Using for loop
For loop is an iterator, which iterate the characters from string.

Example:
```
# write a program to count alphabets, digits and
# special characters within string

str1=input("enter any string")
dc,ac,sc=0,0,0

for ch in str1:
    if (ch>='a' and ch<='z') or (ch>='A' and ch<='Z'):
        ac+=1
    elif ch>='0' and ch<='9':
        dc+=1
    else:
        sc+=1
print("alphabet count",ac)
print("digit count",dc)
print("special character count",sc)
```

Output:
```
enter any stringpython 3.11
alphabet count 6
digit count 3
special character count 2
```

String methods

str.capitalize()
Return a copy of the string with its first character capitalized and the rest lowercased.

```
>>> str1="python language"
>>> str1.capitalize()
'Python language'
>>> names=["naresh","suresh","kishore"]
>>> for name in names:
    print(name.capitalize())
```

```
Naresh
Suresh
Kishore
```

str.casefold()
Return a casefolded copy of the string. Casefolded strings may be used for caseless matching.

```
# Login Application

uname=input("UserName :")
pwd=input("Password :")
if uname.casefold()=='nit' and pwd.casefold()=='nit123':
   print("welcome")
else:
   print("invalid user name or password")
```

Output:

```
UserName :NIT
Password :NIT123
Welcome
```

str.center(*width*[, *fillchar*])
Return centered in a string of length *width*. Padding is done using the specified *fillchar* (default is an ASCII space). The original string is returned if *width* is less than or equal to len(s).

Example:

```
student_list=[['naresh','python'],
              ['suresh','java'],
              ['kishore','c++'],
              ['ramesh','python']]
for name,course in student_list:
   print(name.center(15,"*"),course.center(15,'$'))
```

Output:

```
*****naresh**** $$$$$python$$$$
*****suresh**** $$$$$$java$$$$$
```

```
****kishore**** $$$$$$c++$$$$$$
*****ramesh**** $$$$$python$$$$
```

str.count(*sub*[, *start*[, *end*]])
Return the number of non-overlapping occurrences of substring *sub* in the range [*start*, *end*]. Optional arguments *start* and *end* are interpreted as in slice notation.

```
>>> str1="java python java python c C++"
>>> c=str1.count("java")
>>> print(c)
2
>>> c=str1.count("python",10)
>>> print(c)
1
>>> c=str1.count("c")
>>> print(c)
1
>>> c=str1.count("a")
>>> print(c)
4
```

str.encode()
Return an encoded version of the string as a bytes object.

```
>>> str1="Hello"
>>> type(str1)
<class 'str'>
>>> a=str1.encode()
>>> type(a)
<class 'bytes'>
>>> a
b'Hello'
>>> str1
'Hello'
```

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.

Example:

```
namesList=["naresh","kishore","raman","ramesh","suresh"]
for name in namesList:
    if name.endswith('h'):
        print(name)
```

Output:

```
naresh
ramesh
suresh
```

Example:

```
str1="python language"
b=str1.endswith("n",0,6)
print(b)
True
```

Example:

```
namesList=["naresh","kishore","raman","ramesh","suresh"]
for name in namesList:
    if name.endswith(('h','n')):
        print(name)
```

Output:

```
naresh
raman
ramesh
suresh
```

str.expandtabs(*tabsize=8*)

Return a copy of the string where all tab characters are replaced by one or more spaces, depending on the current column and the given tab size. Tab positions occur every *tabsize* characters (default is 8, giving tab positions at columns 0, 8, 16 and so on).

```
>>> str1="empno\tename\tsalary"
>>> print(str1.expandtabs())
```

```
empno   ename   salary
>>> print(str1.expandtabs(10))
empno     ename     salary
```

str.find(*sub*[, *start*[, *end*]])
Return the lowest index in the string where substring *sub* is found within the slice s[start:end]. Optional arguments *start* and *end* are interpreted as in slice notation. Return -1 if *sub* is not found.

```
>>> str1="python programming language"
>>> i=str1.find("language")
>>> print(i)
19
```

String formatting
String formatting is used to format output. String which contains formatting character or formatting fields is called format string.
Formatting string in python is done 3 ways

1. Old style string formatting
2. New style string formatting
3. F-string (python 3.8 version)

Old style string formatting is also called **c-style** string formatting
“characters and formatting characters”%(value,value,value,..)

Formatting characters

%d → decimal integer
%o → octal integer
%x → hexadecimal integer
%f → float in fixed notation
%e → float in exponent notation
%s → string
%c → character

Example:
```
a=10
b=20
```

```
print("sum of %d and %d is %d"%(a,b,a+b))
print("diff of %d and %d is %d"%(a,b,a-b))
x=65
print("%d %o %x %c"%(x,x,x,x))
f1=1.456
print("%f %e %.2f"%(f1,f1,f1))
```

Output:

```
sum of 10 and 20 is 30
diff of 10 and 20 is -10
65 101 41 A
1.456000 1.456000e+00 1.46
```