# Python OOP Questions & Answers (Set 4 - Operator Overloading)

### *Q1. Which two operator overloading methods can you use in your classes to support iteration?*

Two ways:
1. __iter__ with __next__ (iterator protocol).
2. __getitem__ with successive indexes until IndexError (sequence protocol).

### *Q2. In what contexts do the two operator overloading methods manage printing?*

__str__: user-friendly string, used by print(), str(), f-strings.
__repr__: developer/debug string, used by repr(), REPL, inside containers. Fallback if __str__ not defined.

### *Q3. In a class, how do you intercept slice operations?*

Implement __getitem__, __setitem__, and __delitem__.
Inside, check if key is a slice instance, then handle accordingly.

### *Q4. In a class, how do you capture in-place addition?*

Implement __iadd__(self, other) for +=.
Mutable types typically mutate self and return it.
If missing, Python falls back to __add__ and rebinding.

### *Q5. When is it appropriate to use operator overloading?*

Use when it makes code more natural and aligns with semantics:
- Math/structured types (vectors, matrices, complex numbers).
- Container-like behavior (iteration, indexing, slicing).
- String/debug views (__repr__, __str__).
Avoid when it surprises users or makes code less readable.