OOP via Python: Session 04 Recap

Stephen Leach, Nov 2021

Topics

- Code Organisation
- Public vs Non-Public
- Core vs Non-Core

Organisation

- Key point - OOP languages usually enforce a discipline of methods inside classes
- From personal experience, this was in practice a huge improvement for most teams because it enforced a non-stupid policy
- In all modern languages this causes problems with non-small projects because additional methods cannot be added to a class subsequently, leading to the proliferation of non-extensible functions
- As there's no alternative, there's not much point in worrying about it

Public vs Non-public Methods

- Behaviour vs Implementation - separating the methods that correspond to operations in the problem domain and essential from the artefacts of design
- Access Control - an attempt to guarantee implementation hiding, causing more problems than it solves
- Scope - an attempt to export public methods widely but limit the visibility of non-public methods, which also has drawbacks
- Python very weakly supports scoping via name-mangling/import-hiding by hijacking the underscore

Core vs Surface Methods

- Implementations are typically layered
- You only need to alter the bottom layer ("core") to change the implementation
- Abstract classes, which are supported in Python, exemplify this principle
- Core methods are usually said to be public (no leading underscore) or protected (single underscore)

Exercise

- How did we get on?