Module 3: Introduction to File Handling

Overview

File handling in Python is a fundamental aspect of programming that allows you to work with files on your computer. This includes reading data from files, writing data to files, and manipulating file contents. This document provides an introduction to file handling in Python, covering the basics of opening, reading, and writing files.

Opening Files

Before you can read from or write to a file, you need to open it. Python provides the `open()` function for this purpose. The `open()` function takes two arguments: the file path and the mode in which you want to open the file.

Modes:

- `"r"`: Read mode (default). Opens the file for reading.
- `"w"`: Write mode. Opens the file for writing. If the file doesn't exist, creates a new file. If the file exists, truncates it.
- `"a"`: Append mode. Opens the file for writing. If the file doesn't exist, creates a new file. If the file exists, appends data to it.
- `"b"`: Binary mode. Opens the file in binary mode (e.g., `"rb"`, `"wb"`).

Reading from Files:

Once you've opened a file for reading, you can read its contents using various methods such as `read()`, `readline()`, or `readlines()`. These methods allow you to read the entire file, one line at a time, or all lines into a list, respectively.

Writing to Files:

When a file is opened in write mode, you can write data to it using the `write()` method. You can also use the `writelines()` method to write a sequence of lines to the file.

Example:

Let's consider an example of reading from and writing to a text file.

```
# Open a file for writing
with open("sample.txt", "w") as file:
 # Write data to the file
 file.write("Hello, world!\n")
 file.write("This is a sample file.")

# Open the file for reading
with open("sample.txt", "r") as file:
 # Read and print the contents of the file
 print(file.read())
```

Conclusion:

File handling is an essential part of programming, allowing you to interact with external data stored in files. Understanding how to open, read, and write files in Python is crucial for various tasks, such as data processing, log file analysis, and more. With the knowledge gained from this introduction, you'll be able to perform basic file operations in Python effectively.

Exercise 1:
Create a text file named "example.txt" and write the following content to it:

```
Python is a powerful programming language.
It is widely used for various applications.
```

Answer 1:

```
# Write content to the file
with open("example.txt", "w") as file:
 file.write("Python is a powerful programming language.\n")
 file.write("It is widely used for various applications.\n")
```

Exercise 2:
Read the contents of the "example.txt" file and print each line.

Answer 2:

```
# Read and print the contents of the file
with open("example.txt", "r") as file:
 for line in file:
 print(line.strip()) # Strip newline characters
```

Exercise 3:
Append the following line to the "example.txt" file:

```
Python has a large ecosystem of libraries and frameworks.
```

Answer 3:

```
# Append content to the file
with open("example.txt", "a") as file:
 file.write("Python has a large ecosystem of libraries and frameworks.\n")
```

Exercise 4:
Create a new text file named "numbers.txt" and write numbers from 1 to 10, each on a new line.

Answer 4:

```
# Write numbers to the file
```

```
with open("numbers.txt", "w") as file:
 for i in range(1, 11):
 file.write(str(i) + "\n")
```

Exercise 5:
Read the numbers from the "numbers.txt" file, calculate their sum, and print the result.

Answer 5:

```
# Read numbers from the file and calculate sum
total = 0
with open("numbers.txt", "r") as file:
 for line in file:
 total += int(line.strip()) # Convert string to integer and strip newline characters

# Print the sum
print("Sum of numbers:", total)
```