[]
Used to indicate a set of characters.

- Characters can be listed individually, e.g. [amk] will match 'a', 'm', or 'k'.
- Ranges of characters can be indicated by giving two characters and separating them by a '-', for example [a-z] will match any lowercase ASCII letter, [0-5][0-9] will match all the two-digits numbers from 00 to 59, and [0-9A-Fa-f] will match any hexadecimal digit. If - is escaped (e.g. [a\-z]) or if it's placed as the first or last character (e.g. [-a] or [a-]), it will match a literal '-'.

Example:

```
import re
def main():
    names_list=['naresh','kishore','ramesh','rajesh','kiran','raman','suresh']
    for name in names_list:
        m=re.search('^[nk]',name)
        if m!=None:
            print(name)

    for name in names_list:
        m=re.search('^[nk].*[hn]$',name)
        if m!=None:
            print(name)

main()
```

Output:

```
naresh
kishore
kiran
naresh
kiran
>>>
```

Example:

```
import re
def main():
    str1="current date is 17-01-2022"
```

```
    print(str1)
    m=re.search(r'([0-9]{2})-([0-9]{2})-([0-9]{4})',str1)
    if m!=None:
        print(m)
        print(m.group(0))
        print(m.group(1))
        print(m.group(2))
        print(m.group(3))

main()
```

Output:

```
current date is 17-01-2022
<re.Match object; span=(16, 26), match='17-01-2022'>
17-01-2022
17
01
2022
>>>
```

Search pattern is group(0), this group can divided into number of sub groups using (), each group is having unique number which starts at 1

Example:

```
import re
def main():
    email="email id is naresh@nareshit.com"
    m=re.search(r'([a-z]+)@([a-z]+)\.([a-z]{3})',email)
    if m!=None:
        print(m.group(0))
        print(m.group(1))
        print(m.group(2))
        print(m.group(3))

main()
```

Output:

```
naresh@nareshit.com
naresh
nareshit
```

com
>>>

The special sequences consist of '\' and a character from the list below

\A
Matches only at the start of the string.
\b
Matches the empty string, but only at the beginning or end of a word

Example:

```
import re
def main():
    str1="a aa aaa aaaa"
    l=re.findall(r'\baa\b',str1)
    print(l)

main()
```

Output:
['aa']
>>>

Example:

```
import re
def main():
    str1=input("enter any string")
    s=input("enter search pattern")
    pattern=re.compile(s)
    l=pattern.findall(str1)
    print(l)

main()
```

Output:

```
enter any stringpython jpython ironpython rpython
enter search patternpython
['python', 'python', 'python', 'python']
>>>
```

```
========== RESTART: C:/Users/user/Desktop/python6pm/py224.py
==========
enter any stringpython jpython ironpython rpython
enter search pattern\bpython\b
['python']
>>>
```

\B

Matches the empty string, but only when it is *not* at the beginning or end of a word. This means that r'py\B' matches 'python', 'py3', 'py2', but not 'py', 'py.', or 'py!'. \B is just the opposite of \b

Example:

```
import re
def main():
    str1="python py2 py3"
    l=re.findall(r'py\B',str1)
    print(l)
    str2="py py. py!"
    l=re.findall(r'py\B',str2)
    print(l)
    l=re.findall(r'py\b',str2)
    print(l)

main()
```

Output:

```
['py', 'py', 'py']
[]
['py', 'py', 'py']
>>>
```

\d

Matches any decimal digit; this is equivalent to [0-9].

```
import re
def main():
    str1="current date is 17-01-2022"
    m=re.search(r'\b(\d{2})-(\d{2})-(\d{4})\b',str1)
    if m!=None:
```

```
        print(m.group(0))
        print(m.group(1))
        print(m.group(2))
        print(m.group(3))

main()
```
Output:
```
17-01-2022
17
01
2022
>>>
```

\D

Matches any character which is not a decimal digit. This is the opposite of \d.

Example:
```
import re
def main():
    name=input("enter any name")
    m=re.search('^\D',name)
    if m!=None:
        print(f'{name} is valid')
    else:
        print(f'{name} should not start with digit')

main()
```

Output:
```
enter any namenaresh
naresh is valid
>>>
```

\s

Matches characters considered whitespace in the ASCII character set; this is equivalent to [\t\n\r\f\v].

```
[a-z]+\s [a-z]+
```