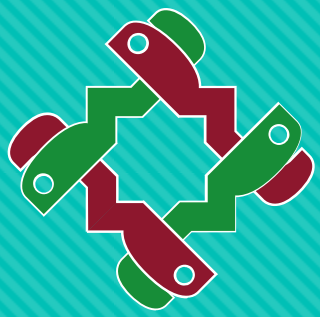


PYCONES
GRANADA 2022

# Inyección de dependencias, ~~fácil!~~

@panicoenlaxbox

https://www.panicoenlaxbox.com/

https://analyticalways.com/

https://github.com/panicoenlaxbox/pycones2022

# ¿Por qué?

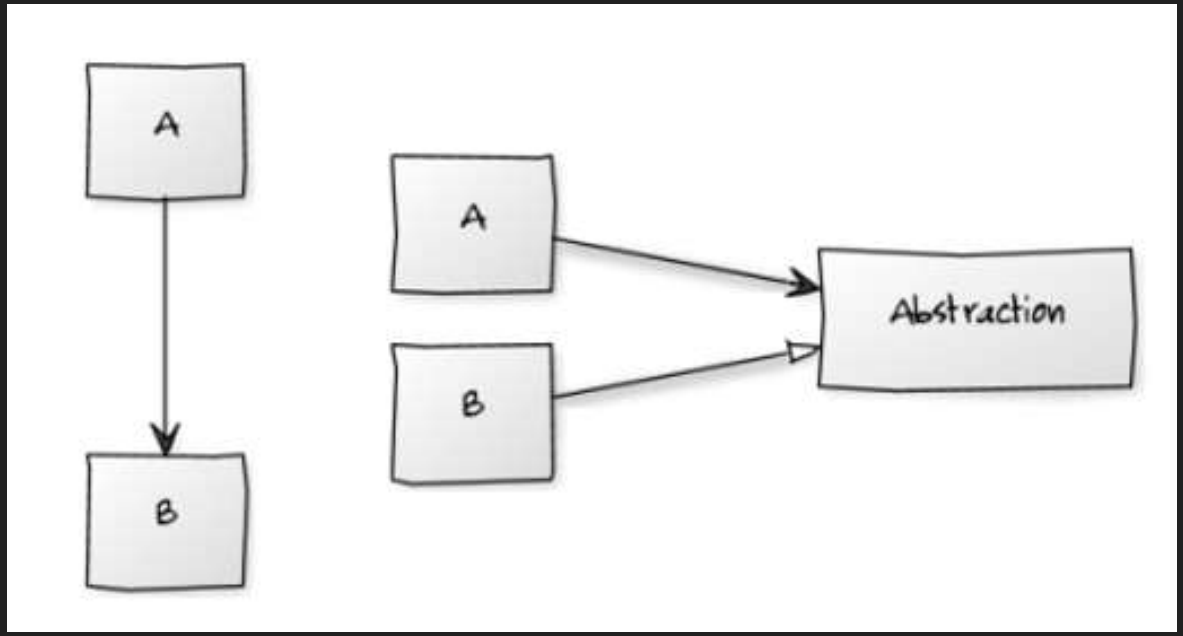
- Código mantenible
  - Que sea relativamente fácil poder cambiar, actualizar, añadir o quitarle funcionalidad al código y corregir los problemas cuando aparezcan.
- Entrega de valor continua.

# Dependency inversion principle

- High-level modules should not depend on low-level modules. Both should depend on abstractions.[1]
- Abstractions should not depend on details. Details should depend on abstractions.[2]

1. https://blog.koalite.com/2015/02/cohesion-y-acoplamiento/
2. https://www.joelonsoftware.com/2002/11/11/the-law-of-leaky-abstractions/

# Dependency inversion principle




dependency_inversion_principle*.py

# Dependency inversion principle

- Program to an interface, not an implementation
  - https://www.amazon.es/Design-Patterns-Object-Oriented-professional-computing/dp/0201633612
- Depend on abstractions
  - http://principles-wiki.net/principles:dependency_inversion_principle

# 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.
  - Late binding.
  - 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.

```
benefits\*.py
```

# Testing

- Se puede aislar el SUT y mockear las dependencias.
  - https://opensource.com/article/17/5/30-best-practices-software-development-and-testing
- Patch
  - Se hace patch a la implementación y se mockea la abstracción.
  - Tu código podría no tener un buen diseño y no ser fácilmente testeable.
- 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

- ~~Poor's man DI~~ Pure DI
  - https://blog.ploeh.dk/2014/06/10/pure-di/
- DI container
  - Responsabilidades
    - Object composition.
    - Lifetime management.
    - Interception.

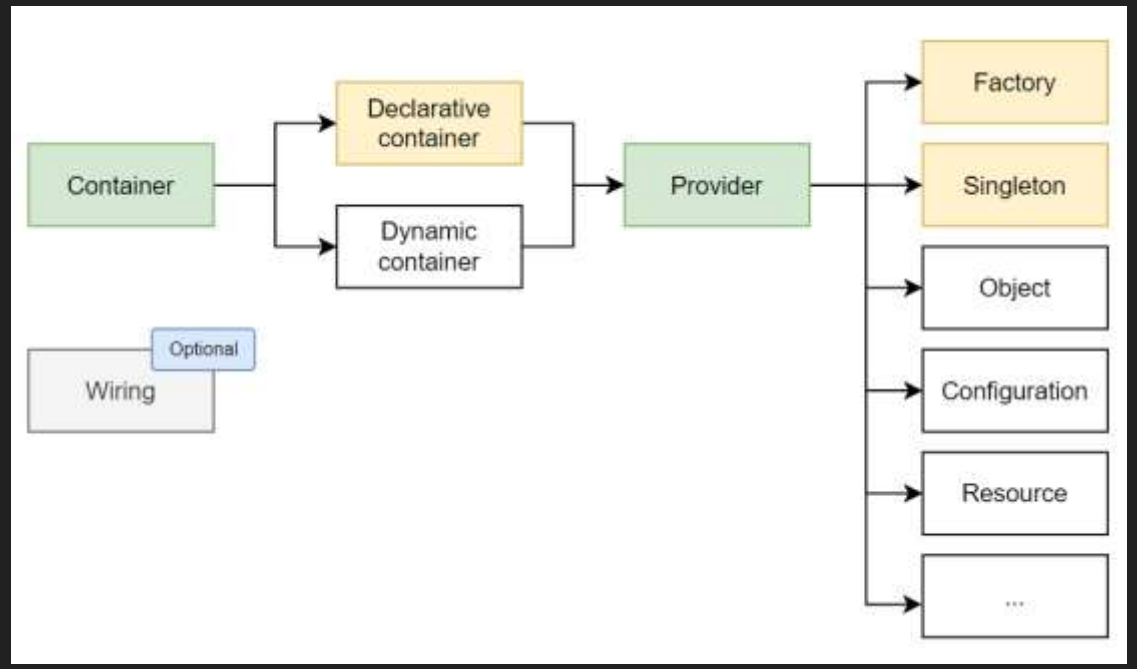
`interception\*.py`

# ¿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 mientras que no se demuestre lo contrario
  - ¿Vamos a querer inyectar distintas implementaciones?
  - ¿Es necesario parallel development?
  - Testing
    - ¿Vamos a querer mockear la dependencia?

# Dependency Injector

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




Dependency Injector

`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.