Regex -

The Regex or Regular Expression is a way to define a pattern for searching or manipulating strings. We can use a regular expression to match, search, replace, and manipulate inside textual data.

Raw String

In [1]:
```
print('kiran')
print(r'kiran') # we can use r or R
print(r'kiran'=='kiran')
```

```
kiran
kiran
True
```

In [2]:
```
print(r'the \n the')
```

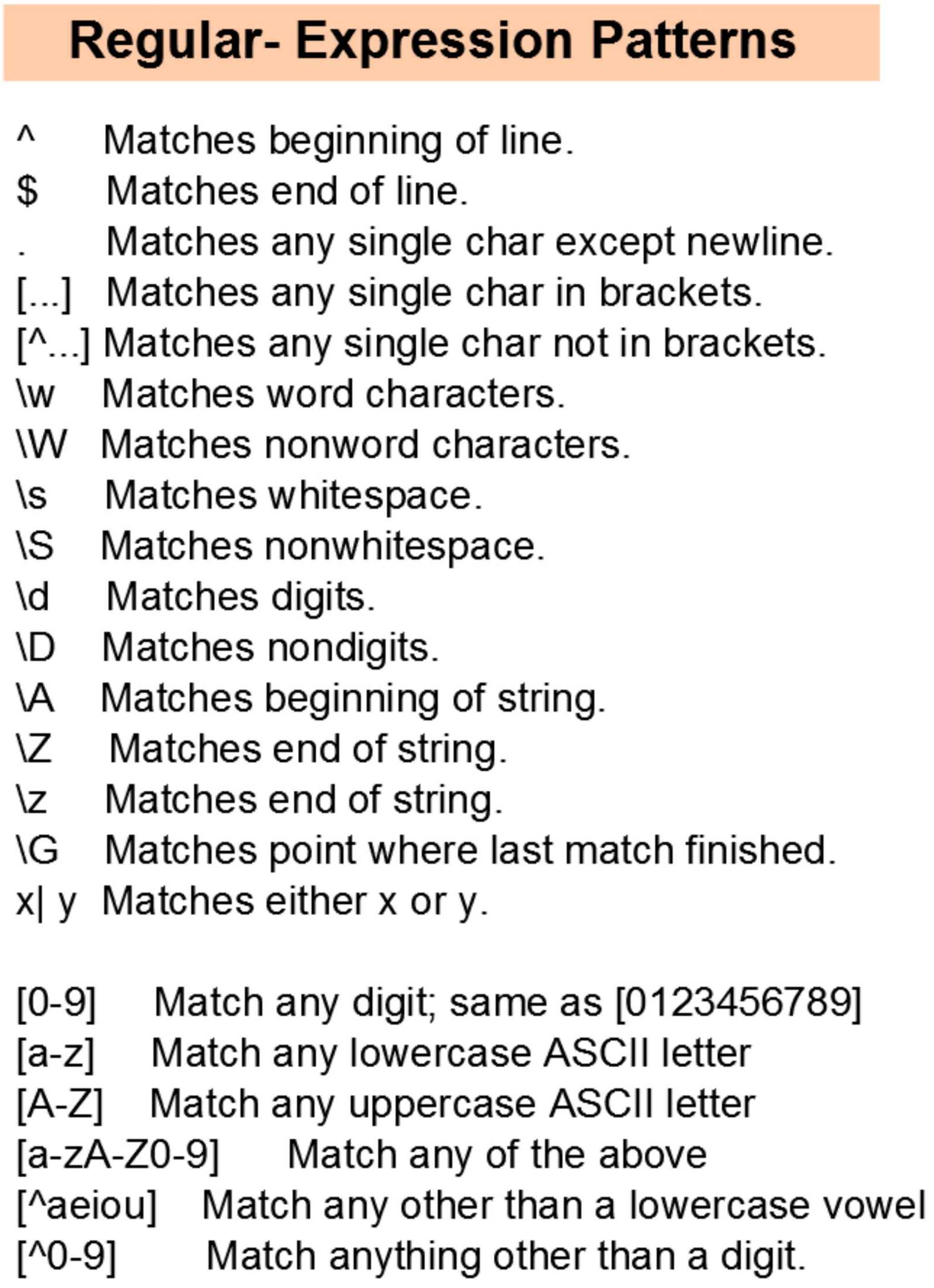
```
the \n the
```

In [3]:
```
print('the \n the')
```

```
the
 the
```

In [4]:
```
print(ord("A"))
print(ord("Z"))
print(ord('a'))
print(ord('z'))
```

```
65
90
97
122
```



Regular- Expression Patterns

^	Matches beginning of line.
$	Matches end of line.
.	Matches any single char except newline.
[...]	Matches any single char in brackets.
[^...]	Matches any single char not in brackets.
\w	Matches word characters.
\W	Matches nonword characters.
\s	Matches whitespace.
\S	Matches nonwhitespace.
\d	Matches digits.
\D	Matches nondigits.
\A	Matches beginning of string.
\Z	Matches end of string.
\z	Matches end of string.
\G	Matches point where last match finished.
x| y	Matches either x or y.

[0-9]	Match any digit; same as [0123456789]
[a-z]	Match any lowercase ASCII letter
[A-Z]	Match any uppercase ASCII letter
[a-zA-Z0-9]	Match any of the above
[^aeiou]	Match any other than a lowercase vowel
[^0-9]	Match anything other than a digit.

Regex Metacharacters and Operators

Find all

re.findall() method scans the regex pattern through the entire target string and returns all the matches that were found in the form of a list.

In [5]:
```
import re
```

In [6]:
```
s = '''
<html>
<head>
<title>Current IP Address Allocations
</title>
</head>
<body>
IP Address are 172.45.78.109
LoopBack Address: 127.0.0.1
Computer 1: 10.67.89.101
Computer 2: 11.67.98.102
Computer 3: 12.68.98.102
</body>
</html>
'''
```

In [7]:
```
ip_s=re.findall(r'\d{1,3}.\d{1,3}.\d{1,3}.\d{1,3}',s)
# 123.4.55.66
print(f'ip address are -: {ip_s}')
```

```
ip address are -: ['172.45.78.109', '127.0.0.1', '10.67.89.101', '11.67.98.102', '12.
68.98.102']
```

In [8]:
```
### 10  Or 11
ip_s1=re.findall(r"1[0-1]\.\d{1,3}\.\d{1,3}\.\d{1,3}", s)
print(f'ip address are -: {ip_s1}')
```

```
ip address are -: ['10.67.89.101', '11.67.98.102']
```

In [9]:
```
print("Find all matches for format Month day")

matches = re.findall(r"[A-Z][a-z]+\s\d{1,2}","These are the match dates June 24, Augus
print(f'gives Month Date format - {matches}')
```

```
Find all matches for format Month day
gives Month Date format - ['June 24', 'August 9', 'Dec 12']
```

In [10]:
```
s = "purple alice@google.com abcde helloab@abc.com ---@gmail.com 23@gmail.com my23@gma

emails = re.findall(r"\w+@\w+\.\w+", s)
print(emails)
```

```
['alice@google.com', 'helloab@abc.com', '23@gmail.com', 'my23@gmail.com', '_@gmail.co
m']
```

In [11]:
```
s = "purple alice@google.com abcde helloab@abc.com ---@gmail.com 23@gmail.com my23@gma

emails = re.findall(r"[A-Za-z]+@\w+\.\w+", s)
print(f'starts with alphabets only {emails}')
```

```
# \w => A-Za-z0-9_
```

```
starts with alphabets only ['alice@google.com', 'helloab@abc.com']
```

In [12]:
```
s = "purple alice@google.com abcde helloab@abc.com ---@gmail.com 23@gmail.com my23@gma

emails = re.findall(r"[\d\w]+@\w+\.\w+", s)
print(f'starts with alphabets only {emails}')

# \w => A-Za-z0-9_
```

```
starts with alphabets only ['alice@google.com', 'helloab@abc.com', '23@gmail.com', 'm
y23@gmail.com', '_@gmail.com']
```

In [13]:
```
# findall - digit 0ne or more
new_st2 = 'Friend in need is 23 friend in 453214 deed'
nr6 = re.findall('\d+',new_st2)
print(nr6)
```

```
['23', '453214']
```

search

The re.search() returns only the first match to the pattern from the target string

In [18]:
```
target_string = "Em is a Python developer \n Emma also knows ML and AI"

result = re.search(r"\b\w{4}\b", target_string)
print(result)
print(result.group())
```

```
<re.Match object; span=(27, 31), match='Emma'>
Emma
```

In [19]:
```
str1 = "Emma is a Python developer \nEmma also knows ML and AI"
# dollar sign($) to match at the end of the string
result = re.search(r"\w{2}$", str1)
print(result.group())
```

```
AI
```

In [20]:
```
source_str  = 'we need to inform him with the latest information'

info = re.search('inform', source_str)
info
```

Out[20]:
```
<re.Match object; span=(11, 17), match='inform'>
```

In [21]:
```
randomstr = 'here is \\kane'

print(randomstr)

re.search(r'\\kane',randomstr)
```

```
here is \kane
```

Out[21]:
```
<re.Match object; span=(8, 13), match='\\kane'>
```

Compile

The re.compile() method changed the string pattern into a re.Pattern object that we can work upon.

```
In [22]: a = 'hat mat rat pat '

         reg = re.compile('[r]at')
         reg
```

```
Out[22]: re.compile(r'[r]at', re.UNICODE)
```

```
In [23]: rplce = reg.sub('FOOD',a)
         rplce
```

```
Out[23]: 'hat mat FOOD pat '
```

```
In [24]: #replacing

         rplc = re.sub('rat','FOOD',a)
         rplc
```

```
Out[24]: 'hat mat FOOD pat '
```

working with white spaces

```
In [25]: chelsea = '''keep the blue flag
         flying high
         chelsa
         '''
         chelsea
```

```
Out[25]: 'keep the blue flag\nflying high\nchelsa\n'
```

```
In [26]: new_str = re.sub('\n',' ',chelsea)
         new_str
```

```
Out[26]: 'keep the blue flag flying high chelsa '
```

```
In [27]: # other method using compile

         comp = re.compile('\n')

         new =comp.sub(' ',chelsea)
         new
```

```
Out[27]: 'keep the blue flag flying high chelsa '
```

- \b : backspace
- \f : formfeed
- \r: carriage return

- \t: tab
- \v:vertical

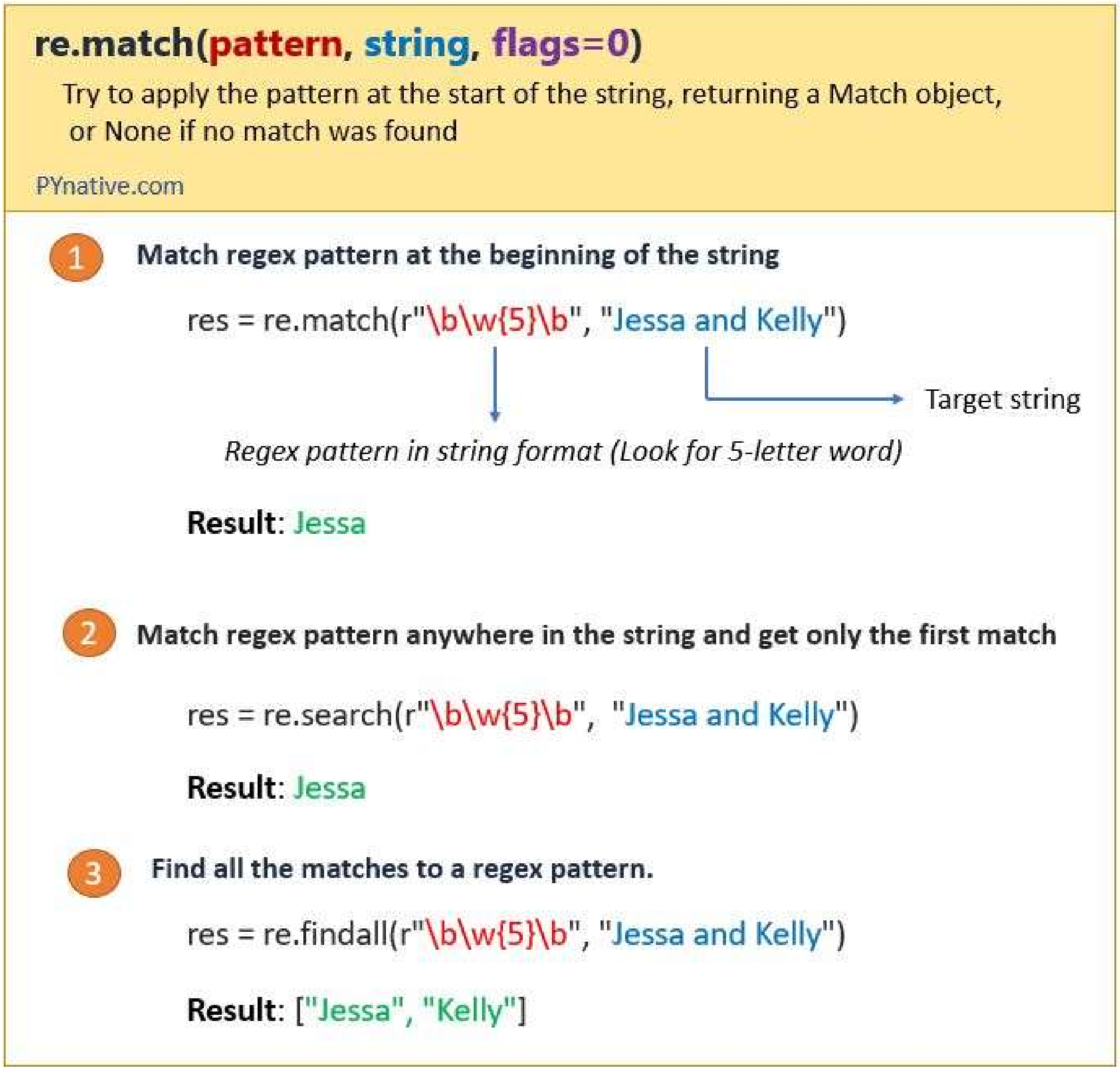
In [28]:
```
phone_no = '''
444-122-1234
123-122-78999
111-123-23
67-7890-2019
'''

reg = re.findall(r'\b\d{2}\b\-\d{4}\-\b\d{4}\b',phone_no)
reg
```

Out[28]:
```
['67-7890-2019']
```

Match

re.match() method looks for the regex pattern only at the beginning of the target string and returns match object if match found; otherwise, it will return None.



re.match(pattern, string, flags=0)

Try to apply the pattern at the start of the string, returning a Match object, or None if no match was found

PYnative.com

1 **Match regex pattern at the beginning of the string**

```
res = re.match(r"\b\w{5}\b", "Jessa and Kelly")
```

Result: Jessa

2 **Match regex pattern anywhere in the string and get only the first match**

```
res = re.search(r"\b\w{5}\b",  "Jessa and Kelly")
```

Result: Jessa

3 **Find all the matches to a regex pattern.**

```
res = re.findall(r"\b\w{5}\b", "Jessa and Kelly")
```

Result: ["Jessa", "Kelly"]

In [30]:
```
import re

target_string = "kiran loves Python and pandas fives"

pattern = r"\b\w{5}\b"

# # match() method
result = re.match(pattern, target_string)
print(result)


# search() method
result = re.search(pattern, target_string)
print(result.group())


# findall() method
result = re.findall(pattern, target_string)
print(result)
```

```
<re.Match object; span=(0, 5), match='kiran'>
kiran
['kiran', 'loves', 'fives']
```

In [ ]:

In [ ]:
```
 # \+91\s\d{10}$
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: