[]
Used to indicate a set of characters. In a set:

Characters can be listed individually, e.g. [amk] will match 'a', 'm', or 'k'.

Example:

```
import re

names=["ramesh","naresh","kiran","raman","rajesh","kishore"]
for name in names:
    m=re.search(r'^[nk]',name)
    if m!=None:
        print(name)
for name in names:
    m=re.search(r'[en]$',name)
    if m!=None:
        print(name)
```

Output

```
naresh
kiran
kishore
kiran
raman
kishore
```

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

```
>>> str1="python 3.13"
>>> import re
>>> A=re.findall(r'[a-z]',str1)
>>> print(A)
['p', 'y', 't', 'h', 'o', 'n']
>>> B=re.findall(r'[0-9]',str1)
>>> print(B)
['3', '1', '3']
```

```
>>> C=re.findall(r'[a-z0-9]',str1)
>>> print(C)
['p', 'y', 't', 'h', 'o', 'n', '3', '1', '3']
```

(?P<name>...)
Similar to regular parentheses, but the substring matched by the group is accessible via the symbolic group name *name*

(...)
Matches whatever regular expression is inside the parentheses, and indicates the start and end of a group;

Example:

```
import re

str1="current date is 17-06-2025"
str2="current time is 10:40:50"
m1=re.search(r'(?P<dd>[0-9]{2})-(?P<mm>[0-9]{2})-(?P<yy>[0-9]{4})',str1)
m2=re.search(r'(?P<hh>[0-9]{2}):(?P<mm>[0-9]{2}):(?P<ss>[0-9]{2})',str2)
print(m1)
print(m2)
print(m1.group(0))
print(m2.group(0))
print(m1.group(1),m1.group(2),m1.group(3))
print(m2.group(1),m2.group(2),m2.group(3))
print(m1.group('dd'),m1.group('mm'),m1.group('yy'))
print(m2.group('hh'),m2.group('mm'),m2.group('ss'))
```

Output

```
<re.Match object; span=(16, 26), match='17-06-2025'>
<re.Match object; span=(16, 24), match='10:40:50'>
17-06-2025
10:40:50
17 06 2025
10 40 50
17 06 2025
10 40 50
```

Special characters lose their special meaning inside sets. For example, [(+*)] will match any of the literal characters '(', '+', '*', or ')'.

```
import re

str1="def fun1():"
m=re.fullmatch(r'^def ([a-zA-Z_]+[0-9]*[(]{1}[)]{1}:$)',str1)
print(m)
print(m.group(0))
print(m.group(1))
```

Output

```
<re.Match object; span=(0, 11), match='def fun1():'>
def fun1():
fun1():
```

|

A|B, where *A* and *B* can be arbitrary REs, creates a regular expression that will match either *A* or *B*.

Example:

```
import re


str1="current date is 12/06/2025 and 10:30:40"
A=re.findall(r'[0-9]{2}/[0-9]{2}/[0-9]{4}|[0-9]{2}:[0-9]{2}:[0-9]{2}',str1)
print(A)
```

Output

```
['12/06/2025', '10:30:40']
```

Character classes

\A

Matches only at the start of the string.

\Z

Matches only at the end of the string.

```
>>> str1="python"
```

```
>>> m=re.search(r'\Apy',str1)
>>> print(m)
<re.Match object; span=(0, 2), match='py'>
>>> m=re.search(r'on\Z',str1)
>>> print(m)
<re.Match object; span=(4, 6), match='on'>
```

\b

Matches the empty string, but only at the beginning or end of a word. A word is defined as a sequence of word characters.

```
>>> str1="python"
>>> m=re.search(r'\Apy',str1)
>>> print(m)
<re.Match object; span=(0, 2), match='py'>
>>> m=re.search(r'on\Z',str1)
>>> print(m)
<re.Match object; span=(4, 6), match='on'>
>>> str1="python pypy py jpython rpython"
>>> A=re.findall(r'py',str1)
>>> print(A)
['py', 'py', 'py', 'py', 'py', 'py']
>>> A=re.findall(r'\bpy\b',str1)
>>> print(A)
['py']
>>> A=re.findall(r'python',str1)
>>> print(A)
['python', 'python', 'python']
>>> A=re.findall(r'python\b',str1)
>>> print(A)
['python', 'python', 'python']
>>> A=re.findall(r'\bpython\b',str1)
>>> print(A)
['python']
>>> str2="ab ab. ab! ab"
>>> A=re.findall(r'\bab\b',str1)
>>> print(A)
[]
>>> A=re.findall(r'\bab\b',str2)
>>> print(A)
```

['ab', 'ab', 'ab', 'ab']

\B
Matches the empty string, but only when it is *not* at the beginning or end of a word.

```
>>> str1="python pypy py jpython rpython"
>>> A=re.findall(r'\Bpy\B',str1)
>>> print(A)
['py', 'py']
```

\d Matches any decimal digit in the ASCII character set; this is equivalent to [0-9].

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

```
import re
str1="python 3.13"
A=re.findall(r'\d',str1)
print(A)
B=re.findall(r'\D',str1)
print(B)
```

Output
['3', '1', '3']
['p', 'y', 't', 'h', 'o', 'n', ' ', '.']

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

```
import re
str1="python java\toracle\vmysql\naws"
A=re.split(r'\s',str1)
print(A)
```

Output
['python', 'java', 'oracle', 'mysql', 'aws']

\S
Matches any character which is not a whitespace character. This is the opposite of \s.

```
import re
str1="python java\toracle\vmysql\naws"
A=re.findall(r'\S',str1)
print(A)
A=re.findall(r'\s',str1)
print(A)
```

Output

```
['p', 'y', 't', 'h', 'o', 'n', 'j', 'a', 'v', 'a', 'o', 'r', 'a', 'c', 'l', 'e', 'm', 'y', 's', 'q', 'l', 'a', 'w', 's']
[' ', '\t', '\x0b', '\n']
```

\w
Matches characters considered alphanumeric in the ASCII character set; this is equivalent to [a-zA-Z0-9_]

```
# email validation
# aaa@bbb.cc
import re

email=input("Enter EmailID :")
m=re.fullmatch(r'^\w+@\w+\.\w{2,3}',email)
print(m)
```

Output

```
Enter EmailID :naresh@nit.com
<re.Match object; span=(0, 14), match='naresh@nit.com'>
```

\W
Matches any character which is not a word character. This is the opposite of \w

```
# Password Validation

import re
```

```
password=input("Password :")
m=re.fullmatch(r'^[a-zA-Z]{4,}\d{2,}\W{2,}$',password)
print(m)
```

Output

```
Password :abcd123@$
<re.Match object; span=(0, 9), match='abcd123@$'>
```

Python Database Communication (PDBC)

Every application required to store data permanently. Data can be stored permanently using 2 systems.

1. File System
2. Database System

Limitations of File System

1. Files cannot hold large amount data
2. Files cannot provide security because files are managed by operating system
3. Files are application dependent
4. Files cannot provide Query language

To overcome these limitations we use database systems or database application.

Advantage

1. Database can hold large amount data
2. Data stored inside database is secured, which is provided by database system
3. Database is application independent

4. Database provides a query language for manipulating data (SQL)

Database software's

Database software name	**Company**
Oracle	Oracle corporation
MySQL	Oracle corporation
SQLServer	Microsoft
PostgreSQL	PostgreSQL
SQLlite	PSF (Python Software Foundation)
MongoDB	MongoDB

SQL

SQL stands for Structured Query Language. It is a standard developed by ASNI (American National Standard Institute) to communicate with database.

SQL is used by database software for manipulating data.
SQL provides the command to perform operation on database.
SQL commands are classified categories

1. DDL → Data Definition Language
    a. CREATE
    b. ALTER
    c. DROP
2. DML → Data Manipulation Language
    a. INSERT
    b. UPDATE
    c. DELETE
3. DCL → Data Control Language
    a. GRANT
    b. REVOKE
4. TCL → Transaction Control Language
    a. COMMIT
    b. ROLLBACK
5. DRL → Data Reading Language
    a. SELECT