STRING MANIPULATION

VINOD KUMAR VERMA, PGT(CS), KV OEF KANPUR &
SACHIN BHARDWAJ, PGT(CS), KV NO.1 TEZPUR

TRAVERSING STRING

- It means accessing the individual characters of string i.e. from first character to last character.
- Every character in string is at different index position i.e. from 0 to size-1
- For loop can be used to traverse the string very easily
- For .e.g

```
name="lovely"
for ch in name:
        print(ch,'-',end='')
```

The above code will print l-o-v-e-l-y-

PROGRAM TO READ STRING AND PRINT IN REVERSE ORDER

```
string1 = input("Enter any string ")
print("The Reverse of ", string1 , " is :")
length=len(string1)
for ch in range(-1,(-length-1),-1):
        print(string1[ch])
```

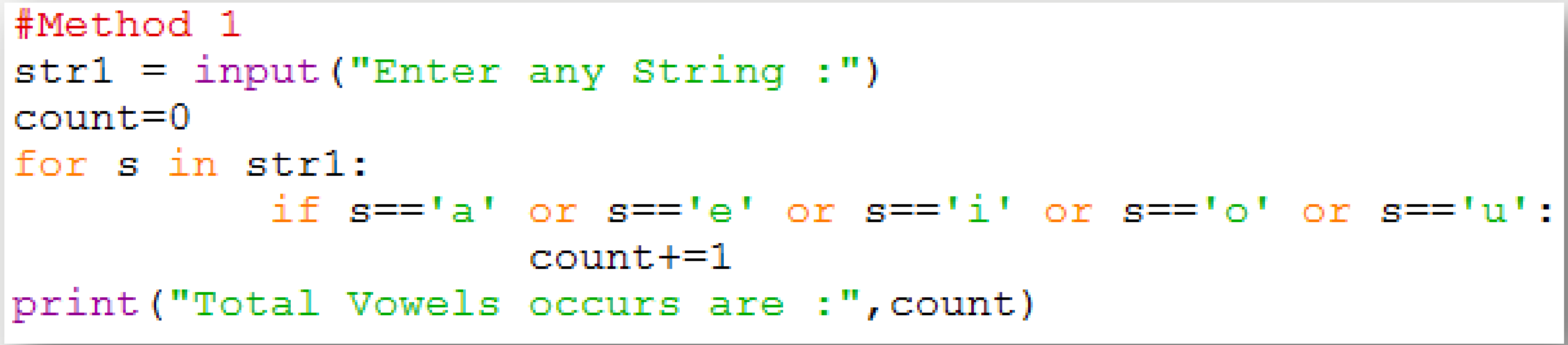
The above code will print

```
Enter any string: karan
n
a
r
a
k
```

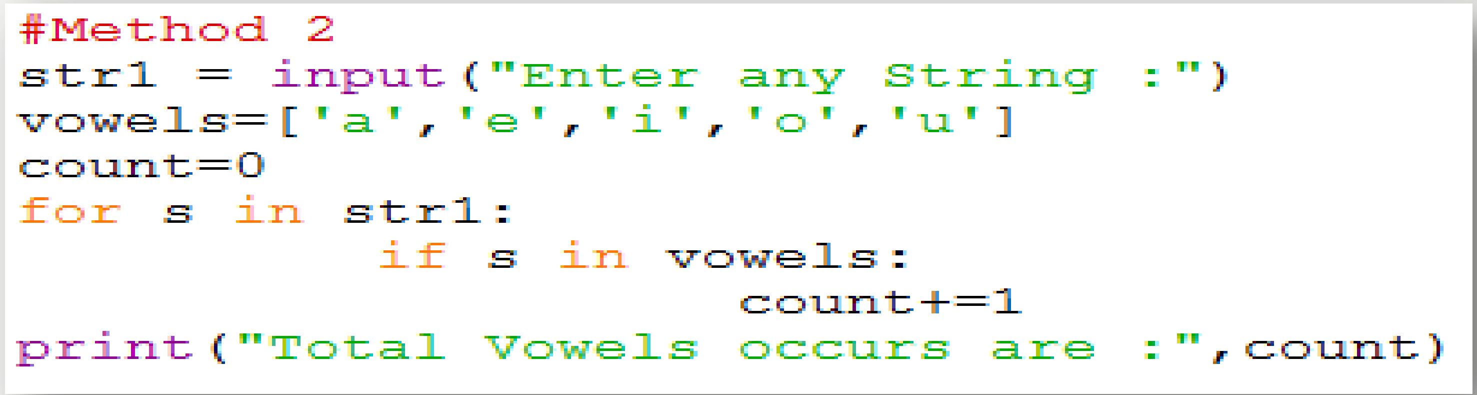
PROGRAM TO INPUT STRING AND PRINT SHORT FORM

```
string=input("Enter any string ")
print(string[0],".",end='')
for ch in range(1,len(string)):
    if string[ch]==' ':
        print(string[ch+1],".",end='')
```

PROGRAM TO INPUT ANY STRING AND COUNT HOW MANY VOWELS IN IT



```
#Method 1
str1 = input("Enter any String :")
count=0
for s in str1:
         if s=='a' or s=='e' or s=='i' or s=='o' or s=='u':
                  count+=1
print("Total Vowels occurs are :",count)
```



```
#Method 2
str1 = input("Enter any String :")
vowels=['a','e','i','o','u']
count=0
for s in str1:
          if s in vowels:
                    count+=1
print("Total Vowels occurs are :",count)
```

STRING OPERATORS

Two basic operators + and * are allowed

+ is used for concatenation (joining)

* Is used for replication (repetition)

EXAMPLE

```
A="Tom"
B="Jerry"
C=A+" & "+B
print(C)
```

Note: you cannot add number and string using +

EXAMPLE

Line=“ go”

print(Line*3, ” Govinda”)

Note: you cannot multiply string and string using *

Only number*number or string*number is allowed

MEMBERSHIP OPERATORS

- Membership operators (in and not in) are used to check the presence of character(s) in any string.

Example	Output
‘a’ in ‘python’	False
‘a’ in ‘java’	True
‘per’ in ‘operators’	True
‘men’ in ‘membership’	False
‘Man’ in ‘manipulation’	False
‘Pre’ not in ‘presence’	True

COMPARISON OPERATORS

- We can apply comparison operators (==, !=,>,<,>=,<=) on string. Comparison will be character by character.

str1='program'

str2='python'

str3='Python'

Comparison of string will be based on ASCII code of the characters

Example	Output
str1==str2	False
str1!=str2	True
str2=='python'	True
str2>str3	True
str3<str1	True

Characters	Ordinal/ ASCII code
A-Z	65-90
a-z	97-122
0-9	48-57

DETERMINING ORDINAL / UNICODE OF A SINGLE CHARACTER

Python allows us to find out the ordinal position single character using ord() function.

>>>ord('A') **output will be 65**

We can also find out the character based on the ordinal value using chr() function

>>>chr(66) **output will be 'B'**

STRING SLICING

As we know slice means ‘part of’, so slicing is a process of extracting part of string. In previous chapters we already discussed that string characters have their unique index position from 0 to length-1 and -1 to –length(backward)

Forward indexing	0	1	2	3	4	5	6
message	W	E	L	C	O	M	E
	-7	-6	-5	-4	-3	-2	-1	Backward indexing

STRING SLICING

```
>>> str1="wonderful"
>>> str1[0:6]
'wonder'
>>> str1[0:3]
'won'
>>> str1[3:6]
'der'
>>> str1[-1:-3]
''
>>> str1[-3:-1]
'fu'
>>> str1[-3:0]
'‘
>>>str1[0::2]
‘Wnefl’
```

```
>>> str1[3:3]
''
>>> str1[3:4]
'd'
>>> str1[-5:-2]
'erf'
>>> str1[:-2]
'wonderf'
>>> str1[:4]
'wond‘
>>> str1[-3:]
'ful‘
>>>str1[::-1]
lufrednow
```

Reverse of string

INTERESTING STRING SLICING

For any index position n: str1[:n]+str1[n:] will give you the original string

>>>str1="wonderful"

>>>str1[:n]+str[n:] **output will be wonderful**

String slicing will never return error even if you pass index which is not in the string . For e.g.

>>>str1[10] **will give error, but**

>>>str1[10:15] **will not give error but return empty string**

PROGRAMS

Program to print the pattern

```
@
@@
@@@
@@@@
@@@@@

string='#'
pattern=''
for i in range(5):
        pattern+=string
        print(pattern)
```

Program to input name and print as (if name is 'AAMIR'), output

```
A
AA
AAM
AAMI
AAMIR

name=input("Enter any name")
for i in range(0,len(name)+1):
        print(name[0:i])
```

STRING FUNCTIONS AND METHODS

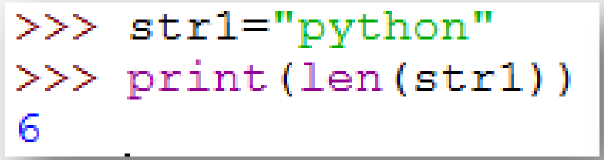
Python offers many built-in function for string manipulation. One method len() we have already used. Let us understand other methods also.

<u>To use string manipulation function the syntax is:</u>

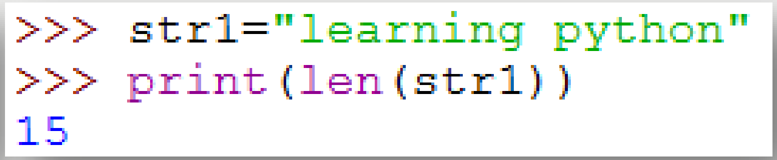
String_Object.functionName()

STRING FUNCTIONS/METHODS

- **len(string)** : this function returns the number of characters in any string including spaces.

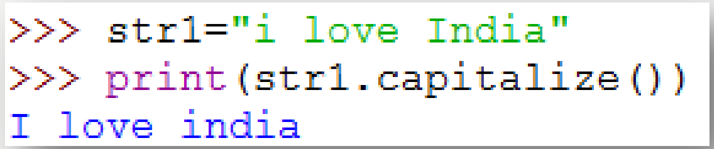


```
>>> str1="python"
>>> print(len(str1))
6
```



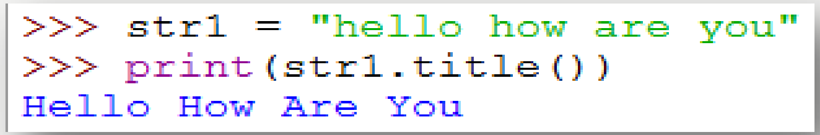
```
>>> str1="learning python"
>>> print(len(str1))
15
```

- **capitalize()** : this function is used to convert the first letter of sentence in capital letter.



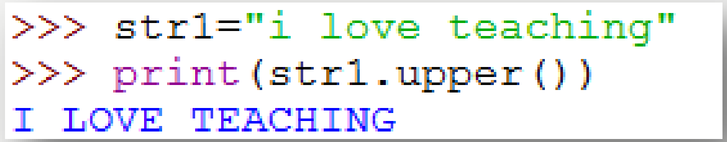
```
>>> str1="i love India"
>>> print(str1.capitalize())
I love india
```

- **title()** : this function is used to convert first letter of every word in string in capital letters.



```
>>> str1 = "hello how are you"
>>> print(str1.title())
Hello How Are You
```

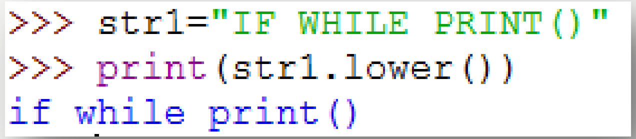
- **upper()** : this function is used to convert the entire string in capital letter.



```
>>> str1="i love teaching"
>>> print(str1.upper())
I LOVE TEACHING
```

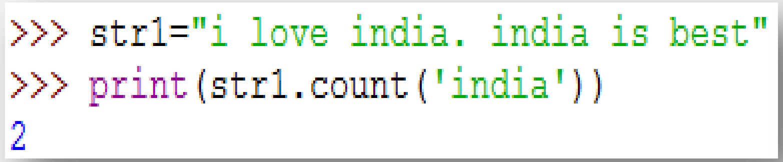
STRING FUNCTIONS/METHODS

- **lower()** : this function is used to convert the entire string in small letter.

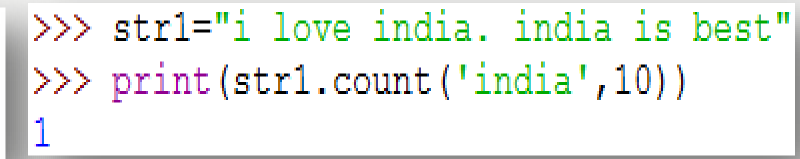


```
>>> str1="IF WHILE PRINT()"
>>> print(str1.lower())
if while print()
```

- **count(substring, [start,[end]])** : this function is used to find the number of occurrence of substring in our string. We can give optional starting index from where searching begin and also end index i.e. upto what index in string it will search and count. It return 0 if substring not found.

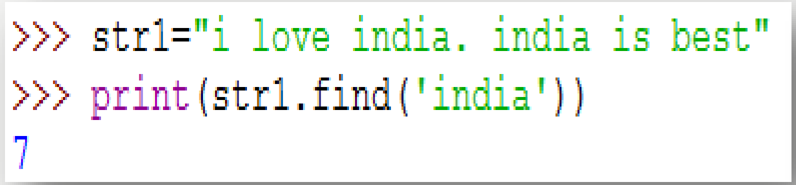


```
>>> str1="i love india. india is best"
>>> print(str1.count('india'))
2
```

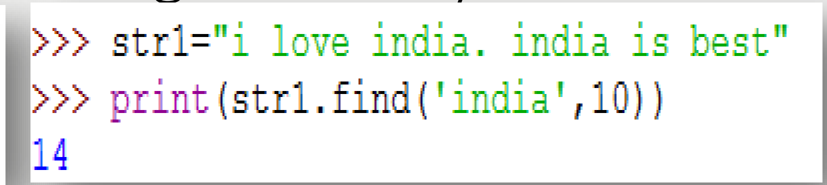


```
>>> str1="i love india. india is best"
>>> print(str1.count('india',10))
1
```

- **find(substring,[start,[end]])** : this function returns the index position of substring in the given string. We can also specify start and end index just like count() to modify search criteria. It return -1 if substring not found/



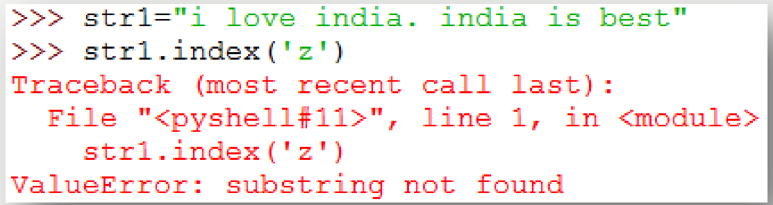
```
>>> str1="i love india. india is best"
>>> print(str1.find('india'))
7
```



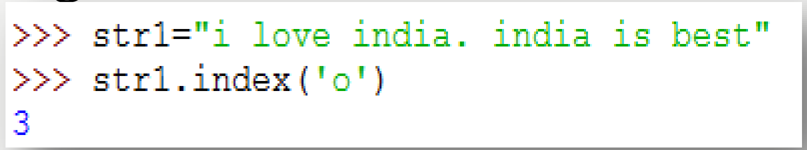
```
>>> str1="i love india. india is best"
>>> print(str1.find('india',10))
14
```

STRING FUNCTIONS/METHODS

- **index(substring)** : this function return index position of substring. If substring not it returns error 'substring not found'

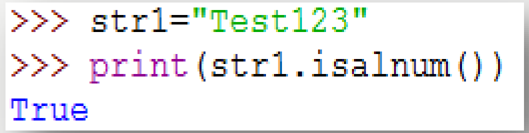


```
>>> str1="i love india. india is best"
>>> str1.index('z')
Traceback (most recent call last):
  File "<pyshell#11>", line 1, in <module>
    str1.index('z')
ValueError: substring not found
```

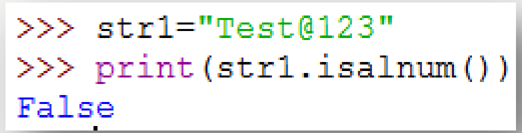


```
>>> str1="i love india. india is best"
>>> str1.index('o')
3
```

- **isalnum()** : this function is used to check where string contain all character as alphanumeric or not. Its return is either True or False.



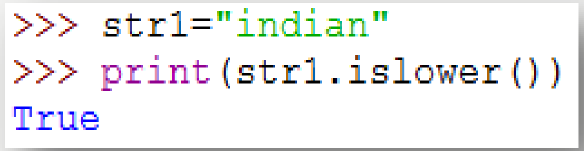
```
>>> str1="Test123"
>>> print(str1.isalnum())
True
```



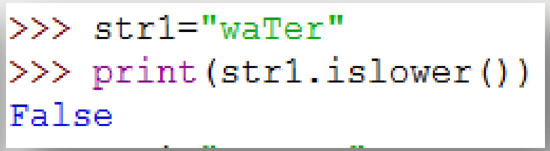
```
>>> str1="Test@123"
>>> print(str1.isalnum())
False
```

- **islower():** this function return True if all the characters in string is in small letters
- **isupper() :** this function return True if all the characters in string is in capital letters. Both islower() and isupper() will check the case only for letter, if any symbol present in string it will be ignored.

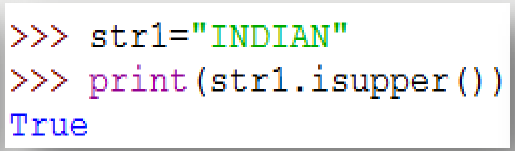
STRING FUNCTIONS/METHODS



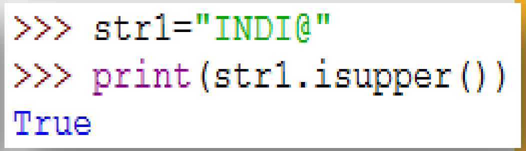
```
>>> str1="indian"
>>> print(str1.islower())
True
```



```
>>> str1="waTer"
>>> print(str1.islower())
False
```

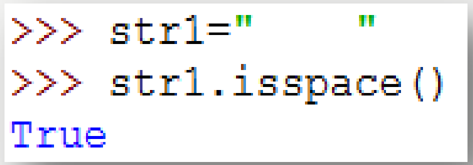


```
>>> str1="INDIAN"
>>> print(str1.isupper())
True
```

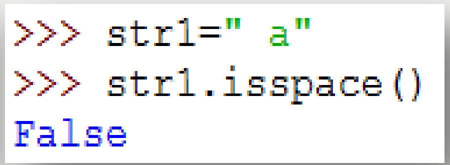


```
>>> str1="INDI@"
>>> print(str1.isupper())
True
```

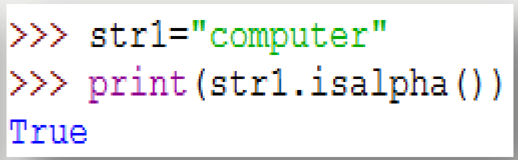
- **isspace() :**.it return True if the string contains only space.
- **isalpha()** : it return True if all the characters in string is alphabet.
- **isdigit()** : it returns True if all the characters in string is digit.



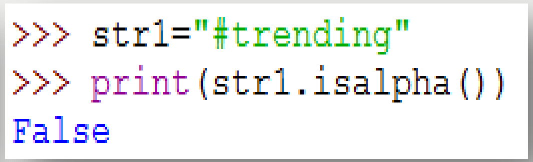
```
>>> str1="      "
>>> str1.isspace()
True
```



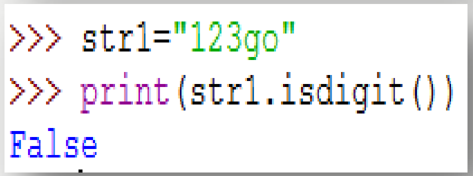
```
>>> str1=" a"
>>> str1.isspace()
False
```



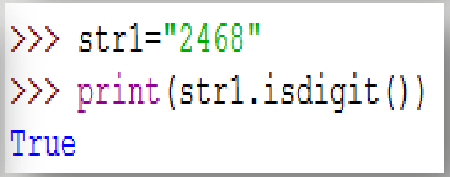
```
>>> str1="computer"
>>> print(str1.isalpha())
True
```



```
>>> str1="#trending"
>>> print(str1.isalpha())
False
```



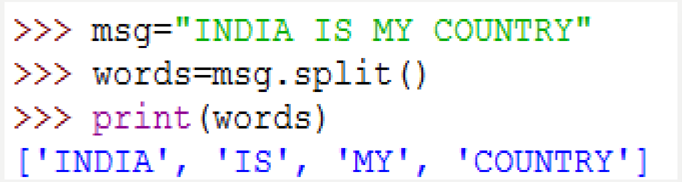
```
>>> str1="123go"
>>> print(str1.isdigit())
False
```



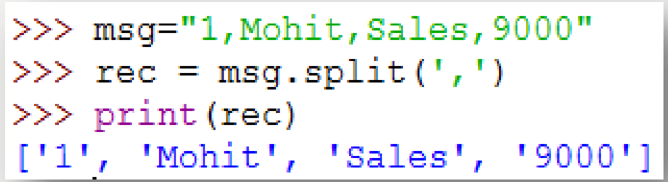
```
>>> str1="2468"
>>> print(str1.isdigit())
True
```

STRING FUNCTIONS/METHODS

- split() : this function is used to split the string based on delimiter and store the result in the form of list. Default delimiter is space.

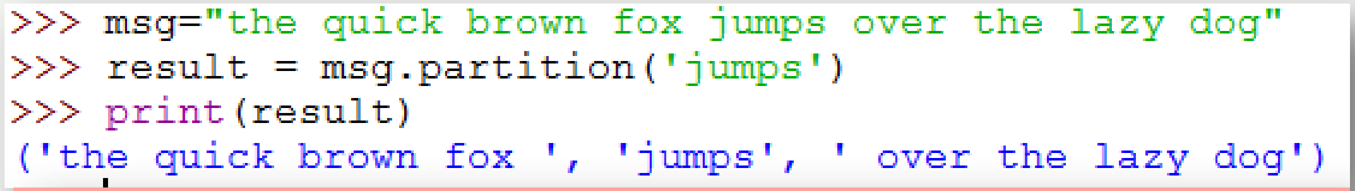


```
>>> msg="INDIA IS MY COUNTRY"
>>> words=msg.split()
>>> print(words)
['INDIA', 'IS', 'MY', 'COUNTRY']
```

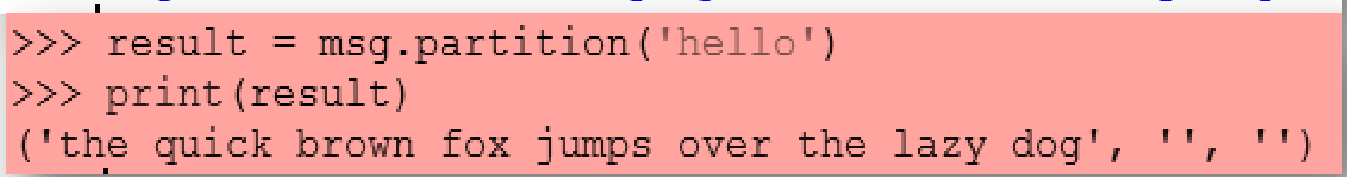


```
>>> msg="1,Mohit,Sales,9000"
>>> rec = msg.split(',')
>>> print(rec)
['1', 'Mohit', 'Sales', '9000']
```

- **partition(sep)** : this function divides the string in the form of tuples of 3 elements known as **head, sep and tail**. All the string **before sep becomes head** and all the string **after sep becomes tail**. If **sep** is not present in the string then everything will becomes head, **sep** and tail will be empty.



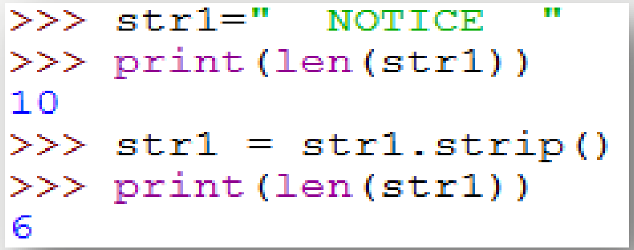
```
>>> msg="the quick brown fox jumps over the lazy dog"
>>> result = msg.partition('jumps')
>>> print(result)
('the quick brown fox ', 'jumps', ' over the lazy dog')
```



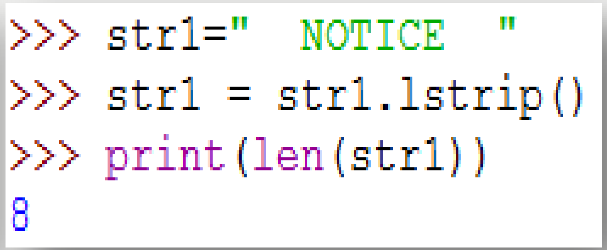
```
>>> result = msg.partition('hello')
>>> print(result)
('the quick brown fox jumps over the lazy dog', '', '')
```

STRING FUNCTIONS/METHODS

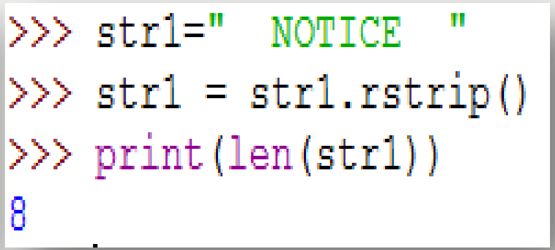
- **strip([chars]) :** it return a copy of string with leading and trailing whitespaces removed. If chars is given, it remove characters instead.
- **lstrip([chars]) :** it return a copy of string with leading whitespaces removed. If chars is given, it remove characters instead.
- **rstrip([chars]) :** it return a copy of string with trailing whitespaces removed. If chars is given, it remove characters instead.



```
>>> str1="   NOTICE   "
>>> print(len(str1))
10
>>> str1 = str1.strip()
>>> print(len(str1))
6
```



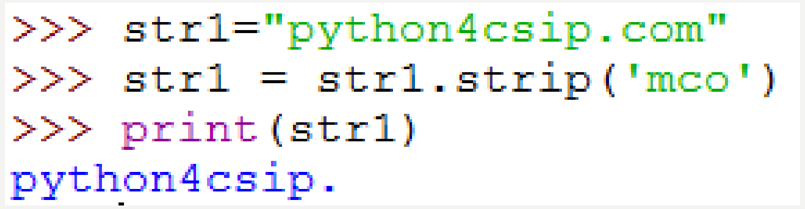
```
>>> str1="   NOTICE   "
>>> str1 = str1.lstrip()
>>> print(len(str1))
8
```



```
>>> str1="   NOTICE   "
>>> str1 = str1.rstrip()
>>> print(len(str1))
8
```

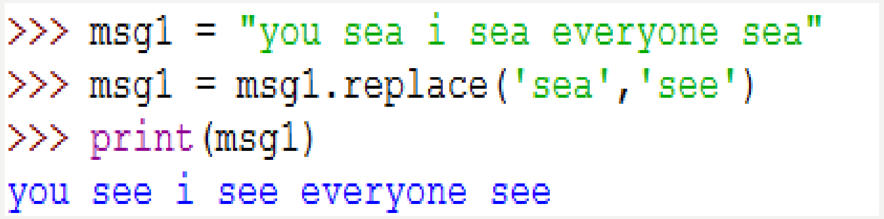
- Note: if chars is given in any strip() function, the chars are checked for all the possible combination that can be formed with the given chars. For e.g. if NOT is passed as char then NOT, OTN, TON, TNO, ONT,NT like this every possible combination will be checked.

STRING FUNCTION / METHODS



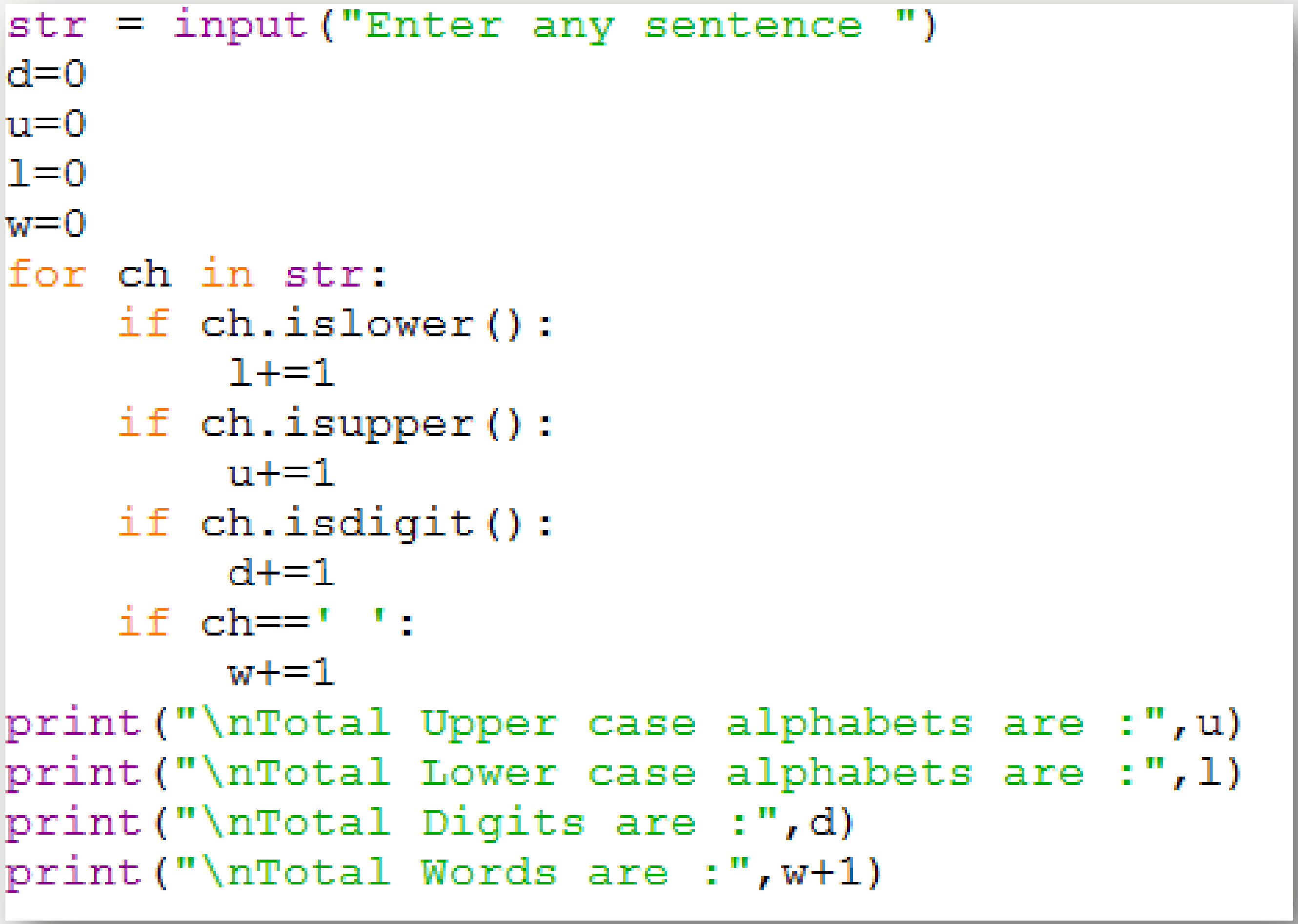
```
>>> str1="python4csip.com"
>>> str1 = str1.strip('mco')
>>> print(str1)
python4csip.
```

- **replace(old, new) :** this function is used to replace old text inside the string with new text.

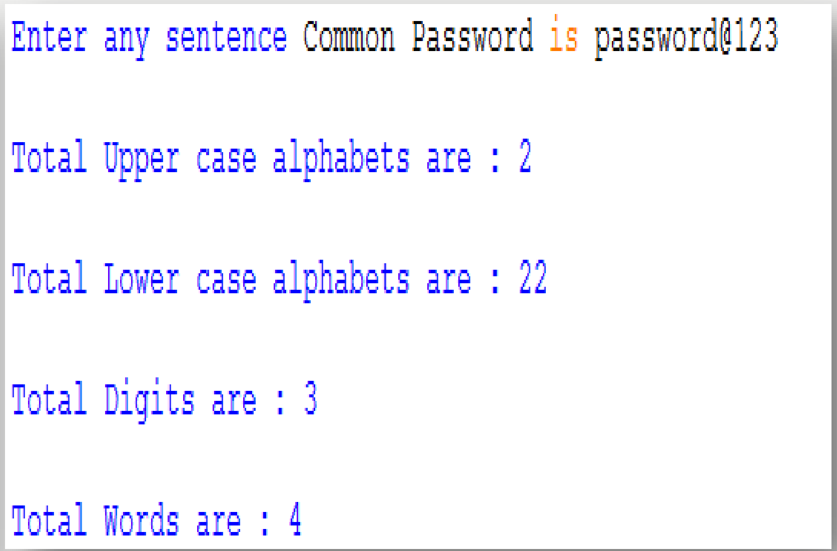


```
>>> msg1 = "you sea i sea everyone sea"
>>> msg1 = msg1.replace('sea','see')
>>> print(msg1)
you see i see everyone see
```

PROGRAM TO ENTER STRING AND COUNT HOW MANY UPPERCASE, LOWERCASE, DIGITS, WORDS PRESENT IN IT.

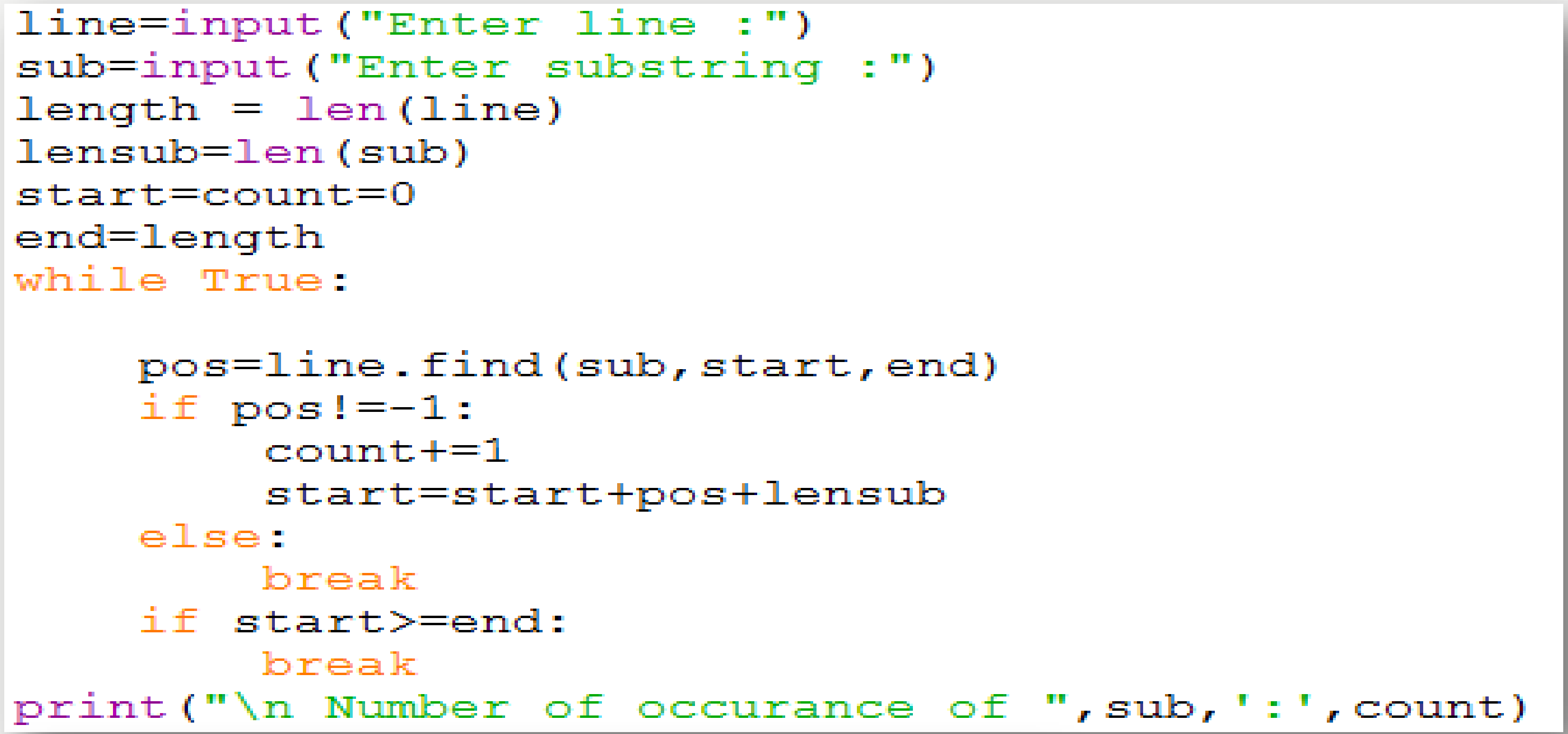


```
str = input("Enter any sentence ")
d=0
u=0
l=0
w=0
for ch in str:
    if ch.islower():
        l+=1
    if ch.isupper():
        u+=1
    if ch.isdigit():
        d+=1
    if ch==' ':
        w+=1
print("\nTotal Upper case alphabets are :",u)
print("\nTotal Lower case alphabets are :",l)
print("\nTotal Digits are :",d)
print("\nTotal Words are :",w+1)
```

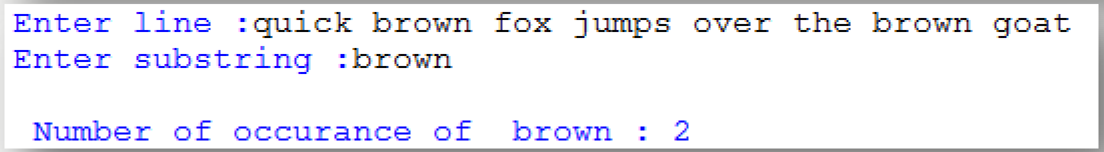


```
Enter any sentence Common Password is password@123

Total Upper case alphabets are : 2

Total Lower case alphabets are : 22

Total Digits are : 3

Total Words are : 4
```

PROGRAM TO ENTER STRING AND FIND THE NUMBER OF OCCURANCE OF ANY WORD



```
line=input("Enter line :")
sub=input("Enter substring :")
length = len(line)
lensub=len(sub)
start=count=0
end=length
while True:

    pos=line.find(sub,start,end)
    if pos!=-1:
        count+=1
        start=start+pos+lensub
    else:
        break
    if start>=end:
        break
print("\n Number of occurance of ",sub,':',count)
```



```
Enter line :quick brown fox jumps over the brown goat
Enter substring :brown

 Number of occurance of  brown : 2
```

JUST A MINUTE…

Find out output of following code fragment:

```
s1='try'
s2='again'
n1=8
n2=5
print(s1+s2)
print(s2*n2)
print(s1+n1)
print(s2*s1)
```

CONSIDER THE FOLLOWING CODE: WHAT WILL BE THE OUTPUT IF INPUT IS

(i) aabbcc **(ii) aaccbb** **(iii) abcc**

```
string = input("Enter a string :")
count=3
while True:
        if string[0]=='a':
                string = string[2:]
        elif string[-1] == 'b':
                string=string[:2]
        else:
                count+=1
                break
print(string)
print(count)
```

CONSIDER THE FOLLOWING CODE: WHAT WILL BE THE OUTPUT IF INPUT IS (i) aabbcc (ii) aaccbb (iii) abcc

```
string = input("Enter a string :")
count=3
while True:
        if string[0]=='a':
                string = string[2:]
        elif string[-1] == 'b':
                string=string[:2]
        else:
                count+=1
                break
print(string)
print(count)
```

If input is aabbcc, output will be

```
bbcc
4
```

If input is aaccbb, output will be

```
cc
4
```

If input is abcc, output will be

```
cc
4
```

WHAT WOULD FOLLOWING EXPRESSION RETURN?

a) “Hello World”.upper().lower()

b) “Hello World”.lower().upper()

c) “Hello World”.find(“Wor”,1,6)

d) “Hello World”.find(‘Wor’)

e) “Hello World”.find(‘wor’)

f) “Hello World”.isalpha()

g) “Hello World”.isalnum()

h) “1234”.isdigit()

i) “123GH”.isdigit()

OUTPUT?

(a)

```
s="0123456789"
print(s[3],',',s[0:3],'-',s[2:5])
print(s[:3],',',s[3:],'-',s[3:100])
print(s[20:],',',s[2:1],'-',s[1:1])
```

(b)

```
y=str(123)
x="hello"*3
print(x,y)
x = "hello" + "world"
y=len(x)
print(y,x)
```