Задание 1.
Исходные данные:
Создать класс TrafficLight (светофор).
• определить у него один атрибут color (цвет) и метод running (запуск);
• атрибут реализовать как приватный;
• в рамках метода реализовать переключение светофора в режимы: красный, жёлтый, зелёный;
• продолжительность первого состояния (красный) составляет 7 секунд, второго (жёлтый) — 2 секунды, третьего (зелёный) — на ваше усмотрение;
• переключение между режимами должно осуществляться только в указанном порядке (красный, жёлтый, зелёный);
• проверить работу примера, создав экземпляр и вызвав описанный метод.
Задачу можно усложнить, реализовав проверку порядка режимов. При его нарушении выводить соответствующее сообщение и завершать скрипт.
Решение:

```
class TrafficLight:
    def __init__(self, __color):
        self.color = __color
    def running(self):
        self.color = __color
        __color = red
        __color = ellow
        __color = green
    def running_start(self):
        self.color = __color
        __color = red
        a = 7
        # seconds
        b = running_start = a
        return b
    def running_flag(self):
        self.color = __color
        __color = red
        c = running_flag_1(__color = red)
        __color = ellow
        d = running_flag_2(__color = ellow)
        f = running_flag_1 in running_flag_2
        return f
    def running_second(self):
        self.color = __color
        __color = ellow
        e = 2
        # seconds
        g = running_second(__color = ellow)
        return g
    def running_flag1(self):
        self.color = __color
        __color = ellow
        h = running_flag_3(__color = ellow)
        __color = green
        n = running_flag_4(__color = green)
        r = running_flag_3 in running_flag_4
```

```
        return r
    def running_treid(self):
        self.color = __green
        __color = green
        k = 51
# seconds
        l = running_treid = k
        return l


class TrafficLight1(TrafficLight):
    def running_start(self):
        self.color = __color
```

Задание 2.
Исходные данные:
Реализовать класс Road (дорога).

- определить атрибуты: length (длина), width (ширина);
- значения атрибутов должны передаваться при создании экземпляра класса;
- атрибуты сделать защищёнными;
- определить метод расчёта массы асфальта, необходимого для покрытия всей дороги;
- 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: 20 м*5000 м*25 кг*5 см = 12500 т.
Решение:

```
class Road:
    def __init__(self, length, width):
        self.length = _length
        self.winth = _winth
    def trassa1(self):
        self.length = _length
        _length = kilometrs
    def trassa2(self):
        self.winth = _winth
        _winth = metrs

class Road1(Road):
    def trassa1(self):
        self.lingth = _length

class Road2(Road):
    def trassa2(self):
        self.winth = _winth

a = 20
# winth, metrs.
b = 5000
# length, metrs.
c = 25
# kilogramm.
d = 5
```

```
# santimetrs.
def my_func(a, b, c, d):
    try:
        a = int(input("Введите значение ширины полотна: "))
        b = int(input("Введите значение длины полотна: "))
        c = int(input("Введите массу асфальта в килограммах: "))
        d = int(input("Введите толщину полотна в сантиметрах: "))
    except zerodivisionerror:
        return
    s = a * b * c * d
    return s

print(my_func(a, b, c, d))
```

Задание 3.
Исходные данные:
Реализовать базовый класс Worker (работник).
• определить атрибуты: name, surname, position (должность), income (доход);
• последний атрибут должен быть защищённым 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 учётом премии (get_total_income);
• проверить работу примера на реальных данных: создать экземпляры класса Position, передать данные, проверить значения атрибутов, вызвать методы экземпляров.
Решение:

```
class Worker:
    def __init__(self, name, surname, position, incone):
        self.name = name
        self.surname = surname
        self.position = position
        self.incone = _incone

class Position(Worker):
    def __init__(self, get_full_name, get_total_incone):
        self.get_full_name = get_full_name
        self.get_total_incone = get_total_incone
    def position1(self):
        self.get_full_name = get_full_name
        self.name = name
        self.surname = surname
    def position2(self):
        self.get_total_incone = get_total_incone
        self.position = position
        self.incone = _incone
        self.incone_full = {}
    def incone_full(self, incone_full):
        self.incone_full.append(wage)
        self.incone_full.append(bonus)


a = 'name'
```

```
b = 'surname'
def my_fullname(a, b):
    try:
        a = input("Введите Ваше имя: ")
        b = input("Введите Вашу фамилию: ")
    except valueerror:
        return
    c = a, b
    return c

print(my_fullname(a, b))


x = 50000
y = 25000
def my_total(x, y):
    try:
        x = int(input("Введите сумму дохода: "))
        y = int(input("Введите сумму бонуса: "))
    except zerodivisionerror:
        return
    z = x + y
    return z

print(my_total(x, y))
```

Задание 4.
Исходные данные:
Реализуйте базовый класс Car.
• у 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. Вызовите методы и покажите результат.
Решение:

```
class Car:
    def __init__(self, speed, color, name, is_police, show_speed):
        self.speed = speed
        self.color = color
        self.name = name
        self.is_police = is_police
        self.show_speed = show_speed

def show_speed(self):
        self.show_speed = show_speed
        self.speed = speed
```

```
def go(self):
        self.go = go
        self.show_speed = show_speed
        self.speed = speed

def stop(self):
        self.stop = stop
        self.show_speed = show_speed
        self.speed = speed

def turn_direction(self):
        self.turn_direction = turn_direction
        self.show_speed = show_speed
        self.speed = speed

class SportCar(Car):
    def __init__(self, speed, color, name, is_police, show_speed):
        self.speed = speed
        self.color = color
        self.name = name
        self.is_police = is_police
        self.show_speed = speed

def show_speed(self):
        self.show_speed = show_speed
        self.speed = speed
        s > 300 == stop

def color(self):
        self.color = color
        color = red

def name(self):
        self.name = name
        name = subaru

def go(self):
        self.go = go
        self.name = name
        name = subaru
        go == 80

def turn_direction(self):
        self.turn_direction = turn_direction
        self.show_speed = show_speed
        self.speed = speed
        turn_direction = right
        right == 170
        turn_direction = left
        left ==130
        turn_direction = back
```

```
        back == 90
        turn_direction = forward
        forward ==270

def stop(self):
        self.stop = stop
        self.show_speed = show_speed
        self.speed = speed
        s > 300 == stop

class PoliceCar(Car):
    def __init__(self, speed, color, name, is_police, show_speed):
        self.speed = speed
        self.color = color
        self.name = name
        self.is_police = is_police
        self.show_speed = show_speed

def show_speed(self):
        self.show_speed = show_speed
        self.speed = speed
        s > 200 == stop

def color(self):
        self.color = color
        color = black

def name(self):
        self.name = name
        name = bmv

def go(self):
        self.go = go
        self.show_speed = show_speed
        self.speed = speed
        self.name = name
        name = bmv
        go ==40

def turn_direction(self):
        self.turn_direction = turn_direction
        self.show_speed = show_speed
        self.speed = speed
        turn_direction = right
        right = 170
        turn_direction = left
        left = 120
        turn_direction = back
        back = 60
        turn_direction = forward
        forward = 190
```

```
def stop(self):
        self.stop = stop
        self.show_speed = show_speed
        self.speed = speed
        s >200 == stop

class TownCar(Car):
    def __init__(self, speed, color, name, is_police, show_speed):
        self.speed = speed
        self.color = color
        self.name = name
        self.is_police = is_police
        self.show_speed = show_speed

def show_speed(self):
        self.show_speed = show_speed
        self.speed = speed
        s > 120 == stop

def color(self):
        self.color = color
        color = grey

def name(self):
        self.name = name
        name = mazda

def go(self):
        self.go = go
        self.show_speed = show_speed
        self.speed = speed
        self.name = name
        name = mazda
        go = 20

def turn_direction(self):
        self.turn_direction = turn_direction
        self.show_speed = show_speed
        self.speed = speed
        turn_direction = right
        right = 100
        turn_direction = left
        left = 90
        turn_direction = back
        back = 50
        turn_direction = forward
        forward = 115

def stop(self):
        self.stop = stop
        self.show_speed = show_speed
        self.speed = speed
```

```
        s > 120 == stop

class WorkCar(Car):
    def __init__(self, speed, color, name, is_police, show_speed):
        self.speed = speed
        self.color = color
        self.name = name
        self.is_police = is_police
        self.show_speed = show_speed

def show_speed(self):
        self.show_speed = show_speed
        self.speed = speed
        s > 90 == stop

def color(self):
        self.color = color
        color = green

def name(self):
        self.name = name
        name = kraz

def go(self):
        self.go = go
        self.show_speed = speed
        self.speed = speed
        self.name = name
        name = kraz
        go = 10

def turn_direction(self):
        self.turn_direction = turn_direction
        self.show_speed = show_speed
        turn_direction = right
        right = 70
        turn_direction = left
        left = 50
        turn_direction = back
        back = 30
        turn_direction = forward
        forward = 85

def stop(self):
        self.stop = stop
        self.show_speed = show_speed
        self.speed = speed
        s > 90 == stop

a = 50
b = 60
c = 70
```

```
def my_speed(a, b, c):
    try:
        a = int(input("Введите текущую скорость городской машины: "))
        if a>=60:
            b = input("Вы привысили допустимую скорость: ")
        if a==60:
            c = input("Вы находитесь на максимально разрешенной скорости в данной местности:
")
        if a<=60:
            d = input("Вы едите на допустимой скорости: ")
    except zerodivisionerror:
        return
    return a

print(my_speed(a, b,c))

x = 30
y = 40
z = 50
def my_speed1(x, y, z):
    try:
        f = int(input("Введите текущую скорость рабочей машины: "))
        if f>=40:
            g = input("Вы привысили допустимую скорость: ")
        if f==40:
            h = input("Вы двигаетесь на максимально разрешённой скорости: ")
        if f<=40:
            k = input("Вы двигаетесь на разрешённой скорости: ")
    except zerodivisionerror:
        return
    return f

print(my_speed1(x, y, z))
```

Задание 5.
Исходные данные:
Реализовать класс Stationery (канцелярская принадлежность).
• определить в нём атрибут title (название) и метод draw (отрисовка). Метод выводит сообщение «Запуск отрисовки»;
• создать три дочерних класса Pen (ручка), Pencil (карандаш), Handle (маркер);
• в каждом классе реализовать переопределение метода draw. Для каждого класса метод должен выводить уникальное сообщение;
• создать экземпляры классов и проверить, что выведет описанный метод для каждого экземпляра.
Решение:

```
class Stationery:
    def __init__(self, title):
        self.title = title

def draw(self):
        self.draw = draw
```

```
a = "name"
def my_name(a):
    try:
        a = input("Выбирите инструмент отрисовки(ручка, карандаш, маркер): ")
    except valueerror:
        return
    return a

print(my_name(a))

class Pen(Stationery):
    def __init__(self, title):
        self.title = title

def draw(self):
        self.draw = draw

b = "name"
def my_name1(b):
    try:
        b = input("Ручкой мы ресуем основные линии и обозначения на чертеже: ")
    except valueerror:
        return
    return b

print(my_name1(b))

class Pencil(Stationery):
    def __init__(self, title):
        self.title = title

def draw(self):
        self.draw = draw

c = "name"
def my_name2(c):
    try:
        c = input("Карандашом наносим мелкие детали на чертеже: ")
    except valueerror:
        return
    return c

print(my_name2(c))

class Handle(Stationery):
    def __init__(self,title):
        self.title = title

def draw(self):
        self.draw = draw

d = "name"
```

```
def my_name3(d):
    try:
        d = input("Маркером мы наносим самые важные детали на чертеже: ")
    except valueerror:
        return
    return d

print(my_name3(d))
```