

**PYCONES**  
GRANADA 2022

# Inyección de dependencias, fácil!

@panicoenlaxbox

<https://www.panicoenlaxbox.com/>

<https://analyticalways.com/>

<https://github.com/panicoenlaxbox/pycones2022>

# ¿Por qué?

- Código mantenible.
- Código testeable.
- Entrega de valor continua.
- DI es sólo el comienzo.

# Dependency inversion principle

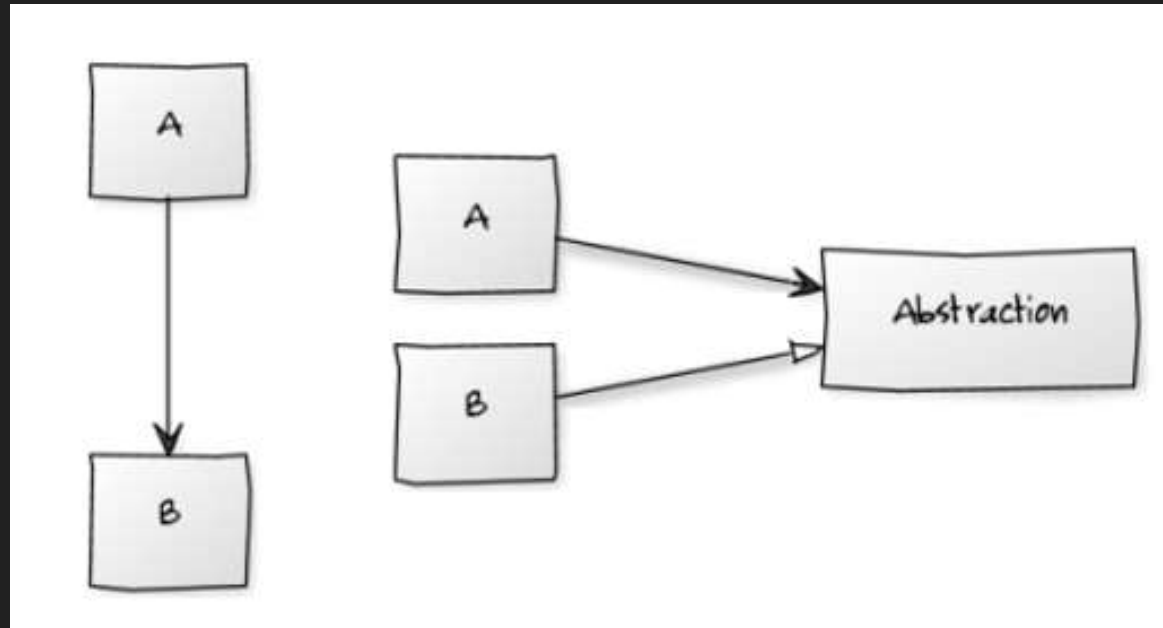
- High-level modules should not depend on low-level modules. Both should depend on abstractions.<sup>1</sup>
- Abstractions should not depend on details. Details should depend on abstractions.<sup>2</sup>

1. <https://blog.koalite.com/2015/02/cohesion-y-acoplamiento/>

2. <https://www.joelonsoftware.com/2002/11/11/the-law-of-leaky-abstractions/>

2. <https://www.serginet.com/2018/03/todas-las-abstracciones-fugan.html>

# Dependency inversion principle



# DIP, en resumen

- Depend on abstractions
  - [http://principles-wiki.net/principles:dependency\\_inversion\\_principle](http://principles-wiki.net/principles:dependency_inversion_principle)
- Program to an interface, not an implementation
  - <https://www.amazon.es/Design-Patterns-Object-Oriented-professional-computing/dp/0201633612>

# Abstracción

- ABC
- Duck typing
- typing.Protocol

abstractions\\*.py

# Beneficios

- Ambos módulos pueden cambiar por separado.
- Se pueden reutilizar los módulos de alto nivel.
- Se puede cambiar el comportamiento del sistema en función de que implementación se inyecta.
  - Open-Closed Principle (OCP).
- Código más claro.
  - The Zen of Python "Explicit is better than implicit."
- Nos ayuda a detectar code-smells.
  - Single Responsibility Principle (SRP).
- Parallel development.

# Testing

- Se puede aislar el SUT, mockeando las dependencias.
  - <https://opensource.com/article/17/5/30-best-practices-software-development-and-testing>
- Podemos hacer menos monkey-patching.
  - IMHO, mucho patch es sinónimo de un mal diseño.
  - El patch se hace de la implementación y el mock de la abstracción.
  - En cualquier caso, nadie haría patch en el código de producción.
- Sin problemas con...
  - responses, <https://github.com/getsentry/responses>
  - freezegun, <https://github.com/spulec/freezegun>
  - pyfakefs, <https://github.com/jmcgeheeiv/pyfakefs/>

testing\\*.py



# DI container

~~Peer's man DI~~ Pure DI

<https://blog.ploeh.dk/2014/06/10/pure-di/>

DI container

- Object composition.
- Lifetime management.
- Interception.

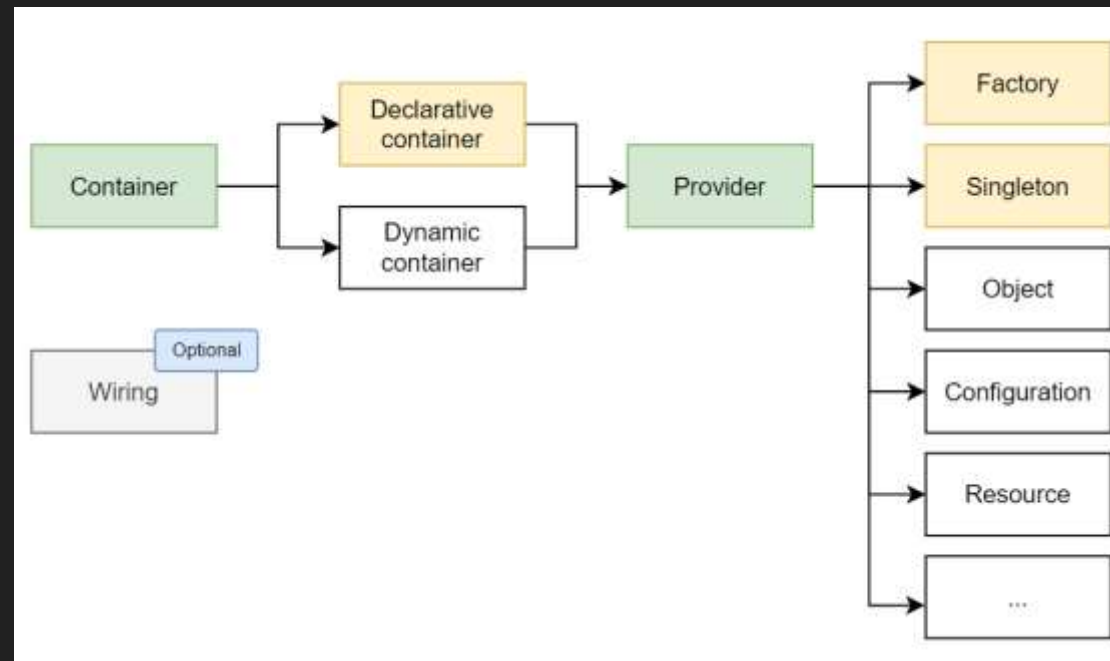
# ¿Qué hay que inyectar?

- Dependencias estables vs volátiles
  - <https://www.manning.com/books/dependency-injection-in-dot-net>
- Se inyectan las dependencias volátiles
- Una dependencia es estable mientras que no se demuestre que es volátil.
  - ¿Queremos inyectar distintas implementaciones?
  - ¿Es necesario para hacer parallel development?
  - ¿Queremos mockear la dependencia sin usar patch?



# Dependency Injector

○ <https://python-dependency-injector.ets-labs.org/>



`pure_di /, dependency_injector_example/*.py, tests/dependency_injector_example/*.py`

# ¿Hay que hacer siempre DIP?

- Depende...
  - El beneficio es directamente proporcional al tamaño y complejidad de la aplicación.

# That's all folks

- @panicoenlaxbox

- <https://www.panicoenlaxbox.com/>

- <https://github.com/panicoenlaxbox/pycones2022>