Python 1st code

- LETS WORK WITH NUMBER

```
In [1]: 10+5
```

```
Out[1]: 15
```

```
In [2]: 10-5
```

```
Out[2]: 5
```

```
In [3]: 10*5
```

```
Out[3]: 50
```

```
In [4]: 10/5
```

```
Out[4]: 2.0
```

```
In [5]: 10//5
```

```
Out[5]: 2
```

```
In [6]: 5+(5*5)
```

```
Out[6]: 30
```

```
In [7]: (5+5)*5
```

```
Out[7]: 50
```

```
In [8]: _ + 3
```

```
Out[8]: 53
```

```
In [9]: import sys
        sys.version
```

```
Out[9]: '3.12.7 | packaged by Anaconda, Inc. | (main, Oct  4 2024, 13:17:27) [MSC v.1929 6
        4 bit (AMD64)]'
```

```
In [10]: 1+1
         2+1
         3+1
```

```
Out[10]: 4
```

In [11]:
```
print(1+1)
print(1+2)
print(1+3)
```
```
2
3
4
```

In [12]:
```
a=10
b=20

c=a+b
print(c)
```
```
30
```

In [13]:
```
print(a)
print(b)
print(c)
```
```
10
20
30
```

In [14]:
```
num1=20
num2=30

add=num1+num2

print(add)
#print('the addition of ',num1,'and',num2,'is=',add)
```
```
50
```

In [15]:
```
num1=20
num2=30

add=num1+num2
print('the addition of--',num1,'and',num2,'is==',add)
```
```
the addition of-- 20 and 30 is== 50
```

In [16]:
```
22nd oct
```
```
  Cell In[16], line 1
    22nd oct
     ^
SyntaxError: invalid decimal literal
```

python variable concept=pythonidentifier concept .syntax of define variable //(variable name=variable value)// (identifier=value)

In [ ]:
```
NIT=15
NIT
```

In [ ]:
```
NIT=20
```

```
NIT
```

In [ ]:
```
V=15
V
```

In [ ]:
```
print(V)
print(NIT)
```

In [ ]:
```
NIT
```

In [ ]:
```
Nit
```

In [ ]:
```
v
```

In [ ]:
```
V
```

In [ ]:
```
1var=20
1var
```

In [ ]:
```
var$=56
var$
```

In [ ]:
```
var=67
var
```

In [ ]:
```
x_train, x_test=80,20
print(x_train)
print(x_test)
```

In [ ]:
```
a,b,c,d=10,20,30,40
print(a)
print(b)
print(c)
print(d)
```

In [ ]:
```
'e'=45
```

In [ ]:
```
aaaaaaaaaaaaaaaaaaa=78
a
```

In [ ]:
```
ABC=100
abc
```

In [ ]:
```
nit_=50
nit_
```

python identifier is completed

In [ ]:
```
1nit=20
1nit
```

In [ ]:
```
python identifier completed
```

***************TYPE CASTING*************** Type casting-converting each og data type to other data types Integer-covert

In [ ]:
```
1.Integer-converts float, boolean, and string, (only if string is a number)to but d
(if string is word it doesnot accept)and compare


2.Float  -converts int, string (only if string is a number)and boolean,but
3.Boolean-converts into all data types
4.Complex-converts into int, float, boolean, and string (only if string is a number
5.String -converts into all data types
```

In [ ]:
```
1.INTEGER-Other data types to int
```

In [ ]:
```
int(5.3) #float to int
```

In [ ]:
```
int(True)
```

In [ ]:
```
int('NIT')     #string to int doesnot work
```

In [ ]:
```
int('10')
```

In [ ]:
```
int(20+30j)    #complex to int does not work
```

In [ ]:
```
2.FLOAT -Converting other data types to float
```

In [ ]:
```
float(5)
```

In [ ]:
```
float(True)
```

In [ ]:
```
float('NIT')     # string to float doesnot work
```

In [ ]:
```
float('10.6')
```

In [ ]:
```
float(10+20j)  #cmplex to float does not work
```

In [ ]:
```
3.BOOLEAN-Converting other data types
```

In [ ]:
```
a.bool()-False
a.bool of non zero is True
```

In [ ]:
```
bool(1)
```

In [ ]:
```
bool(0)  #int to bool
```

In [ ]:
```
bool()
```

```
In [ ]: bool(1.5)
```

```
In [ ]: bool('NIT')
```

```
In [ ]: bool(10+20j)  #complex to bool
```

```
In [ ]: 4.COMPLEX
        a.In complex first value is taken as a and 2nd value as b
```

```
In [ ]: complex(1)
```

```
In [ ]: complex(2.5)
```

```
In [ ]: complex(True,False)  #boolean to complex
```

```
In [ ]: complex(False)
```

```
In [ ]: complex(4,2.3)
```

```
In [ ]: complex ('NIT')
```

```
In [ ]: complex('10')
```

```
In [ ]: 5.STRING
```

```
In [ ]: str(1)
```

```
In [ ]: str(3.5)
```

```
In [ ]: str(True)
```

```
In [ ]: str(10+20j)
```

```
In [ ]: PYTHON DATA STRUCTURES
```

```
In [ ]: 1.LIST
```

```
In [ ]: list=[]
        list
```

```
In [ ]: print(type(list))
```

```
In [ ]: list1=[10,20,30]
        list1  #list is an integer
```

```
In [ ]: list2=[10,73,30,66,60,76]
        list2
```

In [ ]:
```
list3=['one','two','three']
list3  # list of strings
```

In [ ]:
```
list4=['Asif',25,[54,86],[150,90]]
list4  #nested lists
```

In [ ]:
```
list5=[100,'Asif',17.59]
list5  #list of mixed data type
```

In [ ]:
```
list6=['Asif',25,[50,100],[150,90],{'john','david'}]
list6
```

In [ ]:
```
len(list6)
```

In [ ]:
```
list2
```

In [ ]:
```
LIST INDEXING
```

In [ ]:
```
list2[0]
```

In [ ]:
```
list3[0]
```

In [ ]:
```
list3[0][0]
```

In [ ]:
```
list3[-1]
```

In [ ]:
```
list4[-1]
```

In [ ]:
```
LIST SLICING
```

In [ ]:
```
mylist=['one','two','three','four','five','six','seven','eight']
mylist
```

In [ ]:
```
mylist[0:3]
```

In [ ]:
```
mylist[2:5]
```

In [ ]:
```
mylist[:2]
```

In [ ]:
```
mylist[-3:]
```

In [ ]:
```
mylist[-2:]
```

In [ ]:
```
mylist[-1]
```

In [ ]:
```
mylist[:]
```

In [ ]:
```
ADD, REMOVE&CHANGES ITEMS
```

In [ ]:
```
mylist
```

In [ ]:
```
mylist.append('nine')
mylist
```

In [ ]:
```
mylist.remove('nine')
mylist
```

In [ ]:
```
mylist.insert(1,'one')
mylist
```

In [ ]:
```
mylist.remove('one')
mylist
```

In [ ]:
```
mylist.pop()
mylist
```

In [ ]:
```
mylist.pop(7)
mylist
```

In [ ]:
```
del mylist[5]
mylist
```

In [ ]:
```
mylist[0]=1
mylist[1]=2
mylist[2]=3
mylist
```

In [ ]:
```
mylist.clear()
mylist
```

In [ ]:
```
COPYLIST
```

In [ ]:
```
mylist=['one','two','threee','four','five','six','seven','eight','nine']
mylist
```

In [ ]:
```
mylist1=mylist
```

In [ ]:
```
id(mylist),id(mylist1)
```

In [ ]:
```
mylist2=mylist.copy()
```

In [ ]:
```
id(mylist2)
```

In [ ]:
```
mylist[0]=1
```

In [ ]:
```
mylist
```

In [ ]:
```
mylist1
```

```
In [ ]: mylist2
```

```
In [ ]:          JOIN LISTS
```

```
In [ ]: list1=['one','two','three','four']
        list1
```

```
In [ ]: list2=['five','six','seven','eight']
        list2
```

```
In [ ]: list1.extend(list2)
        list1
```

```
In [ ]: LIST MEMBERSHIP
```

```
In [ ]: list1
```

```
In [ ]: 'one'in list1
```

```
In [ ]: 'ten' in list1
```

```
In [ ]: if 'three' in list1:
            print('three is present in the list')
        else:
            print('three is not present in the list')
```

```
In [ ]: if'eleven' in list1:
            print('eleven is present in the list')
        else:
            print('eleven is present in the list')
```

```
In [ ]: REVERSE AND SORT LIST
```

```
In [ ]: list1
```

```
In [ ]: list1.reverse()
        list1
```

```
In [ ]: mylist3=[9,5,2,99,12,88,34]
        mylist3.sort()
```

```
In [ ]: mylist3
```

```
In [ ]: mylist3=[9,5,2,99,12,88,34]
        mylist3.sort(reverse=True)
        mylist3
```

```
In [ ]: mylist4=[88,65,33,21,11,98]
        sorted(mylist4)
```

```
In [ ]: mylist4
```

```
In [ ]: loop through a list
```

```
In [ ]: list1
```

```
In [ ]: for i in list1:
            print(i)
```

```
In [ ]: for i in enumerate (list1):
            print(i)
```

```
In [ ]: list10=['one','two','three','four','one','one','two','three']
        list10
```

```
In [ ]: list10.count('one')
```

```
In [ ]: list10.count('two')
```

```
In [ ]: All/Any
```

```
In [ ]: l1=[1,2,3,4,0]
        l1
```

```
In [ ]: all(l1)
```

```
In [ ]: any(l1)
```

```
In [ ]: l2=[1,2,3,4,True,False]
```

```
In [ ]: all(l2)
```

```
In [ ]: any(l2)
```

```
In [ ]: 2.TUPLE
        tup1=()  # empty tuple
```

```
In [ ]: tup2=(10,20,30)
        tup2 # tuple of integer number
```

```
In [ ]: tup3=(10.5,23.5,30.5)
        tup3  # tuple of float number
```

```
In [ ]: tup4=('one','two','three')
        tup4
```

In [ ]:
```
tup5=('Asif',24,(50,100),(150,90))
tup5  # nested tuple
```

In [ ]:
```
tup6=(200,'Asif',17.98)
tup6    # tup with mixed data type
```

In [ ]:
```
tup7=('Asif',35,[50,100],[150,100],{'John','David'},(99,22,33))
tup7
```

In [ ]:
```
len(tup7)
```

In [ ]:
```
Tuple indexing
```

In [ ]:
```
tup2[0]
```

In [ ]:
```
tup4[0]
```

In [ ]:
```
tup4[-1]
```

In [ ]:
```
tup5[-1]
```

In [ ]:
```
Tuple slicing
```

In [ ]:
```
mytuple=('one','two','three','four','five','six','seven','eight')
mytuple
```

In [ ]:
```
mytuple[0:3]
```

In [ ]:
```
mytuple[2:5]
```

In [ ]:
```
mytuple[:3]
```

In [ ]:
```
mytuple[:2]
```

In [ ]:
```
mytuple[-3:]
```

In [ ]:
```
mytuple[-2:]
```

In [ ]:
```
mytuple[-1:]
```

In [ ]:
```
mytuple[:]
```

In [ ]:
```
mytuple
```

In [ ]:
```
Remove&change items
```

In [ ]:
```
mytuple
```

In [ ]:
```
del mytuple[0] # tuple are immutable which means we can't delete tuple items
```

In [ ]:
```
mytuple[0]=1 #tuple are immutable which means we can't change tuple items
```

In [ ]:
```
del mytuple #Deleting entire tuple object is possible
```

In [ ]:
```
loop through a tuple
```

In [ ]:
```
for i in mytuple:
    print(i)
```

In [ ]:
```
mytuple
```

In [ ]:
```
for i in enumerate (mytuple):
    print(i)
```

In [ ]:
```
Tuple membership
```

In [ ]:
```
mytuple
```

In [ ]:
```
'one' in mytuple # check if 'one' exist in the list
```

In [ ]:
```
'ten' in mytuple   #check if 'ten' exists in the list
```

In [ ]:
```
if 'three' in mytuple: #check if 'three' exists in the tuple
    print('three is present in the tuple.')
else:
    print('three is not present in the tuple.')
```

In [ ]:
```
if 'eleven' in mytuple:
    print('Eleven is present in the mytuple.')
else:
    print('Eleven is not present in mytuple.')
```

In [ ]:
```
index position
```

In [ ]:
```
mytuple
```

In [ ]:
```
mytuple.index('one') #index of first element equal to 'one'
```

In [ ]:
```
mytuple.index('five')
```

In [ ]:
```
mytuple1=('one','two','three','four','one','one','two','three')
```

In [ ]:
```
mytuple1
```

In [ ]:
```
mytuple1.index('three')
```

In [ ]:
```
mytuple.index('three') #repetation of element in tuple,index always print
```

In [ ]:
```
Sorting
```

In [ ]:
```
mytuple2=(789,89,2,35,789,10,40)
mytuple2
```

In [ ]:
```
sorted(mytuple2)
```

In [ ]:
```
sorted(mytuple2,reverse=True)
```

In [ ]:
```
Tuple count
```

In [ ]:
```
mytuple1
```

In [ ]:
```
mytuple1.count('one')
```

In [ ]:
```
mytuple1.count('two')
```

In [ ]:
```
mytuple.count('three')
```

In [ ]:
```
mytuple.count('four')
```

In [ ]:
```
3.SET
```

In [ ]:
```
myset={1,2,3,4,5}
myset
```

In [ ]:
```
len(myset)
```

In [ ]:
```
myset={1,1,2,2,3,4,5,5}
myset
```

In [ ]:
```
myset1={1.79,2.8,3.99,4.56,5.45}
myset1
```

In [ ]:
```
myset2={'ASIF','JOHN','TYRION'}
myset2
```

In [ ]:
```
myset3={10,20,"Hola",[11,22,32]}
myset3
```

In [ ]:
```
myset4=set()
print(type(myset4))
```

In [ ]:
```
myset5=set(('one','two','three','four'))
myset5
```

In [ ]:
```
myset={'one','two','three','four','six','seven','eight'}
myset
```

In [ ]:
```
for i in myset:
    print(i)
```

In [ ]:
```
for i in enumerate (myset):
    print(i)
```

In [ ]:
```
set membership
```

In [ ]:
```
myset
```

In [ ]:
```
'one' in myset
```

In [ ]:
```
'ten' in myset
```

In [ ]:
```
if 'three'in myset:
    print('three is present in the set')
else:
    print('three is not present in the set')
```

In [ ]:
```
if 'eleven'in myset:
    print('eleven is present in the set')
else:
    print('eleven is not present in the set')
```

In [ ]:
```
add &remove items
```

In [ ]:
```
myset={'eight','four','one','seven','six','three','two'}
myset
```

In [ ]:
```
myset.add('NINE')
myset
```

In [ ]:
```
myset.update(['TEN''ELEVEN','TWELVE'])
myset
```

In [ ]:
```
myset.remove('NINE')
myset
```

In [ ]:
```
myset.discard('ten')
myset
```

In [ ]:
```
myset.clear()
```

In [ ]:
```
myset
```

In [ ]:
```
del myset
```

In [ ]:
```
myset
```

In [ ]:
```
copy set
```

In [ ]:
```
myset= {'one','two','three','four','five','six','seven','eight'}
myset
```

In [ ]:
```
myset1=myset
```

In [ ]:
```
myset1
```

In [ ]:
```
id(myset1),id(myset)
```

In [ ]:
```
myset2=myset.copy()
```

In [ ]:
```
myset2
```

In [ ]:
```
id(myset2)
```

In [ ]:
```
myset.add('nine')
myset
```

In [ ]:
```
myset1
```

In [ ]:
```
myset2
```

In [ ]:
```
set operation
```

In [ ]:
```
a={1,2,3,4,5}
b={4,5,6,7,8}
c={8,9,10}
```

In [ ]:
```
a|b
```

In [ ]:
```
a.union(b)
```

In [ ]:
```
a.union(b,c)
```

In [ ]:
```
a.update(b,c)
```

In [ ]:
```
a
```

In [ ]:
```
a={1,2,3,4,5}
b={4,5,6,7,8}
```

In [ ]:
```
a&b
```

In [ ]:
```
a.intersection(b)
```

In [ ]:
```
a.intersection_update(b)
```

In [ ]:
```
a
```

In [ ]:
```
a={1,2,3,4,5}
b={4,5,6,7,8}
```

In [ ]:
```
a-b
```

In [ ]:
```
a.difference(b)
```

In [ ]:
```
b.difference(a)
```

In [ ]:
```
b.difference_update(a)
```

In [ ]:
```
b
```

In [ ]:
```
a={1,2,3,4,5}
b={4,5,6,7,8}
```

In [ ]:
```
a^b
```

In [ ]:
```
a.symmetric_difference(b)
```

In [ ]:
```
a.symmetric_difference_update(b)
```

In [ ]:
```
a
```

In [ ]:
```
a={1,2,3,4,5,6,7,8,9}
b={3,4,5,6,7,8}
c={10,20,30,40}
```

In [ ]:
```
b.issubset(a)
```

In [ ]:
```
b.issuperset(a)
```

In [ ]:
```
a.issuperset(b)
```

In [ ]:
```
a.isdisjoint(b)
```

In [ ]:
```
a.isdisjoint(c)
```

In [ ]:
```
b.isdisjoint(c)
```

In [ ]:
```
other built in function
```

In [ ]:
```
a
```

In [ ]:
```
sum(a)
```

In [ ]:
```
max(a)
```

In [ ]:
```
min(a)
```

In [ ]:
```
list(enumerate(a))
```

In [ ]:
```
d=sorted(a,reverse=True)
d
```

In [ ]:
```
sorted(a)
```

In [ ]:
```
sorted(d)
```

In [ ]:
```
4.dictionary
```

In [ ]:
```
d=dict()
d
```

In [ ]:
```
print(type(d))
```

In [ ]:
```
d={}
d
```

In [ ]:
```
d={1:'one',2:'two',3:'three'}
d
```

In [ ]:
```
d=dict({1:one',2:'two',3:'three')}
```

In [ ]:
```
d
```

In [ ]:
```
d1={'A':'one','B':'two','C':'three'}
d1
```

In [ ]:
```
d2={1:'one','A':'two',3:'three'}
d2
```

In [ ]:
```
d2.keys()
```

In [ ]:
```
d2.values()
```

In [ ]:
```
d2.items()
```

In [ ]:
```
d4={1:'one',2:'two','A':['asif','john','Maria'],'B':('Bat','Cat','hat')}
d4
```

In [ ]:
```
keys={'a','b','c','d'}
d5=dict.fromkeys(keys)
d5
```

In [ ]:
```
keys={'a','b','c','d'}
values=10
d6=dict.fromkeys(keys,values)
d6
```

In [ ]:
```
keys={'a','b','c','d'}
value=[10,20,30]
d6=dict.fromkeys(keys,value)
d6
```

In [ ]:
```
value.append(40)
d6
```

In [ ]:
```
Accessing items
```

In [ ]:
```
d1
```

In [ ]:
```
d1['A']
```

In [ ]:
```
d1.get('A')
```

In [ ]:
```
d7={'Name':'Asif','ID':74123,'DOB':1991,'job':'Analyst'}
d7
```

In [ ]:
```
d7['Name']
```

In [ ]:
```
d7.get('job')
```

In [ ]:
```
add,remove&change items
```

In [ ]:
```
d7
```

In [ ]:
```
d7['DOB']=1992
d7['job']='developer'
d7
```

In [ ]:
```
d8={'DOb':1995}
d7.update(d8)
d7
```

In [ ]:
```
d7['Address']='hyderabad'
d7
```

In [ ]:
```
d7.pop('job')
d7
```

In [ ]:
```
del[d7['Address']]
```

In [ ]:
```
d7
```

In [ ]:
```
d7.clear()
d7
```

In [ ]:
```
del d7
d7
```

In [ ]:
```
copy dictionary
```

In [ ]:
```
d7={'Name':'Asif','ID':74123,'DOB':1991,'job':'Analyst'}
d7
```

In [ ]:
```
d8=d7
```

In [ ]:
```
id(d7),id(d8)
```

In [ ]:
```
d9=d7.copy()
```

In [ ]:
```
d9
```

In [ ]:
```
id(d9)
```

In [ ]:
```
d7['job']='developer'
d7
```

In [ ]:
```
d8
```

In [ ]:
```
d9
```

In [ ]:
```
loop through dictionary
```

In [ ]:
```
d8={'Name':'Asif','ID':74123,'DOB':1991,'job':'developer'}
d8
```

In [ ]:
```
for i in d8:
    print(d8)
```

In [ ]:
```
for i in d8:
    print(d8[i])
```

In [ ]:
```
for i in d8:
    print(i,':',d8[i])
```

In [ ]:
```
Dictionary membership
```

In [ ]:
```
d7
```

In [ ]:
```
'Name' in d7
```

In [ ]:
```
'Asif' in d7
```

In [ ]:
```
'ID' in d7
```

In [ ]:
```
'Address' in d7
```

In [ ]:
```
Any&All
```

In [ ]:
```
d7
```

In [ ]:
```
d7[0]='mark'
d7
```

In [ ]:
```
any(d7)
```

In [ ]:
```
all(d7)
```

In [ ]:
```
4 types of data structures completed
```

In [ ]:
```
STRING
Escape character
```

In [2]:
```
print("Hello there!\nHow are you?\nI\'m doing fine.")
```

```
Hello there!
How are you?
I'm doing fine.
```

In [ ]:
```
row string
```

In [3]:
```
print(r"Hello there!\nHow are you?\'m doing fine.")
```

```
Hello there!\nHow are you?\'m doing fine.
```

In [ ]:
```
multiline string
```

In [4]:
```
print(
    """ Dear Alice,
    Eve's cat has been arrested for catnapping,
     cat burglary,and extoration.

    scicerely,
    Bob"""
)
```

```
 Dear Alice,
    Eve's cat has been arrested for catnapping,
     cat burglary,and extoration.

    scicerely,
    Bob
```

In [ ]:
```
indexing&slicing
```

```
In [5]: spam='Hello world!'
```

```
In [6]: spam
```

```
Out[6]: 'Hello world!'
```

```
In [7]: spam[0]
```

```
Out[7]: 'H'
```

```
In [8]: spam[4]
```

```
Out[8]: 'o'
```

```
In [9]: spam[-1]
```

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

```
In [ ]: slicing
```

```
In [10]: spam[0:5]
```

```
Out[10]: 'Hello'
```

```
In [12]: spam[:5]
```

```
Out[12]: 'Hello'
```

```
In [14]: spam[6:-1]
```

```
Out[14]: 'world'
```

```
In [15]: spam[::-1]
```

```
Out[15]: '!dlrow olleH'
```

```
In [16]: spam[:-1]
```

```
Out[16]: 'Hello world'
```

```
In [ ]: the in &  not operator
```

```
In [17]: 'hello' in 'hello world'
```

```
Out[17]: True
```

```
In [18]: 'HELLO' in 'hello world'
```

```
Out[18]: False
```

```
In [19]: 'cat' not in 'cat and dog'
```

```
Out[19]: False

In [ ]: upper(),lower(),title()

In [20]: spam

Out[20]: 'Hello world!'

In [21]: spam.upper()

Out[21]: 'HELLO WORLD!'

In [22]: spam.lower()

Out[22]: 'hello world!'

In [23]: spam.title()

Out[23]: 'Hello World!'

In [ ]: is upper()&is lower()methods

In [24]: spam='Hello world'
         spam

Out[24]: 'Hello world'

In [25]: spam.isupper()

Out[25]: False

In [26]: spam.islower()

Out[26]: False

In [27]: spam.istitle()

Out[27]: False

In [28]: 'HELLO'.isupper()

Out[28]: True

In [29]: 'hello'.islower()

Out[29]: True

In [ ]: startswith&endswith

In [30]: 'hello world!'.startswith('hello')
```

```
Out[30]: True
```

```
In [31]: 'hello world!'.endswith('with')
```

```
Out[31]: False
```

```
In [32]: 'abc123'.startswith('abcde')
```

```
Out[32]: False
```

```
In [ ]: join()split()
```

```
In [33]: ''.join(['my','name','is','gudu'])
```

```
Out[33]: 'mynameisgudu'
```

```
In [34]: ' '.join(['my','name','is','gudu'])
```

```
Out[34]: 'my name is gudu'
```

```
In [35]: ' abc '.join(['my','name','is','gudu'])
```

```
Out[35]: 'my abc name abc is abc gudu'
```

```
In [36]: 'my name is gudu.'.split()
```

```
Out[36]: ['my', 'name', 'is', 'gudu.']
```

```
In [37]: 'my abc name abc is abc gudu'.split('abc')
```

```
Out[37]: ['my ', ' name ', ' is ', ' gudu']
```

```
In [38]: 'my name is gudu'.split('m')
```

```
Out[38]: ['', 'y na', 'e is gudu']
```

```
In [ ]: rjust(),ljust(),center()
```

```
In [39]: 'hello'.rjust(10)
```

```
Out[39]: '     hello'
```

```
In [41]: 'hello'.ljust(10)
```

```
Out[41]: 'hello     '
```

```
In [42]: 'hello'.center(10)
```

```
Out[42]: '  hello   '
```

```
In [43]: 'hello'.center(20,'*')
```

```
Out[43]: '*******hello********'
```

```
In [ ]: removing & white space
        strip(),rstrip()&lstrip()
```

```
In [44]: spam='   hello world!  '
         spam
```

```
Out[44]: '   hello world!  '
```

```
In [45]: spam.strip()
```

```
Out[45]: 'hello world!'
```

```
In [46]: spam.lstrip()
```

```
Out[46]: 'hello world!  '
```

```
In [47]: spam.rstrip()
```

```
Out[47]: '   hello world!'
```

```
In [ ]: count method
```

```
In [49]: spam='one sheep two sheep three sheep four'
         spam
```

```
Out[49]: 'one sheep two sheep three sheep four'
```

```
In [50]: spam.count('e')
```

```
Out[50]: 9
```

```
In [51]: spam.count('sheep')
```

```
Out[51]: 3
```

```
In [52]: spam.count('e',6)
```

```
Out[52]: 8
```

```
In [ ]: replace method
```

```
In [53]: text="Hello worldl"
         text
```

```
Out[53]: 'Hello worldl'
```

```
In [54]: text.replace("worldl","planet")
```

```
Out[54]: 'Hello planet'
```

In [55]:
```
fruits='apple,banana,cherry,apple'
fruits
```

Out[55]:
```
'apple,banana,cherry,apple'
```

In [56]:
```
fruits.replace('apple','orange',1)
```

Out[56]:
```
'orange,banana,cherry,apple'
```

In [57]:
```
sentence= "I like apples, Apples are my favourite fruit"
sentence.replace("apples","oranges")
```

Out[57]:
```
'I like oranges, Apples are my favourite fruit'
```

In [ ]:
```
string assignment completed
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```