**1. Project Structure & Modular Code**

* You split your project into multiple files for clarity and modularity:
  + main.py — the entry point, controlling the user menu and flow
  + caesar.py — contains functions related to Caesar cipher
  + vigenere.py — functions for Vigenère cipher
  + luhn.py — functions for Luhn algorithm (card validation)
* Each file defines its own functions (e.g., handle\_caesar()), and **main.py imports those functions** to call them when needed.

**2. Menu System in main.py**

* You print a menu with options using print() and newline characters \n for formatting:

print("1- Caesar Cipher\n2- Vigenère Cipher\n3- Check Card Number")

* You get user input using input() and convert it to an integer with int().
* You use if / elif / else statements to handle the user's choice.

**3. New Python Concepts Introduced**

**a) Newline character (\n)**

* Used inside strings to break the text into multiple lines in the output.
* Example:

print("Hello\nWorld")

prints:

Hello

World

**b) Functions**

* Defined blocks of code to perform specific tasks.
* Declared with def function\_name():
* Run only when **called**.
* Example:

def say\_hello():

print("Hello")

**c) Importing Functions From Other Files**

* To keep code modular, Python allows importing functions from other .py files (called modules).
* Syntax:

from filename\_without\_py import function\_name

* The filename must:
  + Be in the same folder (or Python module path)
  + Use valid Python naming conventions (no dashes -, only letters, numbers, and underscores \_)
  + Not include the .py extension in the import statement
* This lets you call that function inside your current file.

**4. How These Concepts Work Together in Your Project**

* You define functions for each cipher or validation in separate files.
* Your main.py shows the menu, takes user input, and **calls the appropriate function** imported from those files.
* This keeps your code clean, readable, and easy to maintain.

**5. Common Mistakes to Avoid**

* File names with dashes (-) — use underscores (\_) instead.
* Forgetting to **call** a function after defining it (defining alone doesn't run it).
* Using import incorrectly by including .py or using invalid filenames.
* Assuming loops or other new concepts are known without learning them first.