

Aufgabe 3 - Unit Tests erstellen in PyCharm

Ziel

Sie sind in der Lage, mit PyCharm Unit Tests auf der Basis von `pytest` zu erstellen und auszuführen.

Auftrag

1. Öffnen Sie die nachfolgende Datei `car.py` via PyCharm.
2. Selektieren Sie die Methode `brake` und klicken Sie `<ctrl+shift+t>` um einen Unit Test mit der Methode `test_car_brake` dafür zu erstellen.
3. Ergänzen Sie den Tests mit den drei Zeilen für Arrange, Act und Assert gemäss der Datei `car_test.py` unten.
4. Führen Sie den Test aus. Orientieren Sie sich dafür am [Pytest Tutorial](#).
5. Halten Sie schriftlich fest, unter welchen Umständen ein Unit Test für die Testausführung gefunden wird. Lesen Sie dazu den Abschnitt über [Conventions for Python test discovery](#). Kleiner Tipp: Beachten Sie Methoden- und Dateinamen des Unit Tests ☐
6. Es gibt bestimmte Dinge, wie z.B. das Erstellen des zu testenden Objekts, welche sich in jedem Test wiederholen. Um redundanten Code zu vermeiden, können `@pytest.fixture` Annotations verwendet werden. Lesen Sie den Abschnitt [Use fixtures](#) und bauen Sie den Unit Test entsprechend um.
7. Optional: Wie Sie einen Test parametrisieren, so dass mehrere Testfälle mit derselben Testmethode ausgeführt werden, lesen Sie im Abschnitt [Apply parametrization](#) nach. Ergänzen Sie Ihren Test, so dass er für die Geschwindigkeiten 45, 50, 55 und 100 ausgeführt wird.

Abgabe

Geben Sie Ihr Protokoll als PDF-Datei in Moodle ab.

Code

`car.py`

```
class Car:

    def __init__(self, speed=0):
        self.speed = speed
        self.odometer = 0
        self.time = 0

    def say_state(self):
        print("I'm going {} kph!".format(self.speed))

    def accelerate(self):
```

```
        self.speed += 5

    def brake(self):
        if self.speed < 5:
            self.speed = 0
        else:
            self.speed -= 5

    def step(self):
        self.odometer += self.speed
        self.time += 1

    def average_speed(self):
        if self.time != 0:
            return self.odometer / self.time
        else:
            pass

if __name__ == '__main__':

    my_car = Car()
    print("I'm a car!")
    while True:
        action = input("What should I do? [A]ccelerate, [B]rake, "
                        "show [O]dometer, or show average [S]peed?".upper())
        if action not in "ABOS" or len(action) != 1:
            print("I don't know how to do that")
            continue
        if action == 'A':
            my_car.accelerate()
        elif action == 'B':
            my_car.brake()
        elif action == 'O':
            print("The car has driven {}
kilometers".format(my_car.odometer))
        elif action == 'S':
            print("The car's average speed was {}
kph".format(my_car.average_speed()))
        my_car.step()
        my_car.say_state()
```

[car_test.py](#)

```
from car import Car

def test_car_brake():
```

```
testee = Car(50)
testee.brake()
assert testee.speed == 45
```

From:

<https://wiki.bzz.ch/> - **BZZ - Modulwiki**

Permanent link:

https://wiki.bzz.ch/modul/archiv/m320/learningunits/lu90/aufgaben/lu4-aufgabe_3

Last update: **2026/08/30 14:06**

