

APP_CT_2- Descriptive Answers

Name : Rahul Goel

Reg No: RA1911030010094

Q5)

```
import threading
import datetime
import time

def thrd_1_time():
    print("starting thread 1")
    for _ in range(3):
        now = datetime.datetime.now()
        print ("Thread-1: " + now.strftime("%a %b %d %H:%M:%S
%Y"))
        time.sleep(1)
    print("exit thread 1")

def thrd_2_time():
    print("starting thread 2")
    for _ in range(3):
        now = datetime.datetime.now()
        print ("Thread-2: " + now.strftime("%a %b %d %H:%M:%S
%Y"))
        time.sleep(1)
    print("exit thread 2")

t1 = threading.Thread(target=thrd_1_time)
t2 = threading.Thread(target=thrd_2_time)
t1.start()
t2.start()
```

The screenshot displays a Replit IDE interface. On the left, a file explorer shows 'main.py'. The main editor area contains the following Python code:

```
1
2 import threading
3 import datetime
4 import time
5
6 def thrd_1_time():
7     print("starting thread 1")
8     for _ in range(3):
9         now = datetime.datetime.now()
10        print ("Thread-1: " + now.strftime("%a %b %d %H:%M:%S
11        %Y"))
12        time.sleep(1)
13    print("exit thread 1")
14
15 def thrd_2_time():
16     print("starting thread 2")
17     for _ in range(3):
18         now = datetime.datetime.now()
19         print ("Thread-2: " + now.strftime("%a %b %d %H:%M:%S
20         %Y"))
21         time.sleep(1)
22     print("exit thread 2")
23
24 t1 = threading.Thread(target=thrd_1_time)
25 t2 = threading.Thread(target=thrd_2_time)
26 t1.start()
27 t2.start()
28 print("exiting main thread")
```

On the right, the console output shows the execution results:

```
> run-project
starting thread 1
Thread-1: Sat Apr 03 04:32:50 2021
starting thread 2
Thread-2: Sat Apr 03 04:32:50 2021
exiting main thread
Thread-1: Sat Apr 03 04:32:51 2021
Thread-2: Sat Apr 03 04:32:51 2021
Thread-1: Sat Apr 03 04:32:52 2021
Thread-2: Sat Apr 03 04:32:52 2021
exit thread 1
exit thread 2
>
```

Q4)

```
import sqlite3
conn = sqlite3 . connect ( 'mydatabase.db' )

cursor = conn . cursor ()

#create the salesman table

cursor.execute("CREATE TABLE salesman(salesman_id n(5), name char(30), city char(35), commission decimal(7,2));")


# inserting data into the table

cursor . execute ( """

INSERT INTO salesman(salesman_id,'name', 'city',commission)

VALUES(5001,'James Hoog', 'New York', 0.15)

""")

cursor . execute ( """

INSERT INTO salesman(salesman_id,'name', 'city',commission)

VALUES(5002,'Nail Knite', 'Paris', 0.25)

""")

cursor . execute ( """

INSERT INTO salesman(salesman_id,'name', 'city',commission)

VALUES(5003,'Pit Alex', 'London', 0.15)

""")

cursor . execute ( """

INSERT INTO salesman(salesman_id,'name', 'city',commission)

VALUES(5004,'Mc Lyon', 'Paris', 0.35)

""")

conn.commit ()

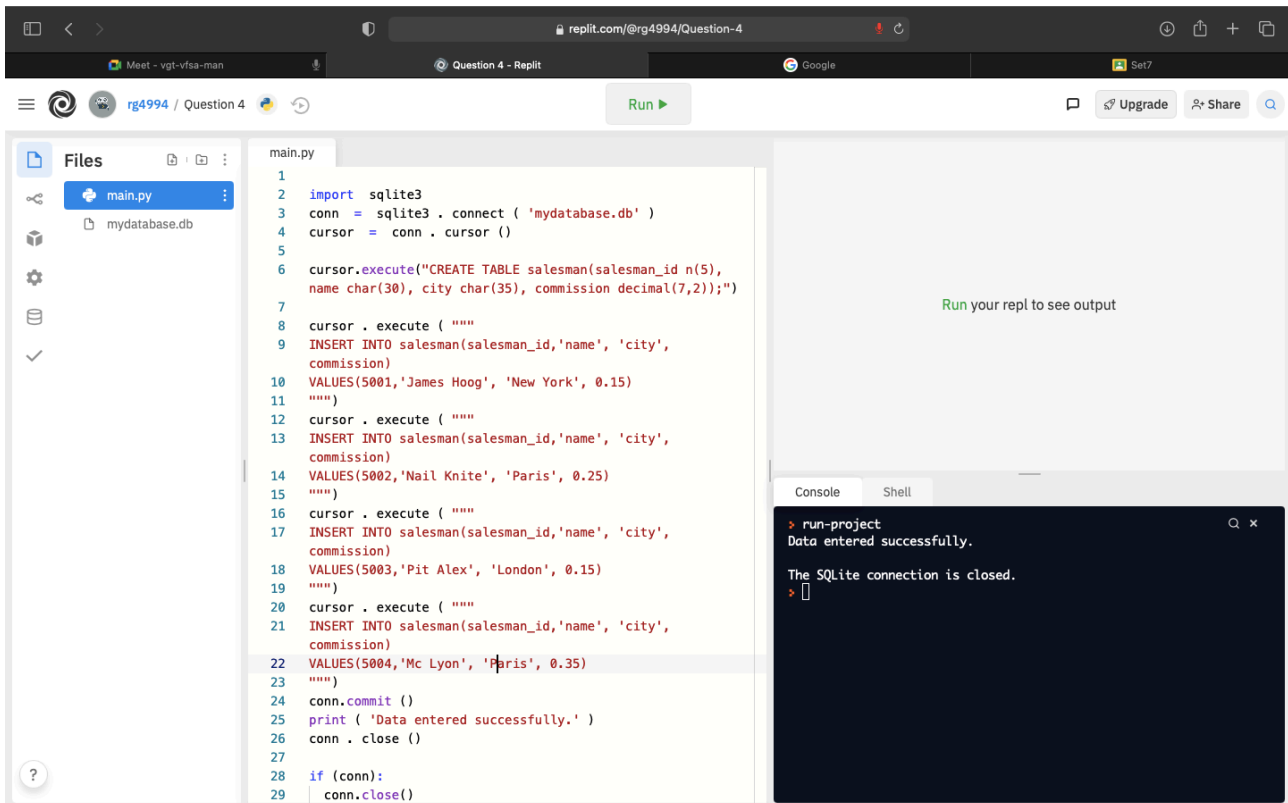
print ( 'Data entered successfully.' )

conn . close ()


if (conn):

    conn.close()

    print("\nThe SQLite connection is closed.")
```



The screenshot shows a Replit environment with a Python file named `main.py` and a SQLite database named `mydatabase.db`. The script performs the following actions:

- Imports `sqlite3`.
- Connects to the database `mydatabase.db`.
- Creates a table `salesman` with columns: `salesman_id` (integer), `name` (string), `city` (string), and `commission` (decimal).
- Inserts four records into the `salesman` table:

salesman_id	name	city	commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.25
5003	Pit Alex	London	0.15
5004	Mc Lyon	Paris	0.35

The console output shows:

```
> run-project
Data entered successfully.

The SQLite connection is closed.
>
```

Q6)

import socket # for socket

import sys

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

print ("Socket successfully created")

port = 80

host_ip = socket.gethostbyname('www.srmist.edu.in')

s.connect((host_ip, port))

print ("the socket has successfully connected to srmist.edu.in")

The screenshot shows a Replit web interface. On the left, a file explorer shows 'main.py' and 'mydatabase.db'. The main editor displays a Python script in 'main.py':

```
1 import socket # for socket
2 import sys
3 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
4 print ("Socket successfully created")
5 port = 80
6 host_ip = socket.gethostname('www.srmist.edu.in')
7 s.connect((host_ip, port))
8 print ("the socket has successfully connected to srmist.edu.in")
9
10
```

On the right, a message says "Run your repl to see output". Below it, a console window shows the output of running the project:

```
> run-project
Socket successfully created
the socket has successfully connected to srmist.edu.in
> 
```

ADVANCED PROGRAMMING PRACTICE - CT-2

Name: Rahul Gred

RegNo: RA1911030010094

Batch: CSE-02.

② (i) 22

(ii) c. execute ('INSERT INTO Customer (Datestamp, Max weight, Reps) VALUES (?, ?, ?)', (date, weight.get(), reps.get()))

(iii) L1 = Label (window, text = "Compound Lift", font = ("arial", 18)). place (x=10, y=100) compound = (Bench, 'squat', 'Deadlift', 'OvH') compd = optionMenu (window, comp, *compound) compd.place (x=220, y=105)

```
l2 = Label(window, text = "Day (dd)", font = ("arial", 18)).  
place(x=10, y=150) dayT = Entry(window, textvariable  
= day) dayT.place(x=220, y=155)
```

(iv) Place geometry manager is used in the above form.

```
button_4 = Button(window, text = "Update")  
button_4.place(x=100, y=400).
```

⑧ Import Threading

```
schedule-1 = get value() # get some initial value.  
lock = threading.Lock()
```

```
def read():
```

```
    lock.acquire()
```

```
    print("Passenger can read schedule-1")
```

```
    time.sleep(10)
```

```
    lock.release()
```

```
def write():
```

```
    global schedule-1
```

```
    lock.acquire()
```

```
    print("Schedule-1 is now modifying")
```

```
    schedule-1 = input("Enter value for schedule-1")
```

```
    time.sleep(10)
```

```
    lock.release()
```

```
    print("Schedule-1 updated")
```

```
scheduler = Thread = threading.Thread(target = write)
```

```
passenger = Thread = threading.Thread(target = read)
```

```
scheduler = Thread.start()
```

```
passenger = Thread.start()
```

```
scheduler = Thread.join()
```

```
passenger = Thread.join()
```

Q1)

```
import socket

import sys

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
port = 60
s.bind(('0.0.0.0', port))
print ('Socket binded to port 60')
s.listen(3)
print ('socket is listening')

while True:
    c, addr = s.accept()
    print ('Got connection from ', addr)
    print (c.recv(1024))
    c.close()
```

Q1 client:

```
import socket

s = socket.socket()
port = 60
s.connect(('localhost', port))
z = """Hi server, I'm ready to transfer my data
This is socket application"""
s.sendall(z.encode())
s.close()
```

Q7)

import socket programming library

```
import socket
```

```
# import thread module
```

```
from _thread import *
```

```
import threading
```

```
print_lock = threading.Lock()
```

```
# thread function
```

```
def threaded(c):  
    while True:  
  
        # data received from client  
        data = c.recv(1024)  
        if not data:  
            print('Bye')  
  
            # lock released on exit  
            print_lock.release()  
            break  
  
        # reverse the given string from client  
        data = data[::-1]  
  
        # send back reversed string to client  
        c.send(data)  
  
        # connection closed  
        c.close()
```

```
def Main():  
    host = ""  
  
    # reverse a port on your computer  
    # in our case it is 12345 but it  
    # can be anything  
    port = 12345  
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
s.bind((host, port))
```

```
print("socket binded to port", port)
```

```
# put the socket into listening mode
```

```
s.listen(5)
```

```
print("socket is listening")
```

```
# a forever loop until client wants to exit
```

```
while True:
```

```
    # establish connection with client
```

```
    c, addr = s.accept()
```

```
    # lock acquired by client
```

```
    print_lock.acquire()
```

```
    print('Connected to :', addr[0], ':', addr[1])
```

```
    # Start a new thread and return its identifier
```

```
    start_new_thread(threaded, (c,))
```

```
s.close()
```

```
if name == 'main':
```

```
    Main()
```


Q3)

```
# importing whole module
from tkinter import *
from tkinter.ttk import *

# importing strftime function to
# retrieve system's time
from time import strftime

# creating tkinter window
root = Tk()
root.title('Clock')

# This function is used to
# display time on the label
def time():
    string = strftime('%H:%M:%S %p')
    lbl.config(text = string)
    lbl.after(1000, time)

# Styling the label widget so that clock
# will look more attractive
lbl = Label(root, font = ('calibri', 40, 'bold'),
            foreground = 'purple')

# Placing clock at the centre
# of the tkinter window
lbl.pack(anchor = 'center')
time()

mainloop()
```

