Module11

October 4, 2020

0.1 Author: Joseph Vargovich

[31]:
```
#import libraries
import numpy as np
import re
```

1 Exercise 1: Regular expression matching descriptions

[19]:
```
#a. Finds a in the string
strings = ['abab', 'ab', 'cb', 'cba', 'cab', 'd', 'cc', 'cfcds']
output = []
for i in strings:
    output.append('a' in i)
print(output)

#b. Finds ab in the string
strings = ['abab', 'ab', 'cb', 'cba', 'cab', 'd', 'cc', 'cfcds']
output = []
for i in strings:
    output.append('ab' in i)
print(output)

#c. This matches strings that contain a or b.
p = re.compile('[ab]')
strings = ['abab', 'ab', 'cb', 'cba', 'cab', 'd', 'cc', 'cfcds']
output = []
for i in strings:
    if(p.match(i)):
        output.append(True)
    else:
        output.append(False)
print(output)

#d. This matches strings that begin with a or b.
p = re.compile('^[ab]')
strings = ['a', 'b', 'cb', 'bc', 'c', 'cd']
output = []
```

```
for i in strings:
    if(p.match(i)):
        output.append(True)
    else:
        output.append(False)
print(output)

#e. Begins with one or more digits and has a space separating them from an a or␣
 ↪A.
p = re.compile('\\d+\\s[aA]')
strings = ['ab', '1a', '1 a', '121221323123434325324 a', 'a 1221', '12 Aa', '12␣
 ↪b']
output = []
for i in strings:
    if(p.match(i)):
        output.append(True)
    else:
        output.append(False)
print(output)


#f.  Begins with one or more digits and has 0 or more spaces seperating digits␣
 ↪from a or A.
p = re.compile('\\d+\\s*[aA]')
strings = ['1212134a', '12231 a', '121323 A', '12213434A', '223443 B', 'aasdf␣
 ↪12', '12 aa']
output = []
for i in strings:
    if(p.match(i)):
        output.append(True)
    else:
        output.append(False)
print(output)

#g. Captures anything as . is a wildcard general capture. Can have any length␣
 ↪due to the Kleene star operator.
p = re.compile('.*')
strings = ['Whatever', 'I', 'Want', '', 'dasfasdfsdffsdafdsfsadafds']
output = []
for i in strings:
    if(p.match(i)):
        output.append(True)
    else:
        output.append(False)
print(output)
```

```
#h. Begins with two alphanumeric characters and ends with bar.
p = re.compile('^\\w{2}bar')
strings = ['aabar', 'dasdfbar', 'bbar', 'bbbar', 'xyz', 'xyzbar', 'xybar']
output = []
for i in strings:
    if(p.match(i)):
        output.append(True)
    else:
        output.append(False)
print(output)

#i. Captures foo.bar OR two alphanumeric chars followed by bar
p = re.compile('(foo\\.bar)|(^\\w{2}bar)')
strings = ['foo.bar', 'foobar', 'aabar', 'aabar', 'bar.foo', 'sdaf', 'gxxxbar',␣
 ↪'gzbar']
output = []
for i in strings:
    if(p.match(i)):
        output.append(True)
    else:
        output.append(False)
print(output)
```

```
[True, True, False, True, True, False, False, False]
[True, True, False, False, True, False, False, False]
[True, True, False, False, False, False, False, False]
[True, True, False, True, False, False]
[False, False, True, True, False, True, False]
[True, True, True, True, False, False, True]
[True, True, True, True, True]
[True, False, False, True, False, False, True]
[True, False, True, True, False, False, False, True]
```

2 Exercise 3: Gettysburg address parsing and average length.

[40]:
```
Gettysburg = """Four score and seven years ago our fathers brought forth on␣
 ↪this
continent, a new nation, conceived in Liberty, and dedicated to the proposition
that all men are created equal.

Now we are engaged in a great civil war, testing whether that nation, or any
nation so conceived and so dedicated, can long endure. We are met on a great
battle-field of that war. We have come to dedicate a portion of that field, as
a final resting place for those who here gave their lives that that nation␣
 ↪might
live. It is altogether fitting and proper that we should do this.
```

```
But, in a larger sense, we can not dedicate -- we can not consecrate -- we can
not hallow -- this ground. The brave men, living and dead, who struggled here,
have consecrated it, far above our poor power to add or detract. The world will
little note, nor long remember what we say here, but it can never forget what
they did here. It is for us the living, rather, to be dedicated here to the
unfinished work which they who fought here have thus far so nobly advanced. It
is rather for us to be here dedicated to the great task remaining before us --
that from these honored dead we take increased devotion to that cause for which
they gave the last full measure of devotion -- that we here highly resolve that
these dead shall not have died in vain -- that this nation, under God, shall
have a new birth of freedom -- and that government of the people, by the␣
 ↪people,
for the people, shall not perish from the earth."""

#Make a list of characters we do not want.
bad_chars = ['.', ',', '?', '!', '-', "\n"]

#Remove our bad characters from any word they appear.
for i in bad_chars:
    Gettysburg = Gettysburg.replace(i,'')

str_list = re.split("\s+", Gettysburg)

filtered_it= filter(lambda item: item != '', str_list )
filtered_list = list(filtered_it)

str_lens = []
#Build our length list
for i in filtered_list:
    str_lens.append(len(i))

#Find the average str length.
print(np.mean(str_lens))
```

```
[4, 5, 3, 5, 5, 3, 3, 7, 7, 5, 2, 4, 9, 1, 3, 6, 9, 2, 7, 3, 9, 2, 3, 11, 4, 3,
3, 3, 7, 8, 2, 3, 7, 2, 1, 5, 5, 3, 7, 7, 4, 6, 2, 3, 6, 2, 9, 3, 2, 9, 3, 4, 6,
2, 3, 3, 2, 1, 5, 11, 2, 4, 3, 2, 4, 4, 2, 8, 1, 7, 2, 4, 5, 2, 1, 5, 7, 5, 3,
5, 3, 4, 4, 5, 5, 4, 4, 6, 5, 4, 2, 2, 10, 7, 3, 6, 4, 2, 6, 2, 7, 2, 1, 6, 5,
2, 3, 3, 8, 2, 3, 3, 10, 2, 3, 3, 6, 4, 6, 3, 5, 3, 6, 3, 4, 3, 9, 4, 4, 11, 2,
3, 5, 3, 4, 5, 2, 3, 2, 7, 3, 5, 4, 6, 4, 3, 4, 8, 4, 2, 3, 4, 3, 2, 3, 5, 6, 4,
4, 3, 4, 2, 2, 3, 2, 3, 6, 6, 2, 2, 9, 4, 2, 3, 10, 4, 5, 4, 3, 6, 4, 4, 4, 3,
2, 5, 8, 2, 2, 6, 3, 2, 2, 2, 4, 9, 2, 3, 5, 4, 9, 6, 2, 4, 4, 5, 7, 4, 2, 4, 9,
8, 2, 4, 5, 3, 5, 4, 4, 3, 4, 4, 7, 2, 8, 4, 2, 4, 6, 7, 4, 5, 4, 5, 3, 4, 4, 2,
4, 4, 4, 6, 5, 3, 5, 4, 1, 3, 5, 2, 7, 3, 4, 10, 2, 3, 6, 2, 3, 6, 3, 3, 6, 5,
3, 6, 4, 3, 5]
4.271375464684015
```

3 Exercise 4 - Creating a variable name regular expression.

[43]:
```
#a. Write a regex to determine if a string starts with a character (upper or␣
 ↪lower case) or underscore and then is followed by zero or more numbers
p = re.compile('^[\\w|\\W|_][0-9a-zA-Z\\._]*$')
strings = ['foo15', 'Bar', '.resid', '_14s',
             '99_Bottles', '.9Arggh', 'Foo!','HIV Rate']
output = []
for i in strings:
    if(p.match(i)):
        output.append(True)
    else:
        output.append(False)
print(output)

#b. Modify regex so the first group can only be [a-zA-Z_]
p = re.compile('^[a-zA-Z_][0-9a-zA-Z\\._]*$')
strings = ['foo15', 'Bar', '.resid', '_14s',
             '99_Bottles', '.9Arggh', 'Foo!','HIV Rate']
output = []
for i in strings:
    if(p.match(i)):
        output.append(True)
    else:
        output.append(False)
print(output)
```

```
[True, True, True, True, True, True, False, False]
[True, True, False, True, False, False, False, False]
```

[ ]:
```

```