

NAME : *answers*

**Open book, open notes, but please do not ask questions.
Write all answers on these sheets.**

question	1	2	3	4	total
points					
maximum	25	25	25	25	100

1. Give Python code for the following function.

```
def PrintForm(D,F):
    """
    Prints a form, given a dictionary D and the name of a python script in F.
    Keys and values in D are all of type string.
    For every key:value pair in D there is one input element of type
    text on the form. The value is used as text to be printed in front
    of the input element. The name of the input element is the key in
    the dictionary. There is one element of type submit.
    Hitting submit activates the script defined by F.
    """
```

Answer:

```
s = "<form method = \"post\" action = \"%s\">" % F
s = s + '\n'
for e in D.keys():
    a = "<p> %s <input type = \"text\" name = \"%s\">" % (D[e],e)
    a = a + '\n'
    s = s + a
a = "<p><input type = \"submit\">\n" + "</form>"
print s + a
```

2. A regional bus company uses MySQL to store the schedules of its bus routes. Busses run every day at somewhat irregular but nevertheless prescribed departure and arrival dates.

There are as many databases as cities served by the bus company. The name of each database is the name of the city. Each database contains as many tables as there are bus routes departing from this city. The names of the tables are the names of the destination cities.

All tables in all databases have two columns. The first column contains the departure time from the city with the same name as the database. The second column stores the schedule arrival item of the city with the name as the table.

No user name nor password is needed to access the databases. All times are formatted in the 24-hour time format, as hh:mm:ss. You may assume that comparison operators for times are available.

Define a function that given two names of two cities and a time searches for the first available bus departing from the first city and heading to the second city. The function returns `None` if there is no bus, otherwise the time of the first available bus is returned.

Answer:

```
import MySQLdb

def Search(source,destination,stime):
    try:
        db = MySQLdb.connect(db=source)
        cs = db.cursor()
    except:
        print 'connecting to database '+ source
    try:
        q = 'select * from ' + destination
        L = cs.execute(q)
        R = cs.fetchall()
    except:
        print 'executed ' + q
        R = [('08:12:00','09:02:12'),('12:23:00','13:55:99')]
    print R
    if len(R) == 0:
        return None
    else:
        for e in R:
            if stime < e[0]: return e[0]
        return R[0][0]
```

3. Write code for a server that keeps a dictionary of its connecting clients. The key of this dictionary is the address of the client. The corresponding value equals the first number the client sends to the server.

After sending first a number to the server, every client is expected to send exactly as many messages to the server as the value of the first number. Every message contains exactly one character. After all messages have been sent, the server closes the connection to the client.

Set up the server to handle an indefinite number of clients indefinitely.

Answer:

```
from SocketServer import StreamRequestHandler, TCPServer

port = 12091

D = {}; S = {}

class OurServer(StreamRequestHandler):

    def handle(self):
        """
        Updates dictionary with address of client
        and then collects data in a string.
        """
        print "connected at", self.client_address
        data = self.rfile.read(10)
        key = str(self.client_address)
        print 'read \'' + data + '\'' from client'
        D[key] = int(data)
        print 'handling %d messages from client %s' % (D[key],key)
        print 'dictionary : ' + str(D)
        for i in range(0,D[key]):
            data = self.rfile.read(1)
            print 'read \'' + data + '\'' from ' + key
            if not S.has_key(key):
                S[key] = data
            else:
                S[key] = S[key] + data
        self.rfile.close()
        print 'received ' + S[key] + ' from ' + key

ss = TCPServer(('',port),OurServer)
print 'server is listening to', port
try:
    print 'press ctrl c to stop server'
    ss.serve_forever()
except KeyboardInterrupt:
    print ' ctrl c pressed, closing server'
    ss.socket.close()
```

4. Define a Thread class to simulate floating disks on a canvas.

Before sleeping one second the disks move their position at random. After each move the disks check whether they have collided with another disk. Disks no longer move after a collision. Disks also stop moving after 15 moves.

Answer:

```
from threading import *
from random import randint
from time import sleep
class Disk(Thread):
    def __init__(self,n,p,L):
        """
        Disk n is initialized with a start position with coordinates in p
        and a list L of all positions of all other disks.
        """
        Thread.__init__(self,name=n)
        self.position = p; self.stopped = False
        self.others = L; self.moves = 15
    def collision(self):
        "Returns k if disk collided with L[k], otherwise returns -1."
        n = int(self.getName())
        for i in range(0,len(self.others)):
            if i != n:
                if self.position == self.others[i]: return i
        return -1
    def run(self):
        """
        If not yet stopped or if no other disk has collided with it,
        the disk will make a random move.
        """
        sleep(1); n = self.getName()
        while not self.stopped and (self.moves > 0):
            k = self.collision()
            if k != -1:
                self.stopped = True
                print 'Thread ' + n + ' collided with ' + str(k)
            else:
                a = randint(-1,+1); b = randint(-1,+1)
                self.position[0] = self.position[0] + a
                self.position[1] = self.position[1] + b
                s = 'Thread ' + n + ' moves to ' + str(self.position)
                self.others[int(n)] = self.position
                self.moves = self.moves - 1
                k = self.collision()
                if k != -1:
                    self.stopped = True
                    s = s + ' collided with ' + str(k)
                print s
```