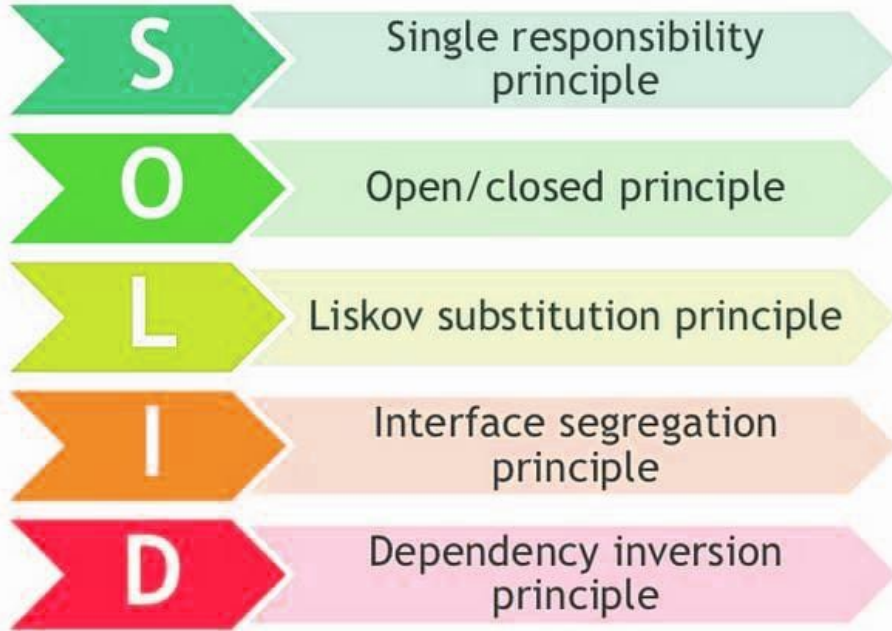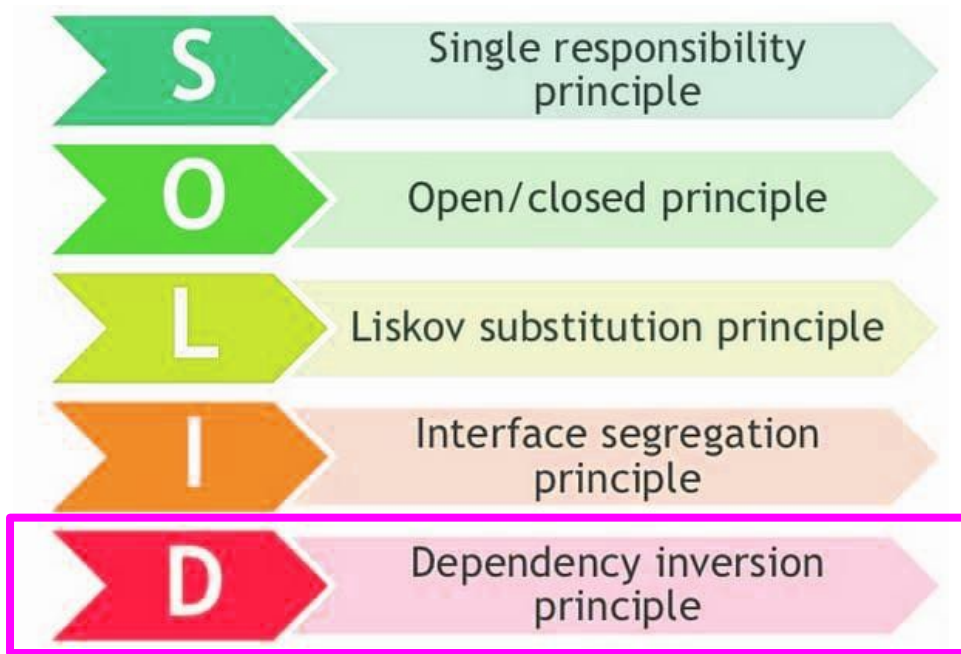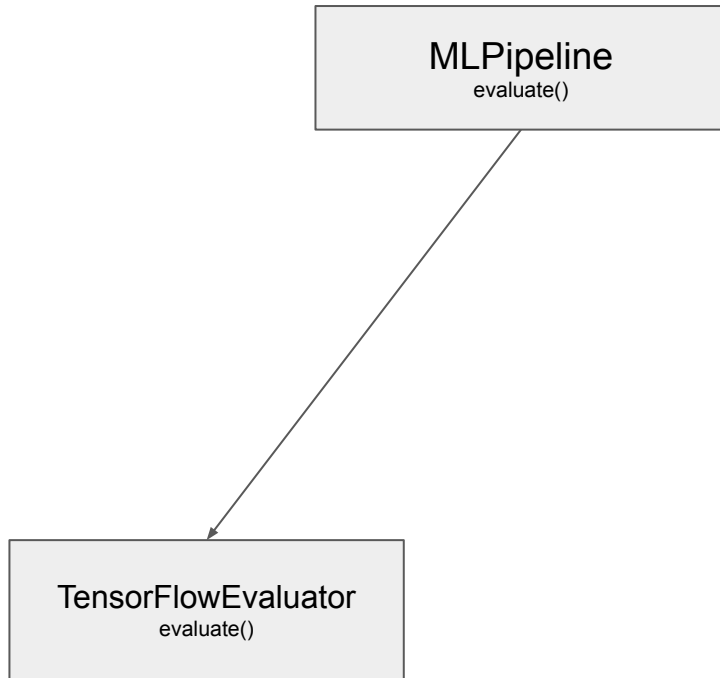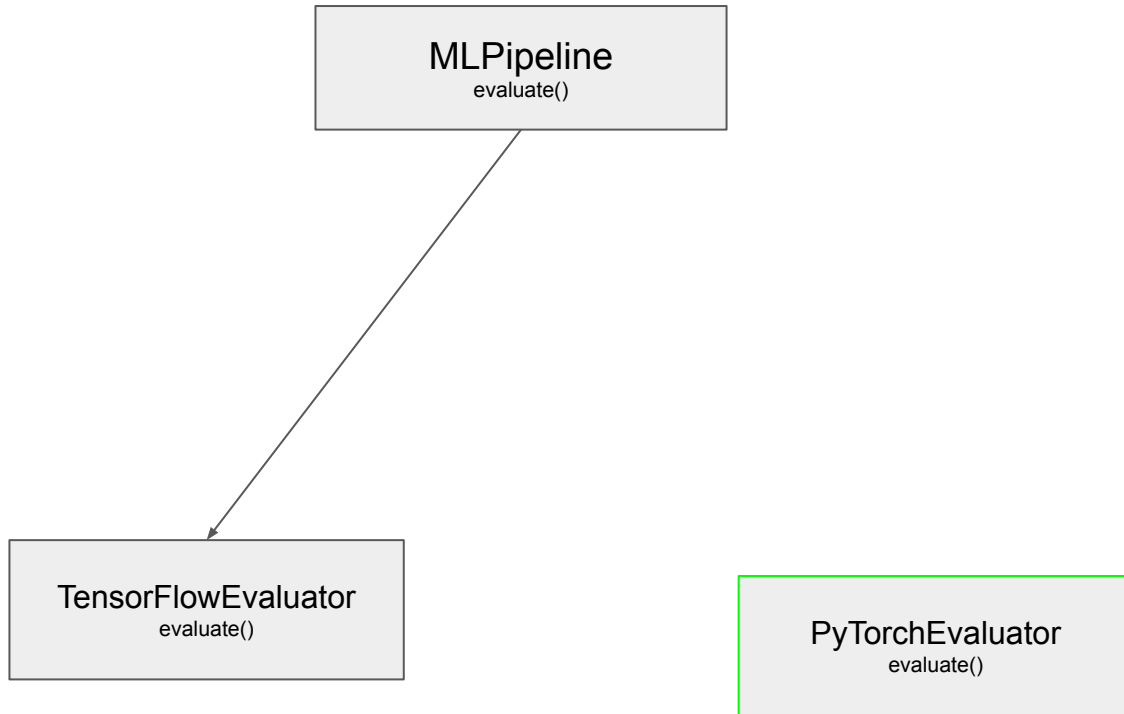# SOLID principles

SOLID principles

Classes should depend on abstractions not concretions

Spotting DIP violations

- Does class depend on concrete objects?

- Are you instantiating a concrete object in the constructor?

```
┌─────────────────────────────┐
│         MLPipeline          │
│         evaluate()          │
└─────────────────────────────┘
              │
              │
              ▼
┌─────────────────────────────┐
│      TensorFlowEvaluator    │
│         evaluate()          │
└─────────────────────────────┘
```

MLPipeline
evaluate()

TensorFlowEvaluator
evaluate()

PyTorchEvaluator
evaluate()

```
MLPipeline
evaluate()
```

```
Evaluator <I>
evaluate()
```

```
TensorFlowEvaluator
evaluate()
```

```
PyTorchEvaluator
evaluate()
```
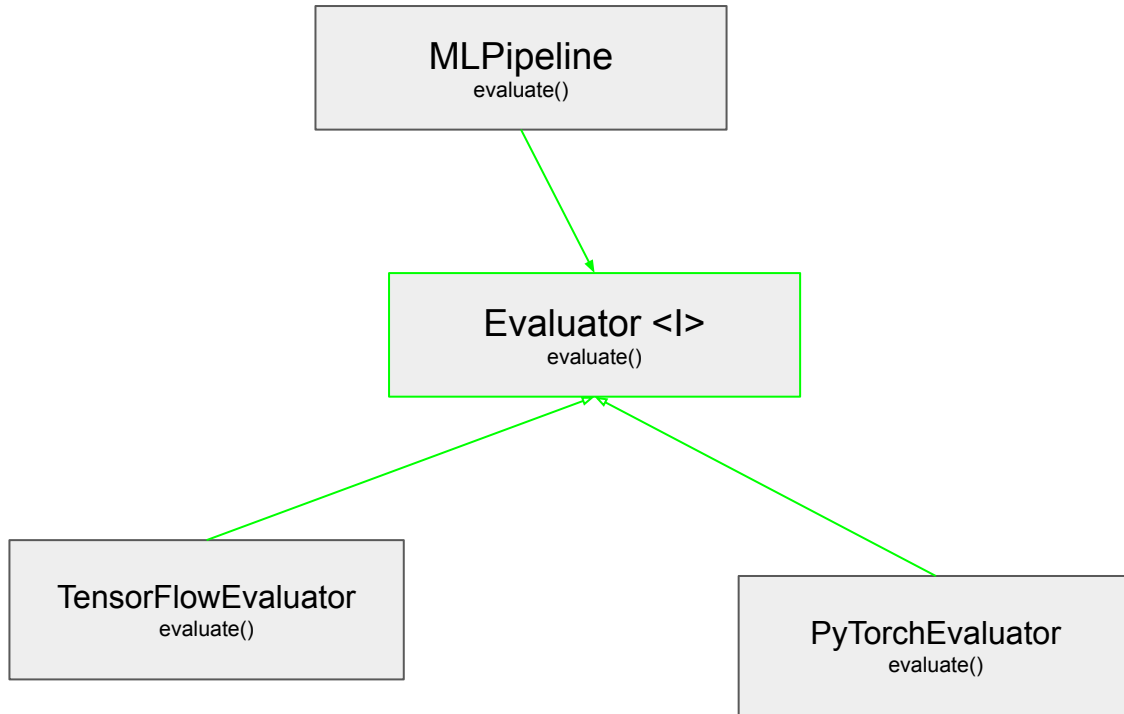
DIP main benefit

- Protect code by making it independent of elements that are fragile / unstable (concretions)

- Components are loosely coupled (open to extension, closed to modification)

- Testing is easy

SOLID principles

S	Single responsibility principle
O	Open/closed principle
L	Liskov substitution principle
I	Interface segregation principle
D	Dependency inversion principle