml
# python.py
re.findall(r'[a-z]+\.[a-z]+', message)
# ---------------------------------------------------------------------------
# Meta Character "?" (matches 0 or 1 occurance of previous expression)
# ---------------------------------------------------------------------------
re.findall(r'colou?r', "what color do you like")

re.findall(r'https?://', 'https://www.google.com')

re.findall(r'https?://', 'http://www.google.com')

re.findall(r'July?', "Jul the 26th day")

re.findall(r'an?a', "ana")

re.findall(r'an?a', "anna")
# ---------------------------------------------------------------------------
# Meta Character "*" (matches 0 or more occurances of previous expression)
# ---------------------------------------------------------------------------
re.findall(r"an*a", "hello ana")

re.findall(r"an*a", "hello aa")

re.findall(r"an*a", "hello annna")

# Regular Expression for Matching Inbox, Inbox(1), .... Inbox(N)
re.findall(r"Inbox\(?\d*\)?", "Inbox(10)")
re.findall(r"Inbox\(?\d*\)?", "Inbox")
# ---------------------------------------------------------------------------
# Negation "^"
# ---------------------------------------------------------------------------
# Matches everything apart from numbers between 0-9
re.findall(r'[^0-9]', 'The cost of the book is Rs.100')

# Matches everything apart from alphabets 'a', 'b', 'c' and 'd'
re.findall(r'[^abcd]', 'abcdefg hijkab')

```

```python
# Matches everything apart from numbers between 0-9
re.findall(r'[^0-9]+', 'The cost of the book is Rs.100')

re.findall(r'[^abcd]+', 'abcdefg hijkab')

# Match only those characters excepts digits
word = '@hello12world34welcome!123'
re.findall(r'[^0-9]', word)

# Count the number of special characters in the below string
sentence = 'hello@world! welcome!!! Python$ hi26 how are you & where are you?
re.findall(r"[^a-zA-Z0-9_\s]", sentence)
# ----------------------------------------------------------------------------
# Starts with "^" and ends with "$"
# ----------------------------------------------------------------------------
re.findall(r"^hello", "hello world")

re.findall(r"^hello", "world hello")

re.findall(r"hello$", "world hello")

re.findall(r"hello$", "hello world")

re.findall(r'hello$', 'hello world welcome to python')

# Matching the only those lines which ends with "UDP"
with open("./data_files/sample.log") as f:
    for line in f:
        match = re.findall(r".*UDP$", line)
        if match:
            print("".join(match))

# string starts with "hello" and ends with "hello" (meaning exactly one word
re.findall(r"^hello$", "hello")

# Phone Number pattern (4DIGITS-3DIGITS-3DIGITS)
re.findall(r'\d{3}-\d{3}-\d{4}', '456-9832-098')

# matching only 800 and 900 numbers
re.findall(r"^[89]00-\d{3}-\d{4}", '800-123-123')
# ----------------------------------------------------------------------------
# Word Boundary (\b) The expression should be a word boundry
# (Transition between non-word character to word character or word character
# ----------------------------------------------------------------------------
# starts with word boundry
```

```python
re.findall(r"\bday", "what a beautiful day today is")

# ends with word boundry
re.findall(r"day\b", "what a beautiful day today is")

# starts and ends with word boundry
re.findall(r"\bday\b", "what a beautiful day today is")

re.findall(r"\b[0-9]{6}\b", 'Pincode of Bangalore is 560001 and the number is

# Regular expression which matches words that starts with "h"
re.findall(r"\bh[a-zA-Z0-9_]+", 'hello world hi hello universe how are you ha

# Regular expression which matches words that starts with "P or J"
re.findall(r"\b[PJ][a-zA-Z0-9_]+", 'Python is a programming language. Python

# Regular expression which matches words that ends with "y"
re.findall(r"[a-zA-Z0-9_]+y\b", 'hello world hi hello universe how are you ha

# print only those words starting with vowel character
sentence = "hello hi american engieers and indian writers officers united sta
words = re.findall(r"\b[aeiou][a-zA-Z0-9_]+", sentence)

# Matches only Capital Letter words
re.findall(r"\b[A-Z]+\b", "This is PYTHON programming LANGUAGE and REGEX")

# Matches only lower case words
re.findall(r"\b[a-z]+\b", "This is PYTHON programming LANGUAGE and REGEX")

# Matching only pdf files
re.findall(r"[a-zA-Z0-9]+\.pdf\b", "downloading apple.pdf to downloads folder

# Regular expression for matching only 3 letter words in the given string
sentence = "hello hi how are you what is your name he is older than me how ol
re.findall(r'\b[a-zA-Z0-9_]{3}\b', sentence)
# o/p ['how', 'are', 'you', 'how', 'old', 'are', 'you']

# Extract only 4 digit numbers from the string
re.findall(r"\b\d{4}\b", "Copyright 1998. All rights reserved")

# Regular expression for matching the words which starts with "he"
sentence = "he helps the community and he is the hero of the day"
re.findall(r"\bhe[a-zA-Z0-9_]*", sentence)

# Regular expression for matching the words which either starts with "he" or
```

```python
sentence = "he helps the community and he is the hero of the day she sells se
re.findall(r"\b(?:he|se)[a-zA-Z0-9_]*", sentence)

# Regular Expression - PAN Number
sentence = "my pan number is ABCDE1234X and the other number is XYZTR3104J id
re.findall(r'\b[A-Z]{5}[0-9]{4}[A-Z]\b', sentence)

# Different Combintations
line = "03/22 08:51:06 WARNING :.....mailslot_create: setsockopt(MCAST_ADD) f
re.findall(r"[A-Z]+", line)
re.findall(r"\b[A-Z]+", line)
re.findall(r"[A-Z]+\b", line)
re.findall(r"\b[A-Z]+\b", line)

# Matches all digits
re.findall(r"\d", "654 this string is starting with 12 and ending with number

# Matches whole numbers
re.findall(r"\d+", "654 this string is starting with 12 and ending with numbe

# Matches only 3 Digit numbers
re.findall(r"\d{3}", "654 this string is starting with 12 and ending with num

# Matches all digit numbers
re.findall(r"\b\d{3}\b", "654 this string is starting with 12 and ending with

# Matches the string that ends with 3 digit number
re.findall(r"\b\d{3}\b$", "4632746327 this string is ending with 235")

# Matches the string that starts with 3 digit number
re.findall(r"^\b\d{3}\b", "654 this string is starting with and ending with n
# -------------------------------------------------------------------------------
# Count the number of white spaces in the file
def count_spaces():
    with open('./data_files/sample.log') as f:
        white_spaces = 0
        for line in f:
            count = len(re.findall(r"\s", line))
            white_spaces += count
    return white_spaces
# -------------------------------------------------------------------------------
# Count the number of Capital Letter words in the file
def count_caps():
    with open('./data_files/sample.log') as f:
        capital_words = 0
        for line in f:
            count = len(re.findall(r"\b[A-Z]+\b", line))
            capital_words += count
    return capital_words
# ------------------------------------------------------------------------------
# Count the number of Capital Letters in the file
def count_cap_letters():
    with open('./data_files/sample.log') as f:
        capital_letters = 0
        for line in f:
            count = len(re.findall(r"[A-Z]", line))
            capital_letters += count
    return capital_letters
# ------------------------------------------------------------------------------
# Count the number of INFO, TRACE, WARNING, EVENT messages in the file
def count_messages(message_name):
    with open('./data_files/sample.log') as f:
        message_count = 0
        for line in f:
            count = len(re.findall(message_name, line))
            message_count += count
    return message_count
# ------------------------------------------------------------------------------
def get_all_ip():
    """Extract all the ip addresses in the sample log file"""
    ips = [ ]
    with open('./data_files/sample.log') as f:
        for line in f:
            ip = re.findall(r'\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}', line)
            if ip:
                for item in ip:
                    ips.append(item)
    return ips
# ------------------------------------------------------------------------------
def count_words():
    """Counts the number of words in the file"""
    word_count = 0
    with open('./data_files/sample.log') as f:
        for line in f:
            words = re.findall(r"\b[A-Za-z]+\b", line)
            word_count += len(words)
    return word_count
# ------------------------------------------------------------------------------
def count_total_letters():
    """Counts number of lowecase and upper case characters in the file"""
    with open('./data_files/sample.log') as f:
        total_letters = 0
        for line in f:
            count = len(re.findall(r"[a-zA-Z]", line))
            total_letters += count
    return total_letters
# ---------------------------------------------------------------------------
def get_log_lines():
    """Prints only those log lines which are logged in the year 98"""
    with open("../data_files/sample.log") as f:
        for line in f:
            matches = findall(r"(?:0[1-9]|1[0-2])/98", line)
            if matches:
                print(line, end="")
# ---------------------------------------------------------------------------
def regex_strip(line):
    """Strips leading and trailing whitespaces and returns a new string"""
    stripped_line = sub(r"(^\s*|\s*$)", "", line)
    return stripped_line
# ---------------------------------------------------------------------------
def count_leading_trailing_spaces(line):
    """Returns a tuple of leading and trailing whitespaces"""
    leading_spaces = findall(r"\s+$", line)
    trailing_spaces = findall(r"^\s+",line)
    # join the list
    count_leading_spaces = len("".join(leading_spaces))
    count_trailing_spaces = len("".join(trailing_spaces))
    return (count_leading_spaces, count_trailing_spaces)
# ---------------------------------------------------------------------------
# Flitering only floating point values from file
with open("./data_files/points.txt") as f:
    floats = [ ]
    for line in f:
        match = re.findall(r"-?[0-9]+\.[0-9]+", line)
        # match = re.findall(r"-?[0-9]+\.[0-9]{3}", line) # matches 3 digits
        for item in match:
            floats.append(item)
# ---------------------------------------------------------------------------
# Groups ( )
# ---------------------------------------------------------------------------
# Matches either "python" or "java"
re.findall('(python|java)', 'python and java are object oriented')

# Matches 09:00 am/pm or 9:00 am/pm
sentence = 'The meeting is between 9:00 am and 12:30 pm'
```

```python
re.findall(r'[0-9]?[0-9]:[0-9][0-9]\s(?:am|pm)', sentence)

# Regular Expression - YYYY-MM-DD date format
_dates = ['2019-01-02', '2019-13-02', '2019-12-26', '26-08-2019', '20-19-20',
re.findall(r'\d{4}-(?:0[1-9]|[12][0-9]|3[01])-(?:0[1-9]|[12][0-9]|3[01])', '2

# Regular Expression - 24hr time format
_formats = ['00:00:00', '23:59:59', '24:00:00', '1:59:20', '12:9:10', '10:20:
re.findall(r"(?:[01]\d|2[0-3]):[0-5]\d:[0-5]\d",'23:59:59')
# ---------------------------------------------------------------------------
# Replacing patterns
# ---------------------------------------------------------------------------
# Replace whitespaces with newline character in the below string
sentence = "Hello world welcome to python"
words = re.sub(r'\s', '\n', sentence)
print(words)

# Replace all vowels with "*"
sentence = "hello world welcome to python"
words = re.sub(r'[aeiou]', '*', sentence)
print(words)

# Replace all occurances of digits with "*"
sentence = 'hello123world welcome456to python012'
words = re.sub(r'\d', '*', sentence)

# Replace all occurances of special characters with "*"
sentence = 'hello#$%world welcome@!#$%to python*&^%'
words = re.sub(r'[^a-zA-Z\s]', "*", sentence)

# Replace "And" with "&"
sentence = "Java and Python are programming languages"
_sentence = re.sub(r"\sAnd\s", " & ", sentence)

# Replace all occurances of "Java" with "Python" in a file
with open('java.txt', 'r') as jf:
    with open('python.txt', 'a') as pf:
        for line in jf:
            new_line = re.sub('Java', 'Python', line)
            pf.write(new_line)
# ---------------------------------------------------------------------------
# re.finditer()
# ---------------------------------------------------------------------------
matches = re.finditer(r"hello", 'hello world welcome to python hello world')
for match in matches:
    print(match.group())

# Write a program to find the index of nth occurrence of a sub-string in a st
sentence = "hello world welcome to python hello hi how are you hello there"
matches = re.finditer(r'hello', sentence)
positions = [ (match.start(), match.end()) for match in matches]

# ---------------------------------------------------------------------------
# re.search()
# ---------------------------------------------------------------------------
match = re.search(r"hello", 'hello world hello world')
print(match.group())

# ---------------------------------------------------------------------------
# dot "." matches with everything
# ---------------------------------------------------------------------------
re.findall(r'.', "hello world")

re.findall(r'h.', "hello")

re.findall(r'h.', "hello world hi how how are you")

re.findall(r'a.b', "acb")

re.findall(r'a.b', "a b")

re.findall(r'a.b', "ab")

re.findall(r'a.b', "a*b a?b")

re.findall(r'a.b', "abcde")

re.findall(r'a.a', "ana")

re.findall(r'a..a', "anna")

re.findall(r'a.*a', "annnnnna")

re.findall(r'a.*a', 'aa')

re.findall(r"^a.*a$", "anna")

re.findall(r"^a.*a$", "hello anna")

re.findall(r'a.*a', 'abcad')

re.findall(r'a.*a$', 'abcad')

re.findall(r'a.*a$', 'abcada')

re.findall(r'a.+a', 'ana')

re.findall(r'a.+a', 'aa')
# -----------------------------------------------------------------------------
# ---------------------------------- Back-referencing -----------------------
"""
1. Back-referencing is a regular-expression feature which allows you to match
matched earlier in the expression without specifically knowing the text when
2. Back-references provide a convenient way to identify a repeated character
For example, if the input string contains multiple occurrences of an arbitrar
first occurrence with a capturing group, and then use a backreference to matc
3. \1 will try to match what ever is matched in the first bracket
"""
# Repeated word sequences
m = re.findall(r"(world)\1", "thethe python hello worldworld the")

# Repeated words
m = re.findall(r"([a-z]+\s)\1", "the the python hello world world the")

# Repeated words for 2 consequtive times
m = re.finditer(r"([a-z]+\s)\1{2}", "the the python hello world world the the

# Repeated characters
m = re.findall(r"([a-z])\1", "hello hurry programming")

# Repeated numbers
n = re.findall(r"([0-9])\1", "hello 123345, welcome to 001 98799")

# Repeated numbers pattern
n = re.findall(r"([0-9]+\s)\1", "hello 12345 12345 , welcome to 001 98799")

n = re.findall(r"([0-9])+", "hello 1234512345 , welcome to 00198799")

# finding the words that are repeated at the beginning and at the end of the
sentence = "hello world welcome to regex hello"
re.findall(r"^([a-z]+).*\1$", sentence)
# -----------------------------------------------------------------------------
# Regular Expression - IP Addresses
# -----------------------------------------------------------------------------
id_address_format = '\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}'
```

```
ips = ['10.1.2.3', '127.0.0.0', '199.99.9.9', '199.9.9999.9', '127-0-0-0']
# ---------------------------------------------------------------------------
# Regular Expression - Email format
# ---------------------------------------------------------------------------
email_pattern = r'[\w-]+\.?[\w-]@[\w]+\.(com|edu|in|gov)'
emails = ['test.user@company.com',
          'test.user2@company.com',
          'test_user@company.com',
          'testing@company.com',
          'test-T.user@company.com',
          'testing@company',
          'testingcompany.com'
          ]
# ---------------------------------------------------------------------------
# Regular Expression - URL Pattern
url_pattern = r'https?://[\w.]+'
urls = ['http://www.youtube.com',
        'https://www.google.com',
        'http://www.amazon.in',
        'https://www.mail.yahoo.com',
        'ftp://test.com'
        'https://www.facebook.com/'
        ]
# ---------------------------------------------------------------------------

# ---------------------------------------------------------------------------
# ---------------------------------- LookAhead and LookBehind Anchors -------
        # Under development ...
# ---------------------------------------------------------------------------
# Misc
# ---------------------------------------------------------------------------
# Finding the all the links in a html document using regex
from requests import request
response = request("GET", "http://demowebshop.tricentis.com/")
html_code = response.text
links = re.findall(r"\s*<a\s.+", html_code)
links = [ link.strip() for link in links ]
# ---------------------------------------------------------------------------
```