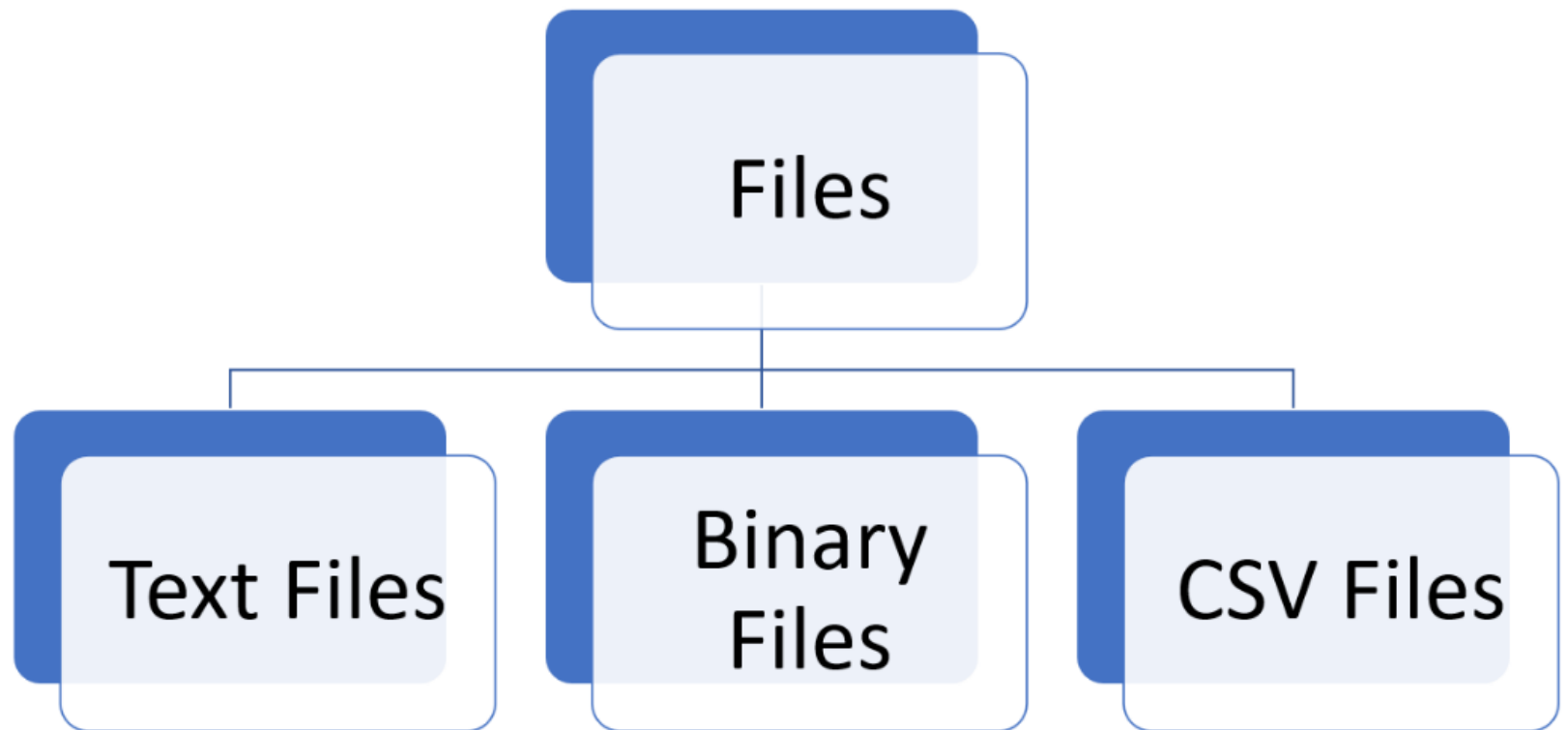


File Handling

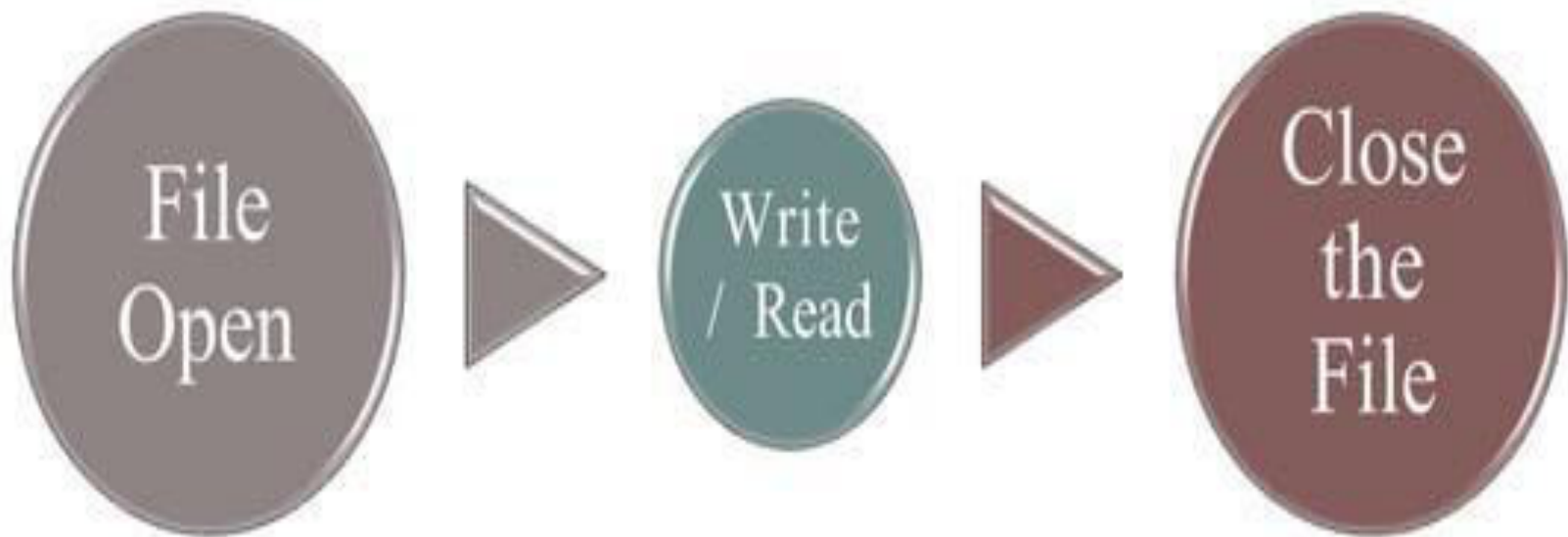
Files

- File handling allows a Python program to read data from or write data to disk files.
- It helps store data permanently for later use.
- Without file handling, data exists only while the program runs.



Operations in file

Three operations



OPENING AND CLOSING FILES

- The basic file manipulation tasks include adding, modifying or deleting data in a file, which in turn include any one or combination of the following operations:



Reading data from files



writing data to files



appending data to files

Open a file

- We have to open a file using built-in function `open()`
- This function returns a file object, also called a `handle`, it is used to read or modify the file
- We can specify the mode while opening a file.

r – Read

w – Write

a – Append

File Handling

- The key function for working with files in Python is the [open\(\)](#) function.
- The [open\(\)](#) function takes two parameters; *filename*, and *mode*.

OPENING FILES

- Syntax:

Identifier to access the file

Built-in Function

Access Mode

`<file_objectname> = open(<filename>, <mode>)`

Name of the file - identifier

The diagram illustrates the syntax for opening a file in Python. It shows the code `<file_objectname> = open(<filename>, <mode>)`. Red curly braces are used to group parts of the code and link them to descriptive text. A brace under `<file_objectname>` points to the text 'Identifier to access the file'. A brace under `open` points to 'Built-in Function'. A brace under `<mode>` points to 'Access Mode'. A larger brace under `<filename>` points to 'Name of the file - identifier'.

There are four different methods (modes) for opening a file:

"r" - Read - Default value. Opens a file for reading, error if the file does not exist

"a" - Append - Opens a file for appending, creates the file if it does not exist

"w" - Write - Opens a file for writing, creates the file if it does not exist

"x" - Create - Creates the specified file, returns an error if the file exists

Create a file, located in the same folder as Python

demofile.txt

**Hello! Welcome to demofile.txt
This file is for testing purposes.
Good Luck!**

```
f = open("demofile.txt")  
print(f.read())
```

Reading File

- File object includes the following methods to read data from the file.
- **read(chars):** reads the specified number of characters starting from the current position.
- **readline():** reads the characters starting from the current reading position up to a newline character.
- **readlines():** reads all lines until the end of file and returns a list object.

Using the with statement

You can also use the with statement when opening a file:

Example

```
with open("demofile.txt") as f:  
    print(f.read())
```

Closing a File

- File closing is done with close() method

Syntax :

```
fileobject.close()
```

Read Lines

You can return one line by using the `readline()` method:

Example

```
f = open("demofile.txt")  
print(f.readline())  
f.close()
```

Read in Loop

By looping through the lines of the file, you can read the whole file, line by line:

Example

#Loop through the file line by line:

```
with open("demofile.txt") as f:
```

```
    for x in f:
```

```
        print(x)
```

Read Only Parts of the File

By default the `read()` method returns the whole text, but you can also specify how many characters you want to return:

Example

```
#Return the 5 first characters of the file:  
with open("demofile.txt") as f:  
    print(f.read(5))
```

Python File Write

Write to an Existing File

To write to an existing file, you must add a parameter to the [open\(\)](#) function:

"a" - Append - will append to the end of the file

"w" - Write - will overwrite any existing content

Append

Example:

```
with open("demofile.txt", "a") as f:  
    f.write("Now the file has more content!")  
  
#open and read the file after the appending:  
with open("demofile.txt") as f:  
    print(f.read())
```

Overwrite Existing Content

- To overwrite the existing content to the file, use the w parameter:

Example

#Open the file "demofile.txt" and overwrite the content:

```
with open("demofile.txt", "w") as f:
```

```
    f.write("Woops! I have deleted the content!")
```

#open and read the file after the overwriting:

```
with open("demofile.txt") as f:
```

```
    print(f.read())
```

Create a New File

To create a new file in Python, use the [open\(\)](#) method, with one of the following parameters:

"x" - Create - will create a file, returns an error if the file exists

"a" - Append - will create a file if the specified file does not exist

"w" - Write - will create a file if the specified file does not exist

Example

Create a new file called "myfile.txt":

```
f = open("myfile.txt", "x")
```

Renaming a file

- The **os** module Python provides methods that help to perform file-processing operations, such as renaming and deleting.
- To rename an existing file, `rename()` method is used
- It takes two arguments, current file name and new file name

Syntax:

```
os.rename(current_file_name, new_file_name)
```

Example:

```
import os  
os.rename("test.txt", "Newtest.txt")  
print("File renamed")
```

Deleting a file

- We can delete a file by using `remove()` method
- Syntax:

```
os.remove(filename)
```

Example:

```
import os  
os.remove("Newtest.txt")  
print("File deleted")
```

1. Writing and Reading a Message

Write message

```
file = open("welcome.txt", "w")
```

```
file.write("Welcome to Python File Handling!\nKeep learning.")
```

```
file.close()
```

Read message

```
file = open("welcome.txt", "r")
```

```
print(file.read())
```

```
file.close()
```

2. Store and Display Student Names

```
students = ["Riya\n", "Amit\n", "John\n"]  
file = open("students.txt", "w")  
file.writelines(students)  
file.close()
```

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

3. Count Lines, Words, and Characters

```
file = open("story.txt", "r")
text = file.read()
lines = text.split("\n")
words = text.split()
print("Lines:", len(lines))
print("Words:", len(words))
print("Characters:", len(text))
file.close()
```

4. Append Data Without Overwriting

```
file = open("feedback.txt", "a")  
feedback = input("Enter your feedback: ")  
file.write(feedback + "\n")  
file.close()  
print("Feedback saved!")
```

5. Copy File Content

```
source = open("data.txt", "r")
destination = open("backup.txt", "w")
for line in source:
    destination.write(line)
source.close()
destination.close()
print("Backup completed.")
```

6. Using seek() and tell()

```
file = open("sample.txt", "r")
print("Position:", file.tell())
data = file.read(10)
print("Read data:", data)
print("After read:", file.tell())
file.seek(0)
print("Pointer reset to:", file.tell())
file.close()
```

7. Write and Read User Details

```
file = open("user_data.txt", "w")  
name = input("Enter your name: ")  
age = input("Enter your age: ")  
file.write(f"Name: {name}\n Age: {age}\n")  
file.close()
```

```
file = open("user_data.txt", "r")  
print(file.read())  
file.close()
```

8. Search for a Word in File

```
word = input("Enter word to search: ")
file = open("story.txt", "r")
found = False
for line in file:
    if word in line:
        print("Found in line:", line.strip())
        found = True
        break
if not found:
    print("Word not found.")
file.close()
```

9. Using 'with' Statement

```
with open("notes.txt", "w") as file:
```

```
    file.write("Always close your files properly!\n")
```

```
with open("notes.txt", "r") as file:
```

```
    print(file.read())
```

10. Display File in Reverse

```
file = open("data.txt", "r")  
lines = file.readlines()  
for line in reversed(lines):  
    print(line.strip())  
file.close()
```

- Write a Python program to reverse a string. Import the module to reverse a string input by the user
- Write a program that asks the user to enter a list of integers. Do the following:
 - (a) Print the total number of items in the list.
 - (b) Print the last item in the list.
 - (c) Print the list in reverse order.
 - (d) Print Yes if the list contains a 5 and No otherwise.
 - (e) Print the number of fives in the list.
 - (f) Remove the first and last items from the list, sort the remaining items, and print the result.