

Name :- Deepawali . B. Mhaisagar
Assignment no 20

1. Set the variable test1 to the string 'This is a test of the emergency text system,' and save test1 to a file named test.txt.

ANS

To set the variable `test1` to the string 'This is a test of the emergency text system,' and save `test1` to a file named `test.txt`, you can follow these steps:

Step 1: Set the variable `test1` to the desired string

```
test1 = 'This is a test of the emergency text system,'
```

Step 2: Open the `test.txt` file in write mode using the `open()` function:

```
file = open('test.txt', 'w')
```

Step 3: Write the value of `test1` to the file using the `write()` method:

```
file.write(test1)
```

Step 4: Close the file to ensure the changes are saved:

```
file.close()
```

By following these steps, you have set the variable `test1` to the specified string and saved it to a file named `test.txt`. The contents of `test1` are written to the file.

2. Read the contents of the file test.txt into the variable test2. Is there a difference between test 1

and test 2?

ANS

To read the contents of the file `test.txt` into the variable `test2` and check if there is a difference between `test1` and `test2`, you can follow these steps:

Step 1: Open the `test.txt` file in read mode using the `open()` function:

```
file = open('test.txt', 'r')
```

Step 2: Read the contents of the file into the variable `test2` using the `read()` method:

```
test2 = file.read()
```

Step 3: Close the file:

```
file.close()
```

Step 4: Compare the values of `test1` and `test2`:

```
difference = test1 == test2
```

The variable `difference` will be `True` if there is no difference between `test1` and `test2`, and `False` if there is a difference.

To verify the result, you can print the values of `test1`, `test2`, and `difference`:

```
print("test1:", test1)
```

```
print("test2:", test2)
```

```
print("Difference:", difference)
```

```
...
```

By following these steps, you have read the contents of the file `test.txt` into the variable `test2` and checked if there is a difference between `test1` and `test2`. The variable `difference` will indicate whether the values are the same or different.

3. Create a CSV file called books.csv by using these lines:

title,author,year

The Weirdstone of Brisingamen,Alan Garner,1960

Perdido Street Station,China Miéville,2000

Thud!,Terry Pratchett,2005

The Spellman Files,Lisa Lutz,2007

Small Gods,Terry Pratchett,1992

ANS

To create a CSV file called `books.csv` with the provided lines:

...

title,author,year

The Weirdstone of Brisingamen,Alan Garner,1960

Perdido Street Station,China Miéville,2000

Thud!,Terry Pratchett,2005

The Spellman Files,Lisa Lutz,2007

Small Gods,Terry Pratchett,1992

...

You can follow these steps:

Step 1: Open a new file named `books.csv` in write mode using the `open()` function

```
file = open('books.csv', 'w')
```

Step 2: Write the lines to the file

```
file.write("title,author,year\n")
```

```
file.write("The Weirdstone of Brisingamen,Alan Garner,1960\n")
```

```
file.write("Perdido Street Station,China Miéville,2000\n")
```

```
file.write("Thud!,Terry Pratchett,2005\n")
```

```
file.write("The Spellman Files,Lisa Lutz,2007\n")
```

```
file.write("Small Gods,Terry Pratchett,1992\n")
```

Step 3: Close the file to ensure the changes are saved:

```
file.close()
```

By following these steps, you have created a CSV file named `books.csv` with the provided lines.

4. Use the `sqlite3` module to create a SQLite database called `books.db`, and a table called `books` with these fields: `title` (text), `author` (text), and `year` (integer).

ANS

To use the `sqlite3` module to create a SQLite database called `books.db` and a table called `books` with the fields `title` (text), `author` (text), and `year` (integer), you can follow these steps:

Step 1: Import the `sqlite3` module:

```
import sqlite3
```

Step 2: Connect to the SQLite database or create it if it doesn't exist:

```
conn = sqlite3.connect('books.db')
```

Step 3: Create a cursor object to execute SQL statements:

```
cursor = conn.cursor()
```

Step 4: Execute an SQL statement to create the `books` table:

```
cursor.execute("""CREATE TABLE books
                (title TEXT, author TEXT, year INTEGER)""")
```

Step 5: Commit the changes to the database:

```
conn.commit()
```

Step 6: Close the connection to the database:

```
conn.close()
```

By following these steps, you have used the `sqlite3` module to create a SQLite database named `books.db` and a table named `books` with the fields `title`, `author`, and `year`. The database file will be created in the current working directory.

5. Read books.csv and insert its data into the book table.

ANS

To read the data from the `books.csv` file and insert it into the `books` table in the `books.db` SQLite database, you can follow these steps:

Step 1: Import the `csv` module and the `sqlite3` module:

```
import csv  
  
import sqlite3
```

Step 2: Connect to the `books.db` SQLite database:

```
conn = sqlite3.connect('books.db')
```

Step 3: Create a cursor object to execute SQL statements:

```
cursor = conn.cursor()
```

Step 4: Read the data from the `books.csv` file using the `csv` module and insert it into the `books` table:

with open('books.csv', 'r') as file:

```
    reader = csv.reader(file)  
  
    next(reader) # Skip the header row  
  
    for row in reader:  
  
        title = row[0]
```

```
author = row[1]
```

```
year = int(row[2])
```

```
cursor.execute('INSERT INTO books (title, author, year) VALUES (?, ?, ?)',  
(title, author, year))
```

Step 5: Commit the changes to the database:

```
conn.commit()
```

Step 6: Close the connection to the database:

```
conn.close()
```

By following these steps, you have read the data from the `books.csv` file and inserted it into the `books` table in the `books.db` SQLite database. The data from the CSV file is now available in the database table.

6. Select and print the title column from the book table in alphabetical order.

ANS

To select and print the `title` column from the `books` table in alphabetical order using SQLite, you can follow these steps:

Step 1: Import the `sqlite3` module:

```
import sqlite3
```

Step 2: Connect to the `books.db` SQLite database:

```
conn = sqlite3.connect('books.db')
```

Step 3: Create a cursor object to execute SQL statements:

```
cursor = conn.cursor()
```

Step 4: Execute an SQL query to select the `title` column from the `books` table and order it alphabetically:

```
cursor.execute('SELECT title FROM books ORDER BY title ASC')
```

Step 5: Fetch all the results from the query:

```
results = cursor.fetchall()
```

```
...
```

Step 6: Print the titles:

for row in results:

```
    print(row[0])
```

Step 7: Close the connection to the database:

```
conn.close()
```

By following these steps, you have selected and printed the `title` column from the `books` table in alphabetical order using SQLite. The titles will be displayed in the console output.

7. From the book table, select and print all columns in the order of publication.

ANS

To select and print all columns from the `books` table in the order of publication using SQLite, you can follow these steps:

Step 1: Import the `sqlite3` module:

```
import sqlite3
```

```
...
```

Step 2: Connect to the `books.db` SQLite database:

```
conn = sqlite3.connect('books.db')
```

```
...
```

Step 3: Create a cursor object to execute SQL statements:

```
cursor = conn.cursor()
```

```
...
```

Step 4: Execute an SQL query to select all columns from the `books` table and order the results by the `year` column:

```
cursor.execute('SELECT * FROM books ORDER BY year')
```

```
...
```

Step 5: Fetch all the results from the query:
`results = cursor.fetchall()`

Step 6: Print the results:
for row in results:
 `print(row)`

Step 7: Close the connection to the database:

`conn.close()`

By following these steps, you have selected and printed all columns from the `books` table in the order of publication using SQLite. The column values will be displayed in the console output.

8. Use the sqlalchemy module to connect to the sqlite3 database books.db that you just made in exercise 6.
ANS

To connect to the SQLite database `books.db` using the `sqlalchemy` module, you can follow these steps:

Step 1: Install the `sqlalchemy` module if you haven't done so already. You can install it using pip:

`pip install sqlalchemy`

Step 2: Import the necessary modules and functions:

```
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker
``
```

Step 3: Create an engine to connect to the SQLite database:
`engine = create_engine('sqlite:///books.db')`

Step 4: Create a session factory using the engine:
`Session = sessionmaker(bind=engine)`

Step 5: Create a session object to interact with the database:

`session = Session()`

Now you can use the `session` object to execute various database operations such as querying, inserting, updating, and deleting records.

By following these steps, you have connected to the SQLite database `books.db` using the `sqlalchemy` module. The `session` object allows you to interact with the database through the SQLAlchemy ORM (Object-Relational Mapping) interface.

9. Install the Redis server and the Python redis library (pip install redis) on your computer. Create a Redis hash called test with the fields count (1) and name (Fester Bestertester). Print all the fields for Test.

ANS

As an AI language model, I am unable to directly install software or libraries on your computer. However, I can guide you through the process of using Redis and the Python Redis library to create a Redis hash and print its fields.

To accomplish this, please follow these steps:

Step 1: Install Redis Server

- Download and install Redis Server from the official Redis website:
<https://redis.io/download>
- Follow the installation instructions specific to your operating system.

Step 2: Install Redis Library for Python

- Open your terminal or command prompt.
- Run the following command to install the Redis library for Python using pip:

```
```shell
pip install redis
```

### Step 3: Start Redis Server

- Start the Redis server by running the appropriate command based on your operating system.
- For example, on Linux, you can start Redis by running the following command in the terminal:

```
redis-server
```

### Step 4: Create the Redis Hash and Print Fields (Python code)

- Open a Python script or use an interactive interpreter.

```
import redis
```

```
Connect to Redis
redis_client = redis.Redis()
```

```
Create the Redis hash
redis_client.hset("test", "count", 1)
redis_client.hset("test", "name", "Fester Bestertester")
```

```
Print all fields of the Redis hash
hash_fields = redis_client.hgetall("test")
for field, value in hash_fields.items():
 print(f'{field}: {value.decode("utf-8")}')
...
```

- Make sure the Redis server is running before executing this code.
- The code uses the `redis` library to connect to Redis, creates a hash named "test" with the fields "count" and "name" and their respective values, and finally prints all the fields and their values.

## **10. Increment the count field of test and print it.**

### **ANS**

To increment the `count` field of the Redis hash named "test" and print its updated value, you can modify the Python code as follows:

```
import redis

Connect to Redis
redis_client = redis.Redis()

Increment the count field
redis_client.hincrby("test", "count", 1)

Print the updated count value
count_value = redis_client.hget("test", "count")
print(f'Updated count: {count_value.decode("utf-8")}')
```

In this code, we use the `hincrby()` method from the `redis` library to increment the `count` field of the "test" hash by 1. Then, we retrieve the updated value using `hget()` and print it.

Make sure that the Redis server is running and the "test" hash with the "count" field exists before executing this code.