Day 6 Tutorial : File Processing

Task 1: Knowledge and understanding (File Processing)

1. What is the use of the open() function?

The open() function in Python is used for opening files. It is a built-in function that allows you to work with files in various modes, such as reading, writing, and appending data.

2. Explain the different modes used when opening a file.

- r (read): This mode opens the file for reading. If the file does not exist, an error will be raised.
- w (write): This mode opens the file for writing. If the file exists, it will be truncated to zero bytes. If the file does not exist, it will be created.
- a (append): This mode opens the file for writing. If the file exists, the data written to the file will be appended to the end of the file. If the file does not exist, it will be created.
- r+ (read and write): This mode opens the file for reading and writing. The file must already exist.
- w+ (write and read): This mode opens the file for writing and reading. If the file exists, it will be truncated to zero bytes. If the file does not exist, it will be created.
- a+ (append and read): This mode opens the file for writing and reading. If the file exists, the data written to the file will be appended to the end of the file. If the file does not exist, it will be created.

The mode used when opening a file should be chosen based on the operation that needs to be performed on the file. For example, if you need to read the contents of a file, you would use the r mode. If you need to write to a file, you would use the w or a mode.

3. How do you read the entire content of a file into a string?

- To read the entire content of a file into a string in Python, you can use the read() method. The read() method reads the entire file and returns the contents of the file as a string.

4. Describe the difference between the read() , readline() and readlines() function

- read(): The read() function reads the entire file into memory and returns the contents of the file as a string.
- readline(): The readline() function reads a single line from the file and returns the line as a string.
- readlines(): The readlines() function reads all of the lines from the file and returns the lines as a list of strings.

5. What is the purpose of the with statement (context manager) when working with files, and why is it recommended?

The with statement in Python is a context manager that can be used to ensure that resources are properly managed and closed. When working with files, the with statement is used to ensure that the file is closed properly, even if an exception is raised.

- Using the with statement is recommended when working with files because it ensures that the file is always closed properly. This can help to prevent data corruption and other problems.

6. Why do you need to close a file after finished working with it.

- To release system resources. When you open a file, the operating system allocates resources to the file, such as memory and file handles. When you close the file, the operating system releases these resources. If you do not close the file, the operating system will eventually run out of resources and your program may crash.
- To prevent data corruption. When you are writing to a file, the data is not actually written to the disk until you close the file. If you do not close the file, the data may be corrupted if the program crashes or the computer is shut down unexpectedly.
- To ensure that the file is saved. When you are writing to a file, the data is not actually saved to the disk until you close the file. If you do not close the file, the data may be lost if the program crashes or the computer is shut down unexpectedly.

Task 2: Practical (File processing)

1) Write the following content into a file called studentlist.txt

Hari Prasanth

Dinesh Raj

Magaraju Mohith

Teja sri

```
# 1) Write the following content into a file called studentlist.txt
# Hari Prasanth
# Dinesh Raj
# Magaraju Mohith
# Teja sri

content = """Hari Prasanth
Dinesh Raj
Magaraju Mohith
Teja sri
"""

# Open the file in write mode ('w') and write the content
with open("studentlist.txt", "w") as file:
    file.write(content)

print("Content has been written to studentlist.txt.")
```

2) Write a program that reads the content of file in q1 and display it on the screen.

```
# 2) Write a program that reads the content of file in q1 and display
# it on the screen.

# Open the file in read mode ('r') and read the content
with open("studentlist.txt", "r") as file:
    content = file.read()

# Display the content on the screen
print("Content of studentlist.txt:")
print(content)
```

3) Write a program that amend the Q2 and counts the number of student in it. Print the student count at the end of the program.

```
# Write a program that amend the Q2 and counts the number of student in it.
Print
# the student count at the end of the program.

# Open the file in read mode ('r') and read the content
with open("studentlist.txt", "r") as file:
    content = file.read()

# Display the content on the screen
print("Content of studentlist.txt:")
print(content)

# Count the number of students by splitting the content into lines
lines = content.split("\n")
student_count = len([line for line in lines if line.strip()])

print(f"Number of students: {student_count}")
```

4) Write a program to prompts user for a new name and appends it to an existing text file.

```
# Write a program to prompts user for a new name and appends it to an
existing text
# file.

# Prompt the user for a new name
new_name = input("Enter a new name to append to the file: ")

# Open the file in append mode ('a') and write the new name
with open("studentlist.txt", "a") as file:
    file.write(new_name + "\n")

print(f"{new_name} has been appended to studentlist.txt.")
```

5) Write a program that update the studentlist file, user sshould be able to specify the search and replacement words, the program will search for searches for a specific word, and replaces it with another word based on user's input.

```
#  Write a program that update the studentlist file, user sshould be able
to specify the
# search and replacement words, the program will search for searches for a
specific
```

```
# word, and replaces it with another word based on user's input.


# Prompt the user for the word to search for
search_word = input("Enter the word to search for: ")

# Prompt the user for the replacement word
replacement_word = input("Enter the replacement word: ")

# Read the content of the file
with open("studentlist.txt", "r") as file:
    content = file.read()

# Perform the search and replace operation
updated_content = content.replace(search_word, replacement_word)

# Write the updated content back to the file
with open("studentlist.txt", "w") as file:
    file.write(updated_content)

print(f"File 'studentlist.txt' has been updated. '{search_word}' has been
replaced with '{replacement_word}'.")
```