

# Datenschutz/-sicherheit

## und die Notwendigkeit von Tests

von Kevin Böhme und Rico Ukro

*- Aufgaben -*

# Aufgabe 1 - Unit tests - Schaltjahr

Ihr Kunde „Karl“ hat für sein Produkt „Karl's Karlender“, bei Ihnen angefragt, die fehlende Implementierung für die Berechnung der Schaltjahre zu übernehmen:

1. Entwickeln Sie Unit-Tests für eine Funktion `is_leap_year(year: int) -> bool`, die überprüft, ob ein gegebenes Jahr ein Schaltjahr ist
2. Erstellen Sie mindestens 5 Testfälle, die verschiedene Szenarien abdecken (z.B. reguläre Jahre, Schaltjahre, Grenzfälle).

*Hinweise: Nutzen Sie die Bibliothek `unittest`*

3. Implementieren Sie die Funktion `is_leap_year(year: int) -> bool`

*Negative Kalenderjahre sind nicht vorgesehen.*

# Aufgabe 1 - Unit tests - Schaltjahr

Als kleine Starthilfe:

- *Template:* `test_leapyear.py`
- *Template:* `leapyear.py`

## Aufgabe 2 - Unit/Integration tests - Sockenversand

Für einen renommierten Online-Sockenversand soll die bisherige stark veraltete und fehleranfällige MS-SQL-Server Implementierung durch ein modernes Python Backend ersetzt werden.

Ihre Aufgabe ist es:

1. Unit-Tests für die Funktionalität des Sockenversands zu entwickeln
2. Implementierung der Funktionalität des Sockenversands zu erstellen
3. Integrationstests für die Funktionalität des Sockenversands zu entwickeln

## Aufgabe 2 - Unit/Integration tests - Sockenversand

Anforderungen:

- Klasse `SockStore` : Verwaltet den Bestand an Socken und stellt die folgenden Schnittstellen bereit
- Methode `search(self, color: str) -> int` : Gibt die Anzahl der Socken einer bestimmten Farbe zurück
- Methode `buy_sock(self, color)` : Kauft ein Paar Socken einer bestimmten Farbe und gibt die gekaufte Farbe zurück
- Methode `add_sock(self, color: str, quantity: int)` : Fügt Socken einer bestimmten Farbe und Menge hinzu

# Aufgabe 2 - Unit/Integration tests - Sockenversand

Als kleine Starthilfe:

- *Template:* `test_sockstore_unit.py`
- *Template:* `sockstore.py`
- *Template:* `test_sockstore_integration.py`

## Aufgabe 3 - System tests - Blog

Einer Ihrer Kunden, die Firma „Bloggify“, hat Sie beauftragt, die System tests für ihre Blogging-Plattform zu entwickeln. Das Plattform-Backend ist mittels REST-API über <https://jsonplaceholder.typicode.com/posts> erreichbar.

Ihre Aufgabe ist es:

1. System tests für die Blogging-Plattform zu entwickeln
2. Schwachstellen durch System tests in der Blogging-Plattform zu finden, um diese dem Kunden zu melden

# Aufgabe 3 - System tests - Blog

Guide für die API:

- <https://jsonplaceholder.typicode.com/guide>

Als kleine Starthilfe:

- *Template:* `test_blog_system.py`