Python Notes – Files and Directories, File I/O, File positioning, File operators.

Prepared by: **Dhananjay Kumar**
□□ Trainer | Java Full-Stack & Python | Contact: 7023577968
LinkedIn: Dhananjay Kumar

Jupyter Notebook Notes: Files & Directories, File I/O, File Positioning, File Operators

1. Files & Directories – Introduction

A **file** is a container used to store data in a computer.

A **directory** (or folder) is used to organize multiple files.

Python provides a built-in module called os for interacting with files and directories, and the shutil module for advanced operations.

1.1 Working with Directories

In [1]:

```
# Example: Checking Current Working Directory

import os

os.getcwd()
```

Out[1]:

```
'C:\\Users\\Ankita Gupta\\Desktop\\Dhananjay\\Chitkara University\\Python\\Students\\ATN\\PythonCodeNotes'
```

In [6]:

```
# Example: Listing Files in a Directory

import os

os.listdir()
```

Out[6]:

```
['.ipynb_checkpoints',
 'AdvanceDataType.ipynb',
 'BasicPythonPrograms.ipynb',
 'control_satement_loop.ipynb',
 'DecisionMakingLoops_Notes.ipynb',
 'File IO',
 'file.ipynb',
 'Files_and_Directories.ipynb',
 'function.ipynb',
 'Introduction.ipynb',
 'lambda.ipynb',
 'loops_control_statements.ipynb',
 'Module',
 'module.ipynb',
 'Operators.ipynb',
 'placementquestion.ipynb',
 'Python_Files_and_Directories_Notes.ipynb',
 'Python_Notes_Operators_Dhananjay_full.ipynb',
 'String.ipynb',
 'student.txt',
 'testpad.ipynb',
 'Two_Dimensional_Lists.ipynb',
 'Types_of_Arguments.ipynb',
 'Untitled.ipynb',
 'Untitled1.ipynb',
 'Untitled2.ipynb']
```

In [3]:

```
# Example: Creating a New Directory

os.mkdir("my_folder")
```

In [5]:

```
# Example: Removing a Directory

os.rmdir("my_folder")
```

2. File I/O (Input/Output)

Python provides the open() function for file operations

Syntax:

open("filename", "mode")

Common File Modes

Mode Meaning

"r" Read (file must exist)

"w" Write (creates new file / overwrites)

"a" Append (adds data at the end)

"r+" Read + Write

"b" Binary mode (images, videos, etc.)

In [8]:

```
# 2.1 Writing to a File ("w" mode)

file = open("students.txt", "w")
file.write("Rahul\n")
file.write("Sita\n")
file.write("Aman\n")
file.close()


"""
Creates a file
Writes 3 names
Overwrites if file already exists

"""
```

Out[8]:

```
'\nCreates a file\nWrites 3 names\nOverwrites if file already exists\n\n'
```

In [ ]:

```
# 2.2 Reading a File ("r" mode)

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

In [ ]:

```
# 2.3 Appending to a File ("a" mode)

file = open("students.txt", "a")
file.write("DHananjay\n")
file.close()

# Adds new name at the end without deleting previous data.
```

3. File Positioning (seek & tell)

Python allows moving the cursor inside a file using:

seek() → move cursor to a position

tell() → returns current cursor position

In [ ]:

```
# Example: Using tell()

file = open("students.txt", "r")
print("Cursor at:", file.tell())
file.close()
```

In [ ]:

```
# Example: Using seek() to move cursor

file = open("students.txt", "r")
file.seek(0)        # move to the beginning
print(file.read(5)) # reads first 5 characters
file.close()
```

Reading Line by Line

In [ ]:

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

4. File Operators (os module)

In [ ]:
```python
# Checking if file exists

import os

os.path.exists("students.txt")
```

In [ ]:
```python
# Renaming a File

os.rename("students.txt", "student_data.txt")
```

In [ ]:
```python
# Deleting a File

os.remove("student_data.txt")
```

5. Real-World Practice Exercise

Q1: Create a file data.txt and write 3 lines.

In [ ]:
```python
file = open("data.txt", "w")
file.write("First line\n")
file.write("Second line\n")
file.write("Third line\n")
file.close()
```

Q2: Read the file and print all data in UPPERCASE

In [ ]:
```python
file = open("data.txt", "r")
print(file.read().upper())
file.close()
```

Q3: Append 2 more lines

In [ ]:
```python
file = open("data.txt", "a")
file.write("Fourth line\n")
file.write("Fifth line\n")
file.close()
```

Q4: Count total characters in file

In [ ]:
```python
file = open("data.txt", "r")
content = file.read()
print("Total characters:", len(content))
file.close()
```

Q5: Move cursor to the beginning and print first line

In [ ]:
```python
file = open("data.txt", "r")
file.seek(0)
print("First line:", file.readline())
file.close()
```

PRACTICE QUESTIONS WITH ANSWERS

import os

print("\n==============================")

print(" PRACTICE QUESTIONS & ANSWERS ")

print("==============================\n")

Q1: Create a file and write 5 lines, then read them.

print("Q1: Create a file 'q1.txt', write 5 lines, then read them.\n")

with open("q1.txt", "w") as f:

```
    f.write("Line 1: Python\n")
    f.write("Line 2: Java\n")
    f.write("Line 3: C++\n")
    f.write("Line 4: SQL\n")
    f.write("Line 5: HTML\n")
with open("q1.txt", "r") as f:
    data = f.read()
print("Output:\n", data)
print("Explanation: 'w' mode creates a file and overwrites existing data. 'read()' returns the entire content.\n")
```

Q2: Count total words in the file q1.txt

```
print("Q2: Count total words in q1.txt\n")
words = data.split()
print("Total words =", len(words))
print("Explanation: split() breaks text into words using spaces.\n")
```

Q3: Append 2 more lines to q1.txt and show full content

```
print("Q3: Append 2 more lines to q1.txt\n")
with open("q1.txt", "a") as f:
    f.write("Line 6: JavaScript\n")
    f.write("Line 7: GoLang\n")
with open("q1.txt", "r") as f:
    print("Updated File Content:\n", f.read())
print("Explanation: 'a' mode adds content at the end without deleting previous data.\n")
```

Q4: Read only the first 10 characters using file positioning

```
print("Q4: Read only the first 10 characters\n")
with open("q1.txt", "r") as f:
    f.seek(0)
    print("First 10 characters:", f.read(10))
print("Explanation: f.seek(0) moves cursor to the beginning, f.read(10) reads 10 characters.\n")
```

Q5: Show cursor position using tell()

```
print("Q5: Show cursor position after reading 10 chars\n")
with open("q1.txt", "r") as f:
    f.seek(0)
    f.read(10)
    print("Cursor position =", f.tell())
print("Explanation: tell() gives the current cursor index inside the file.\n")
```

Q6: Read file line-by-line with line numbers

```
print("Q6: Read file line-by-line with line numbers\n")

with open("q1.txt", "r") as f:

    for i, line in enumerate(f, start=1):

        print(f"Line {i} -> {line.strip()}")

print("Explanation: enumerate() adds line numbers, strip() removes newline.\n")
```

Q7: Check if 'q1.txt' exists using os.path.exists()

```
print("Q7: Check if file exists\n")

print("Does q1.txt exist?", os.path.exists("q1.txt"))

print("Explanation: os.path.exists() returns True if file is present.\n")
```

Q8: Rename file q1.txt → languages.txt

```
print("Q8: Rename q1.txt to languages.txt\n")

os.rename("q1.txt", "languages.txt")

print("Renamed successfully!")

print("Explanation: os.rename(old_name, new_name) renames the file.\n")
```

Q9: Count total characters in languages.txt

```
print("Q9: Count total characters in languages.txt\n")

with open("languages.txt", "r") as f:

    content = f.read()

print("Total characters =", len(content))

print("Explanation: len() on string gives number of characters.\n")
```

Q10: Create a directory and move file into it

```
print("Q10: Create directory 'my_data' and move languages.txt inside it\n")

if not os.path.exists("my_data"):

    os.mkdir("my_data")

import shutil

shutil.move("languages.txt", "my_data/languages.txt")

print("File moved successfully!")

print("Explanation: shutil.move() moves files between folders.\n")
```

Q11: Read file from the new folder

```
print("Q11: Read file from 'my_data' folder\n")
```

```
with open("my_data/languages.txt", "r") as f:

    print(f.read())

print("Explanation: We specify path 'my_data/languages.txt' to read file.\n")
```

Q12: Delete the file (optional)

```
print("Q12: Delete the file languages.txt (Inside my_data)\n")

os.remove("my_data/languages.txt")

print("File deleted successfully!")

print("Explanation: os.remove(filename) deletes the file permanently.\n")
```

SCENARIO-BASED PRACTICE (One Jupyter cell)

```
import os

import shutil

print("="*60 + "\n")

print("SCENARIO-BASED PRACTICE SET — Files & Directories\n")

print("="*60 + "\n")
```

SCENARIO 1: Attendance Log

```
print("SCENARIO 1: Attendance Log")

print("Problem: Your class app writes one student attendance entry per line in 'attendance.log'.")

print("Task: Add today's entry, then display the last 3 entries (most recent last).")
```

Setup / simulate existing log

```
os.makedirs("scenario_files", exist_ok=True)

attendance_path = "scenario_files/attendance.log" with open(attendance_path, "w") as f:\

    f.write("2025-02-10, Rahul, Present\n")

    f.write("2025-02-11, Sita, Absent\n")

    f.write("2025-02-12, Aman, Present\n")

    f.write("2025-02-13, Anita, Present\n")
```

Append today's entry (simulate)

```
with open(attendance_path, "a") as f:

    f.write("2025-02-14, Dhananjay, Present\n")
```

Read and show last 3 entries

```
with open(attendance_path, "r") as f:

    lines = f.readlines()

last_three = lines[-3:]

print("Answer (last 3 entries):")
```

for line in last_three:

```
    print(line.strip())
```

print("\nExplanation: 'a' appends new entries. Reading with readlines() and slicing [-3:] gives last 3 lines.\n")

SCENARIO 2: Inventory Update

print("SCENARIO 2: Inventory Update")

print("Problem: 'inventory.txt' stores item and quantity per line: 'TV, 10'. A sale reduces quantity.")

print("Task: Reduce stock for item 'LED TV' by 2 and save changes safely (create backup).")

inventory_path = "scenario_files/inventory.txt"

backup_path = "scenario_files/inventory.bak"

Setup initial inventory

with open(inventory_path, "w") as f:

```
    f.write("LED TV, 5\n")

    f.write("Refrigerator, 3\n")

    f.write("Washing Machine, 2\n")
```

Make a backup first

shutil.copy(inventory_path, backup_path)

Update quantity (read-modify-write)

with open(inventory_path, "r") as f:

```
    items = [line.strip() for line in f if line.strip()]
```

updated_lines = []

for line in items:

```
    name, qty = [part.strip() for part in line.split(",")]
    qty = int(qty)

    if name == "LED TV":

        qty -= 2  # sold 2 units

        if qty < 0:

            qty = 0

    updated_lines.append(f"{name}, {qty}\n")
```

with open(inventory_path, "w") as f:

```
    f.writelines(updated_lines)
```

print("Answer (updated inventory):")

with open(inventory_path, "r") as f:

```
    for line in f:

        print(line.strip())
```

print("\nExplanation: Create a backup via shutil.copy(), then read all lines, modify in memory, and overwrite using 'w' to ensure atomic-looking update.\n")

SCENARIO 3: Large Log — Read in Chunks

print("SCENARIO 3: Large Log — Read in Chunks")

print("Problem: A server log can be huge. You must read it in 64-byte chunks to process streaming data.")

print("Task: Create a simulated large file and print the first 3 chunks.")

large_path = "scenario_files/server.log"

Create a simulated large file (repeating pattern)

with open(large_path, "w") as f:

```
for i in range(100):

    f.write(f"2025-02-14 10:{i:02d}:00 INFO Request from 192.168.1.{i%255}\n")
```

print("Answer (first 3 chunks of 64 bytes):")

with open(large_path, "r") as f:

```
for _ in range(3):

    chunk = f.read(64)

    print(repr(chunk))   # repr to show exact chunk including newlines
```

print("\nExplanation: f.read(n) reads up to n characters; useful for streaming/bounded-memory processing.\n")

SCENARIO 4: CSV Student Marks — Update One Student

print("SCENARIO 4: CSV Student Marks — Update One Student")

print("Problem: 'marks.csv' contains 'roll,name,marks'. Update marks for roll 102 by +10 (but max 100).")

print("Task: Modify the CSV and show updated entry.")

csv_path = "scenario_files/marks.csv"

with open(csv_path, "w") as f:

```
f.write("101,Asha,78\n")

f.write("102,Bimal,88\n")

f.write("103,Chitra,91\n")
```

Read -> update -> write back

with open(csv_path, "r") as f:

```
rows = [line.strip() for line in f if line.strip()]
```

out_rows = []

for row in rows:

```
roll, name, marks = row.split(",")

marks = int(marks)

if roll == "102":
    marks = min(100, marks + 10)

out_rows.append(f"{roll},{name},{marks}\n")
```

with open(csv_path, "w") as f:

```
    f.writelines(out_rows)
```

print("Answer (updated record for roll 102):")

with open(csv_path, "r") as f:

```
    for line in f:

        if line.startswith("102,"):

            print(line.strip())
```

print("\nExplanation: CSV lines treated as plain text (for simple cases). Read all rows, change the target row, then overwrite file.\n")

\

SCENARIO 5: Binary File Copy (Image)

print("SCENARIO 5: Binary File Copy (Image)")

print("Problem: Copy 'logo.png' to backup 'logo_copy.png' using binary mode.")

print("Task: Simulate an image by writing binary bytes, then copy and verify sizes match.")

orig_bin = "scenario_files/logo.png"

copy_bin = "scenario_files/logo_copy.png"

Simulate binary image (small)

with open(orig_bin, "wb") as f:

```
    f.write(b"\x89PNG\r\n\x1a\n" + b"FAKEPNGDATA" * 10)
```

Copy in binary mode

with open(orig_bin, "rb") as src, open(copy_bin, "wb") as dst:

```
    while True:

        chunk = src.read(1024)

        if not chunk:

            break

        dst.write(chunk)
```

print("Answer: sizes ->", os.path.getsize(orig_bin), "and", os.path.getsize(copy_bin))

print("\nExplanation: For binary files, use 'rb' and 'wb'. Copy in chunks to handle large files efficiently.\n")

SCENARIO 6: File Positioning — Overwrite a Line in-place (fixed-length)

print("SCENARIO 6: File Positioning — Overwrite a Line In-Place (Fixed-Length Record)")

print("Problem: 'fixed.txt' has fixed-length records (20 chars). Update record 2 without rewriting entire file.")

print("Task: Demonstrate seek() to overwrite the second 20-byte record.")

fixed_path = "scenario_files/fixed.txt"

Create 3 fixed-length records (pad/truncate to 20 bytes)

records = ["Alice", "Bob", "Charlie"]

with open(fixed_path, "w") as f:

```
    for name in records:

        rec = name.ljust(20)[:20]

        f.write(rec)
```

Overwrite record 2 (index 1) to 'Bobby' (padded)

with open(fixed_path, "r+") as f:

```
    record_size = 20

    f.seek(record_size * 1)   # move to second record start

    f.write("Bobby".ljust(record_size)[:record_size])
```

print("Answer (all fixed records after overwrite):")

with open(fixed_path, "r") as f:

```
    for i in range(3):

        print(repr(f.read(20)))
```

print("\nExplanation: r+ opens for read/write. Using seek(offset) lets you overwrite a fixed-size record in place.\n")

SCENARIO 7: Temporary Session File Cleanup

print("SCENARIO 7: Temporary Session File Cleanup")

print("Problem: A web app writes temp files named 'session_.tmp'. Remove any .tmp older than a threshold (simulate).")

print("Task: Create temp files and delete all '.tmp' files in directory.")

Create some temp files

for i in range(3):

```
    with open(f"scenario_files/session_{i}.tmp", "w") as f:

        f.write(f"session data {i}\n")
```

List and remove .tmp files

deleted = []

for fname in os.listdir("scenario_files"):

```
    if fname.endswith(".tmp"):

        os.remove(os.path.join("scenario_files", fname))

        deleted.append(fname)
```

print("Answer (deleted files):")

for d in deleted:

```
    print(d)
```

print("\nExplanation: Use os.listdir() and os.remove() to clean temp files. In real apps, check file age using os.path.getmtime().\n")

SCENARIO 8: Safe File Replace (Atomic-ish)

print("SCENARIO 8: Safe File Replace (Write to temp then rename)")

print("Problem: To avoid partial writes, write to a temporary file then atomically rename to target.")

print("Task: Update 'config.txt' safely.")

config = "scenario_files/config.txt"

tmp = "scenario_files/config.tmp"

with open(config, "w") as f:

```
    f.write("version=1\nmode=prod\n")
```

Safe update: write to tmp then replace

with open(tmp, "w") as f:

```
    f.write("version=2\nmode=prod\n")
```

os.replace is atomic on many OSes

os.replace(tmp, config)

print("Answer (config content after safe replace):")

with open(config, "r") as f:

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

print("\nExplanation: Writing to a temp file and using os.replace(tmp, final) reduces window where target file is missing/corrupt.\n")

SCENARIO 9: Search and Report

print("SCENARIO 9: Search and Report")

print("Problem: Search for files containing word 'TODO' in 'scenario_files' and print filenames.")

print("Task: Create files, put 'TODO' in some, then search.")

Create a few txt files

with open("scenario_files/task1.txt", "w") as f:

```
    f.write("Implement feature X\nTODO: add tests\n")
```

with open("scenario_files/task2.txt", "w") as f:

```
    f.write("Documentation only\n")
```

Search

matches = []

for fname in os.listdir("scenario_files"):

```
    if fname.endswith(".txt"):
        with open(os.path.join("scenario_files", fname), "r") as f:

            if "TODO" in f.read():

                matches.append(fname)
```

print("Answer (files with TODO):")

for m in matches:

```
    print(m)
```

print("\nExplanation: Simple content search; for large corpora consider streaming reads and more efficient search/indexing.\n")

SCENARIO 10: Cleanup created folder (optional)

print("SCENARIO 10: Cleanup (optional)")

print("Problem: After exercises, you may want to remove the 'scenario_files' folder and all its contents.")

print("Task: Uncomment the next two lines to delete the folder when you are ready.\n")

print("# To remove the folder and all files, uncomment the following lines:")

print("# import shutil")

print("# shutil.rmtree('scenario_files')\n")

print("END OF SCENARIO SET\n")

print("="*60 + "\n")

================================

30 MCQs

================================

1. Which mode is used to read a file without allowing writing?

A. w
B. r
C. a
D. r+

Answer: B
Explanation: Mode 'r' opens the file only for reading and does not allow writing.

2. What happens when you open a file in 'w' mode?

A. Reads file
B. Appends data
C. Deletes old data
D. Nothing

Answer: C
Explanation: 'w' mode overwrites old data and creates a new empty file.

3. Which function reads the entire file content?

A. readfile()
B. readline()
C. read()
D. scan()

Answer: C
Explanation: read() returns the complete file content as a single string.

4. Which module is used for directory operations?

A. system
B. os
C. shutil
D. path

Answer: B
Explanation: The os module provides functions like mkdir, listdir, etc.

5. What does seek(0) do?

A. Close file
B. Delete file
C. Move cursor to start
D. Move cursor to end

Answer: C
Explanation: seek(0) resets the file pointer to the beginning of the file.

6. Which method shows the current cursor position?

A. where()
B. pos()
C. tell()
D. cursor()

Answer: C
Explanation: tell() returns the current file pointer position.

7. What does 'a' mode do?

A. Overwrites
B. Appends
C. Reads only
D. Renames file

Answer: B
Explanation: Append mode adds content at the end of the file.

8. Which function lists all files in a directory?

A. os.listdir()
B. os.show()
C. os.files()
D. dir.all()

Answer: A
Explanation: os.listdir() returns all files and folders inside a directory.

9. Which method reads one line at a time?

A. read()
B. readline()
C. readlines()
D. scanline()

Answer: B
Explanation: readline() returns the next line from the file.

10. readlines() returns data in which format?

A. string
B. integer
C. list of lines
D. dictionary

Answer: C
Explanation: readlines() returns a list containing all lines.

11. Which symbol represents binary mode?

A. b
B. bin
C. binary
D. rb only

Answer: A
Explanation: Adding 'b' means binary mode, e.g., rb, wb.

12. What happens if you open a non-existing file using 'r' mode?

A. File created
B. Error occurs
C. File deleted
D. Nothing

Answer: B
Explanation: 'r' mode requires file to exist; otherwise FileNotFoundError occurs.

13. Which command deletes a file?

A. os.remove()
B. os.delete()
C. os.erase()
D. os.kill()

Answer: A
Explanation: os.remove() permanently deletes a file.

14. Which is correct syntax to open a file?

A. open.file()
B. file.open()
C. open('a.txt','r')
D. open('r','a.txt')

Answer: C
Explanation: Correct syntax is open(filename, mode).

15. Writing a list of lines to a file uses:

A. write()
B. writelines()
C. writeall()
D. writefile()

Answer: B
Explanation: writelines() writes multiple lines to a file.

16. Which mode is used for both reading and writing?

A. r
B. w
C. a
D. r+

Answer: D
Explanation: r+ allows both reading and writing without clearing the file.

17. After opening a file, which method should be called at the end?

A. end()
B. finish()
C. close()
D. exit()

Answer: C
Explanation: close() is used to close the opened file.

18. Which module is best for copying files?

A. os
B. shutil
C. sys
D. file

Answer: B
Explanation: shutil.copy() is used for copying files.

19. Which Python statement is best for safe file handling?

A. open-close manually
B. with open() as f:
C. try open
D. auto()

Answer: B
Explanation: 'with' automatically closes the file even if an error occurs.

20. What does 'rb' mean?

A. read-back
B. read-binary
C. return binary
D. real byte

Answer: B
Explanation: 'rb' opens the file in read-binary mode.

21. Which command moves a file?

A. os.move()
B. shutil.move()
C. os.rename() only
D. file.move()

Answer: B
Explanation: shutil.move() moves a file from one folder to another.

22. readline() returns:

A. list
B. one line with \n
C. one character
D. dictionary

Answer: B
Explanation: readline() includes the newline character.

23. Which mode prevents overwriting existing data?

A. w
B. a
C. r
D. wb

Answer: B
Explanation: Append mode does not overwrite existing content.

24. os.path.exists('file.txt') returns:

A. File content
B. True/False
C. integer
D. list

Answer: B
Explanation: It returns True if file exists, else False.

25. File pointer means:

A. File location
B. Cursor position
C. File size
D. Memory address

Answer: B
Explanation: File pointer = current reading/writing position.

26. Which function gets file size?

A. os.length()
B. os.getsize()
C. os.path.getsize()
D. file.size()

Answer: C
Explanation: os.path.getsize() returns file size in bytes.

27. Default file opening mode?

A. w
B. r
C. a
D. r+

Answer: B
Explanation: open() defaults to read mode ('r').

28. Which function creates a directory?

A. os.makedir()
B. os.mkdir()
C. make.dir()
D. create.dir()

Answer: B
Explanation: os.mkdir() creates a new directory.

29. To read entire file as list of lines use:

A. read()
B. readline()
C. readlines()
D. readall()

Answer: C
Explanation: readlines() returns a list containing all lines.

30. Which mode clears (truncates) a file?

A. w
B. r
C. rb
D. a

Answer: A
Explanation: Opening a file in 'w' mode clears all previous data.

""")

In [ ]: