✸ SLIP 1 — (I/O Decorator + Wi-Fi Scan)

(File: SADP AND IOT Slips.pdf)

☑ PROGRAM 1: I/O Decorator (Uppercase → Lowercase)
1. What is the Decorator Pattern?

→ A pattern that allows adding new behavior to an object at runtime without changing its original code.

2. Why extend FilterInputStream?

→ Because it already wraps another InputStream, making it easy to modify the data being read.

3. Why override read()?

→ To intercept each byte and convert uppercase letters into lowercase while reading.

4. Does this decorator modify the actual file?

→ No, it only transforms data during reading; the original file stays unchanged.

5. What happens to numbers or symbols?

→ They remain as they are, because lowercase conversion applies only to alphabetic characters.

☑ PROGRAM 2: Wi-Fi Network Scanning (ESP8266)
1. What does WiFi.scanNetworks() do?

→ It scans nearby Wi-Fi networks and returns the count of detected networks.

2. What is an SSID?

→ It is the name of a Wi-Fi network that is broadcast by the router.

3. What is RSSI?

→ Received Signal Strength Indicator, which shows how strong the Wi-Fi signal is.

4. Why use WiFi.mode(WIFI_STA)?

→ To set ESP8266 in Station mode so it can scan networks like a normal Wi-Fi device.

5. Does scanning require a Wi-Fi password?

→ No, scanning only detects networks; it does not connect.

✸ SLIP 2 — (Singleton + Ultrasonic Sensor)
☑ PROGRAM 1: Singleton Pattern
1. What is Singleton Pattern?

→ A pattern that ensures a class has only one instance throughout the program.

2. Why is the constructor private?

→ To stop other classes from creating new objects directly.

3. What is double-checked locking?

→ A technique to create a Singleton safely in multithreading while minimizing synchronization overhead.

4. Why is the instance variable static?

→ So that it belongs to the class and not to any object, making it reusable globally.

5. Where is Singleton used?

→ In logging, configuration managers, or database connection objects.

☑ PROGRAM 2: Ultrasonic Distance Measurement
1. What is the function of the TRIG pin?

→ It sends an ultrasonic pulse when triggered.

2. What does the ECHO pin do?

→ It receives the reflected pulse and measures the return time.

3. What formula is used to calculate distance?

→ Distance = (Time × Speed of Sound) ÷ 2, with speed ≈ 0.034 cm/µs.

4. Why divide by 2?

→ Because the sound wave travels to the object and back, so the time measured is double the actual distance.

5. What kind of output does the sensor give?

→ A pulse duration proportional to the distance of the object.

✸ SLIP 3 — (Observable Weather + Vibration Sensor)
☑ PROGRAM 1: Weather Station (java.util.Observable)
1. What is the Observer Pattern?

→ A pattern where multiple observers automatically update when the subject's data changes.

2. What does setChanged() do?

→ It marks that the subject's state has been updated so observers can be notified.

3. What is the role of notifyObservers()?

→ It calls update() on all registered observers.

4. Which values does WeatherData store?

→ Temperature, humidity, and pressure.

5. Why use java.util.Observable?

→ It provides a ready-made framework for implementing the observer pattern in Java.

☑ PROGRAM 2: Vibration Sensor (SW-420)
1. What is the SW-420 sensor?

→ A vibration/shock sensor that detects sudden movements or vibrations.

2. What kind of output does it give?

→ A digital HIGH signal when vibration is detected.

3. Why is an LED used with the vibration sensor?

→ To visually indicate when a vibration event is detected.

4. Is SW-420 analog or digital?

→ It provides a digital output using an onboard comparator.

5. What is a common use of vibration sensors?

→ Security alarms, door knocks, machinery vibration detection.

✸ SLIP 4 — (Factory Pizza + Touch Sensor)
☑ PROGRAM 1: Factory Method (Pizza Store)
1. What is the Factory Method Pattern?

→ A pattern where subclasses decide which object to create through a factory method.

2. What is the use of createPizza()?

→ It returns the specific pizza object (NY, Chicago, etc.).

3. Why is Chicago pizza's cut() different?

→ Chicago-style pizza is traditionally cut into square slices.

4. What steps are done in orderPizza()?

→ prepare → bake → cut → box.

5. Why is PizzaStore abstract?

→ To allow different pizza stores to provide their own pizza types.

☑ PROGRAM 2: Touch Sensor (TTP223)
1. What type of sensor is TTP223?

→ A capacitive touch sensor detecting finger presence.

2. What is its output on touch?

→ Digital HIGH signal.

3. Does it require pressure?

→ No, it detects touch through capacitance changes.

4. Why do we use an LED here?

→ To show a visual indication of touch detection.

5. Can it sense through materials?

→ Yes, it can work through thin plastic or paper.

✸ SLIP 5 — (Adapter Enumeration→Iterator + WiFi Connect)
☑ PROGRAM 1: Adapter (Enumeration → Iterator)
1. Why do we need to adapt Enumeration to Iterator?

→ Because old Java APIs use Enumeration, while modern code expects an Iterator interface.

2. What is the main difference between Enumeration and Iterator?

→ Iterator allows element removal; Enumeration does not.

3. Why does the adapter implement Iterator?

→ To provide the required Iterator interface while internally using Enumeration.

4. What is delegation in this context?

→ The adapter calls methods of the original Enumeration to perform operations.

5. Give a real-world example of an adapter.

→ A phone charger that adapts AC voltage to USB 5V.

☑ PROGRAM 2: Wi-Fi Connection (ESP8266)
1. What does WiFi.begin() do?

→ Starts the WiFi connection process using SSID and password.

2. What is WiFi.status()?

→ Returns the current connection status (e.g., WL_CONNECTED).

3. What does WL_CONNECTED mean?

→ The ESP8266 successfully connected to the WiFi network.

4. What is WiFi.localIP()?

→ It returns the IP address assigned to the ESP8266.

5. Why use Serial Monitor?

→ To view connection progress and debug WiFi status.

✸ SLIP 6 — (Command Pattern + Temperature Notification)

(source: SADP AND IOT Slips.pdf)

☑ PROGRAM 1: Command Pattern (Remote Control)
1. What is the Command Pattern?

→ A pattern where a request is wrapped as an object, allowing easy execution and undoing.

2. What is the role of execute() method?

→ It performs the actual action on the receiver (e.g., turning on a light).

3. What is an Invoker?

→ The object that triggers the execute() method on the command.

4. Who is the Receiver in this pattern?

→ The device that performs the actual operation (e.g., Light, GarageDoor).

5. Why does this pattern reduce coupling?

→ Because the invoker doesn't know how the action is performed, only which command to run.

☑ PROGRAM 2: Temperature Notification (LM35 Sensor)
1. What is LM35?

→ A temperature sensor that outputs a voltage proportional to temperature (10 mV per °C).

2. Why use analogRead() with LM35?

→ Because LM35 gives variable analog voltage based on temperature.

3. When does the LED turn ON in this program?

→ When the temperature crosses the threshold value set in code.

4. Why convert ADC values to °C?

→ Because analogRead() gives raw values; they must be converted to actual temperature.

5. What is the purpose of Serial Monitor here?

→ To display live temperature values for testing.

✸ SLIP 7 — (Undo Command + LDR Control)
☑ PROGRAM 1: Undo Command Pattern (Ceiling Fan)
1. What does undo() do?

→ Reverses the last operation (e.g., returns fan to previous speed).

2. Why store previous fan speed?

→ So that undo() can restore the exact earlier state.

3. What states can a ceiling fan have?

→ OFF, LOW, MEDIUM, and HIGH.

4. What is the Invoker in this scenario?

→ The remote control that triggers commands.

5. What is the Receiver?

→ The ceiling fan object that performs speed changes.

☑ PROGRAM 2: LDR-Based LED Brightness Control
1. What is an LDR?

→ A Light Dependent Resistor whose resistance changes with light intensity.

2. Why use analogRead() for LDR?

→ Because LDR produces varying analog values depending on light.

3. What does map() function do?

→ Converts large LDR values (0–1023) into LED brightness values (0–255).

4. What happens when light increases?

→ LDR resistance decreases → LED brightness is adjusted.

5. Why use PWM pin for LED?

→ To control brightness smoothly using analogWrite().

✸ SLIP 8 — (State Pattern + Linux Commands)
☑ PROGRAM 1: State Pattern (Gumball Machine)
1. What is the State Pattern?

→ A pattern where an object changes its behavior based on its internal state.

2. What are the key states in a Gumball Machine?

→ NoCoinState, HasCoinState, SoldState, SoldOutState.

3. What is the Context class?

→ The GumballMachine class which switches between states.

4. Why use interfaces for states?

→ To ensure all states implement the same behavior methods.

5. What triggers a state transition?

→ User actions like insertCoin(), ejectCoin(), and turnCrank().

☑ PROGRAM 2: Basic Linux Commands (Raspberry Pi)
1. What does ls do?

→ Lists files and directories.

2. What does cd command do?

→ Changes the current working directory.

3. What is the purpose of mkdir?

→ Creates a new directory.

4. What does sudo allow?

→ Running commands with superuser privileges.

5. What does ping do?

→ Tests network connectivity to another device/server.

✸ SLIP 9 — (Spring HR App + Python Basic Programs)
☑ PROGRAM 1: Spring HR Application
1. What is a Spring Bean?

→ A Java object that is created and managed by Spring container.

2. What is Dependency Injection?

→ Providing required objects to a class instead of creating them inside.

3. Why use applicationContext.xml?

→ To configure beans and class dependencies externally.

4. What is IOC Container?

→ The Spring system that manages bean lifecycle.

5. Why is Spring used?

→ To reduce coupling and simplify large applications.

☑ PROGRAM 2: Python Basic Programs

1. What does the input() function do?

→ Reads user input as a string.

2. How do you convert input to numbers?

→ Using int() or float().

3. How to print output in Python?

→ Using the print() function.

4. What does len() return?

→ Number of characters or elements in a string/list.

5. How to split a string into words?

→ Using string.split().

✸ SLIP 10 — (Strategy Pattern + Python Programs)

☑ PROGRAM 1: Strategy Pattern (Duck Behaviors)

1. What is the Strategy Pattern?

→ Pattern where different behaviors are interchangeable at runtime.

2. What are behavior interfaces?

→ FlyBehavior and QuackBehavior interfaces defining actions.

3. Why use composition in Strategy Pattern?

→ To change behaviors dynamically without modifying the class.

4. What is performFly()?

→ It calls the fly behavior currently assigned to the duck.

5. Give a real example of Strategy Pattern.

→ Payment method selection (UPI/Card/Net Banking).

☑ PROGRAM 2: Python Programs (Loops, Time, Exceptions)

1. What loop types does Python support?

→ for loop and while loop.

2. What is ZeroDivisionError?

→ Error raised when dividing by zero.

3. What does time.sleep() do?

→ Pauses program execution for given seconds.

4. How to read a file line-by-line?

→ Using a for loop on file object or file.readline().

5. How can exception handling be done in Python?

→ Using try–except blocks.

✸ SLIP 11 — (Heart Model Adapter + Python LED Control)
☑ PROGRAM 1: Heart Model → Beat Model (Adapter Pattern)
1. Why do we need an Adapter for HeartModel?

→ Because HeartModel and BeatModel use different methods, so Adapter converts one interface into the other.

2. What type of Adapter is used here?

→ Class Adapter (using inheritance + implementing BeatModel interface).

3. What does getBeat() return?

→ It returns heart rate converted to beat format using HeartModel.

4. Why is Adapter Pattern useful?

→ It allows two incompatible classes to work together without modifying existing code.

5. Give a real-life example of Adapter.

→ A mobile charger converting AC voltage to USB voltage.

☑ PROGRAM 2: Python — Light LED Using GPIO Pins
1. What is GPIO?

→ General Purpose Input/Output pins on Raspberry Pi used to control LEDs, sensors, etc.

2. Why set GPIO mode?

→ To select pin numbering system (BOARD or BCM).

3. Why do we use GPIO.setup()?

→ To configure the pin as input or output.

4. How do you turn ON an LED?

→ By writing HIGH signal to the output pin.

5. Why is cleanup() needed?

→ It resets all pins to safe mode after program ends.

✸ SLIP 12 — (Car Decorator + WiFi Scan ESP8266)
☑ PROGRAM 1: Decorator Pattern for Car (Sports/Luxury)
1. What is Car Decorator used for?

→ To add new features like SportsCar or LuxuryCar to an existing Car object.

2. Why use composition in decorators?

→ To wrap and extend the behavior of the original Car object.

3. What does assemble() show?

→ It displays the base assembly + additional features from decorators.

4. Why create separate decorators?

→ To add features independently without altering Car class.

5. Real-life example of Car Decorator?

→ Adding sunroof or sports kit to a basic car.

☑ PROGRAM 2: WiFi Scan Using ESP8266
1. What is WiFi.mode(WIFI_STA)?

→ Sets ESP in Station mode to scan networks.

2. What does WiFi.scanNetworks() return?

→ The number of Wi-Fi networks found.

3. What information can we print for each network?

→ SSID, RSSI (signal strength), encryption type.

4. Why don’t we need a password for scanning?

→ Because scanning only detects networks, not connect.

5. What does RSSI indicate?

→ Wi-Fi signal strength; higher negative value means weaker signal.

✸ SLIP 13 — (Mobile Charger Adapter + Ultrasonic Distance)
☑ PROGRAM 1: Adapter Pattern for Mobile Charger
1. Why use an Adapter in mobile charger?

→ To convert 120V supply into lower voltages like 12V or 3V needed for devices.

2. What is the Volt class?

→ A simple class representing voltage value.

3. Why does Adapter divide voltage?

→ To convert the high voltage to required lower voltage levels.

4. What type of Adapter is used?

→ Class Adapter or Object Adapter (depending on the implementation).

5. Real application?

→ Power adapters for phones, laptops, and chargers.

☑ PROGRAM 2: Ultrasonic Distance Sensor (HC-SR04)
1. What is ultrasonic sensor used for?

→ To measure distance using sound waves.

2. Why use pulseIn() function?

→ To measure time taken for echo signal to return.

3. TRIG pin role?

→ Sends 10µs pulse to start measurement.

4. ECHO pin role?

→ Outputs a pulse whose duration indicates distance.

5. Why is 0.034 used in formula?

→ It is the speed of sound in cm/µs.

✸ SLIP 14 — (Command Pattern + Vibration Sensor)
☑ PROGRAM 1: Command Pattern (Light/Garage/Stereo)
1. Why use Command Pattern?

→ To separate request invocation from the request execution.

2. What is a Command object?

→ A wrapper object with an execute() method performing an action.

3. What is the role of the Invoker?

→ It triggers the command without knowing its inner details.

4. What is Receiver?

→ The actual device (Light, GarageDoor, Stereo) that performs the action.

5. What benefit does Command Pattern give?

→ Allows undo, logging, queuing, and flexible control.

☑ **PROGRAM 2: Vibration Sensor (SW-420)**

1. What does SW-420 detect?

→ Vibrations, shocks, or sudden movements.

2. Is it analog or digital?

→ Digital output sensor.

3. Why use an LED?

→ To show alert when vibration is detected.

4. How does it work internally?

→ A spring and comparator change output when motion occurs.

5. Common applications?

→ Security alarms, motion detection, vehicle vibration systems.

✸ **SLIP 15 — (Facade Pattern + Touch Sensor)**

☑ **PROGRAM 1: Facade Pattern (Home Theater)**

1. What is the purpose of Facade Pattern?

→ It simplifies a complex system by providing a single interface to many components.

2. What components are in Home Theater?

→ DVD Player, Projector, Sound System.

3. What does watchMovie() do?

→ Turns on all components and starts playing the movie.

4. Why is facade useful?

→ Hides complexity and offers easy control for the user.

5. Example of a real-life facade?

→ TV remote controlling many internal circuits.

☑ **PROGRAM 2: Touch Sensor (TTP223)**

1. What kind of sensor is TTP223?

→ Capacitive touch sensor detecting finger presence.

2. Output on touching?

→ HIGH digital signal.

3. Why use Serial Monitor?

→ To display “FINGER DETECTED” messages.

4. Can it detect through materials?

→ Yes, through thin non-metal layers.

5. Why use LED along with sensor?

→ For visual touch indication.

✸ SLIP 16 — (Observer Number Conversion + Wi-Fi Connect)
☑ PROGRAM 1: Observer Pattern (Decimal → Hex, Octal, Binary)
1. Why use Observer Pattern here?

→ Because when decimal number changes, all other number systems update automatically.

2. What is the Subject in this program?

→ The number class that stores decimal value.

3. What do observers represent?

→ Conversions: HexObserver, OctalObserver, BinaryObserver.

4. What does update() do?

→ Converts and prints the new values.

5. Why use notifyObservers()?

→ To inform all observers when the number changes.

☑ PROGRAM 2: ESP8266 Wi-Fi Connection
1. What does WiFi.begin() require?

→ SSID and password to start connecting.

2. What is WiFi.status()?

→ Checks whether module is connected or still connecting.

3. What does WL_CONNECTED mean?

→ ESP8266 is successfully connected to Wi-Fi.

4. What is WiFi.localIP()?

→ Returns the IP address assigned by router.

5. Why do we print dots (...) while connecting?

→ To show progress while waiting for connection.

✸ SLIP 17 — (Abstract Factory Pattern + Temperature Alert)
☑ PROGRAM 1: Abstract Factory (Shapes)
1. What is Abstract Factory Pattern?

→ A pattern that creates families of related objects without specifying their classes.

2. What does FactoryProducer do?

→ Returns the required factory (RoundedFactory, NormalFactory).

3. Why separate Rounded and Normal factories?

→ To create different shape types from appropriate factories.

4. What does draw() represent?

→ Action performed by shape object.

5. Why use interface Shape?

→ To ensure different shapes follow same base structure.

☑ PROGRAM 2: Temperature Alert with LM35
1. What is the role of LM35?

→ Converts temperature into analog voltage.

2. Why compare value with threshold?

→ To trigger a notification or LED alert when temperature is high.

3. Why use analogRead()?

→ Reads the sensor's varying voltage.

4. What does LED ON indicate?

→ Temperature above threshold.

5. Why use Serial Monitor?

→ To view temperature output live.

✸ SLIP 18 — (Weather Observable + LDR Control)
☑ PROGRAM 1: Weather Station (java.util.Observable)
1. What triggers notifyObservers()?

→ When new values are set using setMeasurement().

2. Why use setChanged()?

→ To mark that new data is available for observers.

3. What type of pattern is this?

→ Behavioral pattern.

4. What does CurrentConditionsDisplay show?

→ Updated temperature, humidity, and pressure.

5. Why use Observer pattern here?

→ To automatically update multiple displays.

☑ PROGRAM 2: LDR LED Control
1. What is LDR used for?

→ To detect light intensity.

2. Why use analog input?

→ LDR provides variable resistance/voltage.

3. What does map() convert?

→ LDR reading to LED brightness range.

4. What happens when environment becomes bright?

→ LED brightness decreases.

5. Why use PWM?

→ For smooth brightness control.

✸ SLIP 19 — (Factory Pizza + Linux Commands)
☑ PROGRAM 1: Factory Method (Pizza Store)
1. Why is pizza creation inside createPizza()?

→ To allow subclasses to decide which pizza type to produce.

2. What does orderPizza() do?

→ Calls prepare(), bake(), cut(), box() sequentially.

3. Why override cut() in Chicago pizza?

→ Chicago pizza uses square-cut slices.

4. Why use abstract PizzaStore?

→ To define a standard pizza ordering process.

5. Benefit of Factory Method?

→ Flexible to add new pizza types without changing existing code.

☑ PROGRAM 2: Linux Commands on Raspberry Pi

1. What does ls do?

→ Lists files and directories.

2. What does mv do?

→ Moves or renames a file.

3. What does man command show?

→ Manual/help for a command.

4. What does tar do?

→ Creates or extracts archive files.

5. What does ps show?

→ Running processes.

✸ SLIP 20 — (I/O Decorator + Python File Ops)

☑ PROGRAM 1: I/O Decorator (Lowercase Conversion)

1. Why wrap FileInputStream?

→ To modify file data while reading.

2. Why override read(byte[]) version too?

→ To convert larger data blocks to lowercase.

3. Is decorator reusable?

→ Yes, can wrap any InputStream.

4. Why convert to lowercase?

→ To demonstrate filtering behavior using Decorator design pattern.

5. Does decorator store data?

→ No, it only processes data in transit.

☑ PROGRAM 2: Python File Reading & Word Count

1. How to open a file in Python?

→ Using open("file.txt", "r").

2. How to read line-by-line?

→ Using a for loop on the file object.

3. How to count words?

→ Using len(line.split()).

4. What does read() do?

→ Reads entire file content.

5. Why close the file?

→ To release system resources. 🐞 SLIP 16 — (Observer Number Conversion + Wi-Fi Connect)
☑ PROGRAM 1: Observer Pattern (Decimal → Hex, Octal, Binary)
1. Why use Observer Pattern here?

→ Because when decimal number changes, all other number systems update automatically.

2. What is the Subject in this program?

→ The number class that stores decimal value.

3. What do observers represent?

→ Conversions: HexObserver, OctalObserver, BinaryObserver.

4. What does update() do?

→ Converts and prints the new values.

5. Why use notifyObservers()?

→ To inform all observers when the number changes.

☑ PROGRAM 2: ESP8266 Wi-Fi Connection
1. What does WiFi.begin() require?

→ SSID and password to start connecting.

2. What is WiFi.status()?

→ Checks whether module is connected or still connecting.

3. What does WL_CONNECTED mean?

→ ESP8266 is successfully connected to Wi-Fi.

4. What is WiFi.localIP()?

→ Returns the IP address assigned by router.

5. Why do we print dots (...) while connecting?

→ To show progress while waiting for connection.

🐞 SLIP 17 — (Abstract Factory Pattern + Temperature Alert)
☑ PROGRAM 1: Abstract Factory (Shapes)

1. What is Abstract Factory Pattern?

→ A pattern that creates families of related objects without specifying their classes.

2. What does FactoryProducer do?

→ Returns the required factory (RoundedFactory, NormalFactory).

3. Why separate Rounded and Normal factories?

→ To create different shape types from appropriate factories.

4. What does draw() represent?

→ Action performed by shape object.

5. Why use interface Shape?

→ To ensure different shapes follow same base structure.

☑ PROGRAM 2: Temperature Alert with LM35
1. What is the role of LM35?

→ Converts temperature into analog voltage.

2. Why compare value with threshold?

→ To trigger a notification or LED alert when temperature is high.

3. Why use analogRead()?

→ Reads the sensor's varying voltage.

4. What does LED ON indicate?

→ Temperature above threshold.

5. Why use Serial Monitor?

→ To view temperature output live.

✸ SLIP 18 — (Weather Observable + LDR Control)
☑ PROGRAM 1: Weather Station (java.util.Observable)
1. What triggers notifyObservers()?

→ When new values are set using setMeasurement().

2. Why use setChanged()?

→ To mark that new data is available for observers.

3. What type of pattern is this?

→ Behavioral pattern.

4. What does CurrentConditionsDisplay show?

→ Updated temperature, humidity, and pressure.

5. Why use Observer pattern here?

→ To automatically update multiple displays.

☑ PROGRAM 2: LDR LED Control
1. What is LDR used for?

→ To detect light intensity.

2. Why use analog input?

→ LDR provides variable resistance/voltage.

3. What does map() convert?

→ LDR reading to LED brightness range.

4. What happens when environment becomes bright?

→ LED brightness decreases.

5. Why use PWM?

→ For smooth brightness control.

✸ SLIP 19 — (Factory Pizza + Linux Commands)
☑ PROGRAM 1: Factory Method (Pizza Store)
1. Why is pizza creation inside createPizza()?

→ To allow subclasses to decide which pizza type to produce.

2. What does orderPizza() do?

→ Calls prepare(), bake(), cut(), box() sequentially.

3. Why override cut() in Chicago pizza?

→ Chicago pizza uses square-cut slices.

4. Why use abstract PizzaStore?

→ To define a standard pizza ordering process.

5. Benefit of Factory Method?

→ Flexible to add new pizza types without changing existing code.

☑ PROGRAM 2: Linux Commands on Raspberry Pi
1. What does ls do?

→ Lists files and directories.

2. What does mv do?

→ Moves or renames a file.

3. What does man command show?

→ Manual/help for a command.

4. What does tar do?

→ Creates or extracts archive files.

5. What does ps show?

→ Running processes.

✸ SLIP 20 — (I/O Decorator + Python File Ops)
☑ PROGRAM 1: I/O Decorator (Lowercase Conversion)
1. Why wrap FileInputStream?

→ To modify file data while reading.

2. Why override read(byte[]) version too?

→ To convert larger data blocks to lowercase.

3. Is decorator reusable?

→ Yes, can wrap any InputStream.

4. Why convert to lowercase?

→ To demonstrate filtering behavior using Decorator design pattern.

5. Does decorator store data?

→ No, it only processes data in transit.

☑ PROGRAM 2: Python File Reading & Word Count
1. How to open a file in Python?

→ Using open("file.txt", "r").

2. How to read line-by-line?

→ Using a for loop on the file object.

3. How to count words?

→ Using len(line.split()).

4. What does read() do?

→ Reads entire file content.

5. Why close the file?

→ To release system resources.