Regular Expressions & Functional Programming

Day13 Objectives

- Regular Expessions Contd.
- Functional Programming/ Comprehensions
  - Lists
  - Dictionary
  - Set
  - Tuple/ Generator
- Special Functions
  - lambda
  - map()
  - filter()

Patterns for RegEx:

- `^` - StartsWith
- `$` - EndsWith
- `*` - 0 or more
- `\d` - digits
- `\D` - NotDigits
- `\w` - Alphabets
- `\W` - NotAlphabets
- `\s` - Space
- `\S` - NotSpace
- `+` -
- `?` -
- `.` - anyCharacter
- [] - GroupOfCharacters
  - [A-Z] - UpperCase
  - [a-z] - LowerCase
  - [a-zA-z] - Lower& UpperCase
  - [aeiou] - Vowels
  - [^aeiou] - notVowels
  - [asdf]

    - [0-5] - 0-5 digts
- {min, max} - min-max times matches
    - {min} - min times matches
- \ Escape Sequence
    - \. - matching dot
    - * - matching star

In [2]:

```
data = """5enwRaNY41XqvxN
djC43vaYLzjO3F7
CAMLyxgtJqvdEiW
wYlj4kZnLIzIQxG
3vPQxwo0FIdiou7
jB7q0OhmSMXQLRe
QKmktCMos0cVOgG
z6i02Zt8HBiQCCD
uT4AUIkmYTrzGG0
xYghC09wT4y5Z9K""".split()

data
```

Out[2]:

```
['5enwRaNY41XqvxN',
 'djC43vaYLzjO3F7',
 'CAMLyxgtJqvdEiW',
 'wYlj4kZnLIzIQxG',
 '3vPQxwo0FIdiou7',
 'jB7q0OhmSMXQLRe',
 'QKmktCMos0cVOgG',
 'z6i02Zt8HBiQCCD',
 'uT4AUIkmYTrzGG0',
 'xYghC09wT4y5Z9K']
```

In [4]:

```
pattern = r'^\d'

import re
for string in data:
    print(re.findall(pattern, string))
```

```
['5']
[]
[]
[]
['3']
[]
[]
[]
[]
[]
```

In [5]:

```
pattern = r'^.\d'

for string in data:
    print(re.findall(pattern, string))
```

```
[]
[]
[]
[]
[]
[]
[]
['z6']
[]
[]
```

In [6]:

```
pattern = r'.\d'

for string in data:
    print(re.findall(pattern, string))
```

```
['Y4']
['C4', 'O3', 'F7']
[]
['j4']
['o0', 'u7']
['B7', 'q0']
['s0']
['z6', 'i0', 't8']
['T4', 'G0']
['C0', 'T4', 'y5', 'Z9']
```

In [7]:

```
pattern = r'[A-Z]\d'

for string in data:
    print(re.findall(pattern, string))
```

```
['Y4']
['C4', 'O3', 'F7']
[]
[]
[]
['B7']
[]
[]
['T4', 'G0']
['C0', 'T4', 'Z9']
```

In [9]:

```
pattern = r'\w*[A-Z]\d'

for string in data:
    print(re.findall(pattern, string))
```

```
['5enwRaNY4']
['djC43vaYLzjO3F7']
[]
[]
[]
['jB7']
[]
[]
['uT4AUIkmYTrzGG0']
['xYghC09wT4y5Z9']
```

In [10]:

```
pattern = r'[a-z]*[A-Z]\d'

for string in data:
    print(re.findall(pattern, string))
```

```
['Y4']
['djC4', 'zjO3', 'F7']
[]
[]
[]
['jB7']
[]
[]
['uT4', 'G0']
['ghC0', 'wT4', 'Z9']
```

In [11]:

```
pattern = r'[a-z].[A-Z]\d'

for string in data:
    print(re.findall(pattern, string))
```

```
['aNY4']
['djC4', 'zjO3']
[]
[]
[]
[]
[]
[]
['zGG0']
['ghC0', 'y5Z9']
```

In [16]:

```
pattern = r'[A-Z]..[A-Z]\d'

for string in data:
    print(re.findall(pattern, string))
```

```
['RaNY4']
['LzjO3']
[]
[]
[]
[]
[]
[]
[]
['YghC0']
```

In [15]:

```
pattern = r'[A-Z]+..[A-Z]\d'

for string in data:
    print(re.findall(pattern, string))
```

```
['RaNY4']
['YLzjO3']
[]
[]
[]
[]
[]
[]
[]
['YghC0']
```

In [17]:

```
pattern = r'[A-Z]?..[A-Z]\d'

for string in data:
    print(re.findall(pattern, string))
```

```
['RaNY4']
['djC4', 'LzjO3']
[]
[]
[]
[]
[]
[]
['zGG0']
['YghC0', '9wT4', 'y5Z9']
```

In [18]:
```
pattern = r'[a-z]\w{3,5}[A-Z]\d'

for string in data:
    print(re.findall(pattern, string))
```

```
['enwRaNY4']
['vaYLzjO3']
[]
[]
[]
[]
[]
[]
['mYTrzGG0']
['xYghC0', 'wT4y5Z9']
```

Given IPV4 address is valid or not

In [19]:
```
ipv4 = "172.168.255.237"

pattern = r'\d\d\d\.\d\d\d\.\d\d\d\.\d\d\d'

re.match(pattern, ipv4)
```

Out[19]: `<re.Match object; span=(0, 15), match='172.168.255.237'>`

In [21]:
```
ipv4 = "172.168.255.27"

pattern = r'\d\d\d\.\d\d\d\.\d\d\d\.\d\d\d'

print(re.match(pattern, ipv4))
```

```
None
```

Pattern for finding given Indian Mobile Number is valid or not

valid num1

- 10 Digits
- [6-9] - starting with

Valid num2

- +91 started with
- 10 Digits
- [6-9] - starting with

Valid Num3

- 0 startswith

- [6-9] - starting with
- 10 Digits

In [22]:
```
pattern = '^[6-9]\d{9}$'
pattern2 = '^[6789][0-9]\d\d\d[0-9][0-9]\d\d\d$'
```

In [24]:
```
print(re.match(pattern, input()))
print(re.match(pattern2, input()))
```

```
9876543210
<re.Match object; span=(0, 10), match='9876543210'>
987654321o
None
```

In [37]:
```
pattern='^[+91][6-9]\d{10}$'
# [+91] - first char of any +91
# [6-9] - second char b/w 6-9
# \d{10} - 10 digits
```

In [28]:
```
print(re.match(pattern, input()))
```

```
+9876543210
None
```

In [29]:
```
print(re.match(pattern, input()))
```

```
+919876543210
None
```

In [30]:
```
print(re.match(pattern, input()))
```

```
+9198765432101
None
```

In [34]:
```
print(re.match(pattern, input()))
```

```
+69876543210
None
```

In [38]:
```
print(re.match(pattern, input()))
```

```
+99876543210
<re.Match object; span=(0, 12), match='+99876543210'>
```

M2

In [39]:

```
1 pattern = '\+91[6-9]\d{9}'
2
3 print(re.match(pattern, input()))
```

```
+919876543210
<re.Match object; span=(0, 13), match='+919876543210'>
```

In [41]:

```
1 pattern = '^\+91[6-9]\d{9}$'
2
3 print(re.match(pattern, input()))
```

```
+9198765432101
None
```

In [42]:

```
1 pattern = '^\+91[6-9]\d{9}$'
2
3 print(re.match(pattern, input()))
```

```
+91987654321s
None
```

In [44]:

```
1 p = '^(0)[6-9]\d{9}$'
2 print(re.match(p, input()))
```

```
09876543210
<re.Match object; span=(0, 11), match='09876543210'>
```

In [47]:

```
1 p1 = '^0[6-9]\d{10}$'
2 print(re.match(p1, input()))
```

```
09876543210
None
```

In [48]:

```
1 p1 = '^0[6-9]\d{9}$'
2 print(re.match(p1, input()))
```

```
09876543210
<re.Match object; span=(0, 11), match='09876543210'>
```

In [49]:

```
1 pattern1='^\(0)[6-9]\d{9}$'
2
3 print(re.match(pattern1, input()))
```

```
(09876543210

---------------------------------------------------------------------------
error                                     Traceback (most recent call last)
<ipython-input-49-45bfaf105f81> in <module>
      1 pattern1='^\(0)[6-9]\d{9}$'
      2
----> 3 print(re.match(pattern1, input()))

~\anaconda3\lib\re.py in match(pattern, string, flags)
    189     """Try to apply the pattern at the start of the string, returni
ng
    190     a Match object, or None if no match was found."""
--> 191     return _compile(pattern, flags).match(string)
    192
    193 def fullmatch(pattern, string, flags=0):

~\anaconda3\lib\re.py in _compile(pattern, flags)
    302     if not sre_compile.isstring(pattern):
    303         raise TypeError("first argument must be string or compiled
 pattern")
--> 304     p = sre_compile.compile(pattern, flags)
    305     if not (flags & DEBUG):
    306         if len(_cache) >= _MAXCACHE:

~\anaconda3\lib\sre_compile.py in compile(p, flags)
    762     if isstring(p):
    763         pattern = p
--> 764         p = sre_parse.parse(p, flags)
    765     else:
    766         pattern = None

~\anaconda3\lib\sre_parse.py in parse(str, flags, state)
    960     if source.next is not None:
    961         assert source.next == ")"
--> 962         raise source.error("unbalanced parenthesis")
    963
    964     if flags & SRE_FLAG_DEBUG:

error: unbalanced parenthesis at position 4
```

In [50]:

```
1 pattern1='^\(0[6-9]\d{9}$'
2
3 print(re.match(pattern1, input()))
```

```
(09876543210
<re.Match object; span=(0, 12), match='(09876543210'>
```

In [51]:
```
pattern1='^[0][6-9]\d{9}$'

print(re.match(pattern1, input()))
```

```
09876543210
<re.Match object; span=(0, 11), match='09876543210'>
```

In [52]:
```
pattern1='^[/+][9][1][6-9]\d{9}$'

print(re.match(pattern1, input()))
```

```
+919876543210
<re.Match object; span=(0, 13), match='+919876543210'>
```

In [53]:
```
mobilePattern = r'^[/+][9][1][6-9]\d{9}$|^[0][6-9]\d{9}$|^[6-9]\d{9}$'

re.match(mobilePattern, input())
```

```
+919876543210
```

Out[53]:
```
<re.Match object; span=(0, 13), match='+919876543210'>
```

In [54]:
```
mobilePattern = r'^[/+][9][1][6-9]\d{9}$|^[0][6-9]\d{9}$|^[6-9]\d{9}$'

re.match(mobilePattern, input())
```

```
9876543210
```

Out[54]:
```
<re.Match object; span=(0, 10), match='9876543210'>
```

In [56]:
```
mobilePattern = r'^[/+][9][1][6-9]\d{9}$|^[0][6-9]\d{9}$|^[6-9]\d{9}$'
mobile = input()
if re.match(mobilePattern, mobile):
    print(mobile)
else:
    print("Invalid")
```

```
+91aps9876543210
Invalid
```

Task: Find the email is valid or not

- Email Id contains: username@domain.extension (mailto:username@domain.extension)
  - UserName
  - @
  - Domain (gmail, hotmail, srkit, apssdc, w3schools)
  - .
  - Extension - (in, us, gov, com, org, ngo, ml, ai)
- UserName:
  - must starts and contain lowercase/uppercase lettter
  - no special characters except . , _ but should not ends

    - digits
    - {6 - 18} characters
- @
- Domain
    - {3 - 18} characters
    - it can contain Lower/upper/numbers
- .
- Extension
    - 2 - 8
    - it can contain Lower/upper