Python Data Stracture

List -->[]

- Collection Hetrogenoues Data set
  - Example digits,chr,spl char,space...etc
- Reprasents symbol like this []
- List Contain Dupliate data sett
- List Is a Mutable(can Modify,create,update,delete)

Tuple-->()

Set -->{}

In [17]:
```
l=[23,2.5,"vijay",'c','b','!',' ']
```

In [2]:
```
l
```

Out[2]:
```
[23, 2.5, 'vijay', 'c', 'b', '!', ' ']
```

In [5]:
```
len(l)
```

Out[5]:
```
7
```

In [7]:
```
l[-2]
```

Out[7]:
```
'!'
```

In [11]:
```
l[2:5]
```

Out[11]:
```
['vijay', 'c', 'b']
```

In [12]:
```
print(dir(list))
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir_
_', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem
__', '__gt__', '__hash__', '__iadd__', '__imul__', '__init__', '__init_subclass
__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__',
'__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__', '__setat
tr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'append', 'c
lear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'revers
e', 'sort']
```

In [13]:
```
l.append(15)
```

In [14]:
```
1 l
```

Out[14]: [23, 2.5, 'vijay', 'c', 'b', '!', ' ', 15]

In [15]:
```
1 l.clear()
```

In [18]:
```
1 l
```

Out[18]: [23, 2.5, 'vijay', 'c', 'b', '!', ' ']

In [19]:
```
1 l2=l.copy()
2 l2
```

Out[19]: [23, 2.5, 'vijay', 'c', 'b', '!', ' ']

In [20]:
```
1 l.append(30)
```

In [21]:
```
1 l2
```

Out[21]: [23, 2.5, 'vijay', 'c', 'b', '!', ' ']

In [22]:
```
1 l=[1,1,4,5,6,7,1,4,5,7,8]
2
```

In [23]:
```
1 l.count(1)
```

Out[23]: 3

In [24]:
```
1 for i in l:
2     print(i,"--->",l.count(i))
```

```
1 ---> 3
1 ---> 3
4 ---> 2
5 ---> 2
6 ---> 1
7 ---> 2
1 ---> 3
4 ---> 2
5 ---> 2
7 ---> 2
8 ---> 1
```

In [26]:
```
uniq=[]
for i in l:
    if i not in uniq:
        uniq.append(i)
```

In [27]:
```
uniq
```

Out[27]: [1, 4, 5, 6, 7, 8]

In [28]:
```
for i in uniq:
    print(i,"-->",l.count(i))
```

```
1 --> 3
4 --> 2
5 --> 2
6 --> 1
7 --> 2
8 --> 1
```

String with list

In [29]:
```
s="hello apssd cbit proddatur ap"
```

In [30]:
```
l=list(s)
```

In [32]:
```
s="APSSDC"
```

In [33]:
```
list(s)
```

Out[33]: ['A', 'P', 'S', 'S', 'D', 'C']

In [34]:
```
s="hello apssd cbit proddatur ap"
```

In [35]:
```
s.split()
```

Out[35]: ['hello', 'apssd', 'cbit', 'proddatur', 'ap']

In [36]:
```
s="apple 50 banana 8 piapple 50 grape 50"
```

In [38]:
```
l=s.split()
```

In [39]:
```
l
```

Out[39]: ['apple', '50', 'banana', '8', 'piapple', '50', 'grape', '50']

In [40]:
```
amount=[]
fruits=[]
for i in range(len(l)):
    if(i%2==0):
        fruits.append(l[i])
    else:
        amount.append(l[i])
```

In [41]:
```
print(fruits)
print(amount)
```

```
['apple', 'banana', 'piapple', 'grape']
['50', '8', '50', '50']
```

Task:

- prime Series -->10
  - 2 3 5 7 11 13 17 19 23 29
- even Series: -->10
- 2 4 6 8 10 12 14 16 18 20

output: 2 2 3 4 5 6 7 8 11 10 13 12 17 14 19 16 23 18 29 20

In [42]:
```
print(dir(list))
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir_
_', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem
__', '__gt__', '__hash__', '__iadd__', '__imul__', '__init__', '__init_subclass
__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__',
'__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__', '__setat
tr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'append', 'c
lear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'revers
e', 'sort']
```

In [43]:
```
l1=[1,2,3,4,5]
l2=[4,5,6,7,8]
l1.extend(l2)
l1
```

Out[43]: [1, 2, 3, 4, 5, 4, 5, 6, 7, 8]

In [45]:
```
l1.index(3)
```

Out[45]: 2

In [46]:
```
l1
```

Out[46]: [1, 2, 3, 4, 5, 4, 5, 6, 7, 8]

In [47]:
```
1 l1.append(20)
```

In [48]:
```
1 l1
```

Out[48]: [1, 2, 3, 4, 5, 4, 5, 6, 7, 8, 20]

In [52]:
```
1 l1.insert(5,50)# first argument is index second one is  value
```

In [53]:
```
1 l1
```

Out[53]: [1, 2, 3, 4, 5, 50, 4, 5, 6, 7, 8, 20, 5]

In [54]:
```
1 l1.pop()
```

Out[54]: 5

In [56]:
```
1 l1.pop(5)# pop allows only index position
```

Out[56]: 50

In [57]:
```
1 l1.remove(4)
```

In [62]:
```
1 l1
```

Out[62]: [20, 8, 7, 6, 5, 4, 5, 3, 2, 1]

In [59]:
```
1 l1.reverse()
```

In [61]:
```
1 l1
```

Out[61]: [20, 8, 7, 6, 5, 4, 5, 3, 2, 1]

In [63]:
```
1 l1.sort()# Asceding Order
```

In [65]:
```
1 l1.sort(reverse=True)# Descending Order
```

In [66]:
```
1 l1
```

Out[66]: [20, 8, 7, 6, 5, 5, 4, 3, 2, 1]

In [71]:
```
# Task:
# l=[5,13,3,2,6,7,12,4,8,1,6]
# after Sorting -->[1,2,3,4,5,6,6,7,8,12,13]
# ouput:1-->[5,13,3,2,6,1,4,6,7,8,12]
#       2-->[2,3,5,13,6,7,12,4,8,1,6]
#       3-->[13,5,3,2,6,12,8,7,6,4,1]

l=[5,13,3,2,6,7,12,4,8,1,6]
```

In [72]:
```
print(sorted(l))
```

```
[1, 2, 3, 4, 5, 6, 6, 7, 8, 12, 13]
```

In [73]:
```
l
```

Out[73]:
```
[5, 13, 3, 2, 6, 7, 12, 4, 8, 1, 6]
```

In [83]:
```
mid=len(l)//2
l[mid]
```

Out[83]:
```
7
```

In [84]:
```
# 1-->[5,13,3,2,6,1,4,6,7,8,12]
le=l[:mid]
rl=l[mid+1:]
print(le)
print(rl)
```

```
[5, 13, 3, 2, 6]
[12, 4, 8, 1, 6]
```

In [86]:
```
rl.sort()
```

In [87]:
```
rl
```

Out[87]:
```
[1, 4, 6, 8, 12]
```

In [89]:
```
le.append(l[mid])
```

In [91]:
```
le.extend(rl)
```

In [93]:
```
# 1-->[5,13,3,2,6,7,1,4,6,8,12]
le
```

Out[93]:
```
[5, 13, 3, 2, 6, 7, 1, 4, 6, 8, 12]
```

In [94]:
```
1 l
```

Out[94]: [5, 13, 3, 2, 6, 7, 12, 4, 8, 1, 6]

In [106]:
```
1 ll=l[:mid]
```

In [107]:
```
1 ll
```

Out[107]: [5, 13, 3, 2, 6]

In [108]:
```
1 rl=l[mid+1:]
2 rl
```

Out[108]: [12, 4, 8, 1, 6]

In [109]:
```
1 ll.sort()
```

In [110]:
```
1 ll
```

Out[110]: [2, 3, 5, 6, 13]

In [111]:
```
1 ll.append(l[mid])
```

In [112]:
```
1 ll.extend(rl)
```

In [113]:
```
1 ll
```

Out[113]: [2, 3, 5, 6, 13, 7, 12, 4, 8, 1, 6]

In [115]:
```
1 ll=l[:mid]
2 ll.sort(reverse=True)
3 ll
4 rl=l[mid+1:]
5 rl.sort(reverse=True)
6 rl
```

Out[115]: [12, 8, 6, 4, 1]

In [116]:
```
1 ll.append(l[mid])
```

In [117]:
```
1 ll.extend(rl)
```

In [118]:
```
1 ll
```

Out[118]: [13, 6, 5, 3, 2, 7, 12, 8, 6, 4, 1]

In [119]:
```
# st="ECE4EEE3CIV2CSE4"
# ouput:ECEECEECEECE
#       EEEEEEEEE
#       CIVCIV
#       CSECSECSECSE
```

In [121]:
```
data='cbit'
print(data*2)
```

```
cbitcbit
```

In [122]:
```
st="ECE4EEE3CIV2CSE4"
```

In [129]:
```
branches=[]
values=[]
s=''
for i in st:
    if(i.isalpha()):
#         print(i,end=" ")
        s=s+i
    else:
        branches.append(s)
        values.append(i)
        s=''
print(branches)
print(values)
```

```
['ECE', 'EEE', 'CIV', 'CSE']
['4', '3', '2', '4']
```

In [131]:
```
for i in range(len(branches)):
    print(branches[i]*int(values[i]))
```

```
ECEECEECEECE
EEEEEEEEE
CIVCIV
CSECSECSECSE
```

In [132]:
```
st="ECE4EEE3CIV2CSE4"
s=''
for i in st:
    if(i.isalpha()):
#         print(i,end=" ")
        s=s+i#""+E ---S="E",s="EC",s="ECE"
#     s=EEE
    else:
        print(s*int(i))
        s=''
```

```
ECEECEECEECE
EEEEEEEE
CIVCIV
CSECSECSECSE
```

In [134]:
```
s="PPPPPPYYYYYTTTTTTTTOOOOOHHHHHHHNNNNNN"
uniq=[]
us=""
for i in s:
    if i not in uniq:
        uniq.append(i)
        us=us+i
print(uniq)
print(us)
```

```
['P', 'Y', 'T', 'O', 'H', 'N']
PYTOHN
```

In [136]:
```
for i in uniq:
    print(i,"-->",s.count(i))
```

```
P --> 6
Y --> 5
T --> 8
O --> 5
H --> 7
N --> 6
```

In [164]:
```
# st="q2i3a5o2"
# ouput:qsilafoq
st="z2i3a5o2"
s=""
for i in st:
    if(i.isalpha()):
        temp=i
    else:
        s=s+(temp+chr(ord(temp)+int(i)))
print(s)
# print(s[::-1])
```

```
z|ilafoq
```

In [144]:
```
chr(ord('q')+int('2'))
```

Out[144]:
```
's'
```

In [163]:
```
# st="q2i3a5o2"
# ouput:qsilafoq
st="z2i3a5o2"
s=""
nu=''
for i in st:
    if(i.isalpha()):
        temp=i
    elif(i.isnumaric()):
        nu=nu+i
        s=s+(temp+chr(ord(temp)+int(i)))
print(s)
# print(s[::-1])
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-163-4e2049cd4217> in <module>
      7     if(i.isalpha()):
      8         temp=i
----> 9     elif(i.isnumaric()):
     10         nu=nu+i
     11         s=s+(temp+chr(ord(temp)+int(i)))

AttributeError: 'str' object has no attribute 'isnumaric'
```

In [162]:
```
chr(ord('z')+2)
```

Out[162]:
```
'|'
```

In [161]:
```
st="q2i3a10o2"
v=[]
ch=[]
n=''
for i in st:
    if(i.isnumeric()):
#         print(i)
        n=n+i
    elif(i.isalpha()):
        print(n)
        v.append(n)
        ch.append(i)
        n=''
# print(v)
# print(ch)
```

```
2
3
10
```

In [166]:
```
data=["cbit","vbit","pdtr","apssdc","ap"]
data2=[]
for string in data:
    data2.append(string[::-1])
data2
```

Out[166]:
```
['tibc', 'tibv', 'rtdp', 'cdsspa', 'pa']
```

In [167]:
```
names=['srirama','sai','sairam','Veera venkata satyanarayana','venkatesa',"H
```

In [168]:
```
sorted(names)
```

Out[168]:
```
['Hanuman',
 'Veera venkata satyanarayana',
 'ap',
 'sai',
 'sairam',
 'srirama',
 'venkatesa']
```

In [169]:
```
names.sort()
```

In [170]:
```
1 names
```

Out[170]:
```
['Hanuman',
 'Veera venkata satyanarayana',
 'ap',
 'sai',
 'sairam',
 'srirama',
 'venkatesa']
```

In [171]:
```
1 names.sort(key=len)
```

In [172]:
```
1 names
```

Out[172]:
```
['ap',
 'sai',
 'sairam',
 'Hanuman',
 'srirama',
 'venkatesa',
 'Veera venkata satyanarayana']
```

In [176]:
```
1 print(sorted(names,key=len,reverse=True))
```

```
['Veera venkata satyanarayana', 'venkatesa', 'Hanuman', 'srirama', 'sairam', 's
ai', 'ap']
```

Tuple

- It is also contain Collection Hetrogenoues Data set
- Example digits,chr,spl char,space...etc
- Reprasents symbol like this ()
- It is also Contain Dupliate data set
- It is Immutable

In [177]:
```
1 print(dir(tuple))
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__',
'__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewa
rgs__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__l
e__', '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__red
uce_ex__', '__repr__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__s
ubclasshook__', 'count', 'index']
```

In [178]:
```
1 t=(2,3,1,4,5)
```

In [179]:
```
1 t.count(3)
```

Out[179]: 1

In [180]:
```
1 t.index(2)
```

Out[180]: 0

In [181]:
```
1 t[:2]
```

Out[181]: (2, 3)

In [182]:
```
1 sum(t)
```

Out[182]: 15

In [183]:
```
1 max(t)
```

Out[183]: 5

In [184]:
```
1 min(t)
```

Out[184]: 1

Sets()

- It Can be reprasented by symbol like {}
- It Doesn't contains the Duplicate Values
- It also Mutable

In [185]:
```
1 s={3,2,4,7,8,3,2,1,5,0}
2
```

In [186]:
```
1 s
```

Out[186]: {0, 1, 2, 3, 4, 5, 7, 8}

In [187]:
```
1 print(dir(set))
```

```
['__and__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__',
'__eq__', '__format__', '__ge__', '__getattribute__', '__gt__', '__hash__', '__
iand__', '__init__', '__init_subclass__', '__ior__', '__isub__', '__iter__', '_
_ixor__', '__le__', '__len__', '__lt__', '__ne__', '__new__', '__or__', '__rand
__', '__reduce__', '__reduce_ex__', '__repr__', '__ror__', '__rsub__', '__rxor_
_', '__setattr__', '__sizeof__', '__str__', '__sub__', '__subclasshook__', '__x
or__', 'add', 'clear', 'copy', 'difference', 'difference_update', 'discard', 'i
ntersection', 'intersection_update', 'isdisjoint', 'issubset', 'issuperset', 'p
op', 'remove', 'symmetric_difference', 'symmetric_difference_update', 'union',
'update']
```

In [188]:
```
1 s
```

Out[188]: {0, 1, 2, 3, 4, 5, 7, 8}

In [189]:
```
1 s.pop()
```

Out[189]: 0

In [190]:
```
1 s
```

Out[190]: {1, 2, 3, 4, 5, 7, 8}

In [191]:
```
1 s1={10,20,30,2}
```

In [192]:
```
1 s.intersection(s1)
```

Out[192]: {2}

In [194]:
```
1 s.copy()
```

Out[194]: {1, 2, 3, 4, 5, 7, 8}

In [195]:
```
1 s
```

Out[195]: {1, 2, 3, 4, 5, 7, 8}

In [196]:
```
1 s1
```

Out[196]: {2, 10, 20, 30}

In [197]:
```
1 s=s1.copy()
2 s
```

Out[197]: {2, 10, 20, 30}

In [198]:
```
s.add(1000)
```

In [199]:
```
s
```

Out[199]: {2, 10, 20, 30, 1000}

In [202]:
```
avg=sum(s)//len(s)
avg
```

Out[202]: 212

In [203]:
```
s1
```

Out[203]: {2, 10, 20, 30}

In [204]:
```
s
```

Out[204]: {2, 10, 20, 30, 1000}

In [205]:
```
s.difference(s1)
```

Out[205]: {1000}

In [206]:
```
s="wearelearingPythoprogramming"
set(s)
```

Out[206]: {'P', 'a', 'e', 'g', 'h', 'i', 'l', 'm', 'n', 'o', 'p', 'r', 't', 'w', 'y'}

In [207]:
```
ord('a')
```

Out[207]: 97

In [208]:
```
ord('P')
```

Out[208]: 80

Dictionaries

-

- Word-->Meaining Format
- "Key" -->"Value" pair data in format
- mutable
- No Indexing/Slicing
  - Students:
    - pallavolu sai ram--> peddireddy sai ram-->Value

- Roll num->504-->Key
- Reprasent by flower bracess {}
- Short From is dict

In [ ]:
```
d={"Key1":"Value1","Key2 ":'Value2',"key3 ":"value3"}

```

In [209]:
```
nd={"CBIT":2008,"Python":1991,"Apssdc":2014}
```

In [210]:
```
nd
```

Out[210]:
```
{'CBIT': 2008, 'Python': 1991, 'Apssdc': 2014}
```

In [211]:
```
print(dir(dict))
```

```
['__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc_
_', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt_
_', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len_
_', '__lt__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__',
'__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'cle
ar', 'copy', 'fromkeys', 'get', 'items', 'keys', 'pop', 'popitem', 'setdefaul
t', 'update', 'values']
```

In [232]:
```
nd
```

Out[232]:
```
{'CBIT': 2008, 'Python': 1991, 'Apssdc': 2014}
```

In [233]:
```
nd.keys()#Getting all Keys From Dict
```

Out[233]:
```
dict_keys(['CBIT', 'Python', 'Apssdc'])
```

In [234]:
```
nd.values()
```

Out[234]:
```
dict_values([2008, 1991, 2014])
```

In [235]:
```
nd
```

Out[235]:
```
{'CBIT': 2008, 'Python': 1991, 'Apssdc': 2014}
```

In [238]:
```
nd['python']=1989
```

In [240]:
```
nd.popitem()
```

Out[240]:
```
('python', 1989)
```

In [241]:
```
1 nd
```

Out[241]:
```
{'CBIT': 2008, 'Python': 1989, 'Apssdc': 2014}
```

In [242]:
```
1 nd.pop("Python")
```

Out[242]:
```
1989
```

In [243]:
```
1 nd
```

Out[243]:
```
{'CBIT': 2008, 'Apssdc': 2014}
```

In [ ]:
```
1 #Function to create Contacts Application
2
```

In [2]:
```
1 myContacts={}
```

In [8]:
```
1  #Function To create a contact
2
3  def addContact(name,mobileNumber):
4      if name not in myContacts:
5          myContacts[name]=mobileNumber
6          print(" Data Saved Successfully..",name)
7      else:
8          print(name," Already exist")
9
10 name=input("Enter the Name ")
11 mobileNumber=int(input("Enter the mobile number"))
12 addContact(name,mobileNumber)
13
14
```

```
Enter the Name teja
Enter the mobile number152454
 Data Saved Successfully.. teja
```

In [9]:
```
1 myContacts
```

Out[9]:
```
{'muni': 1212121,
 'sirisha': 16257812535781578,
 'jumbooo': 90909,
 'teja': 152454}
```

In [12]:
```
#Update the Contacts
def updateContact(name):
    if name in myContacts:
        mobile=int(input("Enter the Mobile number for update "))
        myContacts[name]=mobile
        print(mobile," Updated Successfully")
    else:
        print("OOps..! Name Not exist for up date")
updateContact('muni')
```

```
Enter the Mobile number for update 5425467276276247
5425467276276247  Updated Successfully
```

In [259]:
```
# muni--->83627862386826
myContacts
```

Out[259]:
```
{'muni': 367387, 'vijay': 983648284858254823747823547784, 'sai': 1211111}
```

In [14]:
```
def deleteContact(name):
    if name in myContacts:
        myContacts.pop(name)
        print(" Deleted Successfully")
    else:
        print("OOps..! Name Not exist for up date")
deleteContact('teja')
```

```
 Deleted Successfully
```

In [15]:
```
myContacts
```

Out[15]:
```
{'muni': 5425467276276247, 'sirisha': 16257812535781578, 'jumbooo': 90909}
```

In [17]:
```
def displayAllContacts(d):
    if len(myContacts)<0:
        print("Empty contacts ")
    else:
        for k,v in myContacts.items():
            print(k,"-->",v)
displayAllContacts(myContacts)
```

```
muni --> 5425467276276247
sirisha --> 16257812535781578
jumbooo --> 90909
```

In [19]:
```
d="12311114321115657111"
d=set(list(d))
d
```

Out[19]:
```
{'1', '2', '3', '4', '5', '6', '7'}
```

In [ ]:
```
# Task:
#     #Search the  Contacts
#     #Find the freqvency of the given string
#         #Ex:number=12311114321115657111
#         #ex:name="wearelaeringPythonProgrmmingggggggg"
#         output1:1:11
#                 g:9
# Task:
#     input:{'g':4,'s':5,'a':3,'cbit':2}
#     ouput:ggggsssssaaacbitcbit
```

In [24]:
```
#Find the freqvency of the given string
#         #Ex:number=12311114321115657111
#         #ex:name="wearelaeringPythonProgrmmingggggggg"
#         output1:1:11
#                 g:9

name="wearelaeringPythonProgrmmingggggggg"
un=[]
frqDict={}
for char in name:
    if char not in un:
        un.append(char)
# print("Unique Char :",un)
for j in un:
    frqDict[j]=name.count(j)
# print(frqDict)
m=max(frqDict.values())
for k,v in frqDict.items():
    if m==v:
        print(k,"-->",v)

```

```
g --> 9
```

In [ ]:
```

```

In [ ]:
```
# python code to demonstrate working of reduce()

# importing functools for reduce()
import functools

# initializing list
lis = [ 1 , 3, 5, 6, 2, ]

# using reduce to compute sum of list
print ("The sum of the list elements is : ",end="")
print (functools.reduce(lambda a,b : a+b,lis))

# using reduce to compute maximum element from list
print ("The maximum element of the list is : ",end="")
print (functools.reduce(lambda a,b : a if a > b else b,lis))

```

In [213]:
```
import functools
```

In [217]:
```
functools.reduce?
```

In [229]:
```
# Python3 code to demonstrate working of
# Alternate vowels and consonents in String
# using zip_longest() + join() + loop
from itertools import zip_longest

# initializing string
test_str = "gaeifgsbou"

# printing original string
print("The original string is : " + test_str)

# Alternate vowels and consonents in String
# using zip_longest() + join() + loop
vowels = ['a', 'e', 'i', 'o', 'u']
test_vow = []
test_con = []
for ele in test_str:
    if ele in vowels:
        test_vow.append(ele)
    elif ele not in vowels:
        test_con.append(ele)
res = ''.join(''.join(ele) for ele in zip_longest(test_vow, test_con, fillva

# printing result
print("Alternate consonents vowels are: " + res)

```

```
The original string is : gaeifgsbou
Alternate consonents vowels are: agefigosub
```

In [227]:

```
# Python3 code to demonstrate working of
# Get positional characters from String
# using loop

# initializing string
test_str = "gfgisbest"

# printing original string
print("The original string is : " + test_str)

# initializing index list
indx_list = [1, 3, 4, 5, 7]

# Get positional characters from String
# using loop
res = ''
for ele in indx_list:
    res = res + test_str[ele]

# printing result
print("Substring of selective characters : " + res)

```

```
The original string is : gfgisbest
Substring of selective characters : fisbs
```

In [226]:

```
# Python3 code to demonstrate working of
# Suffix removal from String list
# using loop + remove() + endswith()

# initialize list
test_list = ['allx', 'lovex', 'gfg', 'xit', 'is', 'bestx']

# printing original list
print("The original list : " + str(test_list))

# initialize suffix
suff = 'x'

# Suffix removal from String list
# using loop + remove() + endswith()
for word in test_list[:]:
    if word.endswith(suff):
        test_list.remove(word)

# printing result
print("List after removal of suffix elements : " + str(test_list))

```

```
The original list : ['allx', 'lovex', 'gfg', 'xit', 'is', 'bestx']
List after removal of suffix elements : ['gfg', 'xit', 'is']
```

In [230]:

```
# Python3 code to demonstrate working of
# String construction from character frequency
# using loop

# initialize list
test_list = [('g', 4), ('f', 3), ('g', 2)]

# printing original list
print("The original list : " + str(test_list))

# String construction from character frequency
# using loop
res = ''
for char, freq in test_list:
    res = res + char * freq

# printing result
print("The constructed string is : " + str(res))

```

```
The original list : [('g', 4), ('f', 3), ('g', 2)]
The constructed string is : ggggfffgg
```

In [231]:
```
# Python3 code to demonstrate working of
# Get Nth word in String
# using loop

# initializing string
test_str = "GFG is for Geeks"

# printing original string
print("The original string is : " + test_str)

# initializing N
N = 3

# Get Nth word in String
# using loop
count = 0
res = ""
for ele in test_str:
    if ele == ' ':
        count = count + 1
        if count == N:
            break
        res = ""
    else :
        res = res + ele

# printing result
print("The Nth word in String : " + res)

```

```
The original string is : GFG is for Geeks
The Nth word in String : for
```

In [25]:
```
s="123456789"
s=list(s)
s
```

Out[25]:
```
['1', '2', '3', '4', '5', '6', '7', '8', '9']
```

In [26]:
```
n=[]
for i in s:
    n.append(int(i))
n
```

Out[26]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [27]:
```
sum(n)
```

Out[27]:
```
45
```

In [33]:
```
s="987654345678"
l=list(map(int,list(s)))
l
```

Out[33]: [9, 8, 7, 6, 5, 4, 3, 4, 5, 6, 7, 8]

In [34]:
```
l2=list(map(str,l))
l2
```

Out[34]: ['9', '8', '7', '6', '5', '4', '3', '4', '5', '6', '7', '8']

In [35]:
```
str(l2)
```

Out[35]: "['9', '8', '7', '6', '5', '4', '3', '4', '5', '6', '7', '8']"

In [39]:
```
''.join(l2)
```

Out[39]: '987654345678'

In [49]:
```
#conversion
#1.List To tuple
li=[1,2,3,4,5]
tu_data=tuple(li)
tu_data
#2.Tuple to List
l_data=list(tu_data)
l_data
#3.List to Set
set_data=set(l_data)
set_data
#4.set to List
l_data=list(set_data)
l_data
#5.tuple to set
s_data=set(tu_data)
s_data
#6.set to tuple
tu_data=tuple(s_data)
tu_data
#7.list to Dictionary # it is not possible
dic_data=dict(l_data)
dic_data
#8.tuple to dictionary # not possible
#9.set to Dictionary not possible'
#10.Dictionary to list # not possible
l=list({1:"a",2:'b',3:"c"})
l
#11 Dictionary to tuple # not possible
#12 Dictionary to set  # not possible
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-49-bb0361c7a93d> in <module>
     20 tu_data
     21 #7.list to Dictionary # it is not possible
---> 22 dic_data=dict(l_data)
     23 dic_data
     24 #8.tuple to dictionary # not possible

TypeError: cannot convert dictionary update sequence element #0 to a sequence
```

In [40]:
```
1
```

Out[40]: tuple

In [ ]:
```
1
```