Autor:
Felix Dehon
Lutz Pfannenschmidt

Einführung in die Programmierung
WS 2025/2026
Prof. Dr. Peter Thiemann

Gedächtnisklausur

19. Februar 2026

Das ist nur eine inoffizielle Gedächtnisklausur. Sie kann Fehler enthalten. Keine Haftung.

- Für die Bearbeitung der Aufgaben haben Sie **150 Minuten** Zeit.
- Es sind keine Hilfsmittel wie Skripte, Bücher, Notizen oder Taschenrechner erlaubt. Des Weiteren sind alle elektronischen Geräte (wie z.B. Handys) auszuschalten. Ausnahme: Fremdsprachige Wörterbücher sind erlaubt.
- **Falls Sie mehrere Lösungsansätze einer Aufgabe erarbeiten, markieren Sie deutlich, welcher gewertet werden soll**. Die geforderten Funktionen dürfen nur einmal in der Abgabe definiert werden, alles andere muss auskommentiert oder gelöscht werden.
- **Verwenden Sie Typannotationen**, um die Typen der Parameter und des Rückgabewertes Ihrer Funktionen anzugeben. Verwenden Sie Typvariablen, falls die Funktion für beliebige Typen gelten soll. Fehlende oder falsche Typannotationen führen zu Punktabzug.
- Bearbeiten Sie die einzelnen Aufgaben in den **vorgegebenen Musterdateien**, z.B. `ex3_strings.py`. Falsch benannte Funktionen werden nicht bewertet. Neu erstellte Dateien werden nicht bewertet.
- **Die Zielfunktionen dürfen ihre Eingaben nicht verändern**, d.h. Methoden wie `list.remove` dürfen nicht auf die Eingaben angewendet werden; es sei denn, die Aufgabenstellung fordert explizit die Eingabe zu verändern.
- Intern darf Ihre Implementierung den vollen Sprachumfang verwenden; **es sei denn, die Aufgabenstellung schließt etwas aus**.
- **Sie dürfen keine Module importieren**. Alle Imports, die benutzt werden dürfen/müssen sind bereits vorgegeben. Zum Lösen der Aufgaben sind keine weiteren Importe/Module notwendig.

Es sind maximal 150 Punkte zu erreichen.

Aufgabe 1 (Warm-Up; 20 Punkte)

(a) (4 Punkte) Definieren Sie eine Funktion `my_len()`, die einen string `a` als Argument entgegennimmt und die länge von `a` zurückgibt.

Sie dürfen die buildtin Methode len(str) nicht verwenden.

(b) (4 Punkte) Implementieren sie die Funktion `fac` die eine natürliche Zahl `n` als Argument entgegennimmt und ihre Fakultät zurückgibt.

Die Fakultät sei definiert als:

$$\text{fac}(n) := \begin{cases} 1 \text{ für } n = 0 \\ n \cdot \text{fac}(n-1) \text{ für } n > 0 \end{cases}$$

Hinweis: Sie dürfen annehmen, dass n größer gleich 0 ist.

(c) (2 Punkte) Definieren Sie einen Typalias MyOptional, der einen Typparameter `T` besitzt und einen Wert beschreibt, der entweder vom Typ `T` oder `None` ist.

(d) (5 Punkte) Gegeben sei die rekursive Definition eines Binärbaumes in Python:

```
@dataclass
class Node[T]:
  mark: T
  left: "Tree[T]"
  right: "Tree[T]"

type Tree[T] = Optional[Node[T]]
```

Ändern sie diese Definition so, dass der Baum eine beliebige, aber endliche Anzahl an Kindern annehmen kann.

(e) (5 Punkte) Implementieren sie eine Funktion die ein Iterierbares Objekt annimmt und einen Iterator zurückgibt, der die Elemente des Eingabeobjektes in der gleichen Reihenfolge wiedergibt.

Aufgabe 2 (Dictionaries; 20 Punkte)
Diese Aufgabe soll Universitätskurse simulieren. Es seien zwei Dictionaries gegeben:

- `courses_rooms` ist ein Dictionary, das jedem Kurs (String) als Key den jeweiligen Raum (String) als Value zuordnet.
- `courses_students` ist ein Dictionary, das jedem Kurs (String) als Key die Menge an Studierenden (String) als Value zuordnet.

(a) (6 Punkte) Implementieren Sie eine Funktion, die `courses_rooms` und `courses_students` annimmt und diejenigen Kurse aus `courses_rooms` zurückgibt, die nicht in `courses_students` vorkommen.

(b) (6 Punkte) Implementieren Sie eine Funktion, die `courses_students` , eine Menge an Studierenden `graduates` sowie eine Mindestmenge `min_students` annimmt. Sie soll das Dictionary `courses_students` *inline* ändern, sodass diejenigen Studierenden entfernt werden, die in `graduates` sind. Die Funktion soll zudem die Menge an Kursen zurückgeben, in denen sich nach der Änderung weniger als `min_students` Studierende befinden.

(c) (8 Punkte) Implementieren Sie eine Funktion, die `courses_rooms` und `courses_students` sowie eine Mindestmenge `min_students` annimmt. Die Funktion soll ein Dictionary zurückgeben, die allen Räumen die Menge derjenigen Kursen zuordnet, die in diesem Raum stattfinden und in denen sich mindestens `min_students` befinden.

Aufgabe 3 (Strings; 10 Punkte)
Diese Aufgabe befasst sich mit CamelCase und snake_case. Diese Formatierungsarten sind so definiert:

- snake_case: Ein String ist im snake_case Format, wenn es nur aus Kleinbuchstaben und Unterstrichen besteht. Wörter sind mit einem Unterstrich getrennt. Beispeil ist `foo`, `foo_bar` oder `ein_ganzer_satz`.
- CamelCase: Strings sind im CamelCase Format, wenn die Wörter zusammengeschrieben sind und, außer im ersten Wort, der erste Buchstabe des Wortes großgeschrieben ist, der Rest kleingeschrieben. Beispeile sind `foo`, `fooBar` oder `einGanzerSatz`.

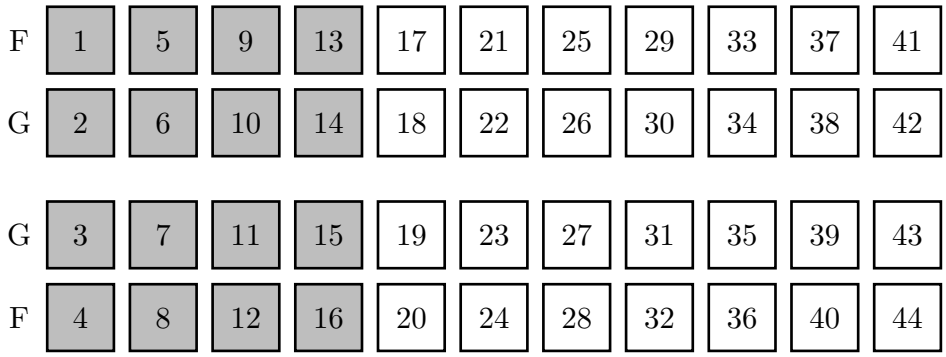
(a) (5 Punke) Implementieren Sie eine Funktion `snake_to_camel`, die einen String im snake_case Format annimmt und diesen Sting in CamelCase format umwandelt und zurückgibt.

```
>>> snake_to_camel("ein_ganzer_satz")
'einGanzerSatz'
```

(b) (5 Punke) Implementieren Sie eine Funktion `camel_to_snake`, die einen String im CamelCase Format annimmt und diesen Sting in snake_case format umwandelt und zurückgibt.

```
>>> snake_to_camel("einGanzerSatz")
'ein_ganzer_satz'
```

Aufgabe 4 (Datenklassen; 30 Punkte)
In dieser Aufgabe werden Sitzplätze in Zügen simuliert. Ein Beispiel einer Sitzplatzverteilung ist unten dargestellt. Die grauen Plätze sind Plätze der Ersten Klasse, die restlichen der Zweiten Klasse. Die äußeren Sitze in Fensterplätze (F), die mittleren die Gangplätze (G).



(a) Implementieren sie die Datenklasse `Seat` die eine Sitznummer `num`, der optionale Name `reversed` (String) der Person, die den Sitz reserviert hat (Wenn niemand den Platz reserviert hat, ist `reverved = None`), und den Wahrheitswert `window`, welcher zeigen soll, ob der Sitz ein Fenstersitz ist, wie im obrigen Diagramm zu sehen ist.

`reserved` soll den Standartwert `None` haben und `window` soll automatisch bei erstellung eines `Seat` anhand von `num` bestimmt werden.

(b) Implementieren sie die Datenklasse `FirstClassSeat`, welche von `Seat` erbt und den Wahrheitswert `desk` speichert, welcher aussagt ob der Sitz einen Tisch hat. Alle Sitze in der ersten Klasse mit num ≤ 8 haben Tische.

(c) Implementieren Sie einen IntEnum `Feedback`, das Rückgabewerte beschreibt. Es soll die Werte `booked`, `alreadyBooked`, und `invalidSeatNumber` geben.

(d) Implementieren Sie eine Datenklasse `Carriage`, die einen Zugwagon beschreibt. Sie soll einen versteckten Parameter `__seat_list` besitzen, der zwar nicht direkt gesetzt wird, sondern durch die zwei Ganzzahlargumente `num_first_class_seats` und `num_second_class_seats` nach dem obigen Schema generiert wird. Der Konstruktor soll mit asserts sicherstellen, dass die beiden Eingaben durch 4 teilbar und nichtnegativ sind.

  (a) Implementieren Sie eine Methode `get_by_name`, die einen Namen annimt und den Seat zurückgibt, der durch diese Person reserviert wurde. Wenn kein Seat mit diesem Namen reserviert wurde, soll die Methode `None` zurückgeben.

  (b) Implementieren Sie eine Methode `get_by_number`, die eine Stizplatzzahl annimt und den Seat mit der Zahl zurückgibt. Wenn kein Seat mit dieser Zahl existiert, soll die Methode `None` zurückgeben.

  (c) Implementieren Sie eine Methode `get_all_free_seats`, die die Menge aller freien Sitzplätze zurückgibt.

  (d) Implementieren Sie eine Methode `book`, die einen Namen `name` und eine Sitzplatzzahl `n` annehmen und den Seat mit der Zahl für diese Person reservieren.

Wenn der Seat schon reserviert ist, soll `alreadyBooked` zurückgegeben werden. Wenn die Reservierung erfolgreich durchgeführt wird, soll `booked` zurückgegeben werden. Gibt es keinen Sitzplatz mit der Nummer, soll `invalidSeatNumber` zurückgegeben werden.

Aufgabe 5 (Endrekursion; 15 Punkte)
In dieser Aufgabe sollen Sie eine Dezimalzahl in Binärdarstellung umwandeln. Die Umwandlung ist rekursiv definiert als:

$$\begin{cases} \text{bin}(0) = \texttt{""} \\ \text{bin}(2n) = \text{bin}(n) + \texttt{"0"} \text{ für } n > 0 \\ \text{bin}(2n+1) = \text{bin}(n) + \texttt{"1"} \end{cases}$$

(a) (5 Punkte) Schreiben Sie eine **rekursive, aber nicht endrekursive** Funktion `to_bin_rec`, die eine Ganzzahl `n` annimmt und die Binärdarstellung von `n`, wie oben beschrieben, als String zurückgibt.

(b) (5 Punkte) Schreiben Sie eine **endrekursive** Funktion `to_bin_tail_rec`, die sich wie `sum_list_rec` verhält, aber endrekursiv implementiert ist. Verwenden Sie hierzu das in der Vorlesung gezeigte Verfahren mit einem Akkumulator acc als zusätzlichem Argument.

(c) (5 Punkte) Schreiben Sie eine **nicht-rekursive** Funktion `to_bin_iter`, die sich wie `to_bin_rec` verhält. Verwenden Sie hierzu das in der Vorlesung gezeigte Verfahren.

```
>>> to_bin_rec(0)
''
>>> to_bin_rec(1)
'1'
>>> to_bin_rec(2)
'10'
>>> to_bin_rec(4)
'100'
>>> to_bin_rec(4) == to_bin_tail_rec(4) == to_bin_iter(4)
True
```

Aufgabe 6 (Rekursion und Bäume; 20 Punkte)
Es sei die rekursive Definition eines Binärbaumes gegeben:

```
@dataclass
class Arm[T]:
  weight: int
  left: "Tree[T]" = None
  right: "Tree[T]" = None

type Mobile[T] = Optional[Arm[T]]
```

(a) (10 Punkte) Implementieren Sie eine Funktion `is_balanced`, die einen Baum entgegennimmt und entscheidet, ob dieser Baum balanziert ist. Die Funktion soll ein Tuple zurückgeben, dessen erstes Element angibt, ob der Baum balanziert ist und dessen zweites Element das Gesamtgewicht des Baumes angibt, also die Summe der Gewichte der Kinder und der des Knoten selbst. Ein Baum ist balanziert, wenn beide Kinder balanziert sind und deren Gewichte gleich sind. **Verwenden Sie Pattern Matching und Rekursion.**

```
>>> t1 = Arm(4, Arm(3), Arm(1, Arm(1), Arm(1)))
>>> is_balanced(t1)
True
>>> t2 = Arm(0, Arm(3), Arm(5, Arm(5)))
>>> is_balanced(t2)
False
```

(b) (10 Punkte) Implementieren sie eine Funktion `is_isomorphic`, die zwei Bäume `m1` und `m2` entgegennimmt und einen Boolean zurückgibt, der angibt, ob die Bäume isomorph sind. Zwei Bäume sind isomorph, wenn Sie die gleiche Struktur haben. Die Isomorphie ist unabhängig von der Reihenfolge der Kinder eines Knotens. **Verwenden Sie Pattern Matching und Rekursion.**

```
>>> m1 = Arm(4, Arm(3), Arm(1, Arm(1), Arm(1)))
>>> m2 = Arm(0, Arm(3), Arm(5, Arm(5)))
>>> m3 = Arm(10, Arm(1, None, Arm(3)), Arm(4))
>>> is_balanced(m1, m2)
False
>>> is_balanced(m2, m3)
True
```

Aufgabe 7 (Generatoren; 20 Punkte)

(a) (10 Punkte) Schreiben Sie eine Generatorfunktion, die ein iterierbares Object aus Ganzzahlen `xs` entgegennimmt und die Indizes zurückgibt, wo die Summe der Elemente von `xs` bis zu diesem Index null ergibt.

Beispiel:

```
>>> list(sum_null([]))
[]
>>> list(sum_null([2, -2, 2, 3, -5, 6]))
[1, 4]
```

(b) (10 Punkte) Schreiben Sie eine Generatorfunktion die das `run-lenght-encoding` implementiert. Die Funktion nimmt ein iterierbares Object `xs` entgegen und gibt ein Tuple zurück, das für jeden Buchstaben der Eingabe die Anzahl an konsekutiven Wiederholungen dieses Buchstaben angibt.

Beispiel:

```
>>> list(run_lenght_encoding(""))
[]
>>> list(run_lenght_encoding("aaaabbcaa"))
[("a", 4), ("b", 2), ("c", 1), ("a", 2)]
>>> list(run_lenght_encoding("161"))
[("1", 1), ("6", 1), ("1", 1)]
```

Aufgabe 8 (Funktionale Programmierung; 20 Punkte)

Implementieren Sie die folgende Aufgabe im funktionalen Stil. Jede Funktion soll nur aus einer `return` Anweisung bestehen.

(a) Implementieren Sie eine Funktion `rev_app`, die eine beliebige Funktion `f` und einen Wert `x` annimt, und die Funktion angewendet auf den Eingabewert zurückgibt.

```
>>> rev_app(lambda x: x * 2, 5)
10
```

(b) Implementieren Sie eine Funktion `isprime`, die eine Ganzzahl `n` annimt, und einen Boolean zurückgibt, der angibt ob die Eingabe prim ist. Eine Zahl ist prim, wenn sie nur durch 1 und sich selbst teilbar ist. **Sie können annehmen, dass die Eingaben größer gleich 2 sind.**

```
>>> isprime(2)
True
>>> isprime(4)
False
```

(c) Implementieren Sie eine Funktion `primes`, die ein iterierbares Objekt aus Ganzzahlen `xs` annimt, und einen Dictionary zurückgibt, der jeder Zahl aus `xs` einen Boolean zuordnet, der angibt ob die Zahl prim ist.

```
>>> primes([2, 3, 4, 5])
{2: True, 3: True, 4: False, 5: True}
```

(d) Implementieren Sie eine Funktion `pipe`, die ein iterierbares Objekt aus Funktionen `fs` und einen Startwert `start` annimt, und alle Funktionen angewendet auf den Startwert, zurückgibt. Mit der Eingabe $[f_1, f_2, f_3, ...]$ soll $...f_3(f_2(f_1(\text{start})))$ ausgegeben werden

```
>>> pipe([lambda x: x * 2, lambda x: x + 3], 5)
13
```