SECTION 1 — Pythonic Classes & Properties

1. Create a class with a property and setter. Create a class with one attribute that can only be accessed and modified through a property and setter. Include a method that performs a calculation using the attribute.

2. Create a class with a clean __str__ representation. Create a class with at least three attributes and implement __str__ to make printed objects readable and nicely formatted.

3. Create a class with a meaningful __repr__. Create a class where __repr__ returns a string that could realistically recreate the object.

4. Create a class that initializes from **kwargs. Write a class where attributes are automatically created from keyword arguments, even when different objects receive different arguments.

5. Create a class that builds its string using a comprehension. Write a class whose __str__ method constructs its output using a comprehension and join().

SECTION 2 — Dunder Methods

6. Implement __eq__. Create a class where two objects are equal if their attributes match.

7. Implement __lt__. Create a class where objects can be compared using < based on one chosen attribute.

8. Implement __add__. Create a class where adding two objects with + produces a new combined object.

9. Implement __len__. Create a class where len(object) returns a meaningful numeric value based on internal data.

10. Implement __contains__. Create a class where you can check "value in object" using __contains__.

11. Implement __getitem__. Create a class that allows indexing with square brackets to access internal data.

SECTION 3 — Comprehension Exercises

12. Rewrite a loop using a list comprehension. Convert any loop-based list transformation into a single list comprehension.

13. Create a filtered list comprehension. Make a comprehension that filters elements based on a condition.

14. Create a dictionary using a dict comprehension. Use two lists and combine them into a dictionary via comprehension.

15. Build a formatted string using a comprehension. Generate a formatted string from object or dictionary data using comprehension and join().

16. Create a nested comprehension. Generate a two-dimensional structure (like a table or grid) with nested comprehensions. SECTION 4 — Combined OOP + Scope + Pythonic Techniques

17. Demonstrate LEGB with nested functions. Create a function that contains another function. Use variables with the same name at multiple levels and demonstrate which is used where. Modify the enclosing variable using nonlocal.

18. Create a class that processes input using a comprehension. Write a class that receives a list or dictionary and transforms or filters the data internally using a comprehension.

19. Create a class that builds itself from a dictionary. Write a class that receives a dictionary and turns every key/value pair into attributes. Build __str__ using a comprehension.

20. Combine kwargs, property, and a dunder method. Create a class that: - accepts all attributes via **kwargs - includes at least one property with getter and setter - implements one or more dunder methods - includes a method that performs a calculation using its data