In [31]:
```
#Lab:7
#Pgm:1

#i)
fruits={'apples':20,'bananas':50,'oranges':100}
```

In [48]:
```
#ii)
for f,q in fruits.items():
    print(f,'->',q)
```

```
apples -> 60
bananas -> 150
oranges -> 100
```

In [33]:
```
#iii)
print("There are",fruits.get('bananas'),"bananas")
```

```
There are 50 bananas
```

In [34]:
```
#iv)
print("No. of keys:",len(fruits))
```

```
No. of keys: 3
```

In [35]:
```
#v)
if 'graphs' in fruits:
    print("Graphs is Available")
else :
    print("Graphs is NOT Available")
```

```
Graphs is NOT Available
```

In [36]:
```
#vi)
if 'pears' in fruits:
    print("Pears is Available")
else :
    fruits['pears']=10
    print(fruits)
```

```
{'apples': 20, 'bananas': 50, 'oranges': 100, 'pears': 10}
```

In [37]:
```
#vii)
print("Asending Order :")
for i in sorted(fruits):
    print(i)
```

```
Asending Order :
apples
bananas
oranges
pears
```

In [47]:
```
#viii)
print("Desending Order :")
for i in sorted(fruits.values(),reverse=True):
    print(i)
```

```
Desending Order :
150
100
60
```

In [39]:
```
#ix)
fruits={'apples': 20, 'bananas': 50, 'oranges': 100, 'pears': 10}
del fruits["pears"]
print(fruits)
```

```
{'apples': 20, 'bananas': 50, 'oranges': 100}
```

In [40]:
```
#x)
def show():
    print(f'{fruits}')

#main:
show()
```

```
{'apples': 20, 'bananas': 50, 'oranges': 100}
```

In [41]:
```
#xi)
def add_fruits(fruits,name,quantity):
    fruits[name]=fruits.get(name)+quantity

#main:
add_fruits(fruits,'apples',40)
show()
```

```
{'apples': 60, 'bananas': 50, 'oranges': 100}
```

In [42]:
```
#xii)
#main:
add_fruits(fruits,'bananas',100)
print(fruits)
```

```
{'apples': 60, 'bananas': 150, 'oranges': 100}
```

In [43]:
```
#xiii)
show()
```

```
{'apples': 60, 'bananas': 150, 'oranges': 100}
```

In [44]:
```
#xiv)
import pickle
fruits={'apples':60,'bananas':150,'oranges':100}
file=open("mypicklefile","wb")
pickle.dump(fruits,file)
file.close()
```

In [45]:
```
import pickle
frut_prc=open("mypicklefile","rb")
fruits=pickle.load(frut_prc)
print(fruits)
```

```
{'apples': 60, 'bananas': 150, 'oranges': 100}
```

In [ ]:

In [ ]:

In [50]:

```
#Lab:7
#Pgm:2

customers={}
while True:
    a=input("Name: ")
    b=int(input("Phone No.: "))
    c=input("Emailid: ")
    d=input("Continue or '(Type Done)' Over: ")
    if d=='done':
        break
    key=a
    contacts=[b,c]
    customers[key]=contacts
    print('\n',customers)
```

```
Name: arul
Phone No.: 8870245563
Emailid: arulkumarark1924@gmail.com
Continue or '(Type Done)' Over: no

 {'arul': [8870245563, 'arulkumarark1924@gmail.com']}
Name: asha
Phone No.: 9595675455
Emailid: asharajkumar@gmai.com
Continue or '(Type Done)' Over: no

 {'arul': [8870245563, 'arulkumarark1924@gmail.com'], 'asha': [9595675455,
'asharajkumar@gmai.com']}
Name: raj
Phone No.: 9442555105
Emailid: rajasha@hotmail.com
Continue or '(Type Done)' Over: no

 {'arul': [8870245563, 'arulkumarark1924@gmail.com'], 'asha': [9595675455,
'asharajkumar@gmai.com'], 'raj': [9442555105, 'rajasha@hotmail.com']}
Name: swathi
Phone No.: 9488220700
Emailid: swathi2220@gmail.com
Continue or '(Type Done)' Over: done
```

In [93]:

```
if "rex" in customers:
    print(customers.get("rex"))
else:
    print("Not exists")
```

```
Not exists
```

In [94]:

```
customers.update({"rex":[9942002764,"rajkumar@bhc.edu"]})
```

In [95]:
```
print(customers)
```

```
{'arul': [8870245563, 'arulkumarark1924@gmail.com'], 'asha': [9595675455,
'asharajkumar@gmai.com'], 'raj': [9442555105, 'rajasha@hotmail.com'], 're
x': [9942002764, 'rajkumar@bhc.edu']}
```

In [96]:
```
for  i,j in customers.items():
    print("Name :",i,"\t","Contect :",j)
```

```
Name : arul      Contect : [8870245563, 'arulkumarark1924@gmail.com']
Name : asha      Contect : [9595675455, 'asharajkumar@gmai.com']
Name : raj       Contect : [9442555105, 'rajasha@hotmail.com']
Name : rex       Contect : [9942002764, 'rajkumar@bhc.edu']
```

In [97]:
```
print("Asending Order :")
for j in sorted(customers):
    print(j)
print()
print("Count of Customers :",len(customers))
```

```
Asending Order :
arul
asha
raj
rex

Count of Customers : 4
```

In [98]:
```
del customers["rex"]
print(customers)
```

```
{'arul': [8870245563, 'arulkumarark1924@gmail.com'], 'asha': [9595675455,
'asharajkumar@gmai.com'], 'raj': [9442555105, 'rajasha@hotmail.com']}
```

In [ ]:

In [ ]:

In [56]:

```
#Lab:7
#Pgm:3

f=open("D:\PSPR\Python Coding\dict_word.txt")
r=f.read()
words = [word.lower() for word in r.split()]
words.sort(reverse=True)
alphabet=str(words).split()

print("Desensing Order :")
for word in words:
    print(word,end=" ")
print()
print()
print()
for word in words:
    for j in word:
        print(j,end=" ")
alphabets = 0
print()
print()
for i in range(len(r)):
    if(r[i].isalpha()):
        alphabets = alphabets + 1

print()
print()
print("Total Number of Alphabets :",alphabets)
```

```
Desensing Order :
written without with with with will which when when well way very very v
ariables, use typing trace. together. to to to to to to time, through th
is therefore there the the the the the the the the the the the the the t
he the testifying syntax supports structures, stepping step, statements
standard stack source: source source so since simple, simple setting sem
antics. segmentation scripting reuse. reduces readability rapid raises q
uickest python's python's python python python python python prov
ides. programs programming programmers program program program program p
roductivity prints print power. platforms, packages, other or or or on.
on often, often of of of of object-oriented, no never modules modularity
makes make major maintenance. love local line library level learn langua
ge language itself, its it it it is is is is is is introspective interpr
eter interpreter interpreter interpreted, instead, inspection input incr
edibly increased in in in in high-level high-level hand, glue global fre
ely form for for for few fault. fast. fast fall extensive expressions, e
xisting exception. exception, evaluation error, encourages emphasizes ef
fective. edit-test-debug edit-test-debug easy: easy dynamic dynamic dyna
mic doesn't distributed. discovers development, debugging debugger debug
ger debug data cycle cycle cost connect components compilation combined
code code charge cause catch can built bug breakpoints, binding, binary
because be bad available attractive at as as as are arbitrary approach a
pplication and and and and and and and and an an an allows all add a a a
a a a a a a


w r i t t e n w i t h o u t w i t h w i t h w i t h w i l l w h i c h w
```

```
h e n w h e n w e l l w a y v e r y v e r y v a r i a b l e s , u s e t
y p i n g t r a c e . t o g e t h e r . t o t o t o t o t o t o t i m e
, t h r o u g h t h i s t h e r e f o r e t h e r e t h e t h e t h e t
h e t h e t h e t h e t h e t h e t h e t h e t h e t h e t h e t h e t h e t
e s t i f y i n g s y n t a x s u p p o r t s s t r u c t u r e s , s t
e p p i n g s t e p , s t a t e m e n t s s t a n d a r d s t a c k s o
u r c e : s o u r c e s o u r c e s o s i n c e s i m p l e , s i m p l
e s e t t i n g s e m a n t i c s . s e g m e n t a t i o n s c r i p t
i n g r e u s e . r e d u c e s r e a d a b i l i t y r a p i d r a i s
e s q u i c k e s t p y t h o n ' s p y t h o n ' s p y t h o n p y t h
o n p y t h o n p y t h o n p y t h o n p y t h o n p r o v i d e s . p
r o g r a m s p r o g r a m m i n g p r o g r a m m e r s p r o g r a m
p r o g r a m p r o g r a m p r o g r a m p r o d u c t i v i t y p r i
n t s p r i n t p o w e r . p l a t f o r m s , p a c k a g e s , o t h
e r o r o r o r o n . o n o f t e n , o f t e n o f o f o f o f o b j e
c t - o r i e n t e d , n o n e v e r m o d u l e s m o d u l a r i t y
m a k e s m a k e m a j o r m a i n t e n a n c e . l o v e l o c a l l
i n e l i b r a r y l e v e l l e a r n l a n g u a g e l a n g u a g e
i t s e l f , i t s i t i t i t i t i s i s i s i s i s i s i s i n t r o s p e
c t i v e i n t e r p r e t e r i n t e r p r e t e r i n t e r p r e t
e r i n t e r p r e t e d , i n s t e a d , i n s p e c t i o n i n p u
t i n c r e d i b l y i n c r e a s e d i n i n i n i n h i g h - l e v
e l h i g h - l e v e l h a n d , g l u e g l o b a l f r e e l y f o r
m f o r f o r f o r f e w f a u l t . f a s t . f a s t f a l l e x t e
n s i v e e x p r e s s i o n s , e x i s t i n g e x c e p t i o n . e
x c e p t i o n , e v a l u a t i o n e r r o r , e n c o u r a g e s e
m p h a s i z e s e f f e c t i v e . e d i t - t e s t - d e b u g e d
i t - t e s t - d e b u g e a s y : e a s y d y n a m i c d y n a m i c
d y n a m i c d o e s n ' t d i s t r i b u t e d . d i s c o v e r s d
e v e l o p m e n t , d e b u g g i n g d e b u g g e r d e b u g g e r
d e b u g d a t a c y c l e c y c l e c o s t c o n n e c t c o m p o n
e n t s c o m p i l a t i o n c o m b i n e d c o d e c o d e c h a r g
e c a u s e c a t c h c a n b u i l t b u g b r e a k p o i n t s , b i
n d i n g , b i n a r y b e c a u s e b e b a d a v a i l a b l e a t t
r a c t i v e a t a s a s a s a r e a r b i t r a r y a p p r o a c h a
p p l i c a t i o n a n d a n d a n d a n d a n d a n d a n d a n d a n
a n a n a l l o w s a l l a d d a a a a a a a a a

Total Number of Alphabets : 1313
```

In [ ]: