Day objectives(6/9/19)

creating contacts using dictionary

search for a contact

packages and modules

regular expression

file expression

In [10]:
```
dict={"k1":"name","k2":"age","k3":"number"}
print(dict)
dict["k1"]
```

```
{'k1': 'name', 'k2': 'age', 'k3': 'number'}
```

Out[10]:
```
'name'
```

In [35]:
```
##name,phone no.
contacts={}
def addcontacts(name,phno):
    if name not in contacts:
        contacts[name]=phno
        print("contact added successfully")
    else:
        print("contact already exists")
addcontacts('name1',9908111125)
addcontacts('name1',9908111125)
addcontacts('name1',9908111126)
addcontacts('name2',9908111126)
addcontacts('name2',9908111125)
```

```
contact added successfully
contact already exists
contact already exists
contact added successfully
contact already exists
```

In [25]:
```
##searching a contact
## using key 'name'
def searchcontact(name):
    if name in contacts:
        print(name,':',contacts[name])
    else:
        print("doesnot exists")
searchcontact('name1')
searchcontact('name2')
searchcontact('name3')
```

```
name1 : 9908111125
name2 : 9908111126
doesnot exists
```

In [47]:
```
##MERGING 2 CONTACTS
##using update method
def mergecontacts(newcontacts):
    contacts.update(newcontacts)
    print(len(newcontacts.keys()),"contacts added successfully")
searchcontact('name2')
newcontacts={'name2':9908111125,'name3':9908111127}
mergecontacts(newcontacts)
searchcontact('name2')
```

```
name2 : 9908111126
2 contacts added successfully
name2 : 9908111125
```

In [62]:
```
#**
##dictionary :mobilecontacts
##add contacts
##key:'a',value:9908111128
##another dictionary
##key:'b',value:9908111129
def addcontacts(name1,phno):
        if name1 not in contacts:
            contacts[name1]=phno
            print("contact added successfully")
        else:
         print("contact already exists")
addcontacts("a",9908111125)
addcontacts('b',9900777777)
```

```
contact added successfully
contact added successfully
```

In [ ]:

packages and Modules

packages->collection of Modules

modules->collection of Modules

In [27]:
```
##prime
def isprime(n):
    for i in range(2,n+1):
        if(n%i==0):
            return False
        else:
            return True
n=int(input("enter n:"))
isprime(n)
```

```
enter n:7
```

Out[27]:
```
True
```

In [26]:
```
def isprime(n):
    for i in range(2,n+1):
        if(n%i==0):
            return False
        else:
            return True
n=int(input("enter n:"))
isprime(n)
```

```
enter n:10
```

Out[26]:
```
False
```

In [24]:
```
#*
import Packages
isprime(7)
```

Out[24]:
```
False
```

In [28]:
```
#*
from Packages.numerical import isprime
isprime(10)
```

Out[28]:
```
False
```

In [29]:
```
from Packages.numerical import isprime
isprime(7)
```

Out[29]:
```
False
```

In [11]:
```
##generating marks using "random" prebuilt function
import random
def generatemarks(n,lb,ub):
    for i in range(0,n):
        print(random.randint(lb,ub))
generatemarks(10,0,100)
```

```
50
92
37
88
40
57
42
3
50
14
```

In [31]:
```
import random
def generatemarks(n,lb,ub):
    for i in range(0,20):
        print(random.randint(lb,ub))
generatemarks(10,0,100)
```

```
58
92
68
37
92
45
77
49
27
29
95
0
72
46
10
91
6
61
99
11
```

Regular Expression

Phone Number:Pattern=^[6-9][0-9]{9}$

In [45]:
```
##Function for phonenumber validation using
##Regular Expression(we need to import re in the 1st step itself)
import re
def phnumvalidator(num):
    pattern='^[0][0-9]{9}$' #pattern is string format
    if re.match(pattern,num):#re.match is a prebuilt method for comparision(here
        print("valid")
    else:
        print("invalid")
phnumvalidator('0258258536')
```

```
valid
```

In [41]:
```
import re
def phnumvalidator(num):
    pattern='^[6-9][0-9]{9}$' #pattern is string format
    if re.match(pattern,num):#re.match is a prebuilt method for comparision(here
        print("valid")
    else:
        print("invalid")
phnumvalidator('9908111100')
```

```
valid
```

In [54]:
```
import re
def phnumvalidator(num):
    pattern='^[6-9][0-9]{9}$' #pattern is string format
    if re.match(pattern,str(num)):#re.match is a prebuilt method for comparision(
        print("valid")
    else:
        print("invalid")
phnumvalidator(9131231311)
```

```
valid
```

In [50]:
```
#*
import re
def phnumvalidator(num):
    pattern='^[0][0-9]{9}$' #pattern is string format
    if re.match(pattern,str(num)):#re.match is a prebuilt method for comparision(
        print("valid")
    else:
        print("invalid")
phnumvalidator(0258258536)
```

```
  File "<ipython-input-50-59d4bfba4318>", line 8
    phnumvalidator(0258258536)
                            ^
SyntaxError: invalid token
```

In [74]:
```
##pattern for email validation
##email id:username(starts with alphabets and numbers),domain(starts with @),exte
##pattern for :'^[a-z0-9][a-z0-9_.]{3,18}[@][a-z0-9]{4,18}[.][a-z]{2,4}$ #lb is f
###[]->extensions
import re
def mail(emailnum):
    pattern="^[a-z0-9][a-z0-9_.]{3,18}[@][a-z0-9]{4,18}[.][a-z]{2,4}$"
    if re.match(pattern,emailnum):
         print("valid")
    else:
        print("invalid")
mail("div3_.@gmail.com")
```

```
valid
```

In [83]:
```
import re
def mail(emailnum):
    pattern="^[a-z0-9][a-z0-9_.]{3,18}[@][a-z0-9]{4,18}[.][a-z]{2,4}$"
    if re.match(pattern,emailnum):
         print("valid")
    else:
        print("invalid")
mail("15ujjdfkbjkabf_.@gmail.com1111")
```

```
invalid
```

File Handling

It is a collection of related information

basic steps

opening a file

performing operations(write,read,append)

closing file

In [138]:
```
#syntax for opening of file: file_obj=open("filepath","mode") #without mode by de
f=open("datafiles/data.txt",'r')
print(f.read())
f.close()
```

```
line 1
line 2
line 3line 5line 5line 5line 6
 revanth is indian idol winner in 2017
 revanth is indian idol winner in 2017
 revanth is indian idol winner in 2017line 2
line 3
line 4line 2
line 3
line 4line 2
line 3
line 4line 2
line 3
line 4line 2
line 3
line 4line 2
line 3
line 4line 6
 revanth is indian idol winner in 2017
```

In [139]:
```
with open("datafiles/data.txt","a") as f:
    f.write("line 6")
```

In [140]:
```
with open("datafiles/data.txt","a") as f:
    f.write("\n revanth is indian idol winner in 2017")
    f.close()
```

In [141]:
```
with open("datafiles/data.txt","a") as f: #a will give unsupported operation so w
    f.read()
    print(f.tell())
    f.write("line 2\nline 3\nline 4")
    print(f.tell())
    f.seek(0)
    print(f.tell())
```

```
---------------------------------------------------------------------------
UnsupportedOperation                      Traceback (most recent call last)
<ipython-input-141-cac907e27e14> in <module>()
      1 with open("datafiles/data.txt","a") as f: #a will give unsupported oper
ation so we need to give
----> 2     f.read()
      3     print(f.tell())
      4     f.write("line 2\nline 3\nline 4")
      5     print(f.tell())

UnsupportedOperation: not readable
```

In [142]:
```
#tell-> curser
#seek-> position change of curser
with open("datafiles/data.txt","a+") as f: #a+ ->read mode,w+ ->read write
    f.read()
    print(f.tell())
    f.write("line 2\nline 3\nline 4")
    print(f.tell())
    f.seek(0)
    print(f.tell())
```

```
381
403
0
```

In [144]:
```
#readline() reads individual line
#read() reads entire file
#readlines() reads every line in entire file
with open("datafiles/data.txt") as f:
    print((f.read()))
    f.close()
```

```
line 1
line 2
line 3line 5line 5line 5line 6
 revanth is indian idol winner in 2017
 revanth is indian idol winner in 2017
 revanth is indian idol winner in 2017line 2
line 3
line 4line 2
line 3
line 4line 2
line 3
line 4line 2
line 3
line 4line 2
line 3
line 4line 2
line 3
line 4line 6
 revanth is indian idol winner in 2017line 6
 revanth is indian idol winner in 2017line 2
line 3
line 4
```

In [146]:
```
with open("datafiles/file.txt") as f: #txt in plaintext
    print(f.readlines())
```

```
['line 1\n', 'line 3']
```

In [147]:
```
with open("datafiles/data.txt") as f: #txt in plaintext
    data=f.read()
    for line in data:
        word=line.split()
        print(word,end="")
```

```
['l']['i']['n']['e'][]['1'][]['l']['i']['n']['e'][]['2'][]['l']['i']['n']['e']
[]['3']['l']['i']['n']['e'][]['5']['l']['i']['n']['e'][]['5']['l']['i']['n']
['e'][]['5']['l']['i']['n']['e'][]['6'][][]['r']['e']['v']['a']['n']['t']['h']
[]['i']['s'][]['i']['n']['d']['i']['a']['n'][]['i']['d']['o']['l'][]['w']['i']
['n']['n']['e']['r'][]['i']['n'][]['2']['0']['1']['7'][][]['r']['e']['v']['a']
['n']['t']['h'][]['i']['s'][]['i']['n']['d']['i']['a']['n'][]['i']['d']['o']
['l'][]['w']['i']['n']['n']['e']['r'][]['i']['n'][]['2']['0']['1']['7'][][]
['r']['e']['v']['a']['n']['t']['h'][]['i']['s'][]['i']['n']['d']['i']['a']['n']
[]['i']['d']['o']['l'][]['w']['i']['n']['n']['e']['r'][]['i']['n'][]['2']['0']
['1']['7']['l']['i']['n']['e'][]['2'][]['l']['i']['n']['e'][]['3'][]['l']['i']
['n']['e'][]['4']['l']['i']['n']['e'][]['2'][]['l']['i']['n']['e'][]['3'][]
['l']['i']['n']['e'][]['4']['l']['i']['n']['e'][]['2'][]['l']['i']['n']['e'][]
['3'][]['l']['i']['n']['e'][]['4']['l']['i']['n']['e'][]['2'][]['l']['i']['n']
['e'][]['3'][]['l']['i']['n']['e'][]['4']['l']['i']['n']['e'][]['2'][]['l']
['i']['n']['e'][]['3'][]['l']['i']['n']['e'][]['4']['l']['i']['n']['e'][]['2']
[]['l']['i']['n']['e'][]['3'][]['l']['i']['n']['e'][]['4']['l']['i']['n']['e']
[]['6'][][]['r']['e']['v']['a']['n']['t']['h'][]['i']['s'][]['i']['n']['d']
['i']['a']['n'][]['i']['d']['o']['l'][]['w']['i']['n']['n']['e']['r'][]['i']
['n'][]['2']['0']['1']['7']['l']['i']['n']['e'][]['6'][][]['r']['e']['v']['a']
['n']['t']['h'][]['i']['s'][]['i']['n']['d']['i']['a']['n'][]['i']['d']['o']
['l'][]['w']['i']['n']['n']['e']['r'][]['i']['n'][]['2']['0']['1']['7']['l']
['i']['n']['e'][]['2'][]['l']['i']['n']['e'][]['3'][]['l']['i']['n']['e'][]
['4']
```

In [148]:
```
with open("datafiles/file.txt") as f: #txt in plaintext
    data=f.read()
    for line in data:
        word=line.split()
        print(word,end="")
```

```
['l']['i']['n']['e'][]['1'][]['l']['i']['n']['e'][]['3']
```

In [150]:
```
with open("datafiles/file.txt") as f: #txt in plaintext
    fh=f.read()
    words=fh.split()
    print(words)
```

```
['line', '1', 'line', '3']
```

In [151]:
```
with open("datafiles/file1.txt") as f: #txt in plaintext
    fh=f.read()
    words=fh.split('$')
    print(words)
```

```
['revanth', 'is', 'my', 'fav', 'singer']
```

In [152]:
```
with open("datafiles/file2.txt") as f: #txt in python
    fh=f.read()
    words=fh.split('$')
    print(words)
```

```
['revanth', 'is', 'one', 'of', 'my', 'role', 'model']
```

In [153]:
```
def readFile(filepath):
    with open(filepath,'r') as f:
        filedata=f.read()  #reads entire file
    return filedata
filepath='datafiles/data.txt'
print(readFile(filepath))
```

```
line 1
line 2
line 3line 5line 5line 5line 6
 revanth is indian idol winner in 2017
 revanth is indian idol winner in 2017
 revanth is indian idol winner in 2017line 2
line 3
line 4line 2
line 3
line 4line 2
line 3
line 4line 2
line 3
line 4line 2
line 3
line 4line 2
line 3
line 4line 6
 revanth is indian idol winner in 2017line 6
 revanth is indian idol winner in 2017line 2
line 3
line 4
```

In [159]:
```
##1
def linecount(filename): #filename or filepath
    count=0 #to increment after every value
    with open(filename,'r') as f:
        for i in f:  #to get whole file
            count=count+1
    return count
filename='datafiles/data.txt'
print(readFile(filename))
```

```
line 1
line 2
line 3line 5line 5line 5line 6
 revanth is indian idol winner in 2017
 revanth is indian idol winner in 2017
 revanth is indian idol winner in 2017line 2
line 3
line 4line 2
line 3
line 4line 2
line 3
line 4line 2
line 3
line 4line 2
line 3
line 4line 2
line 3
line 4line 6
 revanth is indian idol winner in 2017line 6
 revanth is indian idol winner in 2017line 2
line 3
line 4
```

In [161]:
```
##2
def linecount(filename): #filename or filepath
    count=0 #to increment after every value
    with open(filename,'r') as f:
        for i in f:  #to get whole file
            count=count+1
    return count
filename='datafiles/file.txt'
print(readFile(filename))
```

```
line 1
line 3
```

In [162]:
```
##3
def linecount(filename): #filename or filepath
    count=0 #to increment after every value
    with open(filename,'r') as f:
        for i in f:  #to get whole file
            count=count+1
    return count
filename='datafiles/file1.txt'
print(readFile(filename))
```

```
revanth$is$my$fav$singer
```

In [163]:
```
##4
def linecount(filename): #filename or filepath
    count=0 #to increment after every value
    with open(filename,'r') as f:
        for i in f:  #to get whole file
            count=count+1
    return count
filename='datafiles/file2.txt'
print(readFile(filename))
```

```
revanth$is$one$of$my$role$model
```

In [180]:
```
#function to count number of words in a file
import re
def wordCount(filepath):
    pattern='[\n]'
    filedata=readFile(filepath)
    count=len(re.split(pattern,filedata))
    return count
filepath="datafiles/data.txt"
print(wordCount(filepath))
```

```
22
```

In [179]:
```
import re
def spaceCount(filepath):
    pattern='[ ]'
    filedata=readFile(filepath)
    count=len(re.split(pattern,filedata))
    return count
filepath="datafiles/data.txt"
print(spaceCount(filepath))
```

```
66
```

In [190]:

```python
#unique
import re
def uniqueword(filepath):
    with open(filepath,'r') as f:
        fh=f.read()
        words=fh.split()
        print(words)
    item=[]#
    for i in words:
        if i not in item:
            item.append(i)
    print(item)
uniqueword("datafiles/data.txt")

```

```
['line', '1', 'line', '2', 'line', '3line', '5line', '5line', '5line', '6', 're
vanth', 'is', 'indian', 'idol', 'winner', 'in', '2017', 'revanth', 'is', 'india
n', 'idol', 'winner', 'in', '2017', 'revanth', 'is', 'indian', 'idol', 'winne
r', 'in', '2017line', '2', 'line', '3', 'line', '4line', '2', 'line', '3', 'lin
e', '4line', '2', 'line', '3', 'line', '4line', '2', 'line', '3', 'line', '4lin
e', '2', 'line', '3', 'line', '4line', '2', 'line', '3', 'line', '4line', '6',
'revanth', 'is', 'indian', 'idol', 'winner', 'in', '2017line', '6', 'revanth',
'is', 'indian', 'idol', 'winner', 'in', '2017line', '2', 'line', '3', 'line',
'4']
['line', '1', '2', '3line', '5line', '6', 'revanth', 'is', 'indian', 'idol', 'w
inner', 'in', '2017', '2017line', '3', '4line', '4']
```

In [192]:
```
#for every elements in main list
  #checks if it is exists in unique list
  #if it doesnot exists ,add it to unique
  #else if it already exists,move on to the else
##unique
import re
def uniqueword(filepath):
    with open(filepath,'r') as f:
        fh=f.read()
        words=fh.split()
        print(words)
    item=[]#
    for i in words:
        if i not in item:
            item.append(i)
    return item
uniqueword("datafiles/data.txt")
```

```
['line', '1', 'line', '2', 'line', '3line', '5line', '5line', '5line', '6', 're
vanth', 'is', 'indian', 'idol', 'winner', 'in', '2017', 'revanth', 'is', 'india
n', 'idol', 'winner', 'in', '2017', 'revanth', 'is', 'indian', 'idol', 'winne
r', 'in', '2017line', '2', 'line', '3', 'line', '4line', '2', 'line', '3', 'lin
e', '4line', '2', 'line', '3', 'line', '4line', '2', 'line', '3', 'line', '4lin
e', '2', 'line', '3', 'line', '4line', '2', 'line', '3', 'line', '4line', '6',
'revanth', 'is', 'indian', 'idol', 'winner', 'in', '2017line', '6', 'revanth',
'is', 'indian', 'idol', 'winner', 'in', '2017line', '2', 'line', '3', 'line',
'4']
```

Out[192]:
```
['line',
 '1',
 '2',
 '3line',
 '5line',
 '6',
 'revanth',
 'is',
 'indian',
 'idol',
 'winner',
 'in',
 '2017',
 '2017line',
 '3',
 '4line',
 '4']
```

In [203]:
```
def wordfreq(filepath):
    with open(filepath,'r') as f:
        fh=f.read()
        words=fh.split()
        wordfq={}
    for i in words:
        if i not in wordfq:
            wordfq[i]=1
        else:
            wordfq[i]+=1
    return wordfq
wordfreq("datafiles/data.txt")
```

Out[203]:
```
{'line': 17,
 '1': 1,
 '2': 8,
 '3line': 1,
 '5line': 3,
 '6': 3,
 'revanth': 5,
 'is': 5,
 'indian': 5,
 'idol': 5,
 'winner': 5,
 'in': 5,
 '2017': 2,
 '2017line': 3,
 '3': 7,
 '4line': 6,
 '4': 1}
```

In [204]:
```
def wordfreq(filepath):
    with open(filepath,'r') as f:
        fh=f.read()
        words=fh.split()
        wordfq={}
    for i in words:
        if i not in wordfq:
            wordfq[i]=1
        else:
            wordfq[i]+=1
    return wordfq
wordfreq("datafiles/data.txt")
```

Out[204]:
```
{'line': 17,
 '1': 1,
 '2': 8,
 '3line': 1,
 '5line': 3,
 '6': 3,
 'revanth': 5,
 'is': 5,
 'indian': 5,
 'idol': 5,
 'winner': 5,
 'in': 5,
 '2017': 2,
 '2017line': 3,
 '3': 7,
 '4line': 6,
 '4': 1}
```

In [206]:
```
def wordfreq(filepath):
    with open(filepath,'r') as f:
        fh=f.read()
        words=fh.split()
        wordfq={}
    for i in words:
        if i not in wordfq:
            wordfq[i]=1
        else:
            wordfq[i]+=1
    return wordfq
wordfreq("datafiles/file.txt")
```

Out[206]:
```
{'line': 2, '1': 1, '3': 1}
```

In [207]:
```
def wordfreq(filepath):
    with open(filepath,'r') as f:
        fh=f.read()
        words=fh.split()
        wordfq={}
    for i in words:
        if i not in wordfq:
            wordfq[i]=1
        else:
            wordfq[i]+=1
    return wordfq
wordfreq("datafiles/file1.txt")
```

Out[207]:
```
{'revanth$is$my$fav$singer': 1}
```

set and its methods(to use duplicate elements)

add()

union()

intersection()

difference()

update()

In [215]:
```
dir(set)
```

Out[215]:
```
['__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__',
 '__xor__',
 'add',
 'clear',
 'copy',
 'difference',
 'difference_update',
 'discard',
 'intersection',
 'intersection_update',
 'isdisjoint',
 'issubset',
 'issuperset',
 'pop',
 'remove',
 'symmetric_difference',
 'symmetric_difference_update',
 'union',
 'update']
```

In [216]:
```
l=[12,"ppp",333,444,532,55,"vvv"]
print(set(l))
```

```
{'ppp', 12, 333, 532, 55, 'vvv', 444}
```

In [225]:
```
s1={"divija",1203,"iit",1999,"diot"}
s1.add("888")
s1
```

Out[225]:
```
{1203, 1999, '888', 'diot', 'divija', 'iit'}
```

In [241]:
```
s1={"divija",1203,"iit",1999,1999,"diot"}
s2={"nasty",83,1203,1999}
print(s1.union(s2))
print(s1.intersection(s2))
print(s1.difference(s2))
print(s1.difference_update(s2))
print(s1.update(s2))
```

```
{'divija', 'diot', 1203, 83, 'iit', 'nasty', 1999}
{1203, 1999}
{'divija', 'diot', 'iit'}
None
None
```

In [242]:
```
s2=s1.copy()
s2
```

Out[242]:
```
{1203, 1999, 83, 'diot', 'divija', 'iit', 'nasty'}
```

In [245]:
```
v1={28,"p"}
v2={25,"g"}
print(v1.difference(v2))
print(v1.intersection(v2))
```

```
{28, 'p'}
set()
```

In [249]:
```
##tasks
#program for generating multiplication of table
#i/p: (3,5,7)
#o/p: 3*5=15,
#     5*5=25,
#     7*5=35
def mult(n,lb,ub):
    for i in range(lb,ub+1):
        ans=n*i
        print(n,'x',i,'=',ans)
mult(3,5,7)

```

```
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
```

In [272]:
```
#design a program to find maximum,minimum
#and average of numbers in a list
#i/p:l=[1,10,9,9,8,8]
#o/p:1.find unique lst
#    2.find max and min values
#    3.find average
def min_max():
    l=[1,10,9,9,8,8]
    l1=[]
    for i in  l:
        if i not in l1:
            l1.append(i)
    print("unique list=",l1)
    print('max value=',max(l1))
    print("min value=",min(l1))
    j=0
    for i in range(len(l)):
        j=((j+l[i])//(len(l)))
    print("avearge=",j)
min_max()

```

```
unique list= [1, 10, 9, 8]
max value= 10
min value= 1
avearge= 1
```

In [264]:
```
##linear search in list
#if key is found print '1'
#else print '-1'
#l=[10,9,8,7,6],5
def linearsearch():
    l=[10,9,8,7,6]
    for item in range(len(l)):
        if l[item]==5:
            print("1")
        else:
            print("-1")
            break
linearsearch()
```

```
-1
```

In [257]:
```

##sqrt of num w/o using math package
def sqrt():
    l=[1,2,3,4,5]
    for i in l:
        j=i**(0.5)
        print("sqrt of" ,i, "is",j)
sqrt()
```

```
sqrt of 1 is 1.0
sqrt of 2 is 1.4142135623730951
sqrt of 3 is 1.7320508075688772
sqrt of 4 is 2.0
sqrt of 5 is 2.23606797749979
```

In [ ]:
```
#**
#using regular expression add name,phnum,email
import re
def contdict(name,phnum,email)
    name=str(name)
    phnum=str(phnum)
    email=str(email)
    pattern="^[a-zA-Z_.]{3,47}$"
    if(re.match(pattern,name)):
        contdict[name]=name
    else:
        print("invalid")
    pattern="^[6-9][0-9]{10}$"
    if(re.match(pattern,email))
```

In [273]:
```
##**
import re
def contactdictionary(name,phonenum,email):
    name=str(name)
    phonenum=str(name)
    email=str(email)
    contactdictionary={}
    pattern='^[a-zA-Z_.]{3,47}$'
    if(re.match(pattern,name)):
        contactdictionary[name]=name
    else:
        print("invalid")
    pattern='^[6-9][0-9]{10}$'
    if(re.match(pattern,phonenum)):
        contactdictionary[phonenum]=phonenum
    else:
        print("invalid")
    pattern='^[a-z0-9][a-z_.]{3,14}[@][a-z]{3,12}[.][a-z]{2,3}'
    if(re.match(pattern,email)):
        contactdictionary[email]=email
    else:
        print("invalid")
contactdictionary(divija,9223323210,divija@gmail.com)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-273-63d1c2533bd4> in <module>()
     20     else:
     21         print("invalid")
---> 22 contactdictionary(divija,9223323210,divija@gmail.com)

NameError: name 'divija' is not defined
```

In [ ]: