**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.**

Here is one way to accomplish this:

```python

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

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

f.write(test1)

```

This code creates a string `test1` and then writes it to a new file named `test.txt` using the `with` statement and the `open()` function. The `'w'` argument tells Python to open the file in write mode, which will overwrite the file if it already exists or create it if it does not exist. The `f.write()` method is then used to write the contents of `test1` to the file. Once the `with` block is exited, the file is automatically closed.

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

Sure. Please use the following code to read the contents of the file test.txt into the variable test2:

```python

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

test2 = f.read()

```

This code opens the file named test.txt in read mode, reads its contents, and stores them in the variable test2.

There should be no difference between test1 and test2 since we're reading the contents of the file test.txt into test2, which was previously saved with the string 'This is a test of the emergency text system,'.

**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**

I'm sorry, as an AI language model, I don't have the ability to create files on your computer. However, I can provide you with the code that you can use to create a CSV file with the specified content.

You can create a file named `books.csv` in the current directory with the following code:

```python

import csv

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

writer = csv.writer(csvfile)

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

writer.writerow(['The Weirdstone of Brisingamen', 'Alan Garner', 1960])

writer.writerow(['Perdido Street Station', 'China Miéville', 2000])

writer.writerow(['Thud!', 'Terry Pratchett', 2005])

writer.writerow(['The Spellman Files', 'Lisa Lutz', 2007])

writer.writerow(['Small Gods', 'Terry Pratchett', 1992])

```

This code uses the `csv` module to create a new file called `books.csv`. It then writes the headers and data to the file using the `writerow` method. The `newline=''` parameter is used to ensure that the file is written with the correct line endings for your operating system.

**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).**

Here is the code to create a SQLite database called `books.db` and a table called `books` with the specified fields:

```python

import sqlite3

# create a connection to the database

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

# create a cursor to execute SQL commands

c = conn.cursor()

# create the books table

c.execute('''CREATE TABLE books

(title text, author text, year integer)''')

# commit the changes and close the connection

conn.commit()

conn.close()

```

This creates a database file called `books.db` and a table called `books` with three columns: `title`, `author`, and `year`.

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

Here's an example code that reads the data from `books.csv` and inserts it into the `books` table in the `books.db` database:

```python

import csv

import sqlite3

# Connect to the database

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

c = conn.cursor()

# Create the table

c.execute('''CREATE TABLE IF NOT EXISTS books

(title text, author text, year integer)''')

# Read the data from the CSV file and insert it into the table

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

reader = csv.reader(f)

next(reader) # Skip the header row

for row in reader:

c.execute('INSERT INTO books VALUES (?, ?, ?)', row)

# Commit the changes and close the connection

conn.commit()

conn.close()

```

Note that we first create the `books` table with the `title`, `author`, and `year` fields. Then, we use the `csv` module to read the data from the `books.csv` file and insert it into the table using SQL's `INSERT INTO` statement. Finally, we commit the changes and close the connection.

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

Here's the code to select and print the title column from the book table in alphabetical order:

```python

import sqlite3

# connect to the database

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

# create a cursor object

cur = conn.cursor()

# execute a select statement

cur.execute("SELECT title FROM books ORDER BY title ASC")

# fetch and print the results

results = cur.fetchall()

for row in results:

print(row[0])

# close the cursor and connection

cur.close()

conn.close()

```

This code connects to the books.db database, creates a cursor object, executes a SELECT statement to select the title column from the books table in alphabetical order, fetches and prints the results, and closes the cursor and connection.

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

Here's the code to select and print all columns from the `book` table in the order of publication:

```python

import sqlite3

# connect to the database

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

# create a cursor

c = conn.cursor()

# select all columns from the book table in the order of publication

c.execute('SELECT \* FROM book ORDER BY year')

# fetch and print all rows

rows = c.fetchall()

for row in rows:

print(row)

# close the cursor and connection

c.close()

conn.close()

```

This will print all rows from the `book` table in the order of publication, with all columns included.

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

Since I don't have access to files created in previous exercises, I will create a new database called `books.db` and a new table called `books` to proceed with the exercise.

Here's how to do that:

```python

import sqlite3

# create a connection to the database

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

# create a cursor object to execute SQL commands

c = conn.cursor()

# create a table called books with title, author, and year fields

c.execute('CREATE TABLE books (title TEXT, author TEXT, year INTEGER)')

# commit changes and close the connection

conn.commit()

conn.close()

```

Now we have a new database and table to work with. Let's move on to connecting to the `books.db` database using sqlalchemy.

```python

from sqlalchemy import create\_engine

# create an engine to connect to the database

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

# connect to the database using the engine

conn = engine.connect()

# execute a query to select all data from the books table

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

# print the result

for row in result:

print(row)

# close the connection

conn.close()

```

This should connect to the `books.db` database and print out all the data in the `books` table (which should be empty at this point).

**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.**

import redis

# create a connection to Redis

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

# create the hash test with fields count and name

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

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

# print all the fields for test

print(r.hgetall('test'))

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

To increment the count field of a Redis hash called test, you can use the `hincrby` method from the Python redis library. Here's an example:

```python

import redis

# Connect to Redis

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

# Increment the count field of the test hash by 1

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

# Print the new value of the count field

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

```

This code connects to a Redis instance running on the local machine, increments the `count` field of the `test` hash by 1 using the `hincrby` method, and then retrieves and prints the new value of the `count` field using the `hget` method.