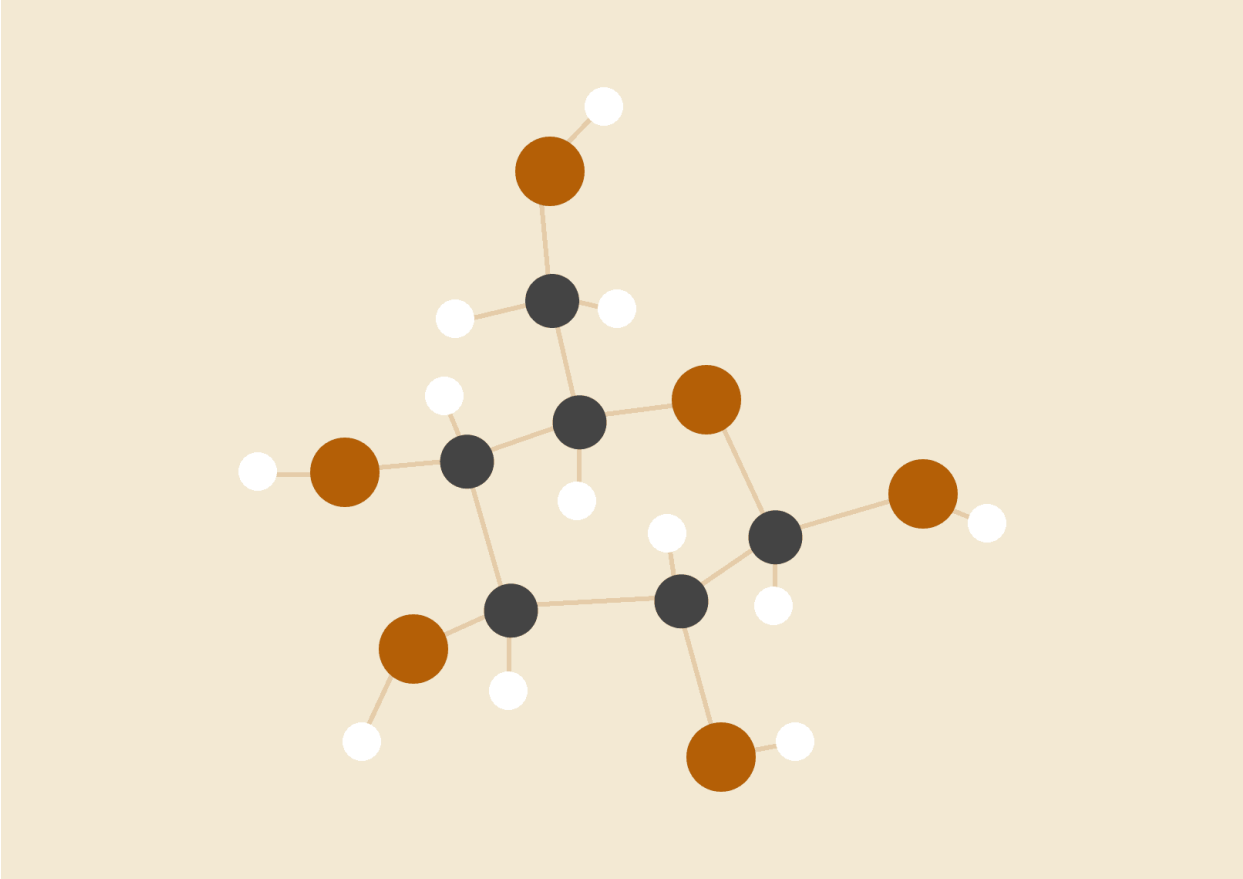
ADV OOP LAB REPORT

CLASS *BCSE II* ***SEM*** *SECOND* ***YEAR*** *2021*



NAME Neeladri Pal

ROLL 001910501015

GROUP A1

PYTHON ASSIGNMENT SET - 2

PROBLEM NUMBER - 1

PROBLEM STATEMENT

Write a single python program to do the following operations on a text file by writing different user defined functions.

a. Remove all the special characters.
b. Remove all single characters.
c. Substitute multiple spaces with single space.
d. Convert all the words into Lowercase.
e. Convert the words into literal form from their contracted form (e.g., Couldn't → Could not)

SOLUTION

We create a text file. We open the file and then extract the content of the file in a string. We then format this string as required. When editing is done, we write the string, i.e, the text content back into the file and close it.

SOURCE CODE

text_formatter.py

```
# import regex module
import re

# class to carry out several formatting operations on a piece of text
class TextFormatter:

   def __init__(self, str) -> None:
       self.text = str     # the string to work on

   def removeSpecialChars(self) -> None:
       """Replace any character which is not alphanumeric or space or dot
with empty character"""
```

```
        self.text = re.sub('[^a-zA-z0-9\n\.\s]', '', self.text)

    def removeSingleChars(self) -> None:
        """Replace a pattern of [space,any character except spa,space] with
[space,space]"""
        self.text = re.sub('\s[^\n\s]\s
', '  ', self.text)

    def removeMultipleSpaces(self) -> None:
        """Remove multiple spaces in the text with a single space"""
        self.text = re.sub('\s+', ' ', self.text)

    def toLowerCase(self) -> None:
        """Convert the text to lowercase"""
        self.text = self.text.lower()

    def expandContractions(self) -> None:
        """Convert the words into literal form from their contracted form
(e.g., Couldn’t -> Could not)"""
        expansions = (("'m"," am"), ("'s"," is"), ("t's", "t us"), ("'re","
are"), ("n't"," not"))
        for c, u in expansions:
            self.text = re.sub(c, u, self.text, flags=re.MULTILINE)

    def __str__ (self) -> str:
        return self.text

    @classmethod
    def operationMenu(cls) -> int:
        """Menu of operations which can be performed on the text"""
        print("Text File Operations -->")
        print("1. Remove special characters.")
        print("2. Remove all single characters.")
        print("3. Remove multiple spaces.")
        print("4. Convert the text into lower case.")
        print("5. Expand the contractions in the text.")
```

```
        op = int(input("Enter option: "))
        return op

filename = input("\nEnter file name: ")

try:
    with open(filename, 'r') as file:
        content = TextFormatter(file.read())    # store content of file
# in case the file does not exist on the path
except FileNotFoundError as e:
    print (e, "\nExiting..")
    exit (0)

ch = 'y'

while ch == 'y' or ch == 'Y' :
    ch == 'n'
    print ("\nBefore editing, text :- ", content, "\n")
    option = TextFormatter.operationMenu()
    if option == 1:
        content.removeSpecialChars()
    elif option == 2:
        content.removeSingleChars()
    elif option == 3:
        content.removeMultipleSpaces()
    elif option == 4:
        content.toLowerCase()
    elif option == 5:
        content.expandContractions()
    else:
        print("Invalid option.")
    print ("\nAfter editing, text :- ", content)
    ch = input("\nDo you want to edit more? (y/n) ")

with open(filename, 'w') as file:
    file.write(str(content))        # write content back to the file
```

INPUT

sample.txt

```
Hello I'm Doge. Would u be    my        friend
```

OUTPUT

```
[neeladripal@Neeladris-Macbook-Air problem_1 % python text_formatter.py

Enter file name: sample.txt

Before editing, text :-  Hello I'm Doge. Would u be    my        friend?#

Text File Operations -->
1. Remove special characters.
2. Remove all single characters.
3. Remove multiple spaces.
4. Convert the text into lower case.
5. Expand the contractions in the text.
Enter option: 5

After editing, text :-  Hello I am Doge. Would u be    my        friend?#

Do you want to edit more? (y/n) y

Before editing, text :-  Hello I am Doge. Would u be    my        friend?#

Text File Operations -->
1. Remove special characters.
2. Remove all single characters.
3. Remove multiple spaces.
4. Convert the text into lower case.
5. Expand the contractions in the text.
Enter option: 4

After editing, text :-  hello i am doge. would u be    my        friend?#

Do you want to edit more? (y/n) y

Before editing, text :-  hello i am doge. would u be    my        friend?#

Text File Operations -->
1. Remove special characters.
2. Remove all single characters.
3. Remove multiple spaces.
4. Convert the text into lower case.
5. Expand the contractions in the text.
Enter option: 3

After editing, text :-  hello i am doge. would u be my friend?#

Do you want to edit more? (y/n) y
```

```
Before editing, text :-  hello i am doge. would u be my friend?#

Text File Operations -->
1. Remove special characters.
2. Remove all single characters.
3. Remove multiple spaces.
4. Convert the text into lower case.
5. Expand the contractions in the text.
Enter option: 2

After editing, text :-  hello  am doge. would  be my friend?#

Do you want to edit more? (y/n) y

Before editing, text :-  hello  am doge. would  be my friend?#

Text File Operations -->
1. Remove special characters.
2. Remove all single characters.
3. Remove multiple spaces.
4. Convert the text into lower case.
5. Expand the contractions in the text.
Enter option: 1

After editing, text :-  hello  am doge. would  be my friend

Do you want to edit more? (y/n) n
```

sample.txt

```
hello  am doge. would  be my friend
```

PROBLEM NUMBER - 2

PROBLEM STATEMENT

Implement one multi-threaded server with socket programming in python.

SOLUTION

We create an Echo Server, a server which relays back the same message to the client, unless the client sends a “bye”. At a time, only MAXIMUM_PROCESSING number of clients can be processed and MAXIMUM_WAITING number of clients can be kept waiting. A waiting server’s connection is accepted as soon as an active client disconnects on a first-come-first-serve basis. No more than MAXIMUM_PROCESSING + MAXIMUM_WAITING clients’ request would be listened by the server.

SOURCE CODE

server.py

```
# import the necessary modules
import socket
from threading import Thread, Lock

lock = Lock ()    # to regulate the limit on number of active threads at a time

activeThreadCount = 0           # number of threads currently active

# maximum number of clients that can be served simultaneously
MAXIMUM_PROCESSING = 3

# maximum number of clients waiting to connect to server
MAXIMUM_WAITING = 2

# specify a host network interface, here we use loopback interface
# whose IPv4 address is 127.0.0.1; if a hostname is used in the host
# portion of IPv4/v6 socket address, the program may show a
# non-deterministic behavior, as Python uses the first address
```

```
# returned from the DNS resolution
HOST = '127.0.0.1'

# reserve a port on local machine for listening to incoming client requests
PORT = 12345



# new thread for a new connection
class ConnectionThread (Thread) :
   def __init__ (self, clientAddress, clientSocket) -> None :
       Thread.__init__ (self)

       # increment the active thread count
       # lock must be acquired to prevent possible race conditions
       lock.acquire ()
       global activeThreadCount
       activeThreadCount += 1
       lock.release ()

       # save the properties of the incoming client communication
       self.csocket = clientSocket
       self.caddr = clientAddress
       print ('Got new connection from', clientAddress)

   # override the __init__ method to specify the code that the thread would run
on
   def run (self) -> None:

       # send a message to indicate that the server is ready to respond
       self.csocket.send(bytes("You are now connected to server.\n\tSay
something and I'll echo the message.\n\tSay bye to terminate the
session.",'utf-8'))
       while True:
           try:
               # a blocking call to receive message from client in bytes form
               msg = self.csocket.recv (1024)
           except:
               # in case the client is abruptly terminated
```

```
                print ('Cannot receive data')

            # if the message contains no data, it must be due to some error
            if not msg:
                break

            # decode the message to string format to interpret
            msg = msg.decode ()
            # if msg is 'bye', the client wants to disconnect
            if msg == 'bye' :
                break

            print ('From client at', self.caddr[1], 'received: ', msg)

            # echo the message sent by client, back to client
            self.csocket.sendall (bytes(msg, 'UTF-8'))

        # message to show that the client has disconnected
        print ("Client at ", self.caddr , " disconnected...")

        # close the socket used for connecting to the client at the specific
address
        self.csocket.close ()

        # decrement the active thread count
        lock.acquire ()
        global activeThreadCount
        activeThreadCount -= 1
        lock.release ()

# create a socket object which supports context manager types
# this is used to listen to incoming client connection requests
# AF_INET refers to the address family ipv4.
# The SOCK_STREAM means connection oriented TCP protocol.
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as server:

   print("Server started")
```

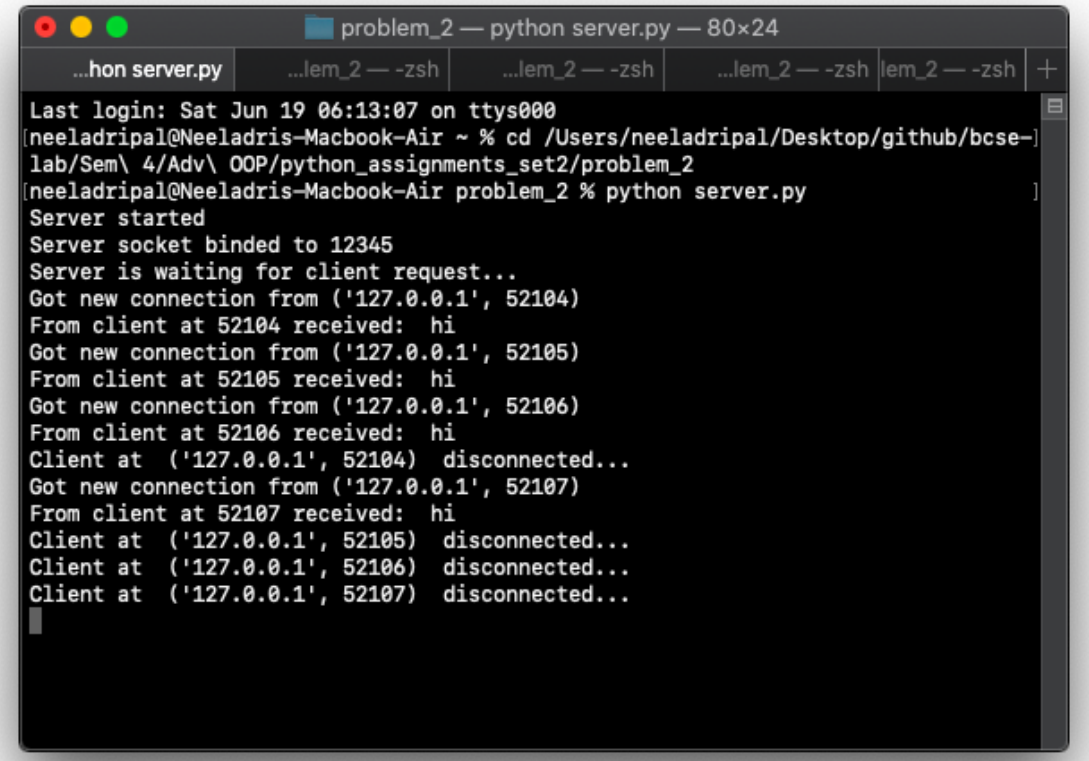
```
# set socket options, SO_REUSEADDR specifies that the local
# address to which the socket binds can be reused
server.setsockopt (socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

# Next bind to the port
server.bind((HOST, PORT))
print ("Server socket binded to %s" %(PORT))

# put the socket into listening mode, Its backlog parameter
# specifies the number of unaccepted connections that the
# system will allow before refusing new connections.
server.listen (MAXIMUM_WAITING)
print ("Server is waiting for client request...")

# a forever loop until we interrupt it or an error occurs
while True:

    # fetch the active thread count
    # lock ensures fetch does not take place while
    # the value is being updated
    lock.acquire ()
    n = activeThreadCount
    lock.release ()

    if n < MAXIMUM_PROCESSING:
        # Establish connection with client
        # a new socket is returned for connecting to the client
        # which is different from the listening socket
        conn, addr = server.accept()

        # interaction with client continues in a new thread
        newthread = ConnectionThread (addr, conn)
        newthread.start ()
```

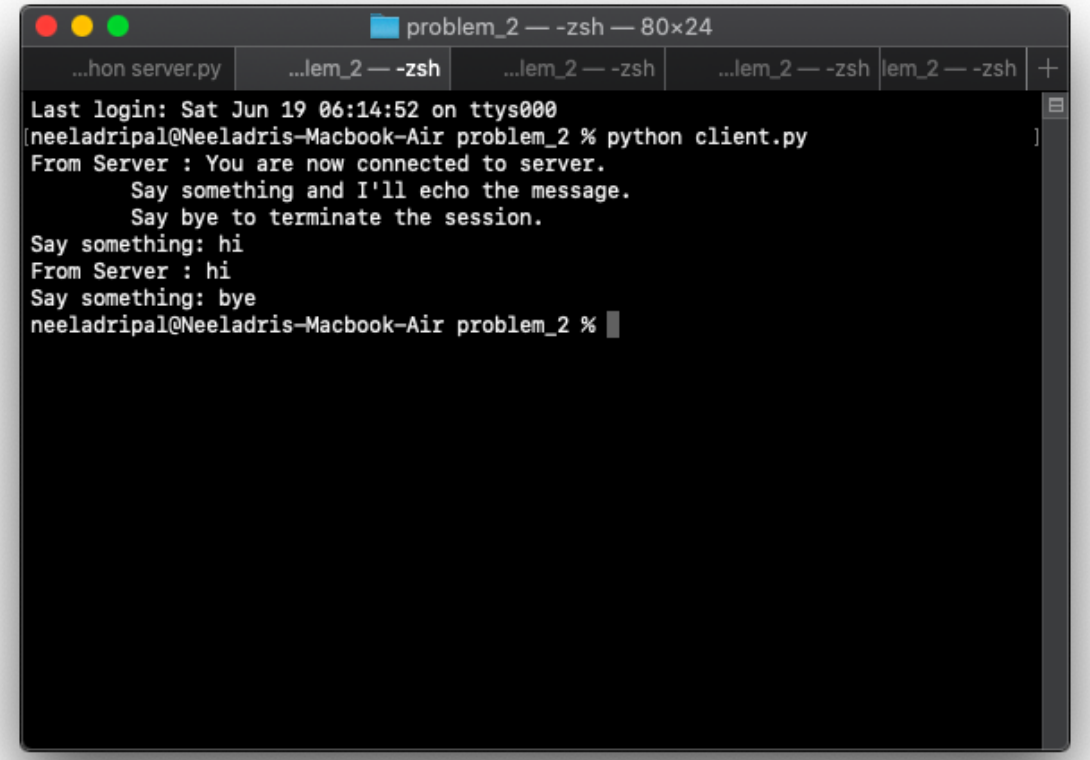
client.py

```
# import socket module
import socket

SERVER = "127.0.0.1"         # host interface of the server
PORT = 12345                 # port on which the server listens

# create a new socket object
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as client :
   client.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

   # connect to server
   client.connect((SERVER, PORT))

   # a forever loop until we interrupt it or an error occurs
   while True:
       # recieve data from server
       in_data =  client.recv(1024)
       print("From Server :" ,in_data.decode())

       # take message from user
       msg = input('Say something: ')

       # send it to server
       client.sendall (bytes(msg,'UTF-8'))

       # if message is 'bye', terminate the connection
       if msg=='bye':
            break
```

OUTPUT

A server is created. 3 new clients connect and say “hi” one-by-one. They immediately connect to the server. When a 4th client tries to connect, it is kept in waiting state. As soon as client #1 disconnects by saying “bye”, client #4 gets connection and it says ‘hi’. After that, clients #2, #3, #4 disconnect by saying “bye”. Here, we show the server side and one of the client’s side output. (All clients have same output)


problem_2 — python server.py — 80×24
...hon server.py
...lem_2 — -zsh
...lem_2 — -zsh
...lem_2 — -zsh
lem_2 — -zsh
Last login: Sat Jun 19 06:13:07 on ttys000
neeladripal@Neeladris-Macbook-Air ~ % cd /Users/neeladripal/Desktop/github/bcse-lab/Sem\ 4/Adv\ OOP/python_assignments_set2/problem_2
neeladripal@Neeladris-Macbook-Air problem_2 % python server.py
Server started
Server socket binded to 12345
Server is waiting for client request...
Got new connection from ('127.0.0.1', 52104)
From client at 52104 received: hi
Got new connection from ('127.0.0.1', 52105)
From client at 52105 received: hi
Got new connection from ('127.0.0.1', 52106)
From client at 52106 received: hi
Client at ('127.0.0.1', 52104) disconnected...
Got new connection from ('127.0.0.1', 52107)
From client at 52107 received: hi
Client at ('127.0.0.1', 52105) disconnected...
Client at ('127.0.0.1', 52106) disconnected...
Client at ('127.0.0.1', 52107) disconnected...


problem_2 — -zsh — 80×24
...hon server.py
...lem_2 — -zsh
...lem_2 — -zsh
...lem_2 — -zsh
lem_2 — -zsh
Last login: Sat Jun 19 06:14:52 on ttys000
neeladripal@Neeladris-Macbook-Air problem_2 % python client.py
From Server : You are now connected to server.
Say something and I'll echo the message.
Say bye to terminate the session.
Say something: hi
From Server : hi
Say something: bye
neeladripal@Neeladris-Macbook-Air problem_2 %