In [1]:
```
# Q1
test1 = 'This is a test of the emergency text system,'
with open('test.txt', 'w') as file:
    file.write(test1)
```

In [2]:
```
# Q2
with open('test.txt', 'r') as file:
    test2 = file.read()
```

In [3]:
```
# Q3
import csv

header = ['title', 'author', 'year']
rows = [
    ['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]
]

with open('books.csv', 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(header)
    writer.writerows(rows)
```

In [4]:
```
# Q4
import sqlite3

# Create a connection to the database
conn = sqlite3.connect('books.db')

# Create a cursor object to execute SQL queries
c = conn.cursor()

# Create the books table with the title, author, and year fields
c.execute('CREATE TABLE books (title TEXT, author TEXT, year INTEGER)')

# Commit the changes to the database and close the connection
conn.commit()
conn.close()
```

In [5]:
```
# Q5
import csv
import sqlite3

# Open the books.csv file and read the data
with open('books.csv', 'r') as f:
    reader = csv.reader(f)
    # Skip the header row
    next(reader)
    # Iterate over the remaining rows and insert the data into the database
    for row in reader:
        title, author, year = row
        conn = sqlite3.connect('books.db')
        c = conn.cursor()
        c.execute('INSERT INTO books (title, author, year) VALUES (?, ?, ?)', (
        conn.commit()
        conn.close()
```

In [6]:
```
# Q6
import sqlite3

# connect to the database
conn = sqlite3.connect('books.db')

# create a cursor object
c = conn.cursor()

# execute the SELECT statement to retrieve the title column
c.execute("SELECT title FROM books ORDER BY title ASC")

# fetch all the rows and print them
rows = c.fetchall()
for row in rows:
    print(row[0])

# close the cursor and the connection
c.close()
conn.close()
```

```
Perdido Street Station
Small Gods
The Spellman Files
The Weirdstone of Brisingamen
Thud!
```

In [9]:
```
# Q7
import sqlite3

# Connect to the database
conn = sqlite3.connect('books.db')

# Create a cursor object
cur = conn.cursor()

# Select all columns from the book table in the order of publication
query = "SELECT * FROM book ORDER BY year ASC;"
cur.execute(query)

# Fetch all rows and print them
rows = cur.fetchall()
for row in rows:
    print(row)

# Close the cursor and database connections
cur.close()
conn.close()
```

In [10]:
```
# Q8
from sqlalchemy import create_engine

engine = create_engine('sqlite:///books.db')
```

In [15]:
```
pip install redis
```

```
Collecting redis
  Downloading redis-4.5.4-py3-none-any.whl (238 kB)
     -------------------------------------- 238.9/238.9 kB 2.1 MB/s eta 0:00:
00
Requirement already satisfied: async-timeout>=4.0.2 in c:\users\em\anaconda3
\lib\site-packages (from redis) (4.0.2)
Installing collected packages: redis
Successfully installed redis-4.5.4
Note: you may need to restart the kernel to use updated packages.
```

In [16]:
```
# Q9
import redis

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

# create the Redis hash
r.hset('test', 'count', 1)
r.hset('test', 'name', 'Fester Bestertester')

# print all fields for test
print(r.hgetall('test'))
```

```
---------------------------------------------------------------------------
ConnectionRefusedError                    Traceback (most recent call last)
~\anaconda3\lib\site-packages\redis\connection.py in connect(self)
    697         try:
--> 698             sock = self.retry.call_with_retry(
    699                 lambda: self._connect(), lambda error: self.disconnec
t(error)

~\anaconda3\lib\site-packages\redis\retry.py in call_with_retry(self, do, fai
l)
     45             try:
---> 46                 return do()
     47             except self._supported_errors as error:

~\anaconda3\lib\site-packages\redis\connection.py in <lambda>()
    698             sock = self.retry.call_with_retry(
--> 699                 lambda: self._connect(), lambda error: self.disconnec
t(error)
    700             )

~\anaconda3\lib\site-packages\redis\connection.py in _connect(self)
    986         if err is not None:
--> 987             raise err
    988         raise OSError("socket.getaddrinfo returned an empty list")

~\anaconda3\lib\site-packages\redis\connection.py in _connect(self)
    974                 # connect
--> 975                 sock.connect(socket_address)
    976 

ConnectionRefusedError: [WinError 10061] No connection could be made because
the target machine actively refused it

During handling of the above exception, another exception occurred:

ConnectionError                           Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_13700\848354280.py in <module>
      6 
      7 # create the Redis hash
----> 8 r.hset('test', 'count', 1)
      9 r.hset('test', 'name', 'Fester Bestertester')
     10 

~\anaconda3\lib\site-packages\redis\commands\core.py in hset(self, name, key,
value, mapping, items)
   4930                 items.extend(pair)
   4931 
-> 4932         return self.execute_command("HSET", name, *items)
   4933 
   4934     def hsetnx(self, name: str, key: str, value: str) -> Union[Awaita
ble[bool], bool]:

~\anaconda3\lib\site-packages\redis\client.py in execute_command(self, *args,
**options)
   1253         pool = self.connection_pool
   1254         command_name = args[0]
-> 1255         conn = self.connection or pool.get_connection(command_name, *
*options)
   1256
   1257         try:

~\anaconda3\lib\site-packages\redis\connection.py in get_connection(self, com
mand_name, *keys, **options)
   1440         try:
   1441             # ensure this connection is connected to Redis
-> 1442             connection.connect()
   1443             # connections that the pool provides should be ready to s
end
   1444             # a command. if not, the connection was either returned t
o the

~\anaconda3\lib\site-packages\redis\connection.py in connect(self)
    702             raise TimeoutError("Timeout connecting to server")
    703         except OSError as e:
--> 704             raise ConnectionError(self._error_message(e))
    705
    706         self._sock = sock

ConnectionError: Error 10061 connecting to localhost:6379. No connection coul
d be made because the target machine actively refused it.
```

In [ ]:
```
#  Q10
import redis

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

# increment the count field
r.hincrby('test', 'count', 1)

# print the updated count field
count = r.hget('test', 'count')
print(count)
```

In [ ]: