Joshna Mariya Jose

(Scroll down for Screenshots)

Problem Statement:

Build a file-based key-value data store that supports the basic CRD (create, read and delete) operations. This data store is meant to be used as a local storage for one single process on one laptop. The data store must be exposed as a library to clients that can instantiate a class and work with the data store.

The data store will support the following functional requirements.

1. It can be initialized using an optional file path. If one is not provided, it will reliably create itself in a reasonable location on the laptop.
2. A new key-value pair can be added to the data store using the create operation. The key is always a string – capped at 32 chars. The value is always a JSON object -capped at 16KB.
3. If Create is invoked for an existing key, an appropriate error must be returned.
4. A Read operation on a key can be performed by providing the key.
5. Every key supports setting a Time-To-Live property when it is created. This property is optional. If provided, it will be evaluated as an integer defining the number of seconds the key must be retained in the data store. Once the Time-To-Live for a key has expired, the key will no longer be available for read or delete operations.
6. Appropriate error responses must always be returned to a client if it uses the data store in unexpected ways or breaches any limits.

The data store will support the following non-functional requirements.

1. The size of the file storing data never exceed 1GB.
2. More than one client process cannot be allowed to use the same file as a data store at any given time.
3. A client process is allowed to access the data store using multiple threads, if it desires to. The data store must therefore be thread-safe.
4. The client will bear as little memory costs as possible to use this data store, while deriving maximum performance with respect to response time for accessing the data store.

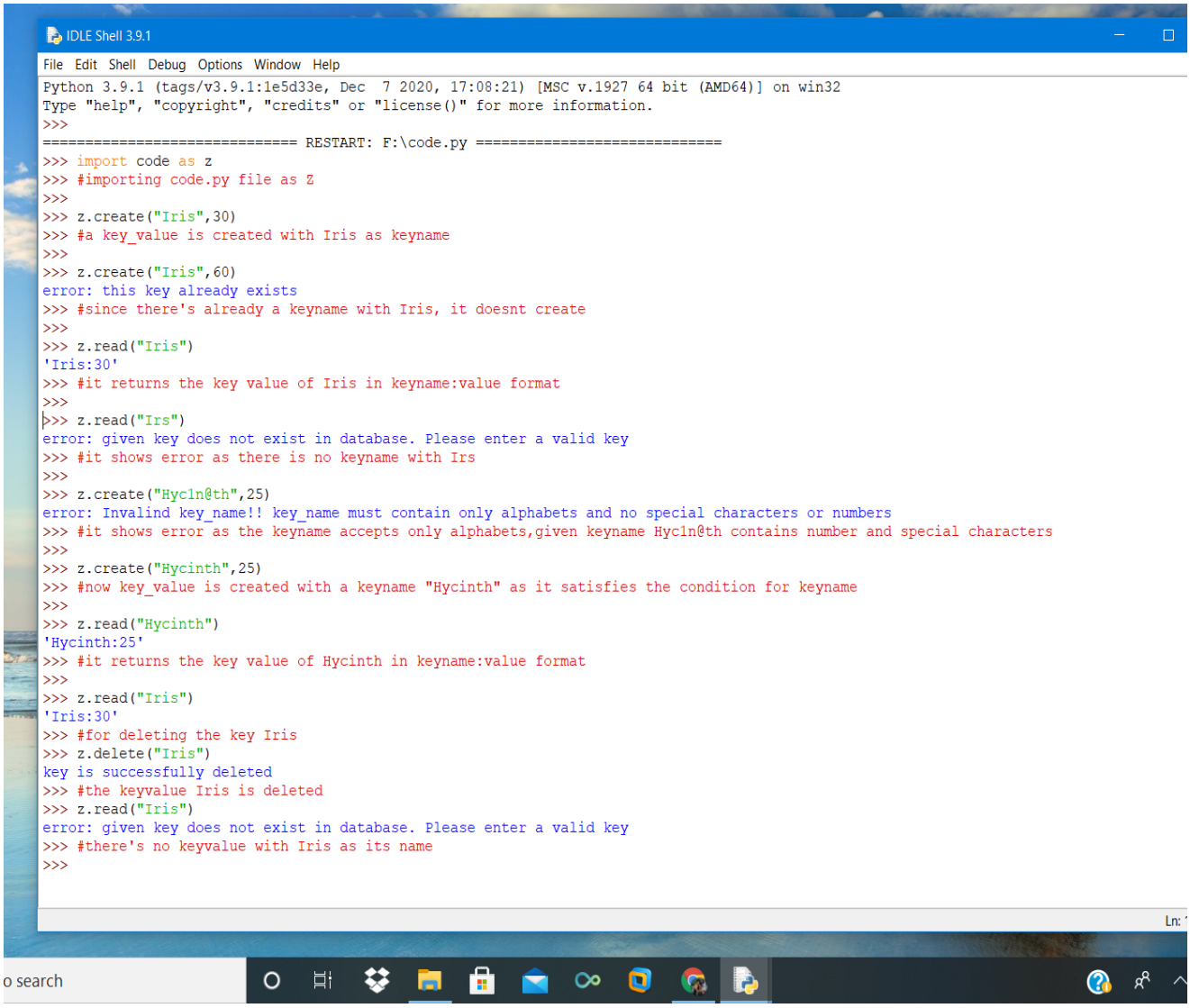
Language preferred – Python

Screenshots have been attached below.

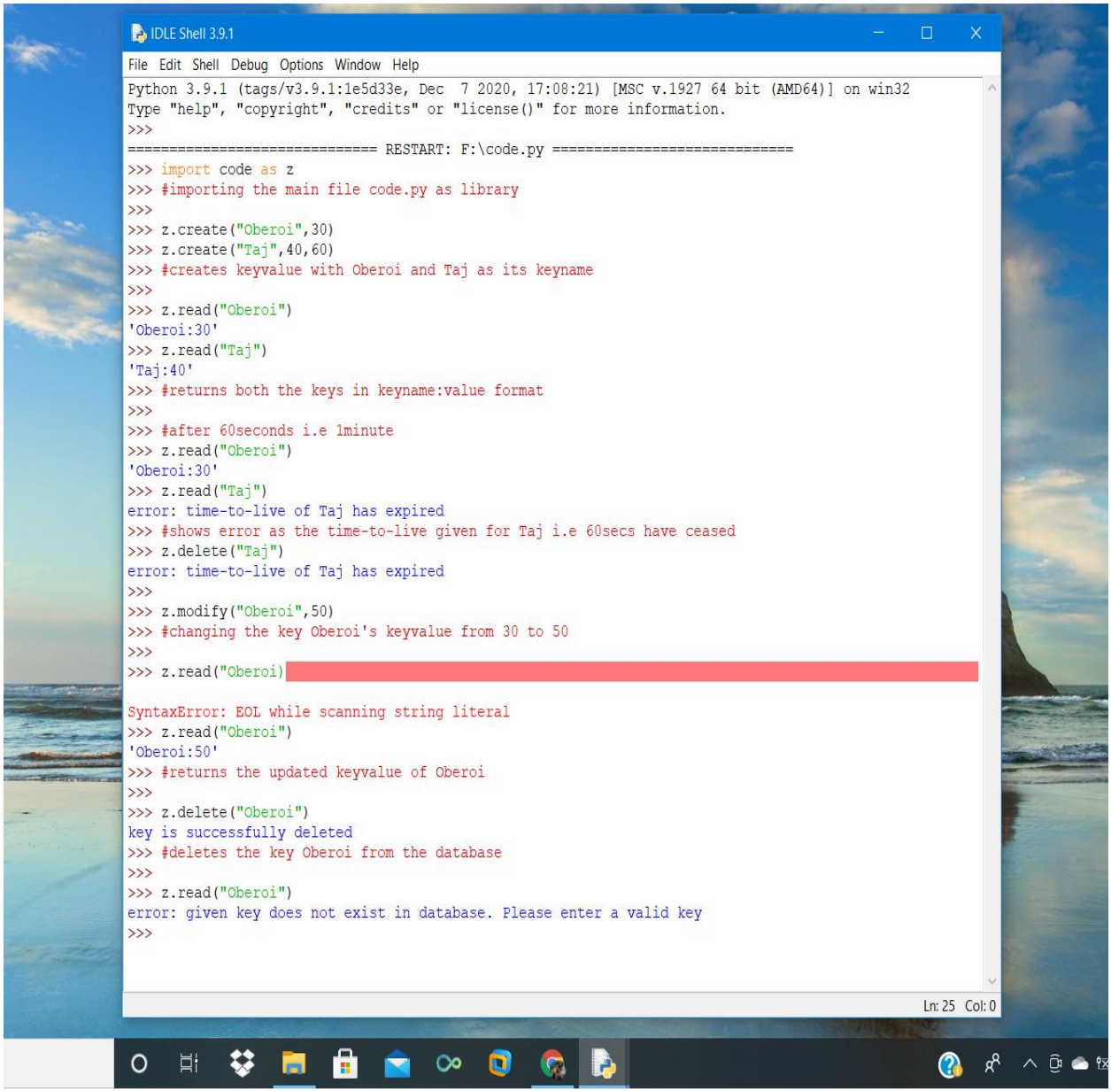
Joshna Mariya Jose

Output Examples:

1.

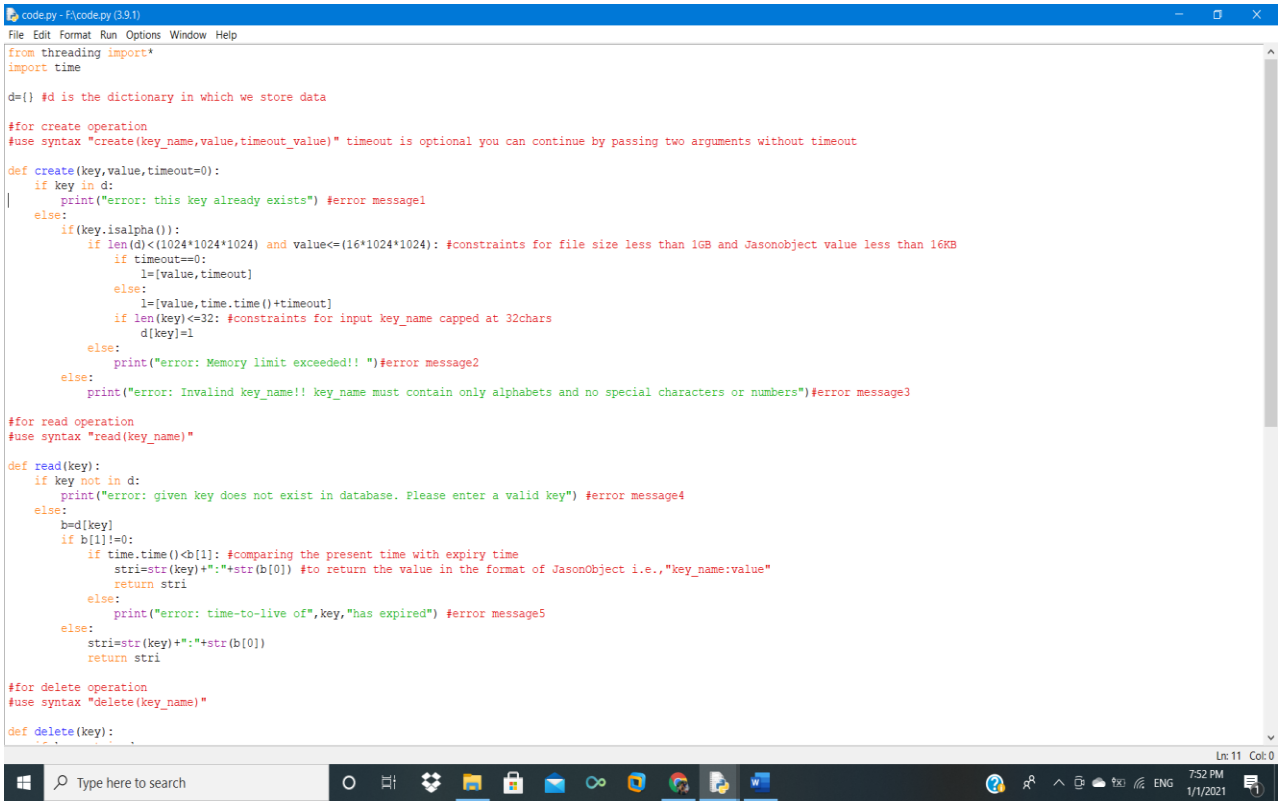


2.

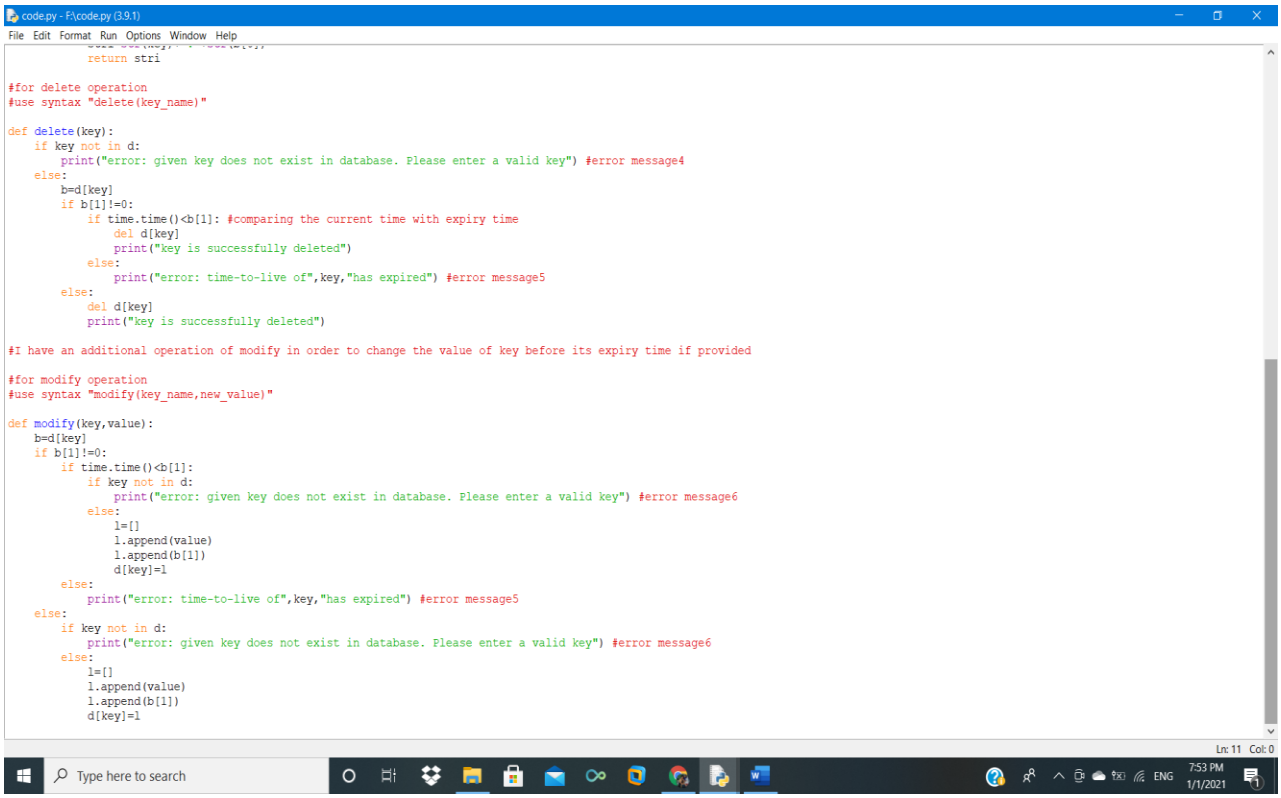

IDLE Shell 3.9.1
File Edit Shell Debug Options Window Help
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: F:\code.py ==========
>>> import code as z
>>> #importing the main file code.py as library
>>>
>>> z.create("Oberoi",30)
>>> z.create("Taj",40,60)
>>> #creates keyvalue with Oberoi and Taj as its keyname
>>>
>>> z.read("Oberoi")
'Oberoi:30'
>>> z.read("Taj")
'Taj:40'
>>> #returns both the keys in keyname:value format
>>>
>>> #after 60seconds i.e 1minute
>>> z.read("Oberoi")
'Oberoi:30'
>>> z.read("Taj")
error: time-to-live of Taj has expired
>>> #shows error as the time-to-live given for Taj i.e 60secs have ceased
>>> z.delete("Taj")
error: time-to-live of Taj has expired
>>>
>>> z.modify("Oberoi",50)
>>> #changing the key Oberoi's keyvalue from 30 to 50
>>>
>>> z.read("Oberoi)
SyntaxError: EOL while scanning string literal
>>> z.read("Oberoi")
'Oberoi:50'
>>> #returns the updated keyvalue of Oberoi
>>>
>>> z.delete("Oberoi")
key is successfully deleted
>>> #deletes the key Oberoi from the database
>>>
>>> z.read("Oberoi")
error: given key does not exist in database. Please enter a valid key
>>>
Ln: 25 Col: 0

Joshna Mariya Jose

MAIN FILE



```
from threading import*
import time

d={} #d is the dictionary in which we store data

#for create operation
#use syntax "create(key_name,value,timeout_value)" timeout is optional you can continue by passing two arguments without timeout

def create(key,value,timeout=0):
    if key in d:
        print("error: this key already exists") #error message1
    else:
        if(key.isalpha()):
            if len(d)<(1024*1024*1024) and value<=(16*1024*1024): #constraints for file size less than 1GB and Jasonobject value less than 16KB
                if timeout==0:
                    l=[value,timeout]
                else:
                    l=[value,time.time()+timeout]
                if len(key)<=32: #constraints for input key_name capped at 32chars
                    d[key]=l
            else:
                print("error: Memory limit exceeded!! ")#error message2
        else:
            print("error: Invalind key_name!! key_name must contain only alphabets and no special characters or numbers")#error message3

#for read operation
#use syntax "read(key_name)"

def read(key):
    if key not in d:
        print("error: given key does not exist in database. Please enter a valid key") #error message4
    else:
        b=d[key]
        if b[1]!=0:
            if time.time()<b[1]: #comparing the present time with expiry time
                stri=str(key)+":"+str(b[0]) #to return the value in the format of JasonObject i.e.,"key_name:value"
                return stri
            else:
                print("error: time-to-live of",key,"has expired") #error message5
        else:
            stri=str(key)+":"+str(b[0])
            return stri

#for delete operation
#use syntax "delete(key_name)"

def delete(key):
```



```
            return stri

#for delete operation
#use syntax "delete(key_name)"

def delete(key):
    if key not in d:
        print("error: given key does not exist in database. Please enter a valid key") #error message4
    else:
        b=d[key]
        if b[1]!=0:
            if time.time()<b[1]: #comparing the current time with expiry time
                del d[key]
                print("key is successfully deleted")
            else:
                print("error: time-to-live of",key,"has expired") #error message5
        else:
            del d[key]
            print("key is successfully deleted")

#I have an additional operation of modify in order to change the value of key before its expiry time if provided

#for modify operation
#use syntax "modify(key_name,new_value)"

def modify(key,value):
    b=d[key]
    if b[1]!=0:
        if time.time()<b[1]:
            if key not in d:
                print("error: given key does not exist in database. Please enter a valid key") #error message6
            else:
                l=[]
                l.append(value)
                l.append(b[1])
                d[key]=l
        else:
            print("error: time-to-live of",key,"has expired") #error message5
    else:
        if key not in d:
            print("error: given key does not exist in database. Please enter a valid key") #error message6
        else:
            l=[]
            l.append(value)
            l.append(b[1])
            d[key]=l
```