**Assignment 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.**

**Sure, here's how you can do that in Python:**

**python**

**Copy code**

**# Set the variable test1 to the string 'This is a test of the emergency text system'**

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

**# Open the file named test.txt in write mode**

**with open('test.txt', 'w') as file:**

**# Write the contents of test1 to the file**

**file.write(test1)**

**This code will create a new file named test.txt in the current directory, and write the contents of test1 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?**

**Sure, here's how you can read the contents of test.txt into test2 variable in Python:**

**python**

**Copy code**

**# Open the file named test.txt in read mode**

**with open('test.txt', 'r') as file:**

**# Read the contents of the file into test2**

**test2 = file.read()**

**# Print the contents of test2 to compare with test1**

**print(test2)**

**This code will read the contents of the test.txt file and store them in the test2 variable. To check if there's any difference between test1 and test2, we can print both of them and compare:**

**python**

**Copy code**

**print(test1 == test2)**

**This code will return True if test1 and test2 have the same content, otherwise, it will return False. If there's no difference between test1 and test2, this will return True.**

**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**

**Sure, here's how you can create a CSV file called books.csv using Python's csv module:**

**python**

**Copy code**

**import csv**

**# Define the list of books data**

**books\_data = [**

**['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]**

**]**

**# Create the books.csv file and write the data to it**

**with open('books.csv', 'w', newline='') as file:**

**writer = csv.writer(file)**

**# Write the header row**

**writer.writerow(['title', 'author', 'year'])**

**# Write each book's data as a row in the CSV file**

**writer.writerows(books\_data)**

**This code will create a new file named books.csv in the current directory, and write the list of books data to it. Each row in the CSV file corresponds to a book, with the book's title, author, and year listed in separate columns. The first row of the file contains the header row with the column names.**

**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).**

**Sure, here's how you can use the sqlite3 module to create a SQLite database called books.db with a table called books and the required fields:**

**python**

**Copy code**

**import sqlite3**

**# Connect to the books.db database (it will be created if it doesn't exist)**

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

**# Create a books table with the required fields**

**conn.execute('''CREATE TABLE books**

**(title TEXT, author TEXT, year INTEGER)''')**

**# Commit the changes to the database and close the connection**

**conn.commit()**

**conn.close()**

**This code will create a new SQLite database named books.db in the current directory, with a single table named books. The books table has three columns: title (text), author (text), and year (integer). The conn.execute() method is used to execute SQL commands on the database, and the conn.commit() method is used to save the changes to the database. Finally, the conn.close() method is used to close the connection to the database.**

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

**Sure, here's how you can use the csv and sqlite3 modules to read the data from books.csv and insert it into the books table in the books.db database:**

**python**

**Copy code**

**import csv**

**import sqlite3**

**# Connect to the books.db database**

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

**# Open the books.csv file and read its contents**

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

**# Create a CSV reader object**

**reader = csv.reader(file)**

**# Skip the header row**

**next(reader)**

**# Iterate over the rows of the CSV file**

**for row in reader:**

**# Extract the values from the row**

**title, author, year = row**

**# Insert the values into the books table**

**conn.execute(f"INSERT INTO books (title, author, year) VALUES ('{title}', '{author}', {year})")**

**# Commit the changes to the database and close the connection**

**conn.commit()**

**conn.close()**

**This code will open the books.csv file, read its contents using a csv.reader object, and iterate over the rows of the file. For each row, it will extract the values for the title, author, and year columns, and insert them into the books table in the books.db database using an SQL INSERT INTO command. Finally, it will commit the changes to the database and close the connection.**

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

**Sure, here's how you can use the sqlite3 module to select and print the title column from the books table in alphabetical order:**

**python**

**Copy code**

**import sqlite3**

**# Connect to the books.db database**

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

**# Select the title column from the books table and print the results in alphabetical order**

**for row in conn.execute('SELECT title FROM books ORDER BY title ASC'):**

**print(row[0])**

**# Close the connection to the database**

**conn.close()**

**This code will connect to the books.db database and use an SQL SELECT statement to retrieve the title column from the books table. The ORDER BY clause is used to sort the results in alphabetical order. The conn.execute() method returns a cursor object, which can be used to iterate over the results of the query. For each row in the result set, the title column is printed to the console. Finally, the connection to the database is closed.**

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

**Sure, here's how you can use the sqlite3 module to select and print all columns from the books table, ordered by publication year:**

**python**

**Copy code**

**import sqlite3**

**# Connect to the books.db database**

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

**# Select all columns from the books table and print the results ordered by year**

**for row in conn.execute('SELECT \* FROM books ORDER BY year ASC'):**

**print(row)**

**# Close the connection to the database**

**conn.close()**

**This code will connect to the books.db database and use an SQL SELECT statement to retrieve all columns from the books table. The ORDER BY clause is used to sort the results by the year column in ascending order. The conn.execute() method returns a cursor object, which can be used to iterate over the results of the query. For each row in the result set, all columns are printed to the console. Finally, the connection to the database is closed.**

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

**Sure, here's how you can use the sqlalchemy module to connect to the books.db database:**

**python**

**Copy code**

**from sqlalchemy import create\_engine**

**# Create an engine that connects to the books.db database using the sqlite:// protocol**

**engine = create\_engine('sqlite:///books.db')**

**# Test the connection by executing a SELECT statement on the books table**

**result = engine.execute('SELECT \* FROM books')**

**for row in result:**

**print(row)**

**# Close the connection to the database**

**engine.dispose()**

**This code will use the create\_engine() function from the sqlalchemy module to create an engine object that connects to the books.db database using the sqlite:// protocol. The engine.execute() method is used to execute an SQL SELECT statement on the books table, and the results are printed to the console using a for loop. Finally, the engine.dispose() method is used to close the connection to the database.**

**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.**

**To install the Redis server and the Python redis library, you can follow these steps:**

**Install the Redis server by following the instructions for your operating system. For example, on Ubuntu Linux, you can run sudo apt-get install redis-server to install Redis.**

**Install the Python redis library by running pip install redis in your terminal or command prompt.**

**Once you have installed Redis and the Python redis library, you can create a Redis hash called test with the fields count (1) and name ('Fester Bestertester') and print all the fields for test using the following Python code:**

**python**

**Copy code**

**import redis**

**# Connect to the Redis server**

**r = redis.Redis(host='localhost', port=6379, db=0)**

**# Create the test hash with count=1 and name='Fester Bestertester'**

**r.hset('test', 'count', 1)**

**r.hset('test', 'name', 'Fester Bestertester')**

**# Print all fields for the test hash**

**print(r.hgetall('test'))**

**This code will use the redis module to connect to the Redis server running on localhost with the default port and database number. The r.hset() method is used to create the test hash with the fields count and name, and their respective values. Finally, the r.hgetall() method is used to retrieve all fields for the test hash, and the result is printed to the console.**

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

**Sure, here's how you can increment the count field of the test hash and print its new value:**

**python**

**Copy code**

**import redis**

**# Connect to the Redis server**

**r = redis.Redis(host='localhost', port=6379, db=0)**

**# Increment the count field of the test hash and print its new value**

**r.hincrby('test', 'count', 1)**

**print(r.hget('test', 'count'))**

**This code will use the hincrby() method from the redis module to increment the count field of the test hash by 1. The new value of the count field is then retrieved using the hget() method and printed to the console.**