(3.2) String

1. Introduction

str represents String data type.

A String is a sequence of characters enclosed within single quotes or double quotes.

1. Immutable
2. Ordered
3. Ierable / Sequential / Collection

In [22]:
```
s = "Hello World!"
print(s, type(s),id(s))
```

```
Hello World! <class 'str'> 1932864627056
```

Note:

In most of other languges 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.

In [1]:
```
a = 'hELLO wOLRLD'
print(a)
```

```
hELLO wOLRLD
```

In [9]:
```
s='hello \nworld!'
print(s)
```

```
hello
world!
```

In Python,we can represent char values also by using str type and explicitly char type is not available.

Eg:

In [14]:
```
c = 'a'
print(c, type(c))
```

```
a <class 'str'>
```

2. Multiline String

For this requirement we should go for triple single quotes(''') or triple double quotes(""")

```
s1='''durga
 soft'''
s1="""durga
 soft"""
```

In [12]:
```
#Multiline String

s= """hello
    be hurry up
bye bye"""

print(s)
```

```
hello
    be hurry up
bye bye
```

following is multiline comment but actual it is multiline string since it is not store any where in variable so it have been discarded as garbage value

In [14]:
```
"""
    comment1
    comment2
        ...
    commentn
"""
#  single line comment
print("Hello")
```

```
Hello
```

We can also use triple quotes to use single quote or double quote in our String.

In [2]:
```
#print hello world, 'Hi'

s = "hello world, 'Hi'"
print(s)
s = """hello world, 'Hi'"""
print(s)
```

```
hello world, 'Hi'
hello world, 'Hi'
```

In [7]:
```
#print hello world, "Hi"

s = 'hello world, "Hi"'
print(s)
s = '''hello world, "Hi"'''
print(s)
```

```
hello world, "Hi"
hello world, "Hi"
```

3. Escape sequence

\n, \t, \a

\ - line break, escape sequence

In [17]:
```
s = "Hello \
world"

print(s)
```

```
Hello world
```

In [21]:
```
#print he didn't said, "She is beautiful."

s = 'he didn\'t said, "she is beautiful."'
s1 = "he didn't said, \"she is beaytiful.\""
s2 = "he didn\'t said, \"she is beaytiful.\""

print(s)
print(s1)
print(s2)
```

```
he didn't said, "she is beautiful."
he didn't said, "she is beaytiful."
he didn't said, "she is beaytiful."
```

4. String Method

In [16]:
```
print(dir(str))
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'capitalize', 'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isascii', 'isdecimal', 'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']
```

A. Case Related 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

In [24]:
```
s = 'hello'

print(s, id(s))
s = s.upper()

print(s, id(s)) #id will be change bcz string is immutable

#string is immutable means value can not be changed
#other string will be create instead of doing change in same string
```

```
hello 1932860428016
HELLO 1932864446256
```

In [25]:
```
s = "HELLO WORLD!"
```

```
s.lower() #since string is immutable so data will not be changed,
          #other string will be create but we are not storing here

print(s)
```

```
HELLO WORLD!
```

In [26]:

```
s = "HELLO WORLD!"

s = s.lower()

print(s)
```

```
hello world!
```

In [27]:

```
s = "HeLLo WORlD!"
print("Original: ", s)
s1 = s.upper()
print("Upper   : ", s1)
s2 = s.lower()
print("Lower   : ", s2)
s3 = s.swapcase()
print("Swap    : ", s3)
s4 = s.title()
print("Title   : ", s4)
s5 = s.capitalize()
print("Capitalize: ",s5)
```

```
Original:  HeLLo WORlD!
Upper   :  HELLO WORLD!
Lower   :  hello world!
Swap    :  hEllO worLd!
Title   :  Hello World!
Capitalize:  Hello world!
```

B. Justification Methods

```
"Pankaj          " # Left justification
"          pankaj" # right justification
"     pankaj     " # center justification

"pankaj__________" # fillchar="_"
```

str.rjust(width, fillchar='') -> right

str.ljust(width, fillchar='') -> left

str.center(width, fillchar='') -> Center

```
Standard Output -> __str__ (print)

Raw Output      -> __repr__
```

In [28]:

```
s = "hello\n\n\tworld"
print(s) # standard output / __str__

print("-----------------------------------------")
```

```
print(repr(s)) # raw output / __repr__
```

```
hello

        world
-----------------------------------------
'hello\n\n\tworld'
```

In [30]:

```
s = "Pankaj Yadav"
print(repr(s))

s1 = s.ljust(30)
print(repr(s1))

s2 = s.center(30)
print(repr(s2))

s3 = s.rjust(30)
print(repr(s3))
```

```
'Pankaj Yadav'
'Pankaj Yadav                  '
'         Pankaj Yadav         '
'                  Pankaj Yadav'
```

In [33]:

```
s = "Hello World!".center(30, '^')
print(s)
```

```
^^^^^^^^^Hello World!^^^^^^^^^
```

In [34]:

```
for var in range(1, 6):
    print(("*"*var).rjust(5))
```

```
    *
   **
  ***
 ****
*****
```

In [35]:

```
s = "Arya College"

width = 30

print("Original : ", repr(s))
s1 = s.ljust(width, "_")
print("Left Just: ", repr(s1))
s2 = s.rjust(width, "-")
print("Right Just:", repr(s2))
s3 = s.center(width, "*")
print("Center    :", repr(s3))
```

```
Original :  'Arya College'
Left Just:  'Arya College__________________'
Right Just: '------------------Arya College'
Center    : '*********Arya College*********'
```

In [38]:

```
b = "1001"

print(b.zfill(10))
```

```
0000001001
```

In [40]:
```
b1 = "1"
b2 = "101"
b3 = "10001"
b4 = "110111"

for value in [ b1, b2, b3, b4]:
    print(value.rjust(6, '0'))
```

```
000001
000101
010001
110111
```

In [9]:
```
b1 = "1"
b2 = "101"
b3 = "10001"
b4 = "110111"

for value in [ b1, b2, b3, b4]:
    print(value.zfill(6))
```

```
000001
000101
010001
110111
```

Note:-

- < -> left
- > -> right
- ^ -> center

In [54]:
```
names = ['Sachin','Rajat','Ani','Yadvendra']
lang = ['python','c','java','go']

for n, l in zip(names,lang):
    s = f"Hello {n:^10} welcome to {l:^10} enjoy coding."
    print(s)
```

```
Hello   Sachin   welcome to   python   enjoy coding.
Hello   Rajat    welcome to     c      enjoy coding.
Hello    Ani     welcome to    java    enjoy coding.
Hello Yadvendra  welcome to     go     enjoy coding.
```

In [28]:
```
for n, l in zip(names, lang):
    s = f"Hello {n:<10} welcome to {l:<10} enjoy coding."
    print(s)
```

```
Hello Sachin     welcome to python     enjoy coding.
Hello Rajat      welcome to c          enjoy coding.
Hello Ani        welcome to java       enjoy coding.
Hello Yadvendra  welcome to go         enjoy coding.
```

In [29]:
```
for n, l in zip(names, lang):
    s = f"Hello {n:>10} welcome to {l:>10} enjoy coding."
    print(s)
```

```
Hello     Sachin welcome to     python enjoy coding.
Hello      Rajat welcome to          c enjoy coding.
Hello        Ani welcome to       java enjoy coding.
Hello  Yadvendra welcome to         go enjoy coding.
```

In [30]:
```
t = "Hello {:<10} welcome to {:<10} enjoy coding."

names = ['Sachin', 'Rajat', 'Ani', 'Yadevndra']
lang = ['python','c','java','go']

for n, l in zip(names, lang):
    print(t.format(n,l))
```

```
Hello Sachin     welcome to python     enjoy coding.
Hello Rajat      welcome to c          enjoy coding.
Hello Ani        welcome to java       enjoy coding.
Hello Yadevndra  welcome to go         enjoy coding.
```

C. Stripping / Trimming

It remove character form left,right, both side

str.lstrip(chars) -> left
str.rstrip(chars) -> right
str.strip(chars) -> left and right

bydefault chars = " ", "\n", "\t"

In [55]:
```
choice = input("choice: (yes/no)").strip().lower()

print(repr(choice))

if choice == 'yes' or choice == 'y':
    print("Continue Lecture")
else:
    print("We can Go Home")
```

```
choice: (yes/no)           y
'y'
Continue Lecture
```

In [40]:
```
s = "          hello        world          "
ls = s.lstrip() # will remove spaces from left side
rs = s.rstrip() # will remove spaces from right side
cs = s.strip()  # will remove spaces from begining and end
print("Original: ", repr(s))
print("left Strip: ",repr(ls))
print("right Strip: ",repr(rs))
print("Strip: ",repr(cs))
```

```
Original:  '          hello        world          '
left Strip:  'hello        world          '
right Strip:  '          hello        world'
Strip:  'hello        world'
```

In [41]:
```
s = "   \n\n\t\t    hello world\n\n\t\t\t"
print(repr(s))
print(repr(s.lstrip()))
print(repr(s.rstrip()))
print(repr(s.strip()))
```

```
'   \n\n\t\t    hello world\n\n\t\t\t'
'hello world\n\n\t\t\t'
'   \n\n\t\t    hello world'
'hello world'
```

In [44]:
```
s = "   \n\t\thello world\t\n\n    "
print(repr(s))
s1 = s.strip()
print(repr(s1))
s2 = s.strip(" ")
print(repr(s2))
```

```
'   \n\t\thello world\t\n\n    '
'hello world'
'\n\t\thello world\t\n\n'
```

In [45]:
```
s = "----****____sachin____****----"

s1 = s.strip('-').strip('*').strip("_")
print(s1)
```

```
sachin
```

In [46]:
```
s = "!@$#$&^$#@#@#%$%@#Sachin Yadav@$#%%^&%$#%"
# output = "Sachin Yadav"

s1 = s.strip("!@$^@%#&")
print(s1)
```

```
Sachin Yadav
```

D. Split

In [56]:
```
s = "let's break\nstring\tinto words"
# split - returns list of words
# words = [ "let's", "break", "string","into", "words"]
words = s.split()
print(words)
```

```
["let's", 'break', 'string', 'into', 'words']
```

In [57]:
```
s = "let's break\nstring\tinto words"
# split - returns list of words
# words = [ "let's", "break", "string",
#"into", "words"]
words = s.split(maxsplit=2)
print(words)
```

```
["let's", 'break', 'string\tinto words']
```

In [58]:
```
s = "hello-world-python-is-awesome"
words = s.split('-')
words
```

Out[58]:
```
['hello', 'world', 'python', 'is', 'awesome']
```

E. Join

In [59]:
```
words = ['hello', 'world', 'python', 'is', 'awesome']

char = " "
```

```
ss = char.join(words)

print(ss)
```

```
hello world python is awesome
```

In [64]:

```
s = "    s    a  chi   n     "

# op = "sachin"

words = s.split(" ")
print(words)
print("-----------------------------------------------------------------------------------------")
ns = "".join(words)
print(ns)
print("-----------------------------------------------------------------------------------------")
print("".join(s.split(' ')))
```

```
['', '', '', '', 's', '', '', '', 'a', '', 'chi', '', '', 'n', '', '', '', '', '']
-----------------------------------------------------------------------------------------
--------
sachin
-----------------------------------------------------------------------------------------
--------
sachin
```

In [65]:

```
s = "".join([ 'hi', 'hello', 'bye'])
print(s)
```

```
hihellobye
```

F. find() and index()

For forward direction:

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

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

For backward direction:

rfind()
rindex()

In [33]:

```
s="Learning Python is very easy"

print(s.find("Python")) #9
print(s.find("Java")) # -1
print(s.find("r"))#3
print(s.rfind("r"))#21
```

```
9
-1
3
21
```

s.find(substring,bEgin,end)

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

In [34]:
```
s="durgaravipavanshiva"
print(s.find('a'))#4
print(s.find('a',7,15))#10
print(s.find('z',7,15))#-1
```

```
4
10
-1
```

In [40]:
```
s = "Pankaj"

print(s.index("P"))
print(s.index("nk"))
print(s.index("a"))
print(s.rindex("a"))
print(s.index(r))
```

```
0
2
1
4
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
C:\Users\PANKAJ~1\AppData\Local\Temp/ipykernel_6800/3664114226.py in <module>
      5 print(s.index("a"))
      6 print(s.rindex("a"))
----> 7 print(s.index(r))

NameError: name 'r' is not defined
```

G. count()

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

In [41]:
```
s="abcabcabcabcadda"
print(s.count('a'))
print(s.count('ab'))
print(s.count('a',3,7))
```

```
6
4
2
```

H. replace()

s.replace(oldstring,newstring)

inside s, every occurrence of oldstring will be replaced with newstring.

In [42]:
```
s="Learning Python is very difficult"
print(s)
s1=s.replace("difficult","easy")
print(s1)
```

```
Learning Python is very difficult
```

```
Learning Python is very easy
```

In [43]:
```
s="abababababbab"
s1=s.replace("a","b")
print(s1)
```

```
bbbbbbbbbbbbbb
```

Q. String objects are immutable then how we can change the content by using replace() method.

Once we creates 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 happend in existing object.
Hence with replace() method also a new object got created but existing object won't be changed.

I. check type of characters

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 aplhabet symbols
6) istitle(): Returns True if string is in title case
7) isspace(): Returns True if string contains only spaces

In [44]:
```
print('Durga786'.isalnum()) #True
print('durga786'.isalpha()) #False
print('durga'.isalpha()) #True
print('durga'.isdigit()) #False
print('786786'.isdigit()) #True
print('abc'.islower()) #True
print('Abc'.islower()) #False
print('abc123'.islower()) #True
print('ABC'.isupper()) #True
print('Learning python is Easy'.istitle()) #False
print('Learning Python Is Easy'.istitle()) #True
print(' '.isspace()) #True
```

```
True
False
True
False
True
True
False
True
True
False
True
True
```

J. startwith(), endwith()

In [46]:
```
s='learning Python is very easy'
print(s.startswith('learning'))
```

```
print(s.endswith('learning'))
print(s.endswith('easy'))
```

```
True
False
True
```

5. Indexing on Strings

In Python Strings follows zero based index.
The index can be either +ve or -ve.
+ve index means forward direction from Left to Right
-ve index means backward direction from Right to Left

-12.....-1

\"HelloWorld"

0.....11

In [16]:
```
s = "Hello World!"
```

In [17]:
```
n = len( s )

print("There are ", n, "characters in string", s)

print("Index values will range from 0 to ", n-1)
```

```
There are  12 characters in string Hello World!
Index values will range from 0 to  11
```

In [18]:
```
print(s[1])

print(s[-11])

print(s[8])

print(s[-4])
```

```
e
e
r
r
```

Note:

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

In [19]:
```
print(s[40])
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
C:\Users\PANKAJ~1\AppData\Local\Temp/ipykernel_6360/3189330757.py in <module>
----> 1 print(s[40])

IndexError: string index out of range
```

6. Slicing

slice means a piece

[ ] operator is called slice operator,which can be used to retrieve parts of String.

string[Start:end:step]

start -> by default 0
end -> by default len(string)
step -> by default +1

if step = -ve
start -> by default -1
end -> by default -(len(string) + 1)

Note :- start(include) , end(exclude)

Behaviour of slice operator:

s[start:end:step]

step value can be either +ve or -ve
if +ve then it should be forward direction(left to right) and we have to consider start to end-1

if -ve then it should be backward direction(right to left) and we have to consider start to end+1

In [8]:
```
s = "HEllo world!"
print(s)
```

```
HEllo world!
```

In [9]:
```
s[ : ]
```

Out[9]:
```
'HEllo world!'
```

In [10]:
```
print(repr(s[3:7])) # start=3,end=7 but it will till 6,step=+1
```

```
'lo w'
```

In [11]:
```
s[0:40]
```

Out[11]:
```
'HEllo world!'
```

In [12]:
```
print(s[0:6:+1]) #s[Start:end:step]
print(s[:5]) #start = 0 , step = +1
print(s[0:5]) #step = +1

print(s[-12:-7]) # -12 to -8 with +1 step
print(s[:-7]) # 0 to -8 with +1 step
```

```
print(s[-12:5]) #-12 to 4 with +1 step
print(s[0:-7])
```

```
HEllo
HEllo
HEllo
HEllo
HEllo
HEllo
HEllo
```

In [13]:
```
print(s[-5: ]) #last five character

print(s[-5:-1])
```

```
orld!
orld
```

In [14]:
```
s[1:10:2] # Jump character by 2
```

Out[14]:
```
'El ol'
```

In [15]:
```
s[ : :2] # s[0:12:2]
```

Out[15]:
```
'Hlowrd'
```

In [17]:
```
s[-5:0]
```

Out[17]:
```
''
```

In [18]:
```
s[4:0]
```

Out[18]:
```
''
```

In [16]:
```
s = "AOWMEES"
print(s[ : :2]+s[1: :2])
```

```
AWESOME
```

In [20]:
```
s = "HELLO WORLD!"
s[4: :-1]
```

Out[20]:
```
'OLLEH'
```

In [21]:
```
s[::-1] #Reverse String (First step will be check)
```

Out[21]:
```
'!DLROW OLLEH'
```

In [22]:
```
s[-8: :-1]
```

Out[22]:
```
'OLLEH'
```

In [23]:
```
s[10:5:-1]
```

Out[23]:
```
'DLROW'
```

In [24]:
```
s[-4:1:-2]
```

Out[24]:
```
'RWOL'
```

Check Palindrome

In [88]:
```
x = input("x: ")

if x == x[::-1]:
    print("Palindrome")
else:
    print("Not Palindrome")
```

```
x: pannap
Palindrome
```

Q. Write a program to access each character of string in forward and backward direction by using while loop?

In [28]:
```
s = "Learning Python Is Very Easy"

print("Forward direction")
for i in s[::]:
    print(i,end='')

print("\nBackward direction")
for i in s[::-1]:
    print(i,end='')
```

```
Forward direction
Learning Python Is Very Easy
Backward direction
ysaE yreV sI nohtyP gninraeL
```

7. Mathematical Operators for String

We can apply the following mathematical operators for Strings.

1. + operator for concatenation
2. * operator for repetition

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

In [4]:
```
print("Pankaj "+"Yadav")
print("Pankaj "*3)
```

```
Pankaj Yadav
Pankaj Pankaj Pankaj
```

8. len() in-built function

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

In [26]:
```
s ="Pankaj"

len(s)
```

Out[26]:
```
6
```

Q. Write a program to access each character of string in forward and backward direction by using while loop?

In [30]:
```
s = "Learning Python Is Very Easy"
n = len(s)
i=0

print("Forward direction")
while i<n:
    print(s[i],end='')
    i+=1

print("\nBackward direction")
i = -1
while i>=-n:
    print(s[i],end='')
    i -= 1
```

```
Forward direction
Learning Python Is Very Easy
Backward direction
ysaE yreV sI nohtyP gninraeL
```

9. 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

In [31]:
```
s = "Pankaj"

print('P' in s)
print('r' in s)
```

```
True
False
```

10. Comparison of Strings

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

In [32]:
```
s1=input("Enter first string:")
s2=input("Enter Second string:")

if s1==s2:
    print("Both strings are equal")
elif s1<s2:
    print("First String is less than Second String")
else:
    print("First String is greater than Second String")
```

```
Enter first string:Pankaj
Enter Second string:Kumar
First String is greater than Second String
```

In [ ]:

In [ ]:

In [ ]:

11. String Formatting

In [5]:
```
application_template = """
To
    The Headmaster,
    _______________
    _______________

Dear Sir/Ma'am,

    Due to ________ I am unable to come to school for _____ days.
    My name is ____ and I am a student of class ____.

    Please Grant me leave for ____ days.

Your Student
_____________
_____________
_____________
"""
```

In [6]:
```
print(application_template)
```

```
To
    The Headmaster,
    _______________
    _______________

Dear Sir/Ma'am,

    Due to ________ I am unable to come to school for _____ days.
    My name is ____ and I am a student of class ____.

    Please Grant me leave for ____ days.

Your Student
_____________
_____________
_____________
```

A. Old Technique

string formating

type specifiers

%d - integers

%f - float

%s - string

In [47]:
```
name = 'Pankaj Yadav'
age = 20
country = "India"

info = "My name is %s and I am %d years old. I live in %s."
print(info)

print(info%(name, age, country))
```

```
My name is %s and I am %d years old. I live in %s.
My name is Pankaj Yadav and I am 20 years old. I live in India.
```

In [48]:
```
name = 'Pankaj Yadav'
age = 20
country = "India"
height = 135.2

info = "My name is %s and I am %d years old. I live in %s.\
My heigh is approx %.2f cm."%(name, age, country, height)
print(info)
# pricision points
```

```
My name is Pankaj Yadav and I am 20 years old. I live in India.My heigh is approx 135.20 c
m.
```

In [49]:
```
s = "Growth of google 2020 year will be approx '%-10.2f', let's whats happens next"%(12.45
print(s)
s = "Growth of google 2019 year will be approx '%10.2f', let's whats happens next"%(112.45
print(s)
```

```
Growth of google 2020 year will be approx '12.46     ', let's whats happens next
Growth of google 2019 year will be approx '    112.46', let's whats happens next
```

B. Python way

format method of string - will work on all versions of python

"{} {}".format(var_1, var_2)

f-string - it will only work on python version >= 3.6

f"{var_1}, {var_2}"

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

{} -> palace holder, buffer space, type specifier, 'replacement fields'

In [93]:
```
name = input("Enter your name: ")

s = "Welcome!! User {}, to the world of Python."
```

```
print(s)
```

```
Enter your name: pankaj
Welcome!! User {}, to the world of Python.
```

In [94]:

```
name = input("Enter your name: ")
s = "Welcome!! User {}, to the world of Python.".format(name)

print(s)
```

```
Enter your name: Pankaj
Welcome!! User Pankaj, to the world of Python.
```

In [50]:

```
x = int( input("X: ") )
y = int( input("Y: ") )
r = x + y

template = """
            x = {}

            y = {}

            {} + {} = {}
"""

print(template.format(x, y, x, y, r ))
#                     0, 1, 2, 3, 4  positional formatting
```

```
X: 5
Y: 4

            x = 5

            y = 4

            5 + 4 = 9
```

In [97]:

```
x = int( input("X: ") )
y = int( input("Y: ") )
r = x + y

template = """

            x = {0}

            y = {1}

            {2} + {3} = {4}

"""

print(template.format(x, y, x, y, r ))
#                     0, 1, 2, 3, 4  positional formatting
```

```
X: 10
Y: 20


            x = 10

            y = 20
```

```
10 + 20 = 30
```

In [15]:

```
x = int( input("X: ") )
y = int( input("Y: ") )
r = x + y

template = """

            x = {0}

            y = {1}

            {0} + {1} = {2}

"""

print(template.format(x, y, r ))
#                     0, 1, 2,   positional formatting
```

```
X: 7
Y: 8


            x = 7

            y = 8

            7 + 8 = 15
```

In [16]:

```
x = int( input("X: ") )
y = int( input("Y: ") )
r = x + y

template = """

            x = {X}

            y = {Y}

            {X} + {Y} = {result}

"""

print(template.format(X=x, Y=y, result=r))
# keyword formatting
```

```
X: 6
Y: 7


            x = 6

            y = 7

            6 + 7 = 13
```

In [54]:

```
name = 'Pankaj Yadav'
```

```
age = 24
country = "India"
height = 135.2

info = "My name is {} and I am {} years old. I live in {}.\
My height is approx {:.2f} cm.".format(name, age, country, height)
print(info)
# pricision points
```

```
My name is Pankaj Yadav and I am 24 years old. I live in India.My height is approx 135.20 cm.
```

In [53]:

```
name = 'Pankaj Yadav'
age = 24
country = "India"
height = 135.2

info = "My name is {0} and I am {1} years old. I live in {2}.\
My height is approx {3:.2f} cm.".format(name, age, country, height)
print(info)
# pricision points
```

```
My name is Pankaj Yadav and I am 24 years old. I live in India.My height is approx 135.20 cm.
```

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)
```

In [55]:

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

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

In [57]:

```
print("The float number is: {}".format(123.4567))
print("The float number is: {:f}".format(123.4567))
print("The float number is: {:8.3f}".format(123.4567))
print("The float number is: {:08.3f}".format(123.4567))
print("The float number is: {:08.3f}".format(123.45))
print("The float number is: {:08.3f}".format(786786123.45))
```

```
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
```

In [58]:

```
print("Binary Form:{0:b}".format(153))
print("Octal Form:{0:o}".format(153))
```

```
print("Hexa decimal Form:{0:x}".format(154))
print("Hexa decimal Form:{0:X}".format(154))
```

```
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.

Case-3: 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
```

In [59]:
```
print("{:5d}".format(12))
print("{:<5d}".format(12))
print("{:<05d}".format(12))
print("{:>5d}".format(12))
print("{:>05d}".format(12))
print("{:^5d}".format(12))
print("{:=5d}".format(-12))
print("{:^10.3f}".format(12.23456))
print("{:=8.3f}".format(-12.23456))
```

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

Case-4: Truncating Strings with format() method

In [63]:
```
print("{:.3}".format("durgasoftware"))
print("{:5.3}".format("durgasoftware"))
print("{:>5.3}".format("durgasoftware"))
print("{:^5.3}".format("durgasoftware"))
print("{:*^5.3}".format("durgasoftware"))
```

```
dur
dur
  dur
 dur
*dur*
```

Case-5: Formatting dictionary members using format()

In [65]:
```
person={'age':20,'name':'Pankaj'}
print("{p[name]}'s age is: {p[age]}".format(p=person))
```

```
Pankaj's age is: 20
```

In [67]:
```
person={'age':20,'name':'Pankaj'}
print("{name}'s age is: {age}".format(**person))
```

```
Pankaj's age is: 20
```

Case-6: Formatting class members using format()

In [69]:
```
class Person:
    age=20
    name="Pankaj"
print("{p.name}'s age is :{p.age}".format(p=Person()))
```

```
Pankaj's age is :20
```

In [70]:
```
class Person:
    def __init__(self,name,age):
        self.name=name
        self.age=age
print("{p.name}'s age is :{p.age}".format(p=Person('Pankaj',20)))
```

```
Pankaj's age is :20
```

Case-7: Dynamic Formatting using format()

In [72]:
```
string="{:{fill}{align}{width}}"
print(string.format('cat',fill='*',align='^',width=5))
print(string.format('cat',fill='*',align='^',width=6))
print(string.format('cat',fill='*',align='<',width=6))
print(string.format('cat',fill='*',align='>',width=6))
```

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

Case-8: Dynamic Float format template

In [73]:
```
num="{:{align}{width}.{precision}f}"
print(num.format(123.236,align='<',width=8,precision=2))
print(num.format(123.236,align='>',width=8,precision=2))
```

```
123.24
  123.24
```

Case-9: Formatting Date values

In [74]:
```
import datetime
#datetime formatting
date=datetime.datetime.now()
print("It's now:{:%d/%m/%Y %H:%M:%S}".format(date))
```

```
It's now:23/01/2022 14:59:44
```

Case-10: Formatting complex numbers

In [76]:
```
complexNumber=1+2j
print("Real Part:{0.real} and Imaginary Part:{0.imag}"\
      .format(complexNumber))
```

```
Real Part:1.0 and Imaginary Part:2.0
```

In [ ]:
```

```