3. Student Activity

Student Activity: Practicing File Handling in Python

Welcome to the hands-on activity session on **File Handling in Python**! In this activity, you'll practice opening, reading, writing, and closing files using Python. Follow the steps and examples provided to reinforce your understanding of file handling concepts.

---

1. Opening a File

Before you can read or write to a file, you need to open it using the `open()` function. Let's practice opening files in different modes.

Example 1: Open a File in Read Mode

```
file = open("example1.txt", "r")
print("File opened in read mode.")
file.close()
```

Example 2: Open a File in Write Mode

```
file = open("example2.txt", "w")
print("File opened in write mode.")
file.close()
```

Example 3: Open a File in Append Mode

```
file = open("example3.txt", "a")
print("File opened in append mode.")
file.close()
```

---

2. Reading from a File

Once a file is opened in read mode, you can read its content. Practice reading files using different methods.

Example 1: Read the Entire File

```
file = open("example1.txt", "r")
content = file.read()
print("File content:\n", content)
file.close()
```

Example 2: Read File Line by Line

```
file = open("example1.txt", "r")
for line in file:
    print("Line:", line.strip())
file.close()
```

Example 3: Read Specific Number of Characters

```
file = open("example1.txt", "r")
content = file.read(10)  # Read the first 10 characters
print("First 10 characters:", content)
file.close()
```

---

3. Writing to a File

To write data to a file, open it in write or append mode. Practice writing data to files.

Example 1: Write to a File in Write Mode

```
file = open("example2.txt", "w")
file.write("Hello, Python!\n")
```

```
file.write("Writing to a file.\n")
file.close()
```

Example 2: Append Data to a File

```
file = open("example3.txt", "a")
file.write("Appending new data.\n")
file.close()
```

Example 3: Overwrite Existing File Content

```
file = open("example2.txt", "w")
file.write("This will overwrite the previous content.\n")
file.close()
```

---

4. Closing a File

Always close a file after you're done with it to free up resources.

Example 1: Close a File After Reading

```
file = open("example1.txt", "r")
content = file.read()
file.close()
print("File closed after reading.")
```

Example 2: Close a File After Writing

```
file = open("example2.txt", "w")
file.write("Closing the file after writing.\n")
file.close()
print("File closed after writing.")
```

Example 3: Ensure File is Closed Using `with` Statement

```
with open("example1.txt", "r") as file:
    content = file.read()
print("File automatically closed after exiting the block.")
```

---

5. Handling Errors with Try-Except

Use try-except blocks to handle errors gracefully when working with files.

Example 1: Handle File Not Found Error

```
try:
    file = open("nonexistent.txt", "r")
except FileNotFoundError:
    print("The file does not exist.")
```

Example 2: Handle Permission Error

```
try:
    file = open("/root/protected.txt", "r")
except PermissionError:
    print("You do not have permission to access this file.")
```

Example 3: General Exception Handling

```
try:
    file = open("example1.txt", "r")
    content = file.read()
    file.close()
except Exception as e:
    print(f"An error occurred: {e}")
```

Conclusion

In this activity, you've practiced the basics of file handling in Python, including opening files in different modes, reading and writing data, and handling errors. Remember to always close files after use and handle potential errors gracefully.

Feel free to experiment with these examples and modify them to deepen your understanding. If you have any questions or need further clarification, don't hesitate to ask!