# Lab: SOLID

Problems for in-class lab for the [Python OOP Course @SoftUni](https://softuni.bg/courses/python-oop).

## Books

**Refactor** the provided code, so there is a separate class called **Library**, which contains **books** and has a method called **find\_book(title)** that returns the book with the given **title**. **Remove** the **unnecessary** code from the **Book** class.

## Animals

**Refactor** the provided code, so you **don't** need to make **any changes** in it when you want to **add** new species to the animals list

## Robots

**Refactor** the provided code, so it is in line with the **Liskov Substitution Principle**. Define a method in the **parent** class. The **subclasses** should **implement** the method, so it **returns** the **count of sensors** for each type of robot.

## Entertainment System

We've been hired to create a game where the player sets up **entertainment systems**. Each piece of the system (television, game console, etc.) uses a specific **cable** to **connect** to another device. The **TV** uses an **HDMI** cable to connect to a game console. Both the **game console** and **TV connect** to a router via an **ethernet cable** so they can access the internet. And lastly, all the devices are connected to the wall via a **power cable** so they can turn on. Your job is to **extend** this behavior in the device classes.

## Print books

We want to be able to **print books**, but before printing the book we should be able to **format it**. To accomplish this we have a class that can print books **called Printer** and a **class Formatter** which is used by Printer. **Refactor** the provided code that breaks the DIP because both Printer and Formatter **depend on concretions**, not abstractions by **creating** some **abstractions** and inject them wherever they are needed.