1. Classes & Objects (Basics)

1. Create a `Student` class with attributes `name`, `age`, and `grade`. Instantiate 3 objects and print their details.
2. Build a `BankAccount` class with deposit and withdraw methods.
3. Create a class `Rectangle` and calculate its area and perimeter using methods.
4. Write a `Book` class with a method to display book info.
5. Develop a `Car` class that tracks speed and model; create multiple objects.
6. Write a class `Laptop` with both default and user-defined values.

---

2. Special Methods (`__init__`, `__str__`, `__eq__`, etc.)

7. Add `__str__` method to `Book` class to customize output.
8. Add `__eq__` to `Student` class to compare students by grade.
9. Implement `__len__` in `Course` class to return number of enrolled students.
10. Use `__repr__` in `Car` class to return debug-friendly string.
11. Add `__call__` to a `FunctionHolder` class and demonstrate how objects become callable.
12. Add `__del__` to `Account` class to show destructor behavior.
13. Implement `__add__` in a `Vector` class to add two vectors.

---

3. Instance, Class, and Static Methods

14. Create a `Counter` class with:
    - instance variable `count`
    - class variable `total_counters`
    - static method to describe what a counter is.
15. Use `@classmethod` to create a user using birth year.
16. Create a `Temperature` class with a static method to convert Celsius to Fahrenheit.
17. Build a `User` class to track number of users using class method.

---

4. Inheritance, `super()`, MRO

18. Create a base class `Person` and derive `Employee` from it.
19. Add a constructor in `Employee` using `super()` to call base constructor.
20. Add a method in both `Person` and `Employee` to understand MRO.
21. Create `Animal → Dog` hierarchy and override a method.

---

5. Multiple Inheritance

22. Build a `SmartPhone` class from `Phone` and `Camera`.
23. Demonstrate MRO in a diamond inheritance problem.
24. Create a `HybridCar` from `ElectricCar` and `PetrolCar` and test method resolution.
25. Add constructors in all parent classes and use `super()` properly.

---

6. Encapsulation & Name Mangling

26. Create a `UserAccount` with private password using `__password`.
27. Write getter and setter methods to access private data.
28. Show how name mangling prevents direct access.
29. Access private variable using mangled name intentionally and explain the output.
30. Create a `Wallet` class with balance encapsulated.

---

7. Abstraction with ABC (Abstract Base Class)

31. Create an abstract class `Shape` with abstract method `area()`.
32. Derive `Circle`, `Square`, and implement `area()` differently.
33. Raise `TypeError` when trying to instantiate abstract class directly.
34. Use `ABCMeta` and `abstractmethod` from `abc` module.
35. Build `EuronDevice` abstract class and implement subclasses like `EuronTablet`, `EuronLaptop`.

---

8. Duck Typing

36. Create a function `describe(obj)` that accepts any object with `name` and `describe()` method.
37. Demonstrate duck typing by passing multiple unrelated classes to a single function.
38. Create two unrelated classes that support `len()`, `str()`, and `eq()` and pass them to a common handler.
39. Build a `Renderer` class that accepts objects with `render()` method (e.g., `PDF`, `HTML`).
40. Demonstrate duck typing with `EuronLogger` that logs anything with a `.log()` method.
41. Implement a polymorphic function that handles both `Student` and `Teacher` without inheritance.
42. Pass a list of mixed objects (from different classes) and call the same method in a loop.