In [ ]:
```
#NAME: ANNAPOORNIMA S
#ROLL NO: 225229101
```

In [1]:
```
#LAB:7
#pgm:1

#1)
fruits={'apple':20,'bananas':50,'orange':100}
```

In [2]:
```
#2)
print(fruits)
```

```
{'apple': 20, 'bananas': 50, 'orange': 100}
```

In [3]:
```
#3)
print("There are",fruits.get('bananas'),"bananas")
```

```
There are 50 bananas
```

In [4]:
```
#4)
print("No. of keys:",len(fruits))
```

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

In [5]:
```
#5)
if 'graphs' in fruits:
    print("graphs is available")
else:
    print("graphs in Not available")
```

```
graphs in Not available
```

In [6]:
```
#6)
if 'pears' in fruits:
    print("pears in available")
else:
    fruits['pears']=10
    print(fruits)
```

```
{'apple': 20, 'bananas': 50, 'orange': 100, 'pears': 10}
```

In [7]:
```
#7)
print("Asending Order:")
for i in sorted(fruits):
    print(i)
```

```
Asending Order:
apple
bananas
orange
pears
```

In [8]:
```
#8)
print("Desending Order:")
for i in reversed(fruits):
    print(i)
```

```
Desending Order:
pears
orange
bananas
apple
```

In [9]:
```
#9)
fruits={'apple': 20, 'bananas': 50, 'oranges': 100, 'pears': 10}
del fruits["pears"]
print(fruits)
```

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

In [11]:
```
#10)
def show():
    print(f'{fruits}')
show()
```

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

In [14]:
```
#11)
def add_fruits(fruits,name,quantity):
    fruits[name]=fruits.get(name,0)+quantity
add_fruits(fruits,'apples',40)
print(fruits)
```

```
{'apple': 20, 'bananas': 50, 'oranges': 100, 'apples': 40}
```

In [15]:
```
#12)
add_fruits(fruits,'bananas',100)
print(fruits)
```

```
{'apple': 20, 'bananas': 150, 'oranges': 100, 'apples': 40}
```

In [16]:
```
#13)
show()
```

```
{'apple': 20, 'bananas': 150, 'oranges': 100, 'apples': 40}
```

In [21]:
```
#14)
import pickle
fruits={'apple':60,'bananas':150,'oranges':100}
file=open("mypicklefile","wb")
pickle.dump(fruits,file)
file.close()
```

In [22]:
```
import pickle
frut_prc=open("mypicklefile","rb")
fruits=pickle.load(frut_prc)
print(fruits)
```

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

In [3]:
```
#pgm:2
customers={}
n=int(input("No. of customers:"))
for i in range(n):
    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)
```

```
No. of customers:3
Name:vijay
phone no:9587567389
Emailid:vijay213@gamil.com
continue or '(Type Done)' over:no

 {'vijay': [9587567389, 'vijay213@gamil.com']}
Name:jo
phone no:7698398798
Emailid:jo1234@gamil.com
continue or '(Type Done)' over:no

 {'vijay': [9587567389, 'vijay213@gamil.com'], 'jo': [7698398798, 'jo1234@gami
l.com']}
Name:raj
phone no:9942002764
Emailid:rajkumar@bhc.edu
continue or '(Type Done)' over:done
```

In [2]:
```
if "rex" in customers:
    print(customers.get("rex"))
else:
    print("Not exists")
```

```
Not exists
```

In [15]:
```
customers.update({"raj":[9942002764,"rajkumar@bhc.edu"]})
```

In [16]:
```
print(customers)
```

```
{'vijay': [9587567389, 'vijay213@gamil.com'], 'jo': [7698398798, 'jo1234@gamil.
com'], 'raj': [9942002764, 'rajkumar@bhc.edu']}
```

In [17]:
```
for i in customers:
    print("Name:",i,"\t","contacts:",customers[i])
```

```
Name: vijay      contacts: [9587567389, 'vijay213@gamil.com']
Name: jo         contacts: [7698398798, 'jo1234@gamil.com']
Name: raj        contacts: [9942002764, 'rajkumar@bhc.edu']
```

In [18]:
```
print("Asending Order:")
for j in sorted(customers):
    print(j)
print()
print("count of customers:",len(customers))
```

```
Asending Order:
jo
raj
vijay

count of customers: 3
```

In [19]:
```
del customers["raj"]
print(customers)
```

```
{'vijay': [9587567389, 'vijay213@gamil.com'], 'jo': [7698398798, 'jo1234@gamil.
com']}
```

In [32]:
```
#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=" ")

alphabets = 0
for i in range(len(r)):
    if(r[i].isalpha()):
        alphabets = alphabets + 1
print()
print()
print()
print("Total Number of Alphabets :",alphabets)
```

```
Desensing Order :
written without with with with will which when when well way very very vari
ables, use typing trace. together. to to to to to to time, through this the
refore there the the the the the the the the the the the the the the the te
stifying syntax supports structures, stepping step, statements standard sta
ck source: source source so since simple, simple setting semantics. segment
ation scripting reuse. reduces readability rapid raises quickest python's p
ython's python python python python python python python provides. programs progra
mming programmers program program program program productivity 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 language language itself, its it it it
is is is is is is is introspective interpreter interpreter interpreter interpr
eted, instead, inspection input incredibly increased in in in in high-level
high-level hand, glue global freely form for for for few fault. fast. fast
fall extensive expressions, existing exception. exception, evaluation erro
r, encourages emphasizes effective. edit-test-debug edit-test-debug easy: e
asy dynamic dynamic dynamic doesn't distributed. discovers development, deb
ugging debugger debugger debug data cycle cycle cost connect components com
pilation combined code code charge cause catch can built bug breakpoints, b
inding, binary because be bad available attractive at as as as are arbitrar
y approach application and and and and and and and an an an allows all
add a a a a a a a a


Total Number of Alphabets : 1313
```

In [ ]: