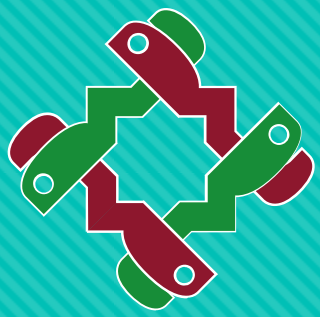


PYCONES
GRANADA 2022

Inyección de dependencias, fácil!

@panicoenlaxbox

https://www.panicoenlaxbox.com/

https://analyticalways.com/

¿Por qué DI?

- Código mantenible.
- Código testeable.
- Entrega de valor continua.

Dependency inversion

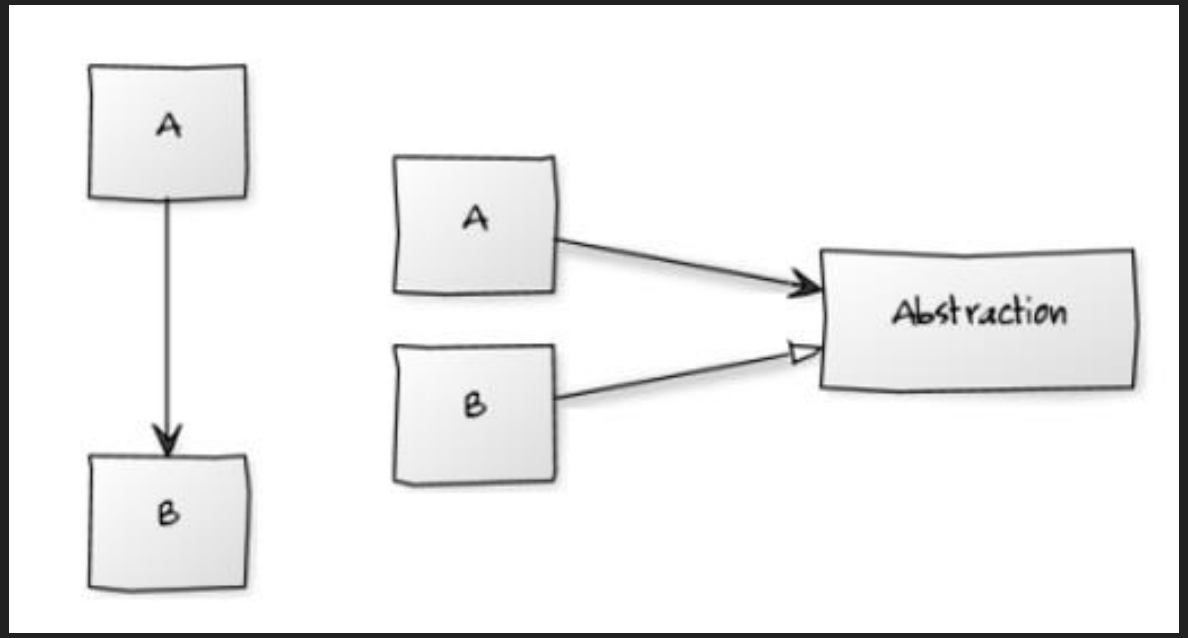
- High-level modules should not depend on low-level modules. Both should depend on abstractions.
- Abstractions should not depend on details. Details should depend on abstractions.

dependency_inversion/abstraction_implementation.py

Dependency inversion v2

- Nuestro código de negocio no debería depender de detalles técnicos. Ambos deberían depender de abstracciones.
- Las abstracciones mandan.

DIP, en la práctica



dependency_inversion/{no_dip|dip}.py

Dependency inversion v3

- Depend on abstractions
  - http://principles-wiki.net/principles:dependency_inversion_principle
  - Program to an interface, not an implementation
    - Design Patterns: Elements of Reusable Object-Oriented Software

Abstracción

- ABC
- Duck typing
- typing.Protocol
  - https://peps.python.org/pep-0544/

abstractions/{duck_typing|protocol}.py

Beneficios

- El módulo de bajo nivel puede cambiar sin impactar al de alto nivel.
- Se pueden reutilizar los módulos de alto nivel.
- Puede cambiar el comportamiento del sistema según la implementación que se inyecte del módulo de bajo nivel.
  - Open-Closed Principle (OCP).
- Parallel development.
- Código más claro.
  - The Zen of Python “Explicit is better than implicit.”
- Nos ayuda a detectar code-smells.
  - Single Responsibility Principle (SRP).

`benefits/{more_implementations|code_smell}.py`

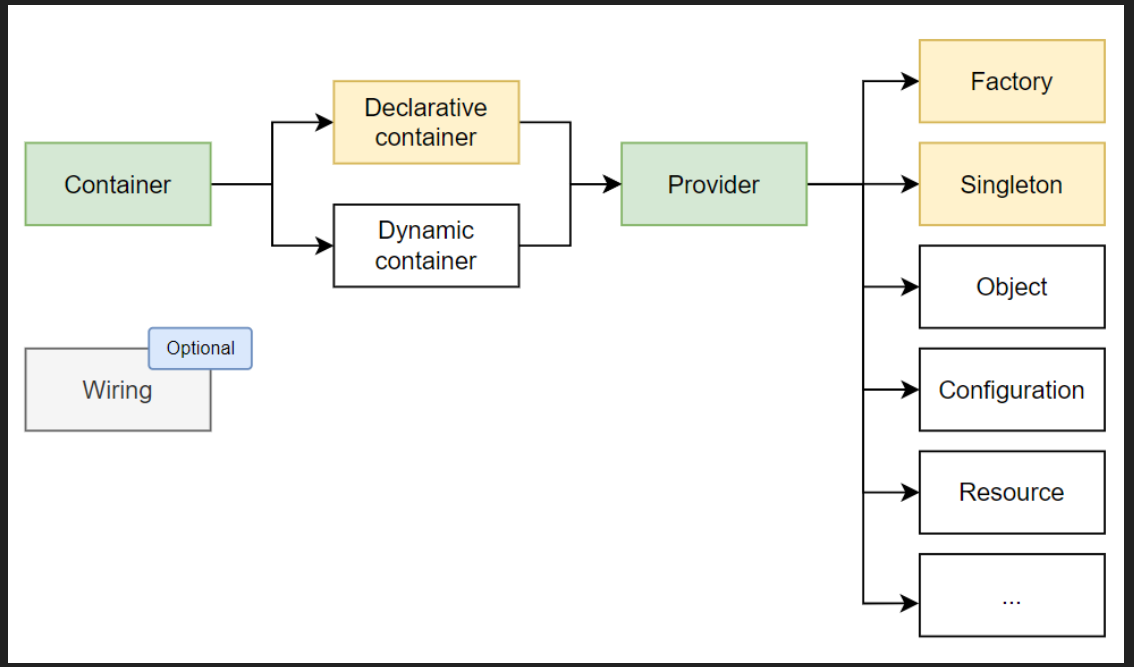
Testing

- Se puede aislar el SUT, mockeando las dependencias.
  - https://opensource.com/article/17/5/30-best-practices-software-development-and-testing
- Monkey-patching vs mocking.
  - El patch se hace de la implementación y el mock de la abstracción.
  - OK
    - responses, https://github.com/getsentry/responses
    - freezegun, https://github.com/spulec/freezegun
    - pyfakefs, https://github.com/jmcgeheeiv/pyfakefs/

`testing/{test_monkey_patching|test_mocking}.py`

Dependency Injector

- https://python-dependency-injector.ets-labs.org/



Dependency Injector

pure_di, dependency_injector_library/{main_with_no_magic|main_with_magic}.py, tests/dependency_injector_library/test_using_container.py

DI container, interception

- Object composition.
- Lifetime management.
- Interception.

interception/{decorator|decorator_pattern}.py

¿Qué hay que inyectar?

- Dependencias estables vs volátiles
  - https://www.manning.com/books/dependency-injection-in-dot-net
- Se inyectan las dependencias volátiles
  - Una dependencia es estable hasta que se demuestre lo contrario.
    - ¿Queremos inyectar distintas implementaciones?
      - ¿Queremos mockear la dependencia?
    - ¿Es necesaria para hacer parallel development?

¿Hay que hacer siempre DI?

- Depende…
  - El beneficio es directamente proporcional al tamaño y complejidad de la aplicación.
  - Si lo hacerlo supera o iguala a la complejidad inherente del problema, entonces no… es complejidad accidental.

That's all folks

- ¡Gracias!
- https://github.com/panicoenlaxbox/pycones2022