Mod_5_strings_regular_expression

October 10, 2024

1 re.match():

Purpose: This function checks for a match only at the beginning of the string. Behavior: It returns a match object if the pattern matches at the start of the string; otherwise, it returns None. Even if the pattern appears later in the string, re.match will not find it.

[4]:
```
import re

text = "Hello, World!"
pattern = "Hello"

# re.match only checks the beginning of the string
match = re.match(pattern, text)
print(match)
print(type(match))
print(match.span())
print(match.group())
print(match.string) #note string is not the variable
```

```
<re.Match object; span=(0, 5), match='Hello'>
<class 're.Match'>
(0, 5)
Hello
Hello, World!
```

[5]:
```
import re

text = "Hello, World!"
pattern = "Hello"

# re.match only checks the beginning of the string
match = re.match(pattern, text)
if match:
    print("Matched:", match.group())  # This will print "Matched: Hello"
else:
    print("No match")
```

```
Matched: Hello
```

[6]:
```
import re

text = "Hello, World!"
pattern = "llo"

# re.match only checks the beginning of the string
match = re.match(pattern, text)
if match:
    print("Matched:", match.group())  # This will print "Matched: Hello"
else:
    print("No match")
```

```
No match
```

[7]:
```
import re

text = "Hello, World!"
pattern = "World"

# re.match only checks the beginning of the string
match = re.match(pattern, text)
if match:
    print("Matched:", match.group())  # This will print "Matched: Hello"
else:
    print("No match")
```

```
No match
```

[13]:
```
import re

text = "Hello, World!"
pattern = "lo, W"

# re.match only checks the beginning of the string
match = re.match(pattern, text)
if match:
    print("Matched:", match.group())  # This will print "Matched: Hello"
else:
    print("No match")
```

```
No match
```

2 re.search():

match = re.search(pattern, str) Purpose: This function searches for a match anywhere in the string, not just at the beginning. Behavior: It returns a match object if the pattern is found anywhere in the string, and returns None if no match is found.

[8]:
```
import re

text = "Hello, World!"
pattern = "World"

# re.search checks the entire string for the pattern
search = re.search(pattern, text)
if search:
    print("Found:", search.group())  # This will print "Found: World"
else:
    print("Not found")
```

```
Found: World
```

[11]:
```
import re

text = "Hello, World!"
pattern = "lo,"

# re.search checks the entire string for the pattern
search = re.search(pattern, text)
if search:
    print("Found:", search.group())  # This will print "Found: World"
else:
    print("Not found")
```

```
Found: lo,
```

[12]:
```
import re

text = "Hello, World!"
pattern = "lo, W"

# re.search checks the entire string for the pattern
search = re.search(pattern, text)
if search:
    print("Found:", search.group())  # This will print "Found: World"
else:
    print("Not found")
```

```
Found: lo, W
```

[14]:
```
import re

text = "Hello, World!"
pattern = "ab"

# re.search checks the entire string for the pattern
search = re.search(pattern, text)
```

```
if search:
    print("Found:", search.group())  # This will print "Found: World"
else:
    print("Not found")
```

```
Not found
```

[19]:
```
import re

text = "Hello, World! Welcome!"
pattern = "el"
x=re.search(pattern,text)
print(x)
print(x.start())
print(x.end())
print(x.group())
```

```
<re.Match object; span=(1, 3), match='el'>
1
3
el
```

[21]:
```
import re

txt = "The rain in Spain"
x = re.findall("ai", txt)
print(x)
print(type(x))
```

```
['ai', 'ai']
<class 'list'>
```

[22]:
```
import re

txt = "The rain in Spain"
x = re.findall("Portugal", txt)
print(x)
```

```
[]
```

[2]:
```
import re
txt=" The rain in spain"
x=re.sub("\s","9",txt)
print(x)
```

```
9The9rain9in9spain
```

[3]:
```
import re
txt=" The rain in spain"
x=re.subn("\s","9",txt)
```

```
print(x) #it displays the number if times 9 has been substituted
```

```
('9The9rain9in9spain', 4)
```

[24]:
```
import re
txt=" The rain in spain"
x=re.subn("\s","9",txt,3)
print(x)
```

```
('9The9rain9in spain', 3)
```

[63]:
```
import re

text = "John Doe 123"
pattern = r"(\w+) (\w+) (\d+)"

match = re.search(pattern, text)

if match:
    print("Full Match:", match.group(0))   # The entire matched string
    print("First Name:", match.group(1))    # First capturing group (John)
    print("Last Name:", match.group(2))     # Second capturing group (Doe)
    print("Number:", match.group(3))         # Third capturing group (123)
```

```
Full Match: John Doe 123
First Name: John
Last Name: Doe
Number: 123
```

[9]:
```
import re

text = "John D@oe 123"
pattern = r"(\w+) (\D+) (\d+)"

match = re.search(pattern, text)

if match:
    print("Full Match:", match.group(0))   # The entire matched string
    print("First Name:", match.group(1))    # First capturing group (John)
    print("Last Name:", match.group(2))     # Second capturing group (Doe)
    print("Number:", match.group(3))         # Third capturing group (123)
```

```
Full Match: John D@oe 123
First Name: John
Last Name: D@oe
Number: 123
```

[64]:
```
import re
```

```
text = "John D@oe a123"
pattern = r"(\w+) (\D+) (\d+)"
match = re.search(pattern, text)
print(match)
```

```
None
```

[4]:
```
import re

# Sample text
text = """
    Hello World! 123-456-7890
    Email: example@example.com
    Visit us at https://www.example.com
    User: John_Doe
    Date: 2024-10-08
"""

# Regex pattern that utilizes most metacharacters
pattern = r"""
    (Hello)\s+              # Matches the word "Hello" followed by one or more␣
 ↪whitespace characters
    (World)                 # Matches the word "World"
    [!?.]                   # Matches any punctuation (!, ?, .)
    \s*                    # Matches zero or more whitespace characters
    (\d{3})                 # Capturing group: Matches exactly 3 digits (area␣
 ↪code in phone number)
    -                       # Matches the dash character literally
    (\d{3})                 # Capturing group: Another 3 digits (prefix in phone␣
 ↪number)
    -                       # Matches another dash
    (\d{4})                 # Capturing group: Four digits for the line number␣
 ↪in phone number
    \s*                     # Matches zero or more whitespace characters
    Email:\s*              # Matches the word "Email:" followed by zero or more␣
 ↪whitespace characters
    ([\w.%+-]+)            # Capturing group: Matches an email user name (before␣
 ↪@)
    @                       # Matches the @ symbol literally
    ([A-Za-z0-9.-]+)       # Capturing group: Matches the domain name
    \.                      # Matches the literal dot "."
    ([A-Za-z]{2,})         # Capturing group: Matches top-level domain (TLD)
"""

# Compile the pattern with VERBOSE for readability
regex = re.compile(pattern, re.VERBOSE)
```

```
# Find all matches
matches = regex.finditer(text)

# Print the results
for match in matches:
    print("Full Match:", match.group(0))         # The entire matched string
    print("First Word:", match.group(1))         # First word (Hello)
    print("Second Word:", match.group(2))        # Second word (World)
    print("Area Code:", match.group(3))          # Area code from phone number
    print("Prefix:", match.group(4))             # First part of phone number
    print("Line Number:", match.group(5))        # Last part of phone number
    print("Email Name:", match.group(6))         # Email user name
    print("Domain:", match.group(7))             # Domain name in email
    print("TLD:", match.group(8))                # Top-level domain (TLD)
    print("-" * 30)                              # Separator between matches
```

```
Full Match: Hello World! 123-456-7890
    Email: example@example.com
First Word: Hello
Second Word: World
Area Code: 123
Prefix: 456
Line Number: 7890
Email Name: example
Domain: example
TLD: com
------------------------------
```

[6]:
```
import re

# Sample text
text = """
    Hello World! 123-456-7890
    Email: example@example.com
    Visit us at https://www.example.com
    User: John_Doe
    Date: 2024-10-08
"""

# Simple regex pattern to match a greeting and phone number
pattern = r"Hello (\w+)! (\d{3})-(\d{3})-(\d{4})"

# Find the first match in the text
match = re.search(pattern, text)

# Check if a match was found
if match:
```

```
    print("Full Match:", match.group(0))        # The entire matched string
    print("Second Word:", match.group(1))       # Second word (World)
    print("Area Code:", match.group(2))         # Area code from phone number
    print("Prefix:", match.group(3))            # First part of phone number
    print("Line Number:", match.group(4))       # Last part of phone number
else:
    print("No match found.")
```

```
Full Match: Hello World! 123-456-7890
Second Word: World
Area Code: 123
Prefix: 456
Line Number: 7890
```

[8]:
```
import re
if re.match("..$","fooo"):
    print("Matched")
else:
    print("Not matched")
```

```
Not matched
```

[10]:
```
import re
if re.match("....$","fooo"):
    print("Matched")
else:
    print("Not matched")
```

```
Matched
```

[11]:
```
import re
if re.match("...","fooo"):
    print("Matched")
else:
    print("Not matched")
```

```
Matched
```

[22]:
```
import re
if re.match("..","abb"):
    print("Matched")
else:
    print("Not matched")
```

```
Matched
```

[65]:
```
import re
if re.match(".....","abb"):
    print("Matched")
else:
```

```
    print("Not matched")
```

```
Not matched
```

[21]:
```
import re
if re.match("...","a"):
    print("Matched")
else:
    print("Not matched")
```

```
Not matched
```

[28]:
```
import re
x = 'My 2 fAvorite numbers are 19 and 42'
y = re.findall('[0-9]+',x)
print(y)

y = re.findall('[AEIOU]+',x)
print(y)
```

```
['2', '19', '42']
['A']
```

[26]:
```
import re
x = 'My 2 favorite numbers are 19 and 42'
y = re.findall('[0-9]',x)
print(y)

y = re.findall('[aeiou]',x)
print(y)
```

```
['2', '1', '9', '4', '2']
['a', 'o', 'i', 'e', 'u', 'e', 'a', 'e', 'a']
```

[27]:
```
import re
x = 'My 2 favorite numbers are 19 and 42'
y = re.findall('[0-9]',x)
print(y)

y = re.findall('[aeiou]+',x)
print(y)
```

```
['2', '1', '9', '4', '2']
['a', 'o', 'i', 'e', 'u', 'e', 'a', 'e', 'a']
```

[11]:
```
import re
x = 'From: Using the : character'
y = re.findall('^F.+:', x)
print(y)
```

```
['From: Using the :']
```

[15]:
```
import re
x = 'From: Using the : character'
y = re.findall('^U.+:', x) #should be from the beginning of the sentence
print(y)
```

```
[]
```

[67]:
```
import re
x = 'From: Using the : character'
y = re.findall('^F.+?:', x)
print(y)
```

```
['From:']
```

[34]:
```
x="From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008"
y = re.findall('\S+@\S+',x)
print(y)
```

```
['stephen.marquard@uct.ac.za']
```

[35]:
```
x="From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008"
y = re.findall('(\S+@\S+)',x)
print(y)
```

```
['stephen.marquard@uct.ac.za']
```

[38]:
```
import re
lin = 'From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008'
y = re.findall('@([^ ]*)',lin)
print(y)
```

```
['uct.ac.za']
```

[42]:
```
"""
.*@ (greedy matching):

The .* is a greedy matcher, meaning it will try to match as many characters
as possible.It starts at the beginning of the string after the ^From part and
continues until it encounters the @ symbol.However, the part before the @ is not
captured, because it is not in a capturing group.

Capturing group ([^ ]*):

This part comes after the @. It is enclosed in parentheses, which means this
portion is captured.[^ ]* matches any sequence of characters that are
not a space, which ensures that it captures the domain after the @.

"""
```

```
import re
lin = 'From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008'
y = re.findall('^From .*@([^ ]*)',lin)
print(y)
```

```
['uct.ac.za']
```

[43]:
```
import re
lin = 'From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008'
y = re.findall('^From .*@([^ ]+)',lin)
print(y)
```

```
['uct.ac.za']
```

2.1 Escape character:

If you want a special regular expression character to just behave normally (most of the time) you prefix it with ''

[50]:
```
import re
x = 'We just received $10.00 for cookies.'
y = re.findall('\$[0-9.]+',x)
print(y)
```

```
['$10.00']
```

[54]:
```
import re

# Example string
test_string = "abc abbc abbbc aac abb ac"

# Using * (0 or more) (because * allows for 0 b),
pattern_star = r'ab*c'  # This will match 'ac', 'abc', 'abbc', 'abbbc', etc.
matches_star = re.findall(pattern_star, test_string)

# Using + (1 or more)(requires at least 1 b),
pattern_plus = r'ab+c'  # This will match 'abc', 'abbc', 'abbbc', but NOT 'ac'
matches_plus = re.findall(pattern_plus, test_string)

print("Matches with * (0 or more):", matches_star)
#in this for test_string aac, the output is ac, because
#The first a is not part of the pattern, so it's skipped.
print("Matches with + (1 or more):", matches_plus)
```

```
Matches with * (0 or more): ['abc', 'abbc', 'abbbc', 'ac', 'ac']
Matches with + (1 or more): ['abc', 'abbc', 'abbbc']
```

2.2 Metacharacter(?)

When placed after a character or group, ? makes that character or group optional, allowing it to appear zero or one time.

[55]:
```
import re

# Example string
test_string = "color colour"

# Pattern with ?
pattern = r'colou?r'  # 'u' is optional

matches = re.findall(pattern, test_string)
print(matches)
```

```
['color', 'colour']
```

[56]:
```
import re

# Example string
test_string = "color colour"

# Pattern with ?
pattern = r'colou?r'  # 'u' is optional

matches = re.search(pattern, test_string)
print(matches)
```

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

[58]:
```
import re

# Example string
test_string = "color colour"

# Pattern with ?
pattern = r'colou?r'  # 'u' is optional

matches = re.match(pattern, test_string)
print(matches)
```

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

[67]:
```
import re
txt="have a happy day, have a happ day, have a hap day"
#pattern=r'happy?' ## Pattern where 'y' is optional , Method 1 of pattern
pattern='happy?'
matches=re.findall(pattern,txt)
```

```
print(matches)
```

```
['happy', 'happ']
```

[70]:
```
import re

# Example strings
txt_with_happy = "have a happy day"
txt_without_happy = "have a day"

# Pattern to match the phrase with 'happy' being optional
pattern = r'(happy\s+)?(have a day)'

# Match with 'happy'
matches_with_happy = re.findall(pattern, txt_with_happy)

# Match without 'happy'
matches_without_happy = re.findall(pattern, txt_without_happy)

# Print the results
print("Matches with 'happy':", matches_with_happy)
print("Matches without 'happy':", matches_without_happy)
```

```
Matches with 'happy': []
Matches without 'happy': [('', 'have a day')]
```

[62]:
```
import re
txt="have a beautiful day"
pattern=r'happy?'
matches=re.findall(pattern,txt)
print(matches)
```

```
[]
```

[63]:
```
import re

# Example string
test_string = "This is <b>bold</b> and <b>another bold</b>."

# Greedy matching (default)
pattern_greedy = r'<b>.*</b>'
greedy_match = re.findall(pattern_greedy, test_string)

# Non-greedy matching with ?
pattern_non_greedy = r'<b>.*?</b>'
non_greedy_match = re.findall(pattern_non_greedy, test_string)

print("Greedy match:", greedy_match)
print("Non-greedy match:", non_greedy_match)
```

```
Greedy match: ['<b>bold</b> and <b>another bold</b>']
Non-greedy match: ['<b>bold</b>', '<b>another bold</b>']
```

[57]:
```
import re
if re.match("f.o$","fooo"):
    print("Matched")
else:
    print("Not matched")
```

```
Not matched
```

2.2.1 explanation

The greedy pattern .* matches everything between the first and the last , so it captures the whole string “bold and another bold”.

The non-greedy pattern .*? captures the smallest possible match, so it matches each tag separately.

[18]:
```
import re
if re.match("f.o$","fooo"):
    print("Matched") #only if the text is foo
else:
    print("Not matched") #if the text is fooo
```

```
Not matched
```

[24]:
```
import re
if re.match("..","fooo "):
    print("Matched")
else:
    print("Not matched")
```

```
Matched
```

[25]:
```
import re
if re.match(".end","bend"):
    print("Matched")
else:
    print("Not matched")
```

```
Matched
```

[42]:
```
import re
pattern=r'["-a*]'
if re.match(pattern,"apple"):
    print("Matched")
else:
    print("Not matched")
```

```
Matched
```

[52]:
```python
import re

pattern = r'0?$'

# Test strings
tests = ["1", "09", "5", "11", "0", "03"]

for test in tests:
    if re.match(pattern, test):
        print(f"Matched: {test}")
    else:
        print(f"Not matched: {test}")
```

```
Not matched: 1
Not matched: 09
Not matched: 5
Not matched: 11
Matched: 0
Not matched: 03
```

[55]:
```python
import re

pattern = r'0?[2-9]'

# Test strings
tests = ["1", "09", "5", "11", "0", "03"]

for test in tests:
    if re.match(pattern, test):
        print(f"Matched: {test}")
    else:
        print(f"Not matched: {test}")
```

```
Not matched: 1
Matched: 09
Matched: 5
Not matched: 11
Not matched: 0
Matched: 03
```

[56]:
```python
import re

pattern = r'0?.[2-9]'

# Test strings
tests = ["1", "09", "5", "11", "0", "03"]

for test in tests:
```

```
    if re.match(pattern, test):
        print(f"Matched: {test}")
    else:
        print(f"Not matched: {test}")
```

```
Not matched: 1
Matched: 09
Not matched: 5
Not matched: 11
Not matched: 0
Matched: 03
```

[ ]: