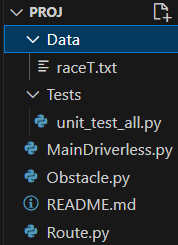
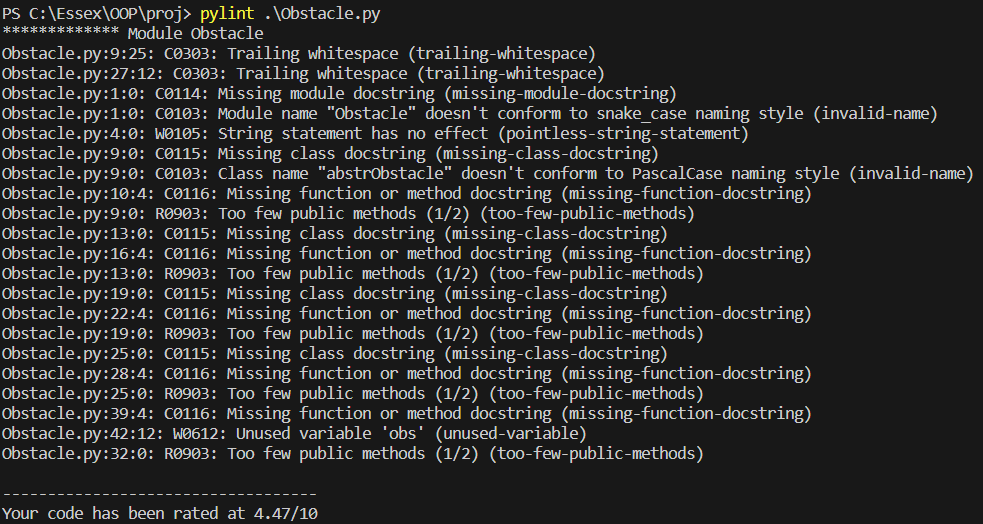
**Question:**Refer to the Mertz (2019) resource. Use some Python code which you have developed in the past (Code in appendix) and apply at least 3 of the strategies presented at the source to improve its quality

**Strategy 1** Directory structure:

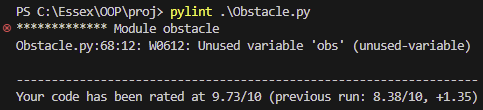


**Strategy 2** Using PyLinters.

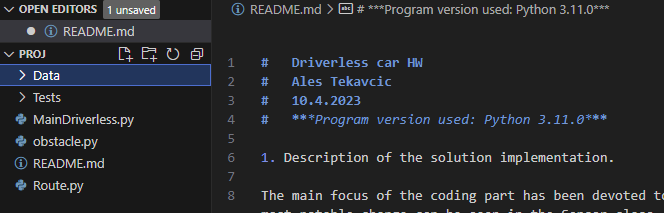
Initial score



After the recommendations made by Pylinters:



**Strategy 3** Proper Documentation:



**Appendix** (Initial Code)

import random

from abc import ABC

"""

The first four classes are called, when generate method of the Obstacle class is invoked.

This first three classes (omitting the first class) depict the types of the obstacles.

"""

class abstrObstacle(ABC):

    def type(self):

        pass

class Cyclist(ABC):

    def \_\_init\_\_(self):

        pass

    def type(self):

        return type(self).\_\_name\_\_

class Pedestrian(ABC):

    def \_\_init\_\_(self):

        pass

    def type(self):

        return type(self).\_\_name\_\_

class Vehicle(ABC):

    def \_\_init\_\_(self):

        pass

    def type(self):

        return type(self).\_\_name\_\_

class Obstacle():

    """

    Class that generates random Obstacle objects in random sequence.

    """

    def \_\_init\_\_(self):

        pass

    def generate(self):

        number\_of\_obstacles = random.randint(0, 2)

        for obs in range(number\_of\_obstacles):

            obstacle\_type = random.randint(1, 3)

            if obstacle\_type == 1:

                obstacle = Pedestrian()

                yield obstacle

            elif obstacle\_type == 2:

                obstacle = Cyclist()

                yield obstacle

            elif obstacle\_type == 3:

                obstacle = Vehicle()

                yield obstacle

            else:

                pass

#o = Obstacle()

#for i in o.generate():

#    print(i.type())