**File handling and input/output operations in python**

File handling and input/output (I/O) operations in Python allow you to work with files for reading, writing, and manipulating data. Python provides built-in functions and methods to handle files efficiently.  
  
---  
  
## \*\*File Handling in Python\*\*  
  
### \*\*1. Opening a File\*\*  
Python uses the `open()` function to open a file. The syntax is:  
  
```python  
file = open(filename, mode)  
```  
  
- \*\*`filename`\*\*: Name of the file to open.  
- \*\*`mode`\*\*: Specifies the mode in which the file is opened. Common modes:  
 - `'r'`: Read (default mode).  
 - `'w'`: Write (overwrites the file if it exists, creates a new file if it doesn't).  
 - `'a'`: Append (adds data to the end of the file).  
 - `'b'`: Binary mode (e.g., `'rb'` for reading binary files).  
 - `'x'`: Create a new file (fails if the file already exists).  
 - `'r+'`: Read and write.  
  
---  
  
### \*\*2. Reading from a File\*\*  
You can read data from a file using methods like `read()`, `readline()`, or `readlines()`.  
  
#### Example: Reading the Entire File  
```python  
# Open the file in read mode  
file = open("example.txt", "r")  
  
# Read the entire content of the file  
content = file.read()  
print(content)  
  
# Close the file  
file.close()  
```  
  
#### Example: Reading Line by Line  
```python  
file = open("example.txt", "r")  
  
# Read one line at a time  
line = file.readline()  
while line:  
 print(line.strip()) # Remove trailing newline  
 line = file.readline()  
  
file.close()  
```  
  
#### Example: Reading All Lines into a List  
```python  
file = open("example.txt", "r")  
  
# Read all lines into a list  
lines = file.readlines()  
print(lines)  
  
file.close()  
```  
  
---  
  
### \*\*3. Writing to a File\*\*  
You can write data to a file using the `write()` or `writelines()` methods.  
  
#### Example: Writing to a File  
```python  
# Open the file in write mode  
file = open("example.txt", "w")  
  
# Write a single line  
file.write("Hello, World!\n")  
  
# Write multiple lines  
file.writelines(["This is line 1.\n", "This is line 2.\n"])  
  
file.close()  
```  
  
#### Example: Appending to a File  
```python  
# Open the file in append mode  
file = open("example.txt", "a")  
  
# Append a new line  
file.write("This is an appended line.\n")  
  
file.close()  
```  
  
---  
  
### \*\*4. Using `with` Statement\*\*  
The `with` statement is used to handle files because it automatically closes the file after the block is executed, even if an exception occurs.  
  
#### Example: Reading a File with `with`  
```python  
with open("example.txt", "r") as file:  
 content = file.read()  
 print(content)  
# No need to call file.close()  
```  
  
#### Example: Writing to a File with `with`  
```python  
with open("example.txt", "w") as file:  
 file.write("This is written using the 'with' statement.\n")  
```  
  
---  
  
### \*\*5. File Modes\*\*  
| \*\*Mode\*\* | \*\*Description\*\* |  
|----------|---------------------------------------------------------------------------------|  
| `'r'` | Read mode (default). File must exist. |  
| `'w'` | Write mode. Overwrites the file if it exists, creates a new file if it doesn't. |  
| `'a'` | Append mode. Adds data to the end of the file. |  
| `'b'` | Binary mode (used with `'r'`, `'w'`, or `'a'`). |  
| `'x'` | Create mode. Fails if the file already exists. |  
| `'r+'` | Read and write mode. File must exist. |  
  
---  
  
### \*\*6. Checking if a File Exists\*\*  
You can use the `os` module to check if a file exists.  
  
```python  
import os  
  
if os.path.exists("example.txt"):  
 print("File exists!")  
else:  
 print("File does not exist.")  
```  
  
---  
  
### \*\*7. Working with Binary Files\*\*  
Binary files store data in binary format (e.g., images, videos).  
  
#### Example: Reading a Binary File  
```python  
with open("image.jpg", "rb") as file:  
 content = file.read()  
 print(content[:10]) # Print the first 10 bytes  
```  
  
#### Example: Writing to a Binary File  
```python  
with open("output.jpg", "wb") as file:  
 file.write(content) # Write binary data  
```  
  
---  
  
## \*\*Input/Output Operations\*\*  
  
### \*\*1. Input from the User\*\*  
The `input()` function is used to take input from the user.  
  
```python  
name = input("Enter your name: ")  
print(f"Hello, {name}!")  
```  
  
#### Example: Taking Integer Input  
```python  
age = int(input("Enter your age: "))  
print(f"You are {age} years old.")  
```  
  
---  
  
### \*\*2. Output to the Console\*\*  
The `print()` function is used to display output.  
  
#### Example: Basic Output  
```python  
print("Hello, World!")  
```  
  
#### Example: Using `sep` and `end` Parameters  
```python  
# Custom separator  
print("Python", "is", "fun", sep="-") # Output: Python-is-fun  
  
# Custom end character  
print("Hello", end=" ")  
print("World!") # Output: Hello World!  
```  
  
#### Example: Formatted Output  
```python  
name = "Alice"  
age = 25  
print(f"My name is {name} and I am {age} years old.") # Output: My name is Alice and I am 25 years old.  
```  
  
---  
  
### \*\*3. Redirecting Output to a File\*\*  
You can redirect the output of the `print()` function to a file.  
  
```python  
with open("output.txt", "w") as file:  
 print("This is written to the file.", file=file)  
```  
  
---  
  
### \*\*4. Reading Input from a File\*\*  
You can use `sys.stdin` to read input from a file instead of the console.  
  
```python  
import sys  
  
# Redirect input from a file  
sys.stdin = open("input.txt", "r")  
data = input() # Reads the first line from the file  
print(data)  
```  
  
---  
  
## \*\*Example: File Handling Program\*\*  
Here’s a complete example that combines reading, writing, and appending to a file:  
  
```python  
# Writing to a file  
with open("example.txt", "w") as file:  
 file.write("This is the first line.\n")  
  
# Appending to the file  
with open("example.txt", "a") as file:  
 file.write("This is an appended line.\n")  
  
# Reading the file  
with open("example.txt", "r") as file:  
 content = file.read()  
 print("File Content:")  
 print(content)  
```  
  
---  
  
## \*\*Summary\*\*  
- Use `open()` to work with files in different modes (`'r'`, `'w'`, `'a'`, etc.).  
- Use `with` to handle files safely and automatically close them.  
- Use `read()`, `readline()`, or `readlines()` to read files.  
- Use `write()` or `writelines()` to write to files.  
- Use `input()` for user input and `print()` for output.  
- Use the `os` module to check file existence or perform file operations.