



# STRING DATA TYPE



The most commonly used object in any project and in any programming language is String only. Hence we should aware complete information about String data type.

## What is String?

Any sequence of characters within either single quotes or double quotes is considered as a String.

### Syntax:

```
s = 'durga'
s = "durga"
```

**Note:** In most of other languages like C, C++, Java, a single character with in single quotes is treated as char data type value. But in Python we are not having char data type. Hence it is treated as String only.

### Eg:

```
>>> ch = 'a'
>>> type(ch)
<class 'str'>
```

## How to define multi-line String Literals?

We can define multi-line String literals by using triple single or double quotes.

### Eg:

```
>>> s = """durga
software
solutions"""
```

We can also use triple quotes to use single quotes or double quotes as symbol inside String literal.

- 1) s = 'This is ' single quote symbol' → Invalid
- 2) s = 'This is \' single quote symbol' → Valid
- 3) s = "This is ' single quote symbol" → Valid
- 4) s = 'This is " double quotes symbol' → Valid
- 5) s = 'The "Python Notes" by 'durga' is very helpful' → Invalid
- 6) s = "The "Python Notes" by 'durga' is very helpful" → Invalid
- 7) s = 'The \'Python Notes\' by \'durga\' is very helpful' → Valid
- 8) s = """The "Python Notes" by 'durga' is very helpful""" → Valid



## How to Access Characters of a String?

We can access characters of a string by using the following ways.

- 1) By using index
- 2) By using slice operator

### 1) Accessing Characters By using Index:

- Python supports both +ve and -ve Index.
- +ve Index means Left to Right (Forward Direction)
- -ve Index means Right to Left (Backward Direction)

Eg: s = 'durga'

```
1) >>> s='durga'
2) >>> s[0]
3) 'd'
4) >>> s[4]
5) 'a'
6) >>> s[-1]
7) 'a'
8) >>> s[10]
9) IndexError: string index out of range
```

**Note:** If we are trying to access characters of a string with out of range index then we will get error saying: IndexError

**Q) Write a Program to Accept some String from the Keyboard and display its Characters by Index wise (both Positive and Negative Index)**

**test.py:**

```
1) s=input("Enter Some String:")
2) i=0
3) for x in s:
4)     print("The character present at positive index {} and at nEgative index {} is {}".fo
       rmat(i,i-len(s),x))
5)     i=i+1
```

**Output:** D:\python\_classes>py test.py

Enter Some String:durga

The character present at positive index 0 and at nEgative index -5 is d

The character present at positive index 1 and at nEgative index -4 is u

The character present at positive index 2 and at nEgative index -3 is r

The character present at positive index 3 and at nEgative index -2 is g

The character present at positive index 4 and at nEgative index -1 is a



## 2) Accessing Characters by using Slice Operator:

- **Syntax:** s[bEginindex:endindex:step]
- **Begin Index:** From where we have to consider slice (substring)
- **End Index:** We have to terminate the slice (substring) at endindex-1
- **Step:** Incremented Value.

### Note:

- If we are not specifying bEgin index then it will consider from bEginning of the string.
- If we are not specifying end index then it will consider up to end of the string.
- The default value for step is 1.

```
1) >>> s="Learning Python is very very easy!!!"
2) >>> s[1:7:1]
3) 'earnin'
4) >>> s[1:7]
5) 'earnin'
6) >>> s[1:7:2]
7) 'eri'
8) >>> s[:7]
9) 'Learnin'
10) >>> s[7:]
11) 'g Python is very very easy!!!'
12) >>> s[::]
13) 'Learning Python is very very easy!!!'
14) >>> s[:]
15) 'Learning Python is very very easy!!!'
16) >>> s[::-1]
17) '!!!ysae yrev yrev si nohtyP gninraeL'
```

## Behaviour of Slice Operator:

- 1) s[bEgin:end:step]
- 2) Step value can be either +ve or -ve
- 3) If +ve then it should be forward direction(left to right) and we have to consider bEgin to end-1
- 4) If -ve then it should be backward direction (right to left) and we have to consider bEgin to end+1.

### **\*\*\*Note:**

- In the backward direction if end value is -1 then result is always empty.
- In the forward direction if end value is 0 then result is always empty.



## In Forward Direction:

default value for bEgin: 0

default value for end: length of string

default value for step: +1

## In Backward Direction:

default value for bEgin: -1

default value for end: -(length of string+1)

**Note:** Either forward or backward direction, we can take both +ve and -ve values for bEgin and end index.

## Slice Operator Case Study:

- 1) S = 'abcdefghij'
- 2) s[1:6:2] → 'bdf'
- 3) s[:1] → 'abcdefghij'
- 4) s[::-1] → 'jihgfedcba'
- 5) s[3:7:-1] → ''
- 6) s[7:4:-1] → 'hgf'
- 7) s[0:10000:1] → 'abcdefghij'
- 8) s[-4:1:-1] → 'gfedc'
- 9) s[-4:1:-2] → 'gec'
- 10) s[5:0:1] → ''
- 11) s[9:0:0] → ValueError: slice step cannot be zero
- 12) s[0:-10:-1] → ''
- 13) s[0:-11:-1] → 'a'
- 14) s[0:0:1] → ''
- 15) s[0:-9:-2] → ''
- 16) s[-5:-9:-2] → 'fd'
- 17) s[10:-1:-1] → ''
- 18) s[10000:2:-1] → 'jihgfed'

**Note:** Slice operator never raises IndexError

## Mathematical Operators for String:

We can apply the following mathematical operators for Strings.

- 1) + operator for concatenation
  - 2) \* operator for repetition
- print("durga"+"soft") → durgasoft
  - print("durga"\*2) → durgadurga



### Note:

- 1) To use + operator for Strings, compulsory both arguments should be str type.
- 2) To use \* operator for Strings, compulsory one argument should be str and other argument should be int.

## len() in-built Function:

We can use len() function to find the number of characters present in the string.

Eg:

```
s = 'durga'  
print(len(s)) → 5
```

## Q) Write a Program to access each Character of String in Forward and Backward Direction by using while Loop?

```
1) s = "Learning Python is very easy !!!"  
2) n = len(s)  
3) i = 0  
4) print("Forward direction")  
5) while i<n:  
6)     print(s[i],end=' ')  
7)     i +=1  
8) print("Backward direction")  
9) i = -1  
10) while i >= -n:  
11)     print(s[i],end=' ')  
12)     i = i-1
```

### Alternative ways:

```
1) s = "Learning Python is very easy !!!"  
2) print("Forward direction")  
3) for i in s:  
4)     print(i,end=' ')  
5) print("Forward direction")  
6) for i in s[::]:  
7)     print(i,end=' ')  
8)  
9) print("Backward direction")  
10) for i in s[::-1]:  
11)     print(i,end=' ')
```



## Checking Membership:

We can check whether the character or string is the member of another string or not by using in and not in operators

```
s = 'durga'
print('d' in s) → True
print('z' in s) → False
```

```
1) s = input("Enter main string:")
2) subs = input("Enter sub string:")
3) if subs in s:
4)     print(subs,"is found in main string")
5) else:
6)     print(subs,"is not found in main string")
```

### Output:

```
D:\python_classes>py test.py
Enter main string:durgasoftwaresolutions
Enter sub string:durga
durga is found in main string
```

```
D:\python_classes>py test.py
Enter main string:durgasoftwaresolutions
Enter sub string:python
python is not found in main string
```

## Comparison of Strings:

- We can use comparison operators (<, <=, >, >=) and equality operators (==, !=) for strings.
- Comparison will be performed based on alphabetical order.

```
1) s1=input("Enter first string:")
2) s2=input("Enter Second string:")
3) if s1==s2:
4)     print("Both strings are equal")
5) elif s1<s2:
6)     print("First String is less than Second String")
7) else:
8)     print("First String is greater than Second String")
```

### Output:

```
D:\python_classes>py test.py
Enter first string:durga
```



Enter Second string:durga  
Both strings are equal

D:\python\_classes>py test.py  
Enter first string:durga  
Enter Second string:ravi  
First String is less than Second String

D:\python\_classes>py test.py  
Enter first string:durga  
Enter Second string:anil  
First String is greater than Second String

## Removing Spaces from the String:

We can use the following 3 methods

- 1) `rstrip()` → To remove spaces at right hand side
- 2) `lstrip()` → To remove spaces at left hand side
- 3) `strip()` → To remove spaces both sides

```
1) city=input("Enter your city Name:")
2) scity=city.strip()
3) if scity=='Hyderabad':
4)     print("Hello Hyderbadi..Adab")
5) elif scity=='Chennai':
6)     print("Hello Madrasi...Vanakkam")
7) elif scity=="Bangalore":
8)     print("Hello Kannadiga...Shubhodaya")
9) else:
10)    print("your entered city is invalid")
```

## Finding Substrings:

We can use the following 4 methods

### For forward direction:

- 1) `find()`
- 2) `index()`

### For backward direction:

- 1) `rfind()`
- 2) `rindex()`





## find():

### **s.find(substring)**

Returns index of first occurrence of the given substring. If it is not available then we will get -1.

```
1) s="Learning Python is very easy"
2) print(s.find("Python")) #9
3) print(s.find("Java")) # -1
4) print(s.find("r"))#3
5) print(s.rfind("r"))#21
```

**Note:** By default find() method can search total string. We can also specify the boundaries to search.

### **s.find(substring,bEgin,end)**

It will always search from bEgin index to end-1 index.

```
1) s="durgaravipavanshiva"
2) print(s.find('a'))#4
3) print(s.find('a',7,15))#10
4) print(s.find('z',7,15))#-1
```

## index():

index() method is exactly same as find() method except that if the specified substring is not available then we will get ValueError.

```
1) s=input("Enter main string:")
2) subs=input("Enter sub string:")
3) try:
4)     n=s.index(subs)
5) except ValueError:
6)     print("substring not found")
7) else:
8)     print("substring found")
```

### **Output:**

```
D:\python_classes>py test.py
Enter main string:learning python is very easy
Enter sub string:python
substring found
```



```
D:\python_classes>py test.py
Enter main string:learning python is very easy
Enter sub string:java
substring not found
```

## **Q) Program to display all Positions of Substring in a given Main String**

```
1) s=input("Enter main string:")
2) subs=input("Enter sub string:")
3) flag=False
4) pos=-1
5) n=len(s)
6) while True:
7)     pos=s.find(subs,pos+1,n)
8)     if pos== -1:
9)         break
10)    print("Found at position",pos)
11)    flag=True
12) if flag==False:
13)    print("Not Found")
```

### **Output:**

```
D:\python_classes>py test.py
Enter main string:abbababababacdefg
Enter sub string:a
Found at position 0
Found at position 3
Found at position 5
Found at position 7
Found at position 9
Found at position 11
```

```
D:\python_classes>py test.py
Enter main string:abbababababacdefg
Enter sub string:bb
Found at position 1
```

## **Counting substring in the given String:**

We can find the number of occurrences of substring present in the given string by using count() method.

- 1) s.count(substring) → It will search through out the string.
- 2) s.count(substring, bEgin, end) → It will search from bEgin index to end-1 index.



```
1) s="abcabcabcabcadda"  
2) print(s.count('a'))  
3) print(s.count('ab'))  
4) print(s.count('a',3,7))
```

### Output:

```
6  
4  
2
```

## Replacing a String with another String:

`s.replace(oldstring, newstring)`

inside s, every occurrence of old String will be replaced with new String.

### Eg 1:

```
s = "Learning Python is very difficult"  
s1 = s.replace("difficult", "easy")  
print(s1)
```

**Output:** Learning Python is very easy

### Eg 2: All occurrences will be replaced

```
s = "ababababababab"  
s1 = s.replace("a", "b")  
print(s1)
```

**Output:** bbbbbbbbbbbbbbb

## Q) String Objects are Immutable then how we can change the Content by using replace() Method

- Once we create a string object, we cannot change the content. This non-changeable behaviour is nothing but immutability. If we are trying to change the content by using any method, then with those changes a new object will be created and changes won't be happen in existing object.
- Hence with replace() method also a new object got created but existing object won't be changed.

### Eg:

```
s = "abab"  
s1 = s.replace("a", "b")  
print(s, "is available at :", id(s))  
print(s1, "is available at :", id(s1))
```



### Output:

abab is available at : 4568672  
bbbb is available at : 4568704

In the above example, original object is available and we can see new object which was created because of `replace()` method.

## Splitting of Strings:

- We can split the given string according to specified separator by using `split()` method.
- `l = s.split(separator)`
- The default separator is space. The return type of `split()` method is List.

```
1) s="durga software solutions"
2) l=s.split()
3) for x in l:
4)     print(x)
```

### Output:

durga  
software  
solutions

```
1) s="22-02-2018"
2) l=s.split('-')
3) for x in l:
4)     print(x)
```

### Output:

22  
02  
2018

## Joining of Strings:

We can join a Group of Strings (List OR Tuple) wrt the given Separator.

`s = separator.join(group of strings)`

### Eg 1:

```
t = ('sunny', 'bunny', 'chinny')
s = '-'.join(t)
print(s)
```

**Output:** sunny-bunny-chinny



### Eg 2:

```
l = ['hyderabad', 'singapore', 'london', 'dubai']  
s = ':'.join(l)  
print(s)
```

**Output:** hyderabad:singapore:london:dubai

## Changing Case of a String:

We can change case of a string by using the following 4 methods.

- 1) `upper()` → To convert all characters to upper case
- 2) `lower()` → To convert all characters to lower case
- 3) `swapcase()` → Converts all lower case characters to upper case and all upper case characters to lower case
- 4) `title()` → To convert all character to title case. i.e first character in every word should be upper case and all remaining characters should be in lower case.
- 5) `capitalize()` → Only first character will be converted to upper case and all remaining characters can be converted to lower case

```
1) s = 'learning Python is very Easy'  
2) print(s.upper())  
3) print(s.lower())  
4) print(s.swapcase())  
5) print(s.title())  
6) print(s.capitalize())
```

### Output:

```
LEARNING PYTHON IS VERY EASY  
learning python is very easy  
LEARNING pYTHON IS VERY eASY  
Learning Python Is Very Easy  
Learning python is very easy
```

## Checking Starting and Ending Part of the String:

Python contains the following methods for this purpose

- 1) `s.startswith(substring)`
- 2) `s.endswith(substring)`

```
1) s = 'learning Python is very easy'  
2) print(s.startswith('learning'))  
3) print(s.endswith('learning'))  
4) print(s.endswith('easy'))
```



### Output:

True  
False  
True

## To Check Type of Characters Present in a String:

Python contains the following methods for this purpose.

- 1) isalnum(): Returns True if all characters are alphanumeric( a to z , A to Z ,0 to9 )
- 2) isalpha(): Returns True if all characters are only alphabet symbols(a to z,A to Z)
- 3) isdigit(): Returns True if all characters are digits only( 0 to 9)
- 4) islower(): Returns True if all characters are lower case alphabet symbols
- 5) isupper(): Returns True if all characters are upper case alphabet symbols
- 6) istitle(): Returns True if string is in title case
- 7) isspace(): Returns True if string contains only spaces

### Eg:

- 1) `print('Durga786'.isalnum())` → True
- 2) `print('durga786'.isalpha())` → False
- 3) `print('durga'.isalpha())` → True
- 4) `print('durga'.isdigit())` → False
- 5) `print('786786'.isdigit())` → True
- 6) `print('abc'.islower())` → True
- 7) `print('Abc'.islower())` → False
- 8) `print('abc123'.islower())` → True
- 9) `print('ABC'.isupper())` → True
- 10) `print('Learning python is Easy'.istitle())` → False
- 11) `print('Learning Python Is Easy'.istitle())` → True
- 12) `print(' '.isspace())` → True

### Demo Program:

```
1) s=input("Enter any character:")
2) if s.isalnum():
3)     print("Alpha Numeric Character")
4)     if s.isalpha():
5)         print("Alphabet character")
6)         if s.islower():
7)             print("Lower case alphabet character")
8)         else:
9)             print("Upper case alphabet character")
10)    else:
11)        print("it is a digit")
12) elif s.isspace():
```



```
13) print("It is space character")
14) else:
15) print("Non Space Special Character")
```

```
D:\python_classes>py test.py
Enter any character:7
Alpha Numeric Character
it is a digit
```

```
D:\python_classes>py test.py
Enter any character:a
Alpha Numeric Character
Alphabet character
Lower case alphabet character
```

```
D:\python_classes>py test.py
Enter any character:$
Non Space Special Character
```

```
D:\python_classes>py test.py
Enter any character:A
Alpha Numeric Character
Alphabet character
Upper case alphabet character
```

## Formatting the Strings:

We can format the strings with variable values by using replacement operator {} and format() method.

```
1) name = 'durga'
2) salary = 10000
3) age = 48
4) print("{} 's salary is {} and his age is {}".format(name,salary,age))
5) print("{0} 's salary is {1} and his age is {2}".format(name,salary,age))
6) print("{x} 's salary is {y} and his age is {z}".format(z=age,y=salary,x=name))
```

### Output:

```
durga 's salary is 10000 and his age is 48
durga 's salary is 10000 and his age is 48
durga 's salary is 10000 and his age is 48
```



## Important Programs regarding String Concept

### Q1) Write a Program to Reverse the given String

Input: durga

Output: agrud

#### 1<sup>st</sup> Way:

```
1) s = input("Enter Some String:")
2) print(s[::-1])
```

#### 2<sup>nd</sup> Way:

```
1) s = input("Enter Some String:")
2) print(''.join(reversed(s)))
```

#### 3<sup>rd</sup> Way:

```
1) s = input("Enter Some String:")
2) i=len(s)-1
3) target=""
4) while i>=0:
5)     target=target+s[i]
6)     i=i-1
7) print(target)
```

### Q2) Program to Reverse Order of Words

Input: Learning Python is very Easy

Output: Easy Very is Python Learning

```
1) s=input("Enter Some String:")
2) l=s.split()
3) l1=[]
4) i=len(l)-1
5) while i>=0:
6)     l1.append(l[i])
7)     i=i-1
8) output=' '.join(l1)
9) print(output)
```

Output: Enter Some String:Learning Python is very easy!!  
easy!!! very is Python Learning





### **Q3) Program to Reverse Internal Content of each Word**

Input: Durga Software Solutions

Output: agruD erawtfoS snoituloS

```
1) s=input("Enter Some String:")
2) l=s.split()
3) l1=[]
4) i=0
5) while i<len(l):
6)     l1.append(l[i][::-1])
7)     i=i+1
8) output=' '.join(l1)
9) print(output)
```

### **Q4) Write a Program to Print Characters at Odd Position and Even Position for the given String?**

#### **1<sup>st</sup> Way:**

```
s = input("Enter Some String:")
print("Characters at Even Position:",s[0::2])
print("Characters at Odd Position:",s[1::2])
```

#### **2<sup>nd</sup> Way:**

```
1) s=input("Enter Some String:")
2) i=0
3) print("Characters at Even Position:")
4) while i< len(s):
5)     print(s[i],end=',')
6)     i=i+2
7) print()
8) print("Characters at Odd Position:")
9) i=1
10) while i< len(s):
11)     print(s[i],end=',')
12)     i=i+2
```



### **Q5) Program to Merge Characters of 2 Strings into a Single String by taking Characters alternatively**

Input: s1 = "ravi"

s2 = "reja"

Output: rtaevjia

```
1) s1=input("Enter First String:")
2) s2=input("Enter Second String:")
3) output=""
4) i,j=0,0
5) while i<len(s1) or j<len(s2):
6)     if i<len(s1):
7)         output=output+s1[i]
8)         i+=1
9)     if j<len(s2):
10)        output=output+s2[j]
11)        j+=1
12) print(output)
```

#### **Output:**

Enter First String:durga

Enter Second String:ravisoft

druarvgiasoft

### **Q6) Write a Program to Sort the Characters of the String and First Alphabet Symbols followed by Numeric Values**

Input: B4A1D3

Output: ABD134

```
1) s=input("Enter Some String:")
2) s1=s2=output=""
3) for x in s:
4)     if x.isalpha():
5)         s1=s1+x
6)     else:
7)         s2=s2+x
8) for x in sorted(s1):
9)     output=output+x
10) for x in sorted(s2):
11)     output=output+x
12) print(output)
```



### **Q7) Write a Program for the following Requirement**

Input: a4b3c2

Output: aaaabbbcc

```
1) s=input("Enter Some String:")
2) output=""
3) for x in s:
4)     if x.isalpha():
5)         output=output+x
6)         previous=x
7)     else:
8)         output=output+previous*(int(x)-1)
9) print(output)
```

**Note:** chr(unicode) → The corresponding character  
ord(character) → The corresponding unicode value

### **Q8) Write a Program to perform the following Activity**

Input: a4k3b2

Output: aeknbd

```
1) s=input("Enter Some String:")
2) output=""
3) for x in s:
4)     if x.isalpha():
5)         output=output+x
6)         previous=x
7)     else:
8)         output=output+chr(ord(previous)+int(x))
9) print(output)
```

### **Q9) Write a Program to Remove Duplicate Characters from the given Input String?**

Input: ABCDABBCDABBBCCCDDEEEF

Output: ABCDEF

```
1) s = input("Enter Some String:")
2) l=[]
3) for x in s:
4)     if x not in l:
5)         l.append(x)
6) output="".join(l)
7) print(output)
```



### **Q10) Write a Program to find the Number of Occurrences of each Character present in the given String?**

Input: ABCABCABBCDE

Output: A-3,B-4,C-3,D-1,E-1

```
1) s=input("Enter the Some String:")
2) d={}
3) for x in s:
4)     if x in d.keys():
5)         d[x]=d[x]+1
6)     else:
7)         d[x]=1
8) for k,v in d.items():
9)     print("{} = {} Times".format(k,v))
```

### **Q11) Write a Program to perform the following Task?**

Input: 'one two three four five six seven'

Output: 'one owt three ruof five xis seven'

```
1) s = input('Enter Some String:')
2) l = s.split()
3) l1 = []
4) i = 0
5) while i<len(l):
6)     if i%2==0:
7)         l1.append(l[i])
8)     else:
9)         l1.append(l[i][::-1])
10)    i=i+1
11) output=' '.join(l1)
12) print('Original String:',s)
13) print('output String:',output)
```

#### **Output:**

D:\durgaclasses>py test.py

Enter Some String:one two three four five six seven

Original String: one two three four five six seven

output String: one owt three ruof five xis seven



## Formatting the Strings:

- ☞ We can format the strings with variable values by using replacement operator {} and format() method.
- ☞ The main objective of format() method to format string into meaningful output form.

### Case- 1: Basic formatting for default, positional and keyword arguments

```
1) name = 'durga'
2) salary = 10000
3) age = 48
4) print("{} 's salary is {} and his age is {}".format(name,salary,age))
5) print("{0} 's salary is {1} and his age is {2}".format(name,salary,age))
6) print("{x} 's salary is {y} and his age is {z}".format(z=age,y=salary,x=name))
```

### Output:

durga 's salary is 10000 and his age is 48  
durga 's salary is 10000 and his age is 48  
durga 's salary is 10000 and his age is 48

### Case-2: Formatting Numbers

d → Decimal IntEger  
f → Fixed point number(float).The default precision is 6  
b → Binary format  
o → Octal Format  
x → Hexa Decimal Format (Lower case)  
X → Hexa Decimal Format (Upper case)

### Eg-1:

```
1) print("The intEger number is: {}".format(123))
2) print("The intEger number is: {:d}".format(123))
3) print("The intEger number is: {:5d}".format(123))
4) print("The intEger number is: {:05d}".format(123))
```

### Output:

The intEger number is: 123  
The intEger number is: 123  
The intEger number is: 123  
The intEger number is: 00123



### Eg-2:

```
1) print("The float number is: {}".format(123.4567))
2) print("The float number is: {:.f}".format(123.4567))
3) print("The float number is: {:.8.3f}".format(123.4567))
4) print("The float number is: {:.08.3f}".format(123.4567))
5) print("The float number is: {:.08.3f}".format(123.45))
6) print("The float number is: {:.08.3f}".format(786786123.45))
```

### Output:

The float number is: 123.4567  
The float number is: 123.456700  
The float number is: 123.457  
The float number is: 0123.457  
The float number is: 0123.450  
The float number is: 786786123.450

### Note:

- ☞ {:.08.3f}
- ☞ Total positions should be minimum 8.
- ☞ After decimal point exactly 3 digits are allowed. If it is less then 0s will be placed in the last positions
- ☞ If total number is < 8 positions then 0 will be placed in MSBs
- ☞ If total number is > 8 positions then all integral digits will be considered.
- ☞ The extra digits we can take only 0

**Note:** For numbers default alignment is Right Alignment (>)

### Eg-3: Print Decimal value in binary, octal and hexadecimal form

```
1) print("Binary Form:{0:b}".format(153))
2) print("Octal Form:{0:o}".format(153))
3) print("Hexa decimal Form:{0:x}".format(154))
4) print("Hexa decimal Form:{0:X}".format(154))
```

### Output:

Binary Form:10011001  
Octal Form:231  
Hexa decimal Form:9a  
Hexa decimal Form:9A

**Note:** We can represent only int values in binary, octal and hexadecimal and it is not possible for float values.



### Note:

- 1) `{:5d}` It takes an integer argument and assigns a minimum width of 5.
- 2) `{:8.3f}` It takes a float argument and assigns a minimum width of 8 including "." and after decimal point exactly 3 digits are allowed with round operation if required
- 3) `{:05d}` The blank places can be filled with 0. In this place only 0 allowed.

### Case-3: Number formatting for signed numbers

- ☯ While displaying positive numbers, if we want to include + then we have to write `{:+d}` and `{:+f}`
- ☯ Using plus for -ve numbers there is no use and for -ve numbers - sign will come automatically.

```
1) print("int value with sign:{:+d}".format(123))
2) print("int value with sign:{:+d}".format(-123))
3) print("float value with sign:{:+f}".format(123.456))
4) print("float value with sign:{:+f}".format(-123.456))
```

### Output:

```
int value with sign:+123
int value with sign:-123
float value with sign:+123.456000
float value with sign:-123.456000
```

### Case-4: Number formatting with alignment

- ☯ `<`, `>`, `^` and `=` are used for alignment
- ☯ `<` → Left Alignment to the remaining space
- ☯ `^` → Center alignment to the remaining space
- ☯ `>` → Right alignment to the remaining space
- ☯ `=` → Forces the signed(+) (-) to the left most position

**Note:** Default Alignment for numbers is Right Alignment.

### Ex:

```
1) print("{:5d}".format(12))
2) print("{:<5d}".format(12))
3) print("{:<05d}".format(12))
4) print("{:>5d}".format(12))
5) print("{:>05d}".format(12))
6) print("{:^5d}".format(12))
7) print("{:=5d}".format(-12))
8) print("{:^10.3f}".format(12.23456))
9) print("{:=8.3f}".format(-12.23456))
```



### Output:

```
12
12
12000
12
00012
12
-12
12.235
- 12.235
```

### Case-5: String formatting with format()

Similar to numbers, we can format String values also with format() method.

`s.format(string)`

```
1) print("{:5d}".format(12))
2) print("{:5}".format("rat"))
3) print("{:>5}".format("rat"))
4) print("{:<5}".format("rat"))
5) print("{:^5}".format("rat"))
6) print("{:*^5}".format("rat")) #Instead of * we can use any character(like +,$,a etc)
```

### Output:

```
12
rat
rat
rat
rat
*rat*
```

**Note:** For numbers default alignment is right where as for strings default alignment is left

### Case-6: Truncating Strings with format() method

```
1) print("{:.3}".format("durgasoftware"))
2) print("{:5.3}".format("durgasoftware"))
3) print("{:>5.3}".format("durgasoftware"))
4) print("{:^5.3}".format("durgasoftware"))
5) print("{:*^5.3}".format("durgasoftware"))
```

### Output:

```
dur
dur
dur
```





dur  
\*dur\*

### Case-7: Formatting dictionary members using format()

```
1) person={'age':48,'name':'durga'}  
2) print("{p[name]}s age is: {p[age]}".format(p=person))
```

#### Output:

durga's age is: 48

Note: p is alias name of dictionary

person dictionary we are passing as keyword argument

More convenient way is to use \*\*person

```
1) person={'age':48,'name':'durga'}  
2) print("{name}s age is: {age}".format(**person))
```

Output: durga's age is: 48

### Case-8: Formatting class members using format()

```
1) class Person:  
2)     age=48  
3)     name="durga"  
4) print("{p.name}s age is :{p.age}".format(p=Person()))
```

Output: durga's age is :48

```
1) class Person:  
2)     def __init__(self,name,age):  
3)         self.name=name  
4)         self.age=age  
5) print("{p.name}s age is :{p.age}".format(p=Person('durga',48)))  
6) print("{p.name}s age is :{p.age}".format(p=Person('Ravi',50)))
```

Note: Here Person object is passed as keyword argument. We can access by using its reference variable in the template string

### Case-9: Dynamic Formatting using format()

```
1) string="{:{fill}{align}{width}}"  
2) print(string.format('cat',fill='*',align='^',width=5))  
3) print(string.format('cat',fill='*',align='^',width=6))
```



```
4) print(string.format('cat',fill='*',align='<',width=6))
5) print(string.format('cat',fill='*',align='>',width=6))
```

### Output:

```
*cat*
*cat**
cat***
***cat
```

### Case-10: Dynamic Float format template

```
1) num="{:align}{width}.{precision}f"
2) print(num.format(123.236,align='<',width=8,precision=2))
3) print(num.format(123.236,align='>',width=8,precision=2))
```

### Output:

```
123.24
123.24
```

### Case-11: Formatting Date values

```
1) import datetime
2) #datetime formatting
3) date=datetime.datetime.now()
4) print("It's now:{:%d/%m/%Y %H:%M:%S}".format(date))
```

Output: It's now:09/03/2018 12:36:26

### Case-12: Formatting complex numbers

```
1) complexNumber=1+2j
2) print("Real Part:{0.real} and Imaginary Part:{0.imag}".format(complexNumber))
```

Output: Real Part: 1.0 and Imaginary Part: 2.0