String examine methods

str.isalnum()
Return True if all characters in the string are alphanumeric and there is at least one character, False otherwise.

```
>>> "abc".isalnum()
True
>>> "abc123".isalnum()
True
>>> "123".isalnum()
True
>>> "abc$".isalnum()
False
>>> "abc123$".isalnum()
False
```

str.isalpha()
Return True if all characters in the string are alphabetic and there is at least one character, False otherwise.

```
>>> "abc".isalpha()
True
>>> "123".isalpha()
False
>>> "abc$".isalpha()
False
```

Example:

```
name=input("Enter name ")
# name must be alphabets
```

```
b=name.isalpha()
if b:
   print('valid name')
else:
   print('invalid name')
```

Output

```
Enter name naresh
valid name

Enter name nit123
invalid name
```

str.isdigit()

Return True if all characters in the string are digits and there is at least one character, False otherwise.

Example:

```
n1=input("Enter first number ")
n2=input("Enter second number ")
if n1.isdigit() and n2.isdigit():
   n3=int(n1)+int(n2)
   print(f'sum of {n1} and {n2} is {n3}')
else:
   print("invalid input only integer type")
```

Output:

```
Enter first number 12
Enter second number 56
sum of 12 and 56 is 68

Enter first number abc
Enter second number 56
```

invalid input only integer type

str.islower()
Return True if all cased characters in the string are lowercase and there is at least one cased character, False otherwise.

```
>>> str1="abc"
>>> str1.islower()
True
>>> str2="Abc"
>>> str2.islower()
False
>>> str3="abc123"
>>> str3.islower()
True
>>> str4="abcD123"
>>> str4.islower()
False
```

str.isspace()
Return True if there are only whitespace characters in the string and there is at least one character, False otherwise.

```
>>> s1="          "
>>> s1.isspace()
True
>>> s2="abc xyz"
>>> s2.isspace()
False
```

str.istitle()

Return True if the string is a titlecased string and there is at least one character, for example uppercase characters may only follow uncased characters and lowercase characters only cased ones. Return False otherwise.

```
>>> s3="Python Programming Language"
>>> s3.istitle()
True
>>> s4="python programming"
>>> s4.istitle()
False
```

str.isupper()
Return True if all cased characters in the string are uppercase and there is at least one cased character, False otherwise.

```
>>> s1="ABC"
>>> s1.isupper()
True
>>> s2="ABC1234"
>>> s2.isupper()
True
>>> s3="abC12"
>>> s3.isupper()
False
```

str.endswith(*suffix*[, *start*[, *end*]])
Return True if the string ends with the specified *suffix*, otherwise return False. *suffix* can also be a tuple of suffixes to look for. With optional *start*, test beginning at that position. With optional *end*, stop comparing at that position.

```
>>> str1="python"
>>> str1.endswith("n")
True
>>> str2="java"
>>> str2.endswith("n")
False
```

Example:

```
names=["naresh","kishore","ramesh","kiran","rajesh","raman"]

for name in names:
    b=name.endswith("h")
    if b:
        print(name)

print("====================")
for name in names:
    b=name.endswith(("h","n"))
    if b:
        print(name)
```

Output:

```
naresh
ramesh
rajesh
====================
naresh
ramesh
kiran
rajesh
raman
```

```
>>> str1="python programming"
>>> str1.endswith("g")
True
>>> str1.endswith("n",0,6)
True
>>> str1.endswith("r",7,12)
True
```

str.startswith(*prefix*[, *start*[, *end*]])
Return True if string starts with the *prefix*, otherwise return False. *prefix* can also be a tuple of prefixes to look for. With optional *start*, test string beginning at that position. With optional *end*, stop comparing string at that position.

Example:

```
names=["naresh","kishore","ramesh","kiran","rajesh","raman"]

for name in names:
    b=name.startswith("r")
    if b:
        print(name)

print("==================")
for name in names:
    b=name.startswith(("r","k"))
    if b:
        print(name)
```

Output:

```
ramesh
rajesh
raman
```

==========
kishore
ramesh
kiran
rajesh
raman

String split methods

str.split(sep=None, maxsplit=- 1)
Return a list of the words in the string, using sep as the delimiter string. If maxsplit is given, at most maxsplit splits are done (thus, the list will have at most maxsplit+1 elements). If maxsplit is not specified or -1, then there is no limit on the number of splits (all possible splits are made).

```
>>> s1="a b c d e"
>>> list1=s1.split()
>>> print(list1)
['a', 'b', 'c', 'd', 'e']
>>> s2="a,b,c,d,e"
>>> list2=s2.split(",")
>>> print(list2)
['a', 'b', 'c', 'd', 'e']
>>> s3="a,b c,d e,f"
>>> list3=s3.split(",")
>>> print(list3)
['a', 'b c', 'd e', 'f']
>>> s4="a,b,c,d,e,f,g"
>>> list4=s4.split(",",2)
>>> print(list4)
['a', 'b', 'c,d,e,f,g']
```

```
>>> list5=s4.split(",",1)
>>> print(list5)
['a', 'b,c,d,e,f,g']
>>> s5="10 20 30 40 50"
>>> list6=s5.split()
>>> print(list6)
['10', '20', '30', '40', '50']
>>> s6="10,20,30,40,50"
>>> a,b,c,d,e=s6.split(",")
>>> print(a,b,c,d,e)
10 20 30 40 50
>>> s7="a\tb\tc\td\te"
>>> list8=s7.split()
>>> print(list8)
['a', 'b', 'c', 'd', 'e']
>>> s8="a\nb\nc\nd\ne"
>>> list9=s8.split()
>>> print(list9)
['a', 'b', 'c', 'd', 'e']
>>> s9="a b\nc\td e"
>>> list10=s9.split()
>>> print(list10)
['a', 'b', 'c', 'd', 'e']
```

str.rsplit(*sep=None*, *maxsplit=- 1*)

Return a list of the words in the string, using *sep* as the delimiter string. If *maxsplit* is given, at most *maxsplit* splits are done, the *rightmost* ones. If *sep* is not specified or None, any whitespace string is a separator.

```
>>> s1="a b c d e"
>>> list1=s1.rsplit()
```

```
>>> print(list1)
['a', 'b', 'c', 'd', 'e']
>>> list2=s1.rsplit(" ",2)
>>> print(list2)
['a b c', 'd', 'e']
```

String strip methods

str.strip([c*hars*])

Return a copy of the string with the leading and trailing characters removed. The *chars* argument is a string specifying the set of characters to be removed. If omitted or None, the *chars* argument defaults to removing whitespace. The *chars* argument is not a prefix or suffix; rather, all combinations of its values are stripped:

```
>>> str1="    nit     "
>>> str1=="nit"
False
>>> str2=str1.strip()
>>> print(str1)
    nit
>>> print(str2)
nit
>>> str2=="nit"
True
>>> s1="*****nit******"
>>> s2=s1.strip("*")
>>> print(s1)
*****nit******
>>> print(s2)
nit
>>> s3="***$$##nit$$##**$$"
```

```
>>> s4=s3.strip("*$#")
>>> print(s3)
***$$##nit$$###**$$
>>> print(s4)
Nit
```

str.rstrip([*chars*])

Return a copy of the string with trailing characters removed.
The *chars* argument is a string specifying the set of characters to be removed. If omitted or None, the *chars* argument defaults to removing whitespace. The *chars* argument is not a suffix; rather, all combinations of its values are stripped:

str.lstrip([*chars*])

Return a copy of the string with leading characters removed.
The *chars* argument is a string specifying the set of characters to be removed. If omitted or None, the *chars* argument defaults to removing whitespace. The *chars* argument is not a prefix; rather, all combinations of its values are stripped: