LU11.A03 - 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. Nehmen Sie die folgende Github-Classroom Aufgabe an: Aufgabe M431_UnitTest oder öffnen Sie die nachfolgende Datei `calculator.py` via PyCharm.
2. Selektieren Sie die Methode `add` und klicken Sie <ctrl+shift+t> um einen Unit Test mit der Methode `test_add` dafür zu erstellen.
3. Ergänzen Sie den Tests mit den drei Zeilen für Arrange, Act und Assert gemäss der Datei `test_calculator.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. Erstellen Sie nun Testfälle für `sub`, `div`, `mult`, `root` und `sqr`
7. **Optional:** Wie Sie einen Test parametrisieren, sodass mehrere Testfälle mit derselben Testmethode ausgeführt werden, lesen Sie im Abschnitt Parametrizing fixtures and test functions nach. Ergänzen Sie Ihren Test so, dass für `add`, `sub`, `div`, `mult` die Rechnungen mit den Zahlenwerten (10,5), (-10,5), (10,-5) und (-10,-5) ausgeführt werden.

Abgabe

- `Commiten` und `Pushen` Sie Ihre Lösung auf Github falls Sie mit Github arbeiten
- Geben Sie einen Printscreen des Testberichtes auf Moodle ab.

Code

calculator.py

```python
import math

def add(num1, num2):
    """
    Addition of two numbers
    :param num1: number 1 for calculation
    :param num2: number 2 for calculation
    :return: result of calculation
    """
    return num1 + num2
```

```
def sub(num1, num2):
    """
    Substracts two numbers
    :param num1: number 1 for calculation
    :param num2: number 2 for calculation
    :return: result of calculation
    """
    return num1 - num2


def mul(num1, num2):
    """
    multiply of two numbers
    :param num1: number 1 for calculation
    :param num2: number 2 for calculation
    :return: result of calculation
    """
    return num1 * num2


def div(num1, num2):
    """
    division of two numbers
    :param num1: number 1 for calculation
    :param num2: number 2 for calculation
    :return: result of calculation
    """
    return num1 / num2


def sqr(num1):
    """
    Squares two numbers
    :param num1: number to square
    :return: result of calculation
    """
    return num1*num1


def root(num1):
    """
    return the root of a number
    :param num1: number to get the root of
    :return: result of calculation
    """
    return math.sqrt(num1)


def main():
```

```
    result_add = add(5, 5.5)
    result_div = div(10,3)
    result_mul = mul(3,3)
    result_sub = sub(10,4.4)
    result_sqr = sqr(23)
    result_root = root(81)

    print(result_add)
    print(result_div)
    print(result_mul)
    print(result_sub)
    print(result_sqr)
    print(result_root)

if __name__ == '__main__':
    main()
```

test_calculator.py

```
import calculator


def test_add():
    calculator.add(10, 5)
    assert calculator.add(10, 5) == 15
```