



|  |
| --- |
| Experiment No.5 |
| Exploring Files and Directories: Python program data to existing file and then display the entire file |
| Date of performance: 14/02/2024 |
| Date of submission: 21/02/2024 |



















**Code:**

f=open("hi1.txt","w")

str=input("Enter a string: ")

f.write(str)

f.close()

f=open("hi1.txt","r")

str1=f.read()

print(str1)

f.close()

f=open("hi2.txt","w")

print("Enter characters till the symbol @ is pressed")

while str!='@':

str=input()

if(str !='@'):

f.write(str+'\n')

else:

f.close()

f=open("hi1.txt","a+")

print("Enter characters till the symbol @ is pressed")

while str!='@':

str=input()

if(str !='@'):

f.write(str+'\n')

else:

f.seek(0,0)

f=open("hi1.txt","r")

str2=f.read()

print(str2)

f.close()

**Output:**

Enter a string: Anjali

Anjali

Enter characters till the symbol @ is pressed

This exp is based on directories

@

Anjali

**Conclusion:**

Working with files and directories is a fundamental aspect of many programming tasks, including data processing, file management, and system administration. Python provides several built-in modules and functions to facilitate these operations.

You can open files using the open( ) function, which takes the file path and an optional mode argument specifying the purpose of opening the file (e.g., read, write, append).

Use methods like read( ) to read data from the file.

If the file is opened in write or append mode, you can use methods like write( ) to write data to the file. Always remember to close the file after you're done with it to release system resources.

Python supports the use of context managers (the with statement) for file handling. It automatically closes the file when the block of code is exited, even if an exception occurs.