All the examples contains SRP principle in background.
Go through app/routes/files.py to read them

1)SRP - get_hero_specs.py

```
from flask import render_template
from app import app


# SRP
class HasHealth:
    def damage_take(self):
        return "Health - 50"

    def is_alive(self):
        return True

    def is_dead(self):
        return False


class CanAttack:
    def damage_make(self):
        return "100"


# We can create all new class for new specs

class Hero:
    def __init__(self, name):
        self.name = name


# It will have only single reason to change now, that is for returning another spec
class ReturnHeroSpec(Hero, CanAttack):

    def __repr__(self):
        return self.name + "has Attack Power:" + self.damage_make()


@app.route('/get_hero_spec')
def get_hero_specs():
    hero_specs = ReturnHeroSpec(name="Superman")
    return render_template('hero.html', hero_specs=hero_specs, title='Solid Principle')
```

2)OCP - get_hero_address.py

```
from flask import render_template
from app import app


# ----------------------------------------------------------------------
# Open Closed Principles
class BlockAddress:
    def return_builing_number(self):
        return "B-28"


class FullAddress(BlockAddress):
    def __repr__(self):
        return "City:- Surat" + "block: " + self.return_builing_number()


# --------------------------------------------------------------------------------
@app.route('/get_hero_address')
def get_hero_address():
    hero_address = FullAddress()
    return render_template('hero_address.html', hero_address=hero_address, title='Solid
Principle')
```

3)LSP
https://github.com/rgunkar/Solid-Principles/blob/main/LSP_correct_code.py

4)ISP - get_animal_properties.py

```
# ISP Broken Code

# from abc import ABC, abstractmethod

# class Animal(ABC):
#     @abstractmethod
#     def eat(self):
#         pass
#     @abstractmethod
#     def walk(self):
#         pass
#     @abstractmethod
#     def swim(self):
#         pass
#     @abstractmethod
#     def fly(self):
#         pass
#
# class Cat(Animal):
#     def eat(self):
#         return True
#     def walk(self):
#         return True
#     def swim(self):
#         raise NotImplemented
#     def fly(self):
#         raise NotImplemented
#
# class Duck(Animal):
#     def eat(self):
#         return True
#     def walk(self):
#         return True
#     def swim(self):
#         return True
#     def fly(self):
#         raise NotImplemented
#
# class Pigeon(Animal):
#     def eat(self):
#         return True
#     def walk(self):
#         return True
#     def swim(self):
#         raise NotImplemented
#     def fly(self):
#         return True


# ISP Correct Code
from abc import ABC, abstractmethod
from flask import render_template
from app import app


class Animal(ABC):
    @abstractmethod
    def eat(self):
        pass

    @abstractmethod
    def walk(self):
        pass


class SwimAbility(ABC):
    @abstractmethod
    def swim(self):
        pass


class FlyAbility(ABC):
    @abstractmethod
    def fly(self):
        pass


class Cat(Animal):
    def eat(self):
        return "eats "

    def walk(self):
        return "walks "

    def __repr__(self):
        return self.eat() + self.walk()

class Duck(Animal, SwimAbility):
    def eat(self):
        return "eats "

    def walk(self):
        return "walks "

    def swim(self):
        return "swims "

    def __repr__(self):
        return self.eat() + self.walk() + self.swim()

class Pigeon(Animal, FlyAbility):
    def eat(self):
        return "eats, "

    def walk(self):
        return "walks, "

    def fly(self):
        return "Fly, "

    def __repr__(self):
        return self.eat() + self.walk() + self.fly()


@app.route('/get_animal_properties/cat')
def get_animal_properties_cat():
    animal_properties = Cat()
    return render_template('animal_properties.html', animal_properties=animal_properties,
title='Solid Principle')


@app.route('/get_animal_properties/duck')
def get_animal_properties_duck():
    animal_properties = Duck()
    return render_template('animal_properties.html', animal_properties=animal_properties,
title='Solid Principle')


@app.route('/get_animal_properties/pigeon')
def get_animal_properties_pigeon():
    animal_properties = Pigeon()
    return render_template('animal_properties.html', animal_properties=animal_properties,
title='Solid Principle')
```

5)DI
Incorrect Code is being used in our repos in most of the places breaking this principle.

For example:-
https://github.com/plangrid/admin-api-2/blob/92f7f22ebb4c8eace8e346f25dd7a20ac1643fc0/service/clients/cacahuete_client.py#L30

Correct code would be in this way:-
https://github.com/rgunkar/Solid-Principles/blob/main/DI_Correct_Code.py