File Handling

Hexavarsity

Objective

- File Handling
- File Creating
- File Reading
- File Writing
- File operation mode
- Database connection and operations

What is Flie Handling?

- To store data temporarily and permanently, we use files. A file is the collection of data stored on a disk in one unit identified by filename.
- Python too supports file handling and allows users to handle files i.e., to read and write files, along with many other file handling options, to operate on files.
- Text File: Text file usually we use to store character data. For example, test.txt
- Binary File: The binary files are used to store binary data such as images, video files, audio files, etc.

Create File in Python

- Create a file using the built-in function open().
- open function contain parameters of file name and access mode to to create a file.
- open() returns the file object. This object is used to read or write the file according to the access mode.
- Accesss mode represents the purpose of opening the file. For example, R is for reading and W is for writing

Syntax: open('file_Path', 'access_mode')	access modes for creating a file.
File Mode	**Meaning**
w	Create a new file for writing. If a file already exists, it truncates the file first. Use to create and write content into a new file.
x	Open a file only for exclusive creation. If the file already exists, this operation fails.
a	Open a file in the append mode and add new content at the end of the file.
b	Create a binary file
t	Create and open a file in a text mode

Creating a file

- Below example create a file and write some text to file.
- The file is created in the same directory where our program/script is running.

Example:

```
# create a empty text file
fp = open('sales_2.txt', 'w')
fp.write('first line')
fp.close()
```

- Create File In A Specific Directory: include the complete directory path required to locate the file. As mentioned below

Example:

```
# create a text file for writing

with open(r'E:\pynative\reports\profit.txt', 'w') as fp: fp.write('This is first line') pass
```

- Note: Using the with statement a file is closed automatically it ensures that all the resources that are tied up with the file are released.

File Access Modes in Python

Mode	Meaning
r	It opens an existing file to read-only mode. The file pointer exists at the beginning.
rb	It opens the file to read-only in binary format. The file pointer exists at the beginning.
r+	It opens the file to read and write both. The file pointer exists at the beginning.
rb+	It opens the file to read and write both in binary format. The file pointer exists at the beginning of the file.
w	It opens the file to write only. It overwrites the file if exists or creates a new if no file exists with the same name.
wb	It opens the file to write only in binary format. It overwrites the file if it exists or creates a new one if no file exists.
w+	It opens the file to write and read data. It will override existing data.
wb+	It opens the file to write and read both in binary format
a	It opens the file in append mode. It will not override existing data. It creates a file if no file exists in same name.
ab	It opens the file in the append mode in binary format.
a+	It opens a file to append and read both.
ab+	It opens a file to append and read both in binary format.

Reading a file

- To read or write a file, we need to open that file. For this purpose, Python provides a built-in function open().

Sample.txt:

```
Welcome to hexavarsity
This is a sample.txt
Line 3
Line 4
Line 5
```

Example:

```
# Opening the file with absolute path
        fp = open('E:\demos\files\sample.txt', 'r')
# read file
        print(fp.read())
# Closing the file after reading
        fp.close()
```

- Another way to read a file is to call a certain number of characters like in the following code the interpreter will read the first five characters of stored data and return it as a string:

Example:

```
# Opening the file with absolute path
        fp = open('E:\demos\files\sample.txt', 'r')
# read file
        print(fp.read(5))
# Closing the file after reading
        fp.close()
```

Writing to a File

- To write content into a file, Use the access mode w to open a file in a write mode.
- If a file already exists, it truncates the existing content and places the filehandle at the beginning of the file. A new file is created if the mentioned file doesn't exist.
- If you want to add content at the end of the file, use the access mode a to open a file in append mode

Example:

```
text = "This is new content"
# writing new content to the file
fp = open("write_demo.txt", 'w')
fp.write(text) print('Done Writing')
fp.close()
```


Move File Pointer seek() and tell()

- The seek() method is used to change or move the cursor to defines where the data must be read or written in the file.
- The position (index) of the first character in files is zero, just like the string index.
- The tell() method to return the current position of the file pointer from the beginning of the file.

Welcome to hexavarsity
This is a sample.txt
Line 3
Line 4
Line 5

Sample.txt:

```
f = open("sample.txt", "r")
# move to 11 character
f.seek(11)              # read from 11th character
print(f.tell())
print(f.read())
f.close()
```

Example:

```
11
hexavarsity
This is a sample.txt
Line 3
Line 4
Line 5
```

Output

Python File Methods

Mode	Meaning
read()	Returns the file content.
readline()	Read single line
readlines()	Read file into a list
truncate(size)	Resizes the file to a specified size.
write()	Writes the specified string to the file.
writelines()	Writes a list of strings to the file.
close()	Closes the opened file.
seek()	Set file pointer position in a file
tell()	Returns the current file location.
fileno()	Returns a number that represents the stream, from the operating system's perspective.
flush()	Flushes the internal buffer.

Rename Files and Delete Files

- the os module provides the functions for file processing operations such as renaming, deleting the file, etc.
- The os module enables interaction with the operating system.
- The os module provides rename() method to rename the specified file name to the new name.

Example:

```
import os # Absolute path of a file
old_name = "E:\demos\files\reports\details.txt"
new_name = "E:\demos\files\reports\new_details.txt"
# Renaming the file
os.rename(old_name, new_name)
```

- the os module provides the remove() function to remove or delete file path.

Example:

```
import os
# remove file with absolute path
os.remove(r"E:\demos\files\sales_2.txt")
```

Implementing all the functions in file handling

```
import os
 def create_file(filename):
   try:
     with open(filename, 'w') as f:
       f.write('Hello, world!\n')
     print("File " + filename + " created successfully.")
   except IOError:
     print("Error: could not create file " + filename)

def read_file(filename):
   try:
     with open(filename, 'r') as f:
       contents = f.read()
       print(contents)
   except IOError:
     print("Error: could not read file " + filename)
def append_file(filename, text):
   try:
     with open(filename, 'a') as f:
       f.write(text)
     print("Text appended to file " + filename )
   except IOError:
     print("Error: could not append to file " + filename)
```

```
def rename_file(filename, new_filename):
   try:
     os.rename(filename, new_filename)
     print("File  renamed to " + new_filename )
   except IOError:
     print("Error: could not rename file " + filename)

def delete_file(filename):
   try:
     os.remove(filename)
     print("File " + filename + " deleted successfully.")
   except IOError:
     print("Error: could not delete file " + filename)

filename,new_filename="example.txt“, "new_example.txt"

create_file(filename)
read_file(filename)
append_file(filename, "This is some additional text.\n")
read_file(filename)
rename_file(filename, new_filename)
read_file(new_filename)
delete_file(new_filename)
```

Example:

Working With binary file

- When we open a file in text mode, that file is decoded from bytes to a string object. when we open a file in the binary mode it returns contents as bytes object without decoding.
- Open the file in binary write mode using wb
- Specify contents to write in the form of bytes.
- Use the write() function to write byte contents to a binary file.

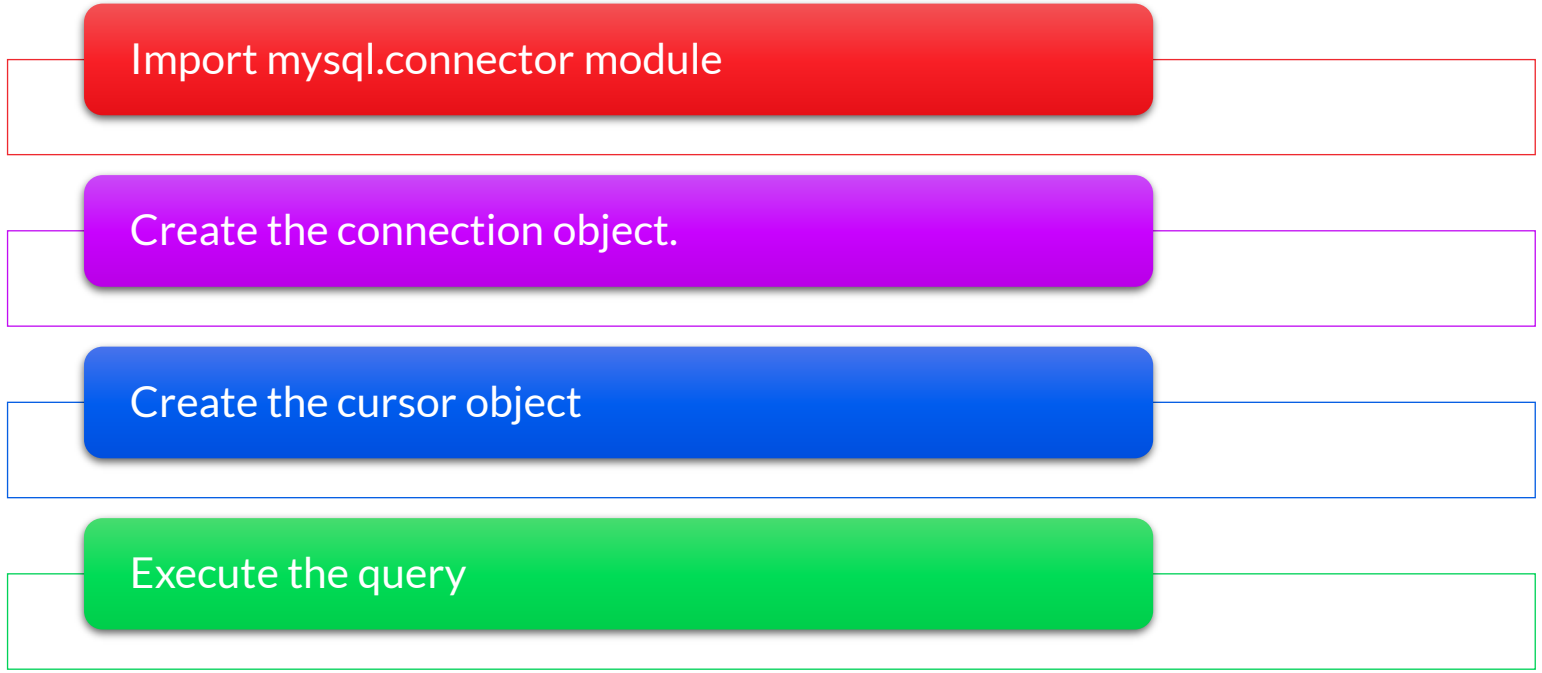
Example:

```
bytes_data = b'\x21’
with open("test.txt", "wb") as fp:
# Write bytes to file
fp.write(bytes_data)
```


Database connection

Database connectivity

- Python allows developers to interact with various relational databases (such as Oracle, SQLite, and MySQL) and relational database systems (RDBMS) through several different libraries.
- To create a connection to a MySQL database in Python download the Python MySQL database connector se PIP with the following command:
    - **$ pip install mysql-connector-python**
- There are the following steps to connect a python application to our database.



Creating the connection

- To create a connection between the MySQL database and the python application, the connect() method of mysql.connector module is used.

- Pass the database details like HostName, username, and the database password in the method call. The method returns the connection object.

user: The username used to connect to a database
password: The users password
host: IP address of the host where the database is hosted. Note that the default is 127.0.0.1 (localhost)
database: Name of the database you are trying to connect to

```
# Python implementation to create a Database in MySQL
import mysql.connector

# connecting to the mysql server
db = mysql.connector.connect(
  host="localhost",
  user="root",
  passwd="password"
)
```

Example:

Execute a MySQL Query in Python

- In order to execute a MySQL query, you need a MySQLCursor object. This object interacts with a MySQLConnection object and the database.

```
c = db.cursor()
```

- then use the execute() method to process a query:

```
my-query = “SELECT * FROM Students”
result = c.execute(my-query)
```

- If you are using the INSERT, DELETE, or UPDATE statements, then you need to commit your connection after executing the query.

```
db.commit()
```

- After you are finished interacting with the database, ensure that you close the connection and the cursor to release system resources. You can do this using the close() method:

```
cursor.close()
conn.close()
```

Create a Database in MySQL

- Creating Database through python

```
import mysql.connector
db = mysql.connector.connect(host="localhost",user="root",passwd="password")
c = db.cursor()
c.execute("CREATE DATABASE employee_db")
c.execute("SHOW DATABASES")
for i in c:
   print(i)
c = db.cursor()
db.close()
```

Example:

Create a table in MySQL

Column name	Data type	Description
empid	INT	Stores the employee id and has auto-increment i.e. increments every time a record is added
empname	VARCHAR(45)	Stores the employee's name
department	VARCHAR(45)	Stores the department to which the employee belongs i.e. accounts, HR.
salary	INT	Stores the salary of the employees.

- Creating above mentioned Table through python

```
import mysql.connector
db = mysql.connector.connect(host="localhost",user="root",passwd="password“,database="employee_db")
c = db.cursor()
employeetbl_create = """CREATE TABLE `employee_db`.`tblemployee` (`empid` INT NOT NULL
                        AUTO_INCREMENT, `empname` VARCHAR(45) NULL,  `department` VARCHAR(45) NULL,
                        `salary` INT NULL, PRIMARY KEY (`empid`))"""“
c.execute(employeetbl_create)
c = db.cursor()
db.close()
```

Example:

Insert a records in table

- Creating above mentioned Table through python

```
import mysql.connector
db = mysql.connector.connect(host="localhost",user="root",passwd="password“,database="employee_db")
c = db.cursor()
employeetbl_insert = """INSERT INTO tblemployee (
  empname,
  department,
  salary)
  VALUES  (%s, %s, %s)"""

# we save all the row data to be inserted in a data variable
data = [("Vani", "HR", "100000"),
      ("Krish", "Accounts", "60000"),
      ("Aishwarya", "Sales", "25000"),
      ("Govind", "Marketing", "40000")]

# execute the insert commands for all rows and commit to the database
c.executemany(employeetbl_insert, data)
db.commit()
db.close()
```

Example:

Reading / Selecting records in table

- Creating above mentioned Table through python

```
import mysql.connector
db = mysql.connector.connect(host="localhost",user="root",passwd="password“,database="employee_db")
c = db.cursor()
# select statement for tblemployee which returns all columns
employeetbl_select = """SELECT * FROM tblemployee"""

# execute the select query to fetch all rows
c.execute(employeetbl_select)

# fetch all the data returned by the database
employee_data = c.fetchall()

# print all the data returned by the database
for e in employee_data:
   print(e)
db.close()
```

Example:

Updating records in table

- Creating above mentioned Table through python

```
import mysql.connector
db = mysql.connector.connect(host="localhost",user="root",passwd="password“,database="employee_db")
c = db.cursor()
# update statement for tblemployee
# which modifies the salary of Vani
employeetbl_update = "UPDATE tblemployee\
SET salary = 115000 WHERE empid = 1"

# execute the update query to modify
# the salary of employee with
# employee id = 1 and commit to the database
c.execute(employeetbl_update)
db.commit()
db.close()
```

Example:

Deleting records in table

- Deleting data from tables has to be done with utmost care as it can lead to the loss of important data at times.
- Often a soft delete is performed where there is an extra column named “active” whose values are 1 or 0 only. 1 means present in the table and 0 means deleted from being displayed.

```
import mysql.connector
db = mysql.connector.connect(host="localhost",user="root",passwd="password“,database="employee_db")
c = db.cursor()
# delete statement for tblemployee
# which deletes employee Aishwarya having empid 3
employeetbl_delete = "DELETE FROM tblemployee WHERE empid=3"

# execute the delete statement and commit to the database
c.execute(employeetbl_delete)
db.commit()
db.close()
```

Example:

Queries

References

- https://www.databasejournal.com/features/python-databases/#:~:text=Python%20allows%20developers%20to%20interact,standards%20defined%20in%20PEP%20249.
- https://www.geeksforgeeks.org/crud-operation-in-python-using-mysql/
- https://pynative.com/python-mysql-database-connection/

Thank you

Innovative Services

Passionate Employees

Delighted Customers