# Tutorium 04 - 2024-11-14

Blatt 04, assert, top-level statements, Blatt 05

# Vorab - LFG Rechnernetze

- Handynummer: +49-160-8080619
- Mail: nils@narl.io

# Vorab - Ausfall Tutorium 12.12.

- Tutorium am 12.12. fällt aus
- Bitte Ersatztutorium auf der Vorlesungswebsite aussuchen
- Ihr könnt da auch Vorrechnen/Anwesenheitspunkte bekommen

# Blatt 04

# `assert` - Was ist das?

- Eine Annahme über einen Zustand
- Wird gerne zum Testen verwendet
- Wenn die Annahme nicht stimmt, dann *stürzt* euer Programm ab
- Wird manchmal als Invariante benutzt

# `assert` als Invariante

- Ihr müsst euch sicher sein, dass die Invariante immer stimmt
- z.B. wenn ihr auf dem Blatt mit nur korrekten Eingaben rechnen könnt

```
def input_number() → int:
    num = input("Please enter a whole number:\n>")
    assert num.isnumeric()
    return int(num)
```

# `assert` als Schleifeninvariante

- Typischeres Beispiel für Invariante in Informatik ist die Schleifeninvariante
- Damit können Beweise über die Korrektheit des Algorithmus gemacht werden

```python
def multiply(a: int, b: int) → int:
    x, y, p = a, b, 0
    # Die Schleifeninvariante gilt vor der Schleife
    assert (x * y) + p == a * b
    while x > 0:
        # Die Schleifeninvariante gilt an jedem Schleifenanfang
        assert (x * y) + p == a * b
        p = p + y
        # Zwischendrin muss diese nicht gelten
        x = x - 1
        # Die Schleifeninvariante muss am Ende jedes Durchlaufs gelten
        assert (x * y) + p == a * b
    # Die Schleifeninvariante muss ganz am Ende gelten
    assert (x * y) + p == a * b
    return p
```

# `assert` als Top-Level Statement

- es ist aber sehr untypisch seinen ganzen Code mit Invarianten voll zu machen
- wir programmieren ja nur korrekte Algorithmen ;)
- stattdessen wird assert gerne zum Testen verwendet
- Hier muss dann `assert` in die `if __name__ == "__main__":`
- Mit genug guten Tests können wir auch davon ausgehen, dass der Algorithmus korrekt ist

```python
if __name__ == "__main__":
    assert multiply(3, 4) == 12
    assert multiply(0, 1) == 0
    assert multiply(1000, 0) == 0
    assert multiply(100, 1000) == 100000
```

# Top-Level Statements

Wann benutzen wir `if __name__ == "__main__":`?

# Wann `if __name__ == "__main__":`?

- eigentlich immer wenn eine der Folgenden Funktionen benutzt:
  - `assert`
  - `print( ... )`
  - `input( ... )`
- außer es ist anders gefordert! Das steht dann aber explizit dabei

# Wann `input()/print()`?

- Es gibt zwei Aufgaben Typen
  i. Implementieren Sie Funktion *f* die `f( ... ) → ...` tut.
     - meistens gibt es `assert` als Beispiele
  ii. Implementieren Sie ein Programm das von Benutzer `x` fordert und `y` ausgibt.
     - es gibt immer eine Beispieleingabe/-ausgabe

# Beispiel Type I

**Schreiben Sie eine Funktion `multiply` die zwei Zahlen multipliziert**

```
def multiply(a: int, b: int) → int:
    x, y, p = a, b, 0
    while x > 0:
        p = p + y
        x = x - 1
    return p
```

**Hier kann man z.B. auch die Tests einfach in die main schreiben:**

```
if __name__ == "__main__":
    assert multiply(3, 4) == 12
    assert multiply(0, 1) == 0
    assert multiply(1000, 0) == 0
    assert multiply(100, 1000) == 100000
```

# Beispiel Type II

Schreiben Sie ein Programm was zwei Zahlen nimmt und das Ergebnis ausgibt, das soll solange geschehen bis der Nutzer `exit` schreibt

# Blatt 05