You are given an example of an inheritance hierarchy, which results in Cartesian-product duplication.

Please refactor this hierarchy, giving the base class Shape  a constructor that takes an interface Renderer  defined as

class Renderer(ABC):

@property

def what\_to\_render\_as(self):

return None

as well as VectorRenderer  and RasterRenderer  classes. Each inheritor of the Shape  abstract class should have a constructor that takes a Renderer  such that, subsequently, each constructed object's \_\_str\_\_()  operates correctly, for example,

str(Triangle(RasterRenderer()) # returns "Drawing Triangle as pixels"