12/04/24

Today’s work

Today at 09.00 we started practicing python for strong basic knowledge.

Regular expression part2

Characters Description

. Matches any character except a newline

Eg: **a.c** would match "abc", "a1c", "a#c", etc

\d Matches any digit (equivalent to [0-9])

Eg: **\d+** would match any sequence of digits, like "123", "4567", etc

\D Matches any non-digit character(equivalent to[^0-9])

Eg: **\D+** would match any sequence of non-digits, like "abc", "!@#", etc.

\w Matches any alphanumeric character(equivalent to [a-zA-z0-9\_]

Eg: **\w+** would match any word, including "hello", "Python123", etc.

\W Matches any non-alphanumeric character(equivalent to [^a-zA-z0-9\_]

Eg: **\W+** would match any sequence of non-alphanumeric characters, like "!@#", "$$$", etc.

\s Matches any whitespaces character (equivalent to [\t\n\r\f\v])

Eg: **\s+** would match any sequence of whitespace characters, like spaces, tabs, newlines, etc.

\S Matches any non-whitespaces character (equivalent to [^\t\n\r\f\v])

Eg: **\S+** would match any sequence of non-whitespace characters, like "hello", "Python123", etc.

| **Quantifier** | **Description** | **Example** |
| --- | --- | --- |
| **\*** | Matches zero or more occurrences | **a\*** matches "", "a", "aa", "aaa", ... |
| **+** | Matches one or more occurrences | **a+** matches "a", "aa", "aaa", ... |
| **?** | Matches zero or one occurrence | **a?** matches "", "a" |
| **{n}** | Matches exactly n occurrences | **a{3}** matches "aaa" |
| **{n,}** | Matches n or more occurrences | **a{2,}** matches "aa", "aaa", "aaaa", ... |
| **{n,m}** | Matches between n and m occurrences | **a{2,4}** matches "aa", "aaa", "aaaa" |
|  |  |  |

import re

text = "My phone number is 408-555-1234"

#phone = re.search(r'\d\d\d-\d\d\d-\d\d\d\d',text)

#phone = re.search(r'\d{3}-\d{3}-\d{3}',text)

phone = re.search(r'(\d{3})-(\d{3})-(\d{3})',text)

print(phone)

print(phone.group(3))

output:

<re.Match object; span=(19, 30), match='408-555-123'>

123

408-555-123

Regex part3

#Additional Regex Syntax

print(re.search(r'cat|dog','The cat is here'))

print(re.findall(r'at','The cat in the hat sat there.'))

print(re.findall(r'.at','The cat in the hat sat there.'))

print(re.findall(r'^\d','1 The cat in the hat sat there.'))

print(re.findall(r'\d$','The cat in the hat sat there 2'))

output:

<re.Match object; span=(4, 7), match='cat'>

['at', 'at', 'at']

['cat', 'hat', 'sat']

['1']

['2']

Meta character chart

| **Metacharacter** | **Description** | **Example** | **Matches** |
| --- | --- | --- | --- |
| **.** | Matches any character except a newline | **a.b** matches "axb", "a4b", but not "a\nb" |  |
| **^** | Matches the start of a string | **^abc** matches "abc" at the start of a string |  |
| **$** | Matches the end of a string | **abc$** matches "abc" at the end of a string |  |
| **\*** | Matches zero or more occurrences of the pattern | **ab\*c** matches "ac", "abc", "abbc", "abbbc", ... |  |
| **+** | Matches one or more occurrences of the pattern | **ab+c** matches "abc", "abbc", "abbbc", ... |  |
| **?** | Matches zero or one occurrence of the pattern | **ab?c** matches "ac", "abc" |  |
| **{}** | Matches a specific number of occurrences | **ab{2}c** matches "abbc" |  |
| **[]** | Matches any one of the characters inside the brackets | **[abc]** matches "a", "b", or "c" |  |
| **[^]** | Matches any character not in the brackets | **[^abc]** matches any character except "a", "b", or "c" |  |
| ` | ` | Alternation, matches either the pattern on its left or right | `a |
| **()** | Groups patterns together | **(ab)+** matches "ab", "abab", "ababab", ... |  |
| **\** | Escapes a metacharacter or creates special sequences | **\d** matches any digit, **\s** matches any whitespace character |  |

text = " Tanuj contact number is 9999999999, and its US contact number is (999)-333-7777"

#Extracting the indian number

pattern = '\d'

matches = re.findall(pattern, text)

print(matches)

output: ['9', '9', '9', '9', '9', '9', '9', '9', '9', '9', '9', '9', '9', '3', '3', '3', '7', '7', '7', '7']

text = " Tanuj contact number is 9999999999, and its US contact number is (999)-333-7777"

#Extracting the indian number

pattern = '\d\d\d\d'

matches = re.findall(pattern, text)

print(matches)

output: ['9999', '9999', '7777']

text = " Tanuj contact number is 9999999999, and its US contact number is (999)-333-7777"

#Extracting the indian number

pattern = '\d{10}'

matches = re.findall(pattern, text)

print(matches)

output: ['9999999999']

text = " Tanuj contact number is 9999999999, and its US contact number is (999)-333-7777"

#Extracting the USA number

pattern = r'\(\d{3}\)-\d{3}-\d{4}'

matches = re.findall(pattern, text)

print(matches)

output: ['(999)-333-7777']

text = " Tanuj contact number is 9999999999, and its US contact number is (999)-333-7777"

#Extracting the both number

pattern = r'\d{10}|\(\d{3}\)-\d{3}-\d{4}'

matches = re.findall(pattern, text)

print(matches)

output: ['9999999999', '(999)-333-7777']

I complete by 1 in the afternoon.

AROUND 2 timming your code:

As I learn python you will discover multiple solutions for a single task and some I tried to find most efficient approach.

Now we will focus on 3 ways of doing this:

Simply tracking time elapsed

Using the timeit module

Special %%timeit “magic” for jupyter Notebooks.

def func\_one(n):

    return[str(num) for num in range(n)]

print(func\_one(10))

def func\_two(n):

    return list(map(str,range(n)))

print(func\_two(10))

import time

# Current time before

start\_time = time.time()

# run code

result = func\_one(100000)

# Current time after running code

end\_time = time.time()

# elapsed time

elapsed\_time = end\_time - start\_time

print(elapsed\_time)

# Current time before

start\_time = time.time()

# run code

result = func\_two(100000)

# Current time after running code

end\_time = time.time()

# elapsed time

elapsed\_time = end\_time - start\_time

print(elapsed\_time)

output

['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']

['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']

0.009997129440307617

0.017612934112548828

import timeit

stmt = '''

func\_one(100)

'''

setup = '''

def func\_one(n):

    return[str(num) for num in range(n)]

'''

print(timeit.timeit(stmt,setup,number=1000000))

stmt2 = '''

func\_two(100)

'''

setup2 = '''

def func\_two(n):

    return list(map(str,range(n)))

'''

print(timeit.timeit(stmt,setup,number=1000000))

output: 9.414344599994365

9.648833000013838

Magic

%%timeit #only for jupyter

func\_one(100)

Unzipping and zipping files.

#f = open("fileone.txt", 'w+')

#f.write("One file")

#f.close()

#f = open("filetwo.txt", 'w+')

#f.write("Two file")

#f.close()

import zipfile

comp\_file = zipfile.ZipFile("comp\_file.zip",'w')

comp\_file.write('filetwo.txt',compress\_type=zipfile.ZIP\_DEFLATED)

comp\_file.close()

zip\_obj = zipfile.ZipFile("comp\_file.zip","r")

zip\_obj.extractall("extracted\_content")

I complete it around 5. After around 7 to 8:30 I had to attend zoom meeting from uic.

After all this I uploaded it on github account

Total sitting over the day – 3

Total study time over the day- approx. 8hr