OOP via Python Session 08 Recap

Stephen Leach, Feb 2022

X-Ray Vision

- Two-layer architecture
- Types/behaviour layered on top of classes/methods
- Types correspond to categories in the problem domain (employees, vehicles, application forms, ...)
- Behaviour is implemented by public methods
- Logical properties in the problem domain are captured as contracts that govern behaviours

S.O.L.I.D.

- Single responsibility principle - "Class has one job to do"
- Open/closed principle - "Happy to be used by others; Unhappy to be changed by others."
- Liskov substitution principle - "Class can be replaced by any of its children"
- Interface segregation principle - "Code should not depend on methods it does not use"
- Dependency inversion principle - "When classes talk to each other in a very specific way, they should use promises (interfaces, parents), so classes can change as long as they keep the promise."

The One Principle: Connect with Weakest Types

- Single responsibility principle - Weak types do single jobs, use weak types
- Open/closed principle - Use types not classes
- Liskov substitution principle - Types have contracts that apply to subtypes
- Interface segregation principle - Use weak types for parameters
- Dependency inversion principle - Use "factories", not constructors

Aside for working in C#

Which is better?

- `List<Stuff> x = GetStuffList();`
- `var x = GetStuffList();`

C#

If you can't stop yourself:

- `IList<IStuff> x = GetStuffList();`