### 1. Создать класс TrafficLight (светофор) и определить у него один атрибут color (цвет) и метод running (запуск). Атрибут реализовать как приватный. В рамках метода реализовать переключение светофора в режимы: красный, желтый, зеленый. Продолжительность первого состояния (красный) составляет 7 секунд, второго (желтый) — 2 секунды, третьего (зеленый) — на ваше усмотрение. Переключение между режимами должно осуществляться только в указанном порядке (красный, желтый, зеленый). Проверить работу примера, создав экземпляр и вызвав описанный метод.

### Задачу можно усложнить, реализовав проверку порядка режимов, и при его нарушении выводить соответствующее сообщение и завершать скрипт.

**from** time **import** sleep  
**from** datetime **import** datetime **as** dt  
  
**class** TrafficLight:  
 \_states = {**'red'**: 7, **'yellow'**: 2, **'green'**: 10}  
 \_color = **''  
  
 def** running(self):  
 **for** color, sw\_time **in** self.\_states.items():  
 self.\_color = color  
 start\_state\_time = dt.now()  
  
 print(**f"Горит {**self.\_color**} светофор"  
 f" {**sw\_time**} секунды"**)  
  
 sleep(sw\_time)  
  
 print(**f"Переключение светофора с {**self.\_color**} после "   
 f"{**(dt.now() - start\_state\_time).seconds**} секунд ожидания"**)  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 tl = TrafficLight()  
 tl.running()

### 2. Реализовать класс Road (дорога), в котором определить атрибуты: length (длина), width (ширина). Значения данных атрибутов должны передаваться при создании экземпляра класса. Атрибуты сделать защищенными. Определить метод расчета массы асфальта, необходимого для покрытия всего дорожного полотна. Использовать формулу: длина\*ширина\*масса асфальта для покрытия одного кв метра дороги асфальтом, толщиной в 1 см\*число см толщины полотна. Проверить работу метода.

**Например:** 20м\*5000м\*25кг\*5см = 12500 т

**class** Road:  
  
 **def** \_\_init\_\_(self, length, width):  
 self.\_length = length  
 self.\_width = width  
 self.weight = 25  
 self.height = 5  
  
 **def** asphalt\_mass(self):  
 asphalt\_mass = self.\_length \* self.\_width \* self.weight \* self.height / 1000  
 print(**f'Для укладки {**self.\_length**} метров дорожного полотна неободимо {**round(asphalt\_mass)**} тонн асфальта'**)  
  
  
r = Road(5000, 20)  
r.asphalt\_mass()

### 3. Реализовать базовый класс Worker (работник), в котором определить атрибуты: name, surname, position (должность), income (доход). Последний атрибут должен быть защищенным и ссылаться на словарь, содержащий элементы: оклад и премия, например, {"wage": wage, "bonus": bonus}. Создать класс Position (должность) на базе класса Worker. В классе Position реализовать методы получения полного имени сотрудника (get\_full\_name) и дохода с учетом премии (get\_total\_income). Проверить работу примера на реальных данных (создать экземпляры класса Position, передать данные, проверить значения атрибутов, вызвать методы экземпляров).

**class** Worker:  
 **def** \_\_init\_\_(self, name: str, surname: str, position: str, wage: int, bonus: int):  
 self.name = name  
 self.surname = surname  
 self.position = position  
 self.\_income = {**'wage'**: wage, **'bonus'**: bonus}  
  
  
**class** Position(Worker):  
 **def** get\_full\_name(self):  
 **return f"{**self.name**} {**self.surname**}"  
  
 def** get\_total\_income(self):  
 **return** sum(self.\_income.values())  
  
  
Worker = Position(**'Петр'**, **'Иванов'**, **'Консультант'**, 90000, 20000)  
print(Worker.get\_full\_name())  
print(Worker.position)  
print(Worker.get\_total\_income())

### 4.Реализуйте базовый класс Car. У данного класса должны быть следующие атрибуты: speed, color, name, is\_police (булево). А также методы: go, stop, turn(direction), которые должны сообщать, что машина поехала, остановилась, повернула (куда). Опишите несколько дочерних классов: TownCar, SportCar, WorkCar, PoliceCar. Добавьте в базовый класс метод show\_speed, который должен показывать текущую скорость автомобиля. Для классов TownCar и WorkCar переопределите метод show\_speed. При значении скорости свыше 60 (TownCar) и 40 (WorkCar) должно выводиться сообщение о превышении скорости.

Создайте экземпляры классов, передайте значения атрибутов. Выполните доступ к атрибутам, выведите результат. Выполните вызов методов и также покажите результат.

**class** Car:  
 **def** \_\_init\_\_(self, color: str, name: str, is\_police: bool):  
 self.speed = 0  
 self.color = color  
 self.name = name  
 self.is\_police = is\_police  
  
 **def** go(self, speed):  
 self.speed = speed  
 print(**f'Разгоняемся до {**speed**} км/ч'**)  
  
 **def** stop(self):  
 self.speed = 0  
 print(**'Останавливаемся'**)  
  
 **def** turn(self, direction: str):  
 **if** direction == **'налево'**:  
 print(**f"Повернули {**direction**}"**)  
 **elif** direction ==**'направо'**:  
 print(**f"Повернули {**direction**}"**)  
  
 **def** show\_speed(self):  
 print(**f'Скорость {**self.speed**} км/ч'**)  
  
  
**class** TownCar(Car):  
 **def** \_\_init\_\_(self, color: str, name: str):  
 self.speed = 0  
 self.color = color  
 self.name = name  
 self.is\_police = **False  
  
 def** show\_speed(self):  
 **if** self.speed > 60:  
 print(**f'Внимание! Превышение скорости {**self.speed**} км/ч'**)  
 **else**:  
 print(**f'Скорость {**self.speed**} км/ч'**)  
  
  
**class** SportCar(Car):  
 **def** \_\_init\_\_(self, color: str, name: str):  
 self.speed = 0  
 self.color = color  
 self.name = name  
 self.is\_police = **False  
  
  
class** WorkCar(Car):  
 **def** \_\_init\_\_(self, color: str, name: str):  
 self.speed = 0  
 self.color = color  
 self.name = name  
 self.is\_police = **False  
  
 def** show\_speed(self):  
 **if** self.speed > 40:  
 print(**f'Внимание! Превышение скорости {**self.speed**} км/ч'**)  
 **else**:  
 print(**f'Скорость {**self.speed**} км/ч'**)  
  
  
**class** PoliceCar(Car):  
 **def** \_\_init\_\_(self, color: str, name: str):  
 self.speed = 0  
 self.color = color  
 self.name = name  
 self.is\_police = **True  
  
  
def** test\_drive(vehicle):  
 print(**f"Тест-драйв {'полицейского' if** vehicle.is\_police **else 'гражданского'} автомобиля {**vehicle.name**}, цвет {**vehicle.color**}"**)  
 vehicle.go(40)  
 vehicle.show\_speed()  
 vehicle.turn(**'направо'**)  
 vehicle.stop()  
 vehicle.show\_speed()  
 vehicle.turn(**'налево'**)  
 vehicle.go(60)  
 vehicle.show\_speed()  
 vehicle.go(120)  
 vehicle.show\_speed()  
 vehicle.stop()  
 print(**"Тест-драйв окончен"**, end=**"\n\n"**)  
  
  
car\_1 = Car(**'белый'**, **'Toyota Camry'**, **False**)  
test\_drive(car\_1)  
  
car\_2 = TownCar(**'серый'**, **'Volkswagen Polo'**)  
test\_drive(car\_2)  
  
car\_3 = SportCar(**'красный'**, **'Ferrari 488 GTB'**)  
test\_drive(car\_3)  
  
car\_4 = WorkCar(**'черный'**, **'Reno Logan'**)  
test\_drive(car\_4)  
  
car\_5 = PoliceCar(**'синий'**, **'Ford Mondeo'**)  
test\_drive(car\_5)

### 5. Реализовать класс Stationery (канцелярская принадлежность). Определить в нем атрибут title (название) и метод draw (отрисовка). Метод выводит сообщение “Запуск отрисовки.” Создать три дочерних класса Pen (ручка), Pencil (карандаш), Handle (маркер). В каждом из классов реализовать переопределение метода draw. Для каждого из классов метод должен выводить уникальное сообщение. Создать экземпляры классов и проверить, что выведет описанный метод для каждого экземпляра.

**class** Stationery:  
 **def** \_\_init\_\_(self, title):  
 self.title = title  
 **def** draw(self):  
 **return f'Запуск отрисовки'  
  
class** Pen(Stationery):  
 **def** draw(self):  
 **return f'Запуск отрисовки {**self.title**}'  
  
class** Pencil(Stationery):  
 **def** draw(self):  
 **return f'Запуск отрисовки {**self.title**}'  
  
class** Handle(Stationery):  
 **def** draw(self):  
 **return f'Запуск отрисовки {**self.title**}'**pen = Pen(**'ручкой'**)  
print(pen.draw())  
pencil = Pencil(**'карандашем'**)  
print(pencil.draw())  
handle = Handle(**'маркером'**)  
print(handle.draw())