```
#Author: Manish Sharma
#importing the boto library
#This is the library that provides an AWS API for python
import boto
import boto.sqs
from boto.sqs.message import Message

def take_input_list():

    #Taking user input as comma-separated list and coverting it into a
list datatype in python
    user_input_list = map(int,raw_input("Enter comma-separated list of
numbers: ").split(','))

    #Printing the value of user list.
    print("The user input list is: %s" % user_input_list)
    return user_input_list

def make_s3_connection(user_input_list):
    #Opening the boto connection with s3
    s3 = boto.connect_s3()

    #Creating the object of my s3 bucket named as 'list-manish-com'
    bucket = s3.create_bucket('list-manish-com')  # bucket names must be
unique

    #Creating a key in which to store the data
    key = bucket.new_key('keys\user_input_list')
    #Printing value of this key.
    print ("Key that stores data is: %s" % key)
    return key

def put_list_in_s3_bucket(user_input_list, s3_connection_key):

    #opens a file handle to the specified sting,
    #and buffers read and write the bytes from the string into the key
object on Amazon S3
    #In this case, the list of user input (i.e. user_input_list) acts as
input string
    s3_connection_key.set_contents_from_string(str(user_input_list))
    return s3_connection_key

def print_list_from_s3_bucket(key):
    val = key.get_contents_as_string()
    print "List/Object fetched from s3 bucket: %s" % val

def create_queue():
    conn = boto.sqs.connect_to_region("us-west-2")
    # Create the queue. This returns an SQS.Queue instance
    queue = conn.create_queue('manish_queue')
    return queue, conn

def write_to_queue(s3_key, queue, process_to_execute):
    '''
```

```
    This function takes operation as user input.
    operation is either addition or multiplication.
    At last, it stores the result as well as operation performed
    in s3 bucket.
    '''

    total = 0 #variable to store sum
    prod = 1  #variable to store product
    msg = ""  #empty message string
    count = 1 #counter variable to iterate through user inputted list
    my_message = Message() #initializing message object

    '''
    This loop iterates through user input list,
    performs the designated operation and
    stores result in a variable.
    '''
    for i in s3_key:
        if i.isdigit():
            if process_to_execute == "addition":
                total += int(i)
            elif process_to_execute == "multiplication":
                prod *= int(i)
        count += 1

    #derive the message to store based on the requested operation
    if process_to_execute == "addition":
        msg =  "sum is %s " % total
    if process_to_execute == "multiplication":
        msg = "product is %s" % prod
    #put the message in the queue and retrun queue as function return
    my_message.set_body("The operation was %s and %s" %
(process_to_execute, msg))
    queue.write(my_message)
    return queue

def read_from_queue(queue):
    message = queue.read()
    print ("Reading from queue")
    print (message.get_body())
    return message, message.get_body()

def main():
    #User inputs a comma-separated list of numbers.
    user_input_list = take_input_list()

    #Puts this list in the S3-bucket (and remembers the key/pointer to
this object).
    s3_connection_key = make_s3_connection(user_input_list)
    s3_connection_key = put_list_in_s3_bucket(user_input_list,
s3_connection_key)
    print_list_from_s3_bucket(s3_connection_key)
```

```
    #Puts a message in the SQS-inbox with a key/pointer to the object in
the S3-bucket
    #along with a process to be executed on these numbers.
    queue, conn = create_queue()
    process_to_execute = raw_input("Process to execute (addition or
multiplication): ")
    queue = write_to_queue(s3_connection_key.get_contents_as_string(),
queue, process_to_execute)

    #Waits until a response is generated in the SQS-outbox (should
contain a pointer to a new, and processed,
    #object in the S3-bucket along with the process executed).
    message_pointer, new_object_after_queue_processsing =
read_from_queue(queue)
    s3_key_new_object_after_queue_processsing =
make_s3_connection(user_input_list)
    s3_key_new_object_after_queue_processsing =
put_list_in_s3_bucket(new_object_after_queue_processsing,
s3_key_new_object_after_queue_processsing)

    #Reads the result from the S3-bucket
    #Prints the results along with the original numbers and the the
process that was done.
    print "Printing the new object "
    print_list_from_s3_bucket(s3_key_new_object_after_queue_processsing)

    #Deletes the result-message from the SQS-outbox.
    print "Deleting message from sqs queue"
    queue.delete_message(message_pointer)
    print "Deleting queue"
    conn.delete_queue(queue)

if __name__ == '__main()__':
    main()
else:
    main()
```