Python Biella Group

S.O.L.I.D.

Python Biella Group

Obiettivi

Migliorare conoscenza e utilizzo della programmazione a oggetti

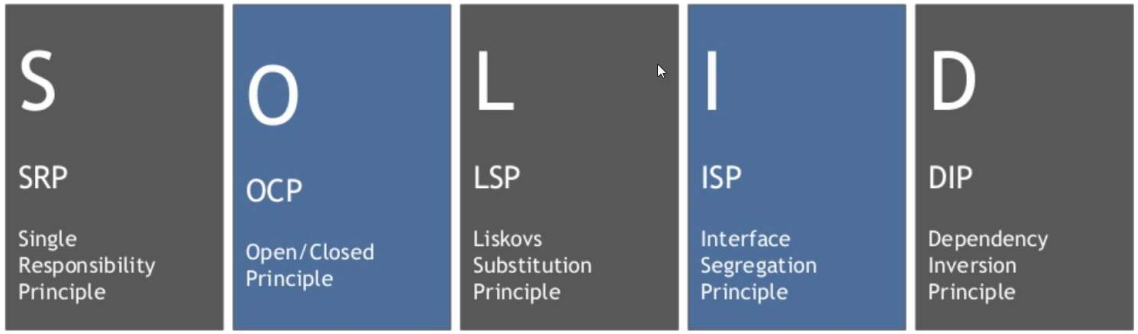
Rendere più leggibile il nostro codice

Python Biella Group

S.O.L.I.D.

- Framework per progettare / fare refactoring con codice object-oriented “migliore”
- Serie di principi che, se rispettati, garantiscono
  - Manutenzione piu’ facile / Piu’ estendibilità
  - Comprensibilita’ / Codice piu’ logico e facile da leggere
  - Stabilita’ / Codice piu’ duraturo

- SOLID è un acronimo facilmente memorizzabile e sta per:
  - Single Responsibility Principle (SRP)
  - Open/Closed Principle (OCP)
  - Liskov Substitution Principle (LSP)
  - Interface Segregation Principle (ISP)
  - Dependency Inversion Principle (DIP)



- Vediamoli separatamente con esempi in Python

Python Biella Group

Single Responsibility Principle (SRP)

"A class should have one and only one reason to change"

- SRP richiede che ogni classe / metodo abbia una singola responsabilita', debba fare un solo lavoro.
- Se una classe ha piu' di una responsabilita', diventa accoppiata e non e' riusabile

Open/Closed Principle (OCP)

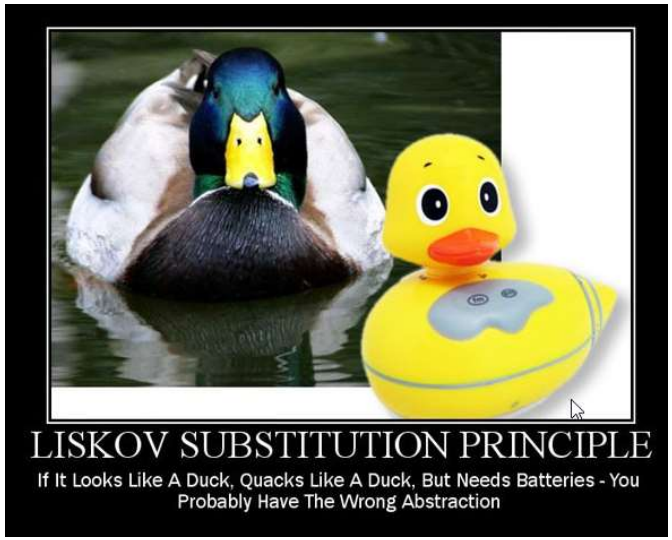
*"Software entities should be **open** for extension, but **closed** for modification."*

- Classi, moduli, funzioni: dovrebbero essere aperte per l'estensione a nuove funzionalita', chiuse per la modifica

Liskov Substitution Principle (LSP)

"A subclass should behave in such a way that it will not cause problems when used instead of the superclass."

- Se S è un sottotipo di T, gli oggetti di tipo T possono essere sostituiti con oggetti di tipo S
- Per qualsiasi classe, un client dovrebbe essere in grado di utilizzare indistinguibilmente uno qualsiasi dei suoi sottotipi, senza nemmeno accorgersene, e quindi senza compromettere il comportamento previsto in fase di esecuzione. Cio' significa che i client sono completamente isolati e inconsapevoli dei cambiamenti nella gerarchia delle classi.

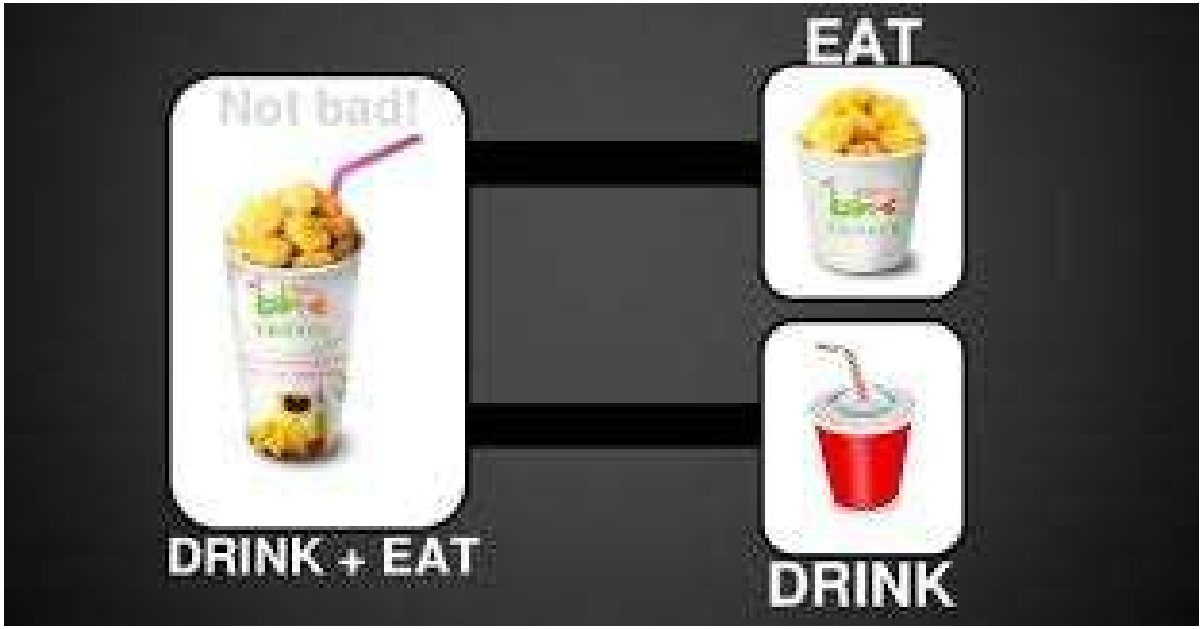


Python Biella Group

Interface Segregation Principle (ISP)

“Clients should not be forced to depend upon interfaces that they don't use”

- Creare interfacce/astrazioni a "grana fine«, specifiche; evitare interfacce «grasse», generiche.
- I client non dovrebbero essere costretti a dipendere da interfacce che non utilizzano.
- Questo principio si occupa degli svantaggi dell'implementazione di grandi interfacce.



Python Biella Group

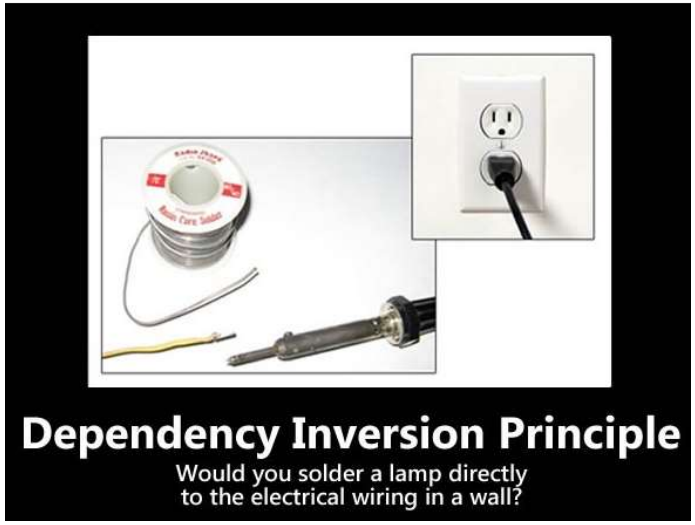
Dependency Inversion Principle (DIP)

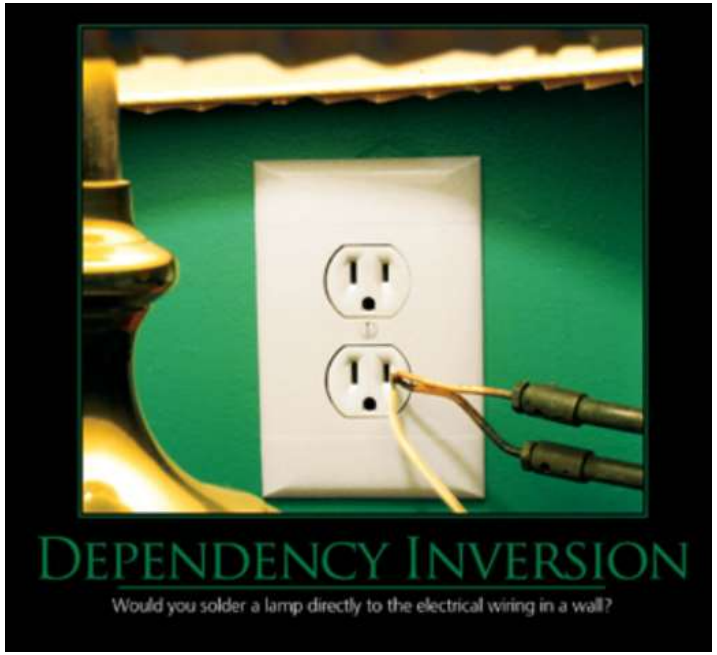
- La dipendenza dovrebbe essere sulle astrazioni, non sugli oggetti concreti.
- I moduli di alto livello non dovrebbero dipendere dai moduli di basso livello.
- Sia le classi di basso che quelle di alto livello dovrebbero dipendere dalle stesse astrazioni.
- Le astrazioni non dovrebbero dipendere dai dettagli.
- I dettagli dovrebbero dipendere dalle astrazioni.

"High-level modules should not depend on low-level modules. Both should depend on abstractions."

"Abstractions should not depend upon details. Details should depend upon abstractions."

- Questo è ciò che lega tutto insieme. Tutto ciò che abbiamo fatto con gli altri principi SOLID è stato quello di arrivare a un punto in cui non siamo più dipendenti da un dettaglio





Python Biella Group

Riferimenti e approfondimenti

- ESEMPIO DA
- https://www.youtube.com/watch?v=pTB30aXS77U&t=262s

- ALTRI RIFERIMENTI
- https://github.com/heykarimoff/solid.python
- https://codingwithjohan.com/blog
- https://medium.com/@dorela/s-o-l-i-d-principles-explained-in-python-with-examples-3332520b90ff
- https://dev.to/ezzy1337/a-pythonic-guide-to-solid-design-principles-4c8i

... c'è sempre qualcosa di bello e utile da imparare insieme

- Sito: https://pythonbiella.herokuapp.com/
- GitHub: https://github.com/PythonGroupBiella/MaterialeLezioni
- YouTube: https://www.youtube.com/c/PythonBiellaGroup/
- Telegram: https://t.me/PythonBiellaGroup

JOIN US!