LB01 - Moodle-Prüfung - 25%

Moodle-Prüfung mit Theorie- und Praxisaufgaben. Die Prüfung findet **ohne Hilfsmittel** statt. Einzige Ausnahme ist ein persönliches Notizblatt.

- Dauer: 40 Minuten
- Hilfsmittel:
  - Persönliche Notizen im Umfang von **max. 2 A4-Seiten** (ein Blatt beidseitig oder zwei Blätter einseitig), handschriftlich oder am Computer erstellt, auf Papier mitgebracht
  - Sonst keine Hilfsmittel: kein Zugriff auf Modulunterlagen (Moodle, Dokuwiki), Internet, IDE oder KI-Werkzeuge
- Praxisaufgaben: Der Code wird direkt in Moodle geschrieben, ohne IDE (kein Ausführen, keine Autovervollständigung)
- Sozialform: Einzelarbeit
- Wann: Abschluss von LU07 (ohne LU06)
- Gewichtung: 25%

Worauf es ankommt: Die Prüfung verwendet Code und Problemstellungen, die Sie noch nicht gesehen haben. Sie müssen das Gelernte **übertragen**: fremden Code lesen und nachvollziehen, Ausgaben ohne Ausführen bestimmen, Fehler und Grenzfälle erkennen, Code unter Randbedingungen anpassen und Ihre Lösung begründen.

Vorbereitete Musterlösungen auf dem Notizblatt helfen deshalb wenig. Was hilft: Syntaxmuster, Begriffe, Unterschiede und typische Fallen, in eigenen Worten zusammengefasst.

Lernziele

Die Kompetenzcodes beziehen sich auf das Kompetenzraster.

1. Paradigmen und Grundkonzepte (LU01, LU02)

Nr	Ich kann …	Kompetenz	Unterlagen
1.1	einem unbekannten Beispiel (Codeausschnitt, SQL, HTML, Schleife, Comprehension usw.) zuordnen, ob es imperativ oder deklarativ ist, und diese Zuordnung begründen.	B1B	LU01a
1.2	den Ablauf eines mir unbekannten Codeausschnitts mit einer Trace Table nachvollziehen und die Ausgabe bestimmen, ohne den Code auszuführen.	C1B	LU01d

Nr	Ich kann …	Kompetenz	Unterlagen
1.3	bei einer gegebenen Funktion entscheiden, ob sie pure ist, und die konkreten Seiteneffekte oder Abhängigkeiten von externem Zustand benennen.	A1B	LU02b
1.4	eine unreine Funktion so umschreiben, dass sie pure ist, ohne ihr Ergebnis zu verändern.	A1B, D1I	LU02b
1.5	vorhersagen, wie sich die Übergabe von veränderlichen (`list`, `dict`, Dataclass) und unveränderlichen Werten (`int`, `str`, `tuple`) an eine Funktion auf die Variablen des Aufrufers auswirkt, und dies begründen.	A1I	LU02c, LU02d, LU02e
1.6	Code so anpassen, dass er bestehende Daten nicht verändert, sondern neue Daten erzeugt, und erklären, warum das den Code vorhersagbarer macht.	A1I	LU02c

2. Funktionen als Werte und Rekursion (LU03, LU05)

Nr	Ich kann …	Kompetenz	Unterlagen
2.1	in unbekanntem Code erkennen, wo Funktionen als Werte gespeichert, übergeben oder zurückgegeben werden, und benennen, welche Funktion Higher-Order Function und welche Callback ist.	C2B	LU03d, LU03e, LU03f
2.2	für eine gegebene rekursive Funktion Basisfall und Rekursionsschritt benennen und die Aufrufreihenfolge sowie die Ausgabe bestimmen, insbesondere wenn Anweisungen vor oder nach dem rekursiven Aufruf stehen.	C1B	LU03b
2.3	eine rekursive Funktion unter vorgegebenen Randbedingungen anpassen (z.B. andere Reihenfolge, anderer Rückgabewert) und meine Lösung in ganzen Sätzen begründen.	C1I	LU03b, LU03c
2.4	für gegebenen Code mit inneren Funktionen bestimmen, welche Werte ein Closure einschliesst, und das Ergebnis mehrerer Aufrufe vorhersagen.	C2A	LU05b, LU05c
2.5	den Ablauf eines gegebenen Decorators nachvollziehen (was vor und nach dem eigentlichen Aufruf passiert) und einen einfachen Decorator nach Vorgabe schreiben, der mit `*args` und `**kwargs` beliebige Funktionen umschliesst.	C2A	LU05a, LU05d, LU05e

3. Funktionales Programmieren mit Iterables (LU04)

Nr	Ich kann …	Kompetenz	Unterlagen
3.1	eine mehrstufige Bedingung als ternären Ausdruck formulieren und bei vorgegebenen Varianten erkennen, welche an den Grenzwerten falsch sind.	C2B	LU04a
3.2	das Ergebnis von List Comprehensions, Slicing, `map`, `filter` und `reduce` für gegebene Daten bestimmen, auch wenn sie kombiniert sind oder das Slicing an unterschiedlichen Stellen steht.	C2B	LU04c, LU04d, LU04e, LU04f, LU04k
3.3	eine in Worten beschriebene Anforderung (filtern, umformen, aggregieren) auf einer neuen Datenstruktur (z.B. Liste von Dictionaries) mit der verlangten Technik umsetzen, ohne die Originaldaten zu verändern.	C2I	LU04c bis LU04f, LU04j

Nr	Ich kann …	Kompetenz	Unterlagen
3.4	Lambda-Ausdrücke schreiben, insbesondere als `key` für `sorted` mit mehreren Sortierkriterien und verschachtelten Datenstrukturen.	C2I, C2A	LU04b, LU04g
3.5	mehrere Verarbeitungsschritte (filtern, umformen, aggregieren, sortieren) zu einer Lösung kombinieren und bei vorgegebenen Varianten beurteilen, welche dasselbe Ergebnis liefern (z.B. Startwert bei `reduce`, leere Liste).	C2A	LU04
3.6	erklären, wann ein Generator gegenüber einer Liste Vorteile hat, das Verhalten eines Generators bestimmen (`next()`, mehrfacher Durchlauf) und eine listenbasierte Funktion in einen Generator umbauen.	D1I	LU04h, LU04i

4. Refactoring und Leistung (LU07)

Nr	Ich kann …	Kompetenz	Unterlagen
4.1	in einem unbekannten Codeausschnitt Code Smells erkennen, benennen und die passende Refactoring-Technik zuordnen.	D1B	LU07b, LU07c
4.2	imperativen Code in eine funktionale Form überführen (z.B. Schleife mit Akkumulator in Comprehension oder Pipeline, `map`/`filter` in eine Comprehension), ohne das Verhalten zu verändern.	D1I, B1A	LU07d
4.3	beurteilen, ob eine refaktorierte Variante sich gleich verhält wie das Original, und unerwünschte Nebeneffekte oder Grenzfälle benennen.	D1A	LU07e
4.4	für eine beschriebene Situation eine geeignete Massnahme zur Leistungsverbesserung auswählen (Datenstruktur, Index statt verschachtelter Schleife, Cache, Generator) und deren Nachteil nennen.	D1I, D1A	LU07f, LU07g

Aufgabentypen

- **Code lesen:** Ausgabe oder Wert einer Variable bestimmen (Auswahl oder Lückentext).
- **Varianten beurteilen:** Aus mehreren Codevarianten die korrekten auswählen. Es können mehrere Antworten richtig sein.
- **Code schreiben oder umbauen:** Direkt in Moodle, ohne IDE. Bewertet wird die Logik; kleine Syntaxfehler, die die Lösung nicht verfälschen, führen nicht zu Abzug.
- **Erklären:** Eine Lösung oder einen Unterschied in ganzen Sätzen begründen.

Hinweise zum Notizblatt

- Wie Sie das Blatt erstellen, ist Ihnen überlassen. Ein Blatt, das Sie selbst zusammenstellen, ist aber bereits ein Teil der Prüfungsvorbereitung.
- Sinnvoller Inhalt: Syntax von Comprehension, Lambda, `sorted(key=…)`, `reduce`, Generator und Decorator; Begriffe mit einem kurzen eigenen Beispiel; Stolperfallen (Grenzwerte, Startwert bei `reduce`, by reference bei Listen, einmaliger Durchlauf von Generatoren).
- Wenig sinnvoll: fertige Lösungen von Übungsaufgaben. Die Prüfungsaufgaben sind neu.

From:
https://wiki.bzz.ch/ - **BZZ - Modulwiki**

Permanent link:
https://wiki.bzz.ch/modul/m323/leistungsbeurteilungen/lb01/start?rev=1790663130

Last update: **2026/09/29 08:25**