Regular Expressions.

Prepared by: Sagun Shakya (https://github.com/sagsshakya)

- GITAM Institute of Science.

Standard Libraries.

- Regular expressions.
  - to validate the Indian mobile.
  - to validate the emailID.
  - to validate the username.
  - to valiate the password.
- Understanding the regular expressions.
  - [0 - 9] -- any digit matching.
  - [a - z] -- any lowercase matching.
  - [A - Z] -- any uppercase matching.
  - cap symbol is used to represent the start of

Function to test the two digit number as input.

In [1]:

```
import re
def twoDigitMatch(n):
    pattern = '^[0-9]{2}$'
    n = str(n)

    if re.match(pattern,n):
        return True
    else:
        return False

twoDigitMatch(136)
```

```
False
```

Validating username for min 5 and max 12 characters.

In [2]:
```
import re

def validateusername(name):
    pattern = '^[a-zA-Z]{4,11}$'
    if re.match(pattern, name):
        return True
    else:
        return False

print(validateusername('Sagun'))
print(validateusername('gitampythnprogramming'))
```

```
True
False
```

Regular Expressions.

- \ Used to drop the special meaning of character following it (discussed below)
- [] Represent a character class
- ^ Matches the beginning
- $ Matches the end
- . Matches any character except newline
- ? Matches zero or one occurrence.
- | Means OR (Matches with any of the characters separated by it.
- Asterisk(*) Any number of occurrences (including 0 occurrences)
- plus(+) One ore more occurrences
- {} Indicate number of occurrences of a preceding RE to match.
- () Enclose a group of REs
    - \d Matches any decimal digit, this is equivalent to the set class [0-9].
    - \D Matches any non-digit character.
    - \s Matches any whitespace character.
    - \S Matches any non-whitespace character
    - \w Matches any alphanumeric character, this is equivalent to the class [a-z
    - \W Matches any non-alphanumeric character.
    - \A - Matches if the specified characters are at the start of a string.
    - \b - Matches if the specified characters are at the beginning or end of a wo
    - \B - Opposite of \b. Matches if the specified characters are not at the begin
    - \Z - Matches if the specified characters are at the end of a string.

In [1]:
```
import re
```

In [2]:
```
#digits only
p = re.compile('\d')
print(p.findall('I went to bed at 10 a.m yesterday on 20th September.'))
```

```
['1', '0', '2', '0']
```

In [3]:
```
#digits in groups
q = re.compile('\d+')
print(q.findall('I went to bed at 10 a.m yesterday on 20th September.'))
```

```
['10', '20']
```

In [4]:
```
#alphanumeric characters.
r = re.compile(('\w'))
print(r.findall('SUper mario bros 1990 still alive 2123 $100'))
```

```
['S', 'U', 'p', 'e', 'r', 'm', 'a', 'r', 'i', 'o', 'b', 'r', 'o', 's', '1', '9', '9', '0', 's', 't', 'i
'e', '2', '1', '2', '3', '1', '0', '0']
```

In [5]:
```
#alphanumeric characters in groups.
p = re.compile('\w+')
print(p.findall('SUper mario bros 1990 still alive 2123 $100'))
```

```
['SUper', 'mario', 'bros', '1990', 'still', 'alive', '2123', '100']
```

In [8]:
```
#non - alphanumeric characters.
q = re.compile('\W')
print(q.findall('SUper mario bros 1990 still alive 2123 $100. @#'))
```

```
[' ', ' ', ' ', ' ', ' ', ' ', ' ', '$', '.', ' ', '@', '#']
```

In [9]:
```
#non - alphanumeric characters in groups.
q = re.compile('\W+')
print(q.findall('SUper mario bros 1990 still alive 2123 $100. @#'))
```

```
[' ', ' ', ' ', ' ', ' ', ' ', ' $', '. @#']
```

In [11]:
```
#'a' followed by any number of 'bb.
p = re.compile('ab*')
print(p.findall('abaababababbababababababababababababbababababab'))
print(p.findall('abb abbb aabb'))
```

```
['ab', 'a', 'ab', 'ab', 'ab', 'abb', 'ab', 'ab', 'ab', 'ab', 'ab', 'ab', 'ab', 'ab', 'ab', 'ab', 'abb',
['abb', 'abbb', 'a', 'abb']
```

In [12]:
```
#Five letters in a given word which starts with 'a' and ends with 's'.
def check(word):
    pattern = '^a...s$'
    result = re.match(pattern, word)

    if result:
        return True
    else:
        return False

check('alias')
```

```
True
```

Function split().

- Syntax:
  - re.split(pattern, string, maxsplit=0, flags=0)

In [13]:
```
import re
from re import split
```

In [14]:
```
'''
Function split()
Split string by the occurrences of a character or a pattern,
upon finding that pattern, the remaining characters from the string are returned
list.

'''


splittedList = split('\W+', 'blah blah blah nah nah nah $ lo - lol ^ty')
print(splittedList)
```

```
['blah', 'blah', 'blah', 'nah', 'nah', 'nah', 'lo', 'lol', 'ty']
```

In [15]:
```
# 'Boy' and 'boy' will be treated same when flags = re.IGNORECASE
# SPlit the string variable when it finds any letters between a and f.
print(re.split('[a-f]+', 'Aey, Boy oh boy, come here', flags = re.IGNORECASE))
print(re.split('[a-f]+', 'Aey, Boy oh boy, come here'))
```

```
['', 'y, ', 'oy oh ', 'oy, ', 'om', ' h', 'r', '']
['A', 'y, Boy oh ', 'oy, ', 'om', ' h', 'r', '']
```

Function sub().

- Syntax:
  - re.sub(pattern, replace, string, count=0, flags=0)

In [16]:
```
# Regular Expression pattern 'ub' matches the string at "Subject" and "Uber".
# As the CASE has been ignored, using Flag, 'ub' should match twice with the stri
# Upon matching, 'ub' is replaced by '~*' in "Subject", and in "Uber", 'Ub' is re
print(re.sub('ub', '~*' , 'Subject has Uber booked already', flags = re.IGNORECAS
```

```
S~*ject has ~*er booked already
```

In [17]:
```
# Consider the Case Senstivity, 'Ub' in "Uber", will not be reaplced.
print(re.sub('ub', '~*' , 'Subject has Uber booked already'))
```

```
S~*ject has Uber booked already
```

In [18]:
```
# As count has been given value 1, the maximum times replacement occurs is 1
print(re.sub('ub', '~*' , 'Subject has Uber booked already', count=1, flags = re.
```

```
S~*ject has Uber booked already
```

In [19]:
```
# 'r' before the patter denotes RE, \s is for start and end of a String.
print(re.sub(r'\sAND\s', ' & ', 'Baked Beans And Spam', flags=re.IGNORECASE))
```

```
Baked Beans & Spam
```

Defining a function that checks if a pattern matches a word.

In [26]:
```
import re
def check(pattern, mystr):
    result = re.match(pattern, mystr)
    p = re.compile(pattern)

    if result:
        print(p.findall(mystr))
        return True

    else:
        return False
```

In [27]:
```
#any five letter string starting with a and ending with s.
p = '^a...s$'
print(check(p, 'alias'))
print(check(p, 'aliaaas'))
```

```
['alias']
True
False
```

In [29]:
```
#The star symbol * matches zero or more occurrences of the pattern left to it.
p1 = 'ma*n'
a11 = 'maaaain'             #No because of 'i'.
a12 = 'maaaaaan'            #Match.

print(check(p1, a11), '\n')
print(check(p1, a12))
```

```
False

['maaaaaan']
True
```

In [33]:
```
#The plus symbol + matches one or more occurrences of the pattern left to it.
p2 = 'ma+n'
a21 = 'maaaain'             #No.
a22 = 'maman'            # no Match.

print(check(p2, a21), '\n')
print(check(p2, a22))
```

```
False

False
```

In [34]:
```
#The question mark symbol ? matches zero or one occurrence of the pattern left tc
p3 = 'ma?n'
a31 = 'maaaan'              #No.
a32 = 'man'                 #Match.

print(check(p3, a31), '\n')
print(check(p3, a32))
```

```
False

['man']
True
```

In [36]:
```
#This RegEx [0-9]{2, 4} matches at least 2 digits but not more than 4 digits.
p41 = '[0-9]{2,4}'
#Same can be done with the following.
p42 = '\d{2,4}'
a41 = '9841sagun123asddd5df34sqqq456789'  #3 matches   9841, 123, 34

print(check(p41, a41), '\n')
print(check(p42, a41))
```

```
['9841', '123', '34', '4567', '89']
True

['9841', '123', '34', '4567', '89']
True
```

In [37]:
```
#Alternation (OR operator).
p51 = 'a|b'
a51 = 'winds of change blows.'  #2 matches.

print(check(p51, a51), '\n')
```

```
False
```

In [38]:
```
#Parentheses () is used to group sub-patterns.
# For example, (a|b|c)xz match any string that matches either a or b or c followe

#The following reg ex validates that the first four numbers are either 9841 or 98
#followed by any six digits.
p52 = '(9841|9808)*\d{6}'
a52 = '9841372490'  #match
a53 = '9808666666'  #match
a54 = '9803702365'  #no match
print(check(p52, a52), '\n')
print(check(p52, a53), '\n')
print(check(p52, a54))
```

```
['9841']
True

['9808']
True

['']
True
```

1. Write a Python program to check that a string contains o characters (in this case a-z, A-Z and 0-9).

In [41]:
```
import re
str = input('Enter a string: ')

pattern = '[0-9]|[a-z]|[A-Z]'    #Also we can use '\w'.
result = re.match(pattern,str)
if result:
    print(True)
else:
    print(False)
```

```
Enter a string: %^%$
False
```

The End.