Reading and Writing Files

Key Concepts:

- Introduction to File Handling
- Reading from a File
- Writing to a File
- Appending to a File
- Closing a File
- Examples of File Operations

Introduction to File Handling

- Up to now, we have used `input()` and `print()` to read program inputs and display results.
- Sometimes, input data is stored in a file, and we want to access it directly without manual entry.
- It is also often useful to save our results in files to access them later. For example, saving a game state in a video game.
- In Python, it is very easy to read and write data in files.

Accessing a Text File

- Before starting to read a file, it must first be opened. Opening a file simply means creating a variable to manipulate it.
- The `open()` function is used to open a file. For example, to open the file called "data.txt", simply do:

```
f = open('data.txt')
```

- By default, `open()` opens a file in "read" mode, which means we cannot modify its content.
- If we try to open a non-existent file in read mode, we get an error:

```
f = open('toto')  # the file 'toto' does not exist
# Traceback (most recent call last):
# IOError: [Errno 2] No such file or directory: 'toto'
```

Reading from a Text File

- Once our text file is opened, there are several ways to read its content:

Read the entire content at once into a string:

```
text = f.read()  # the string 'text' contains all the text in the file
```

Read all the lines at once into a list of strings:

```
lines = f.readlines()  # 'lines' is a list containing the lines of the file
```

Read the file line by line in a `for` loop:

```
for line in f:
    print(line)  # prints each line of the file
```

Writing to a Text File

- To write to a file, it must be opened in write mode:

```
f = open('file.txt', 'w')
```

- If the file opened in write mode does not exist, it will be created. If it already exists, all its content will be erased.
- The function to write to a text file is `write()`:

```
f.write('this text will be written to the file')
```

- Unlike `print()`, the `write()` function does not automatically add a newline. To add a newline, write it manually:

```
f.write('\n')  # this adds a new line
```

- The argument passed to `write()` must be a string. To write an integer or another type, convert it to a string using `str()`.

Appending to a Text File

- To write to the end of a file without erasing its content, open it in append mode:

```
f = open('file.txt', 'a')
```

- If the file opened in append mode does not exist, it will be created. If it exists, unlike opening in write mode, the content will not be erased. All writes will be added to the existing content.
- Use the `write()` method to add text to the end of the file:

```
f.write('this text will be written at the end of the file')
```

Closing a File

- Once reading/writing is finished, close the file using the `close()` function:

```
f = open('file.txt')

for line in f:
    print('a line read:', line)

f.close()
```

Example: Reading from a File

- Consider the file "numbers.txt" which contains integers (one per line).
- We want to calculate the sum of these integers:

```
file = open('numbers.txt')
sum = 0

for number in file:
    sum = sum + int(number)

print(sum)
file.close()
```

Example: Writing to a File

- We want to save the first 10 powers of 2 in a file "powers.txt":

```
file = open('powers.txt', 'w')

for i in range(10):
    file.write(str(2 ** i) + '\n')

file.close()
```