Python Single Responsibility Principle

Summary: in this tutorial, you'll learn about the single responsibility principle and how to implement it in Python.

What is SOLID

SOLID is an abbreviation that stands for five software design principles compiled by Uncle Bob (https://blog.cleancoder.com/) :

- **S** – Single responsibility Principle
- **O** – Open-closed Principle (https://www.pythontutorial.net/python-oop/python-open-closed-principle/)
- **L** – Liskov Substitution Principle (https://www.pythontutorial.net/python-oop/python-liskov-substitution-principle/)
- **I** – Interface Segregation Principle (https://www.pythontutorial.net/python-oop/python-interface-segregation-principle/)
- **D** – Dependency Inversion Principle (https://www.pythontutorial.net/python-oop/python-dependency-inversion-principle/)

The single responsibility is the first principle in the SOLID principles.

Introduction to the single responsibility principle

The single responsibility principle (SRP) states that every class (https://www.pythontutorial.net/python-oop/python-class/) , method, and function should have only one job or one reason to change.

The purposes of the single responsibility principle are to:

- Create high cohesive and robust classes, methods, and functions.
- Promote class composition
- Avoid code duplication

Let's take a look at the following `Person` class:

```
class Person:
    def __init__(self, name):
        self.name = name

    def __repr__(self):
        return f'Person(name={self.name})'

    @classmethod
    def save(cls, person):
        print(f'Save the {person} to the database')


if __name__ == '__main__':
    p = Person('John Doe')
    Person.save(p)
```

This `Person` class has two jobs:

- Manage the person's property.
- Store the person in the database.

Later, if you want to save the `Person` into different storage such as a file, you'll need to change the `save()` method, which also changes the whole `Person` class.

To make the `Person` class conforms to the single responsibility principle, you'll need to create another class that is in charge of storing the `Person` to a database. For example:

```
class Person:
    def __init__(self, name):
        self.name = name

    def __repr__(self):
        return f'Person(name={self.name})'


class PersonDB:
    def save(self, person):
        print(f'Save the {person} to the database')


if __name__ == '__main__':
    p = Person('John Doe')

    db = PersonDB()
    db.save(p)
```

In this design, we separate the `Person` class into two classes: `Person` and `PersonDB` :

- The Person class is responsible for managing the person's properties.
- The PersonDB class is responsible for storing the person in the database.

In this design, if you want to save the `Person` to different storage, you can define another class to do that. And you don't need to change the `Person` class.

When designing classes, you should put related methods that have the same reason for change together. In other words, you should separate classes if they change for different reasons.

This design has one issue that you need to deal with two classes: `Person` and `PersonDB` .

To make it more convenient, you can use the facade pattern so that the `Person` class will be the facade for the `PersonDB` class like this:

```
class PersonDB:
    def save(self, person):
        print(f'Save the {person} to the database')


class Person:
    def __init__(self, name):
        self.name = name
        self.db = PersonDB()

    def __repr__(self):
        return f'Person(name={self.name})'

    def save(self):
        self.db.save(person=self)


if __name__ == '__main__':
    p = Person('John Doe')
    p.save()
```

Summary

- The single responsibility principle (SRP) states that every class, method, or function should have only one job or one reason to change.
- Use the single responsibility principle to separate classes, methods, and functions with the same reason for changes.