In [1]:
```
l=[1, 0, 2, 0, 'hello', '', []]
list(filter(bool, l))
#Incase of boolean 0,"",[],()-->False
```

Out[1]:
```
[1, 2, 'hello']
```

In [ ]:
```
#Inbuild Containers(Collection of element)-->set,dict,list,tuple,string etc...
#Collections -->It is a module in which we have different type of Containers
1.OrderedDict --> ordered collection of keys and values
2.Counter --> will give the frequency of each and every character of a sequence in forom of dict
3.ChainMap -->
4.Dequeue-->
5. DefaultDict
```

In [ ]:
```
Counters --> will return a dictionary
1.You can pass any sequence of items (list,set,tuple)
2.You can pass dictionary(dictionary keys)
3.You can pass keyword argument
```

In [4]:
```
#Example: By passing sequence of items
from collections import Counter
print(Counter(["B","C","a","D","E","B","C"]))
#If you are passing any sequence then it will give you the frequency of each chatavyer
```

```
Counter({'B': 2, 'C': 2, 'a': 1, 'D': 1, 'E': 1})
```

In [6]:
```
#Example: By passing sequence of dictionary
from collections import Counter
print(Counter({"B":1,"C":2,"a":3,"D":4,"E":5,"B":5,"C":5}))
#If you are passing dictionary inside counter then it will give you counter frequence
```

```
Counter({'B': 5, 'C': 5, 'E': 5, 'D': 4, 'a': 3})
```

In [7]:
```
#Example: By passing keyword argument
from collections import Counter
print(Counter(A=3,B=5,C=7))
```

```
Counter({'C': 7, 'B': 5, 'A': 3})
```

In [16]:
```
#Ordered Dict
from collections import OrderedDict
print("Normal Dictionary")
d={}
d["a"]=1
d["b"]=2
d["c"]=98
d["d"]=4
d["e"]=5
print(d)
for k,v in d.items():
    print(k,v)
print("Ordered Dictionary")
od=OrderedDict()
od["a"]=1
od["b"]=2
od["d"]=4
od["c"]=3
print(od)
for k,v in od.items():
    print(k,v)
```

```
Normal Dictionary
{'a': 1, 'b': 2, 'c': 98, 'd': 4, 'e': 5}
a 1
b 2
c 98
d 4
e 5
Ordered Dictionary
OrderedDict([('a', 1), ('b', 2), ('d', 4), ('c', 3)])
a 1
b 2
d 4
c 3
```

In [18]:
```
#deletion and reinsertion
od=OrderedDict()
od["a"]=1
od["b"]=2
od["d"]=4
od["c"]=3
print(od)
od.pop("a")
od["a"]=1
print(od)
```

```
OrderedDict([('a', 1), ('b', 2), ('d', 4), ('c', 3)])
OrderedDict([('b', 2), ('d', 4), ('c', 3), ('a', 1)])
```

In [20]:
```
#Chain map --> Combine two or more dictionary into single one (Encapsulate two or more
#dictionary into single one)
from collections import ChainMap
d1={1:2,2:3}
d2={5:6,8:9,90:9}
d3={70:56,45:4545}
c=ChainMap(d1,d2,d3)
print(c)
```

```
ChainMap({1: 2, 2: 3}, {5: 6, 8: 9, 90: 9}, {70: 56, 45: 4545})
```

In [22]:
```
#Dequeue -->(Double Ended Queue)--> insertion and deletion can be done from both end(front and rear)
from collections import deque
queue=deque(["name","age","DOB"])
print(queue)
```

```
deque(['name', 'age', 'DOB'])
```

In [24]:
```
#Insertion in deque
#append --> add at the last
#appendleft --> add at the first position
from collections import deque
queue=deque(["name","age","DOB"])
queue.append("Hello")
print(queue)
queue.appendleft("World")
print(queue)
```

```
deque(['name', 'age', 'DOB', 'Hello'])
deque(['World', 'name', 'age', 'DOB', 'Hello'])
```

In [25]:
```
#Removing in deque
#pop --> delete from the last
#popleft --> delete from the from
from collections import deque
queue=deque(["name","age","DOB"])
queue.pop()
print(queue)
queue.popleft()
print(queue)
```

```
deque(['name', 'age'])
deque(['age'])
```

In [26]:
```
#DateTime -->This module helps to deal with date and time
#For current date --> date.today()
from datetime import date
today=date.today()
print(today)
```

```
2022-09-15
```

In [30]:
```
from datetime import date
today=date.today()
print(today)
```

```
2022-09-15
```

In [34]:
```
from datetime import datetime
today=datetime.fromtimestamp(1889082384)
print(today)
```

```
2029-11-11 14:36:24
```

In [33]:
```
from datetime import datetime
today=datetime.now()
print(today)
```

```
2022-09-15 21:14:19.465718
```

In [40]:
```
#Specific Time zone data
import pytz
import datetime
current_time=datetime.datetime.now(pytz.timezone("Asia/dhaka"))
print(current_time)
```

```
2022-09-15 21:48:36.795078+06:00
```

In [42]:
```
import datetime
Time=datetime.time(11,12,45,22)
print(Time.hour)
```

```
11
```

In [44]:
```
from datetime import datetime
a=datetime(1999,12,12,12,12)
print(a.year)
print(a.month)
```

```
1999
12
```

In [50]:
```
import time
def factorial(n):
    fact=1
    for i in range(1,n+1):
        fact=fact*i
    return fact
start=time.time()
print(start)
n=int(input("Enter data"))
x=factorial(n)
end=time.time()
print(end)
print(end-start)
```

```
1663257474.666979
Enter data5
1663257475.2088964
0.5419173240661621
```

In [ ]:
```
Timelimitexceed
10sec
```

In [ ]:
```
try except block with Else:
Else --> when no any error is enountered then else part will be executed
```

In [52]:
```
#Example:
try:
    print("try")
    print(10/2)
except:
    print("except")
else:
    print("Else")
finally:
    print("finally")
```

```
try
5.0
Else
finally
```

In [54]:
```
#Default dict vs normal dict
dict={1:2,3:4,5:5,6:9}
print(dict)
print(dict[8])
```

```
{1: 2, 3: 4, 5: 5, 6: 9}
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Input In [54], in <cell line: 4>()
      2 dict={1:2,3:4,5:5,6:9}
      3 print(dict)
----> 4 print(dict[8])

KeyError: 8
```

In [59]:
```
#default dict
from collections import defaultdict

d=defaultdict(lambda:"not present")
d[1]=2
d[2]=3
print(d)
print(d[4])
```

```
defaultdict(<function <lambda> at 0x00000172CC7F7430>, {1: 2, 2: 3})
not present
```

In [ ]:
```

```