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

Example:

```
import re
def main():
    str1="python programming language"
    l=re.findall(r'\s',str1)
    print(l)
    str2="python\tprogramming\nlanguage"
    l=re.findall(r'\s',str2)
    print(l)

main()
```

Output:

```
[' ', ' ']
['\t', '\n']
>>>
```

\S
Matches any character which is not a whitespace character. This is the opposite of \s. If the ASCII flag is used this becomes the equivalent of [^ \t\n\r\f\v].

```
import re
def main():
    str1="python programming language"
    l=re.findall(r'\S',str1)
    print(l)
    str2="python\tprogramming\nlanguage"
    l=re.findall(r'\S',str2)
    print(l)

main()
```

Output:

['p', 'y', 't', 'h', 'o', 'n', 'p', 'r', 'o', 'g', 'r', 'a', 'm', 'm', 'i', 'n', 'g', 'l', 'a', 'n', 'g', 'u', 'a', 'g', 'e']
['p', 'y', 't', 'h', 'o', 'n', 'p', 'r', 'o', 'g', 'r', 'a', 'm', 'm', 'i', 'n', 'g', 'l', 'a', 'n', 'g', 'u', 'a', 'g', 'e']
>>>

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

\W
Matches any character which is not a word character. This is the opposite of \w. If the ASCII flag is used this becomes the equivalent of [^a-zA-Z0-9_].

 # username validation

```
import re
def main():
    uname=input("UserName")
    m=re.search(r'\W',uname)
    if m==None:
        print(f'{uname} is valid')
    else:
        print(f'{uname} is invalid')

main()
```

Output:

```
UserNamenaresh
naresh is valid
>>>
========== RESTART: C:/Users/user/Desktop/python6pm/py229.py
==========
UserNamenit123
nit123 is valid
>>>
========== RESTART: C:/Users/user/Desktop/python6pm/py229.py
==========
UserNamenit_123
nit_123 is valid
```

```
>>>
========== RESTART: C:/Users/user/Desktop/python6pm/py229.py
==========
UserNamenit$
nit$ is invalid
>>>
```

https://www.hackerrank.com/challenges/validating-the-phone-number/problem

```
import re
def main():
    n=int(input())
    for i in range(n):
        mobile=input()
        if len(mobile)!=10:
            print("NO")
        else:
            m=re.search(r'^[789]\d{9}',mobile)
            if m!=None:
                print("YES")
            else:
                print("NO")
main()
```

https://www.hackerrank.com/challenges/introduction-to-regex/problem

```
def main():
    n=int(input())
    for i in range(n):
        value=input()
        m=re.fullmatch(r'[+-]?\d*\.\d*',value)

        if m!=None:
            print(True)
        else:
            print(False)

main()
```

re.split(pattern, string, maxsplit=0, flags=0)
Split string by the occurrences of pattern. If capturing parentheses are used in pattern, then the text of all groups in the pattern are also returned as part of the resulting list. If maxsplit is nonzero, at most maxsplit splits occur, and the remainder of the string is returned as the final element of the list.