

# Kompetenzübersicht

Kompetenzband:	HZ	Grundlagen	Fortgeschritten	Erweitert
Unterschiede zwischen funktionaler Programmierung und anderen Programmierparadigmen aufzeigen.	1	<p>AG1: Ich kann die Eigenschaften von Funktionen beschreiben (z.Bsp. pure function) und den Unterschied zu anderen Programmier-Strukturen erläutern (z.Bsp. zu Prozedur).</p> <ul style="list-style-type: none"> <li>• <a href="#">LU01a - Deklarative vs. Imperative Programmierung</a></li> <li>• <a href="#">LU01b - Strukturierte Programmierung</a></li> <li>• <a href="#">LU01c - Funktionale Programmierung</a></li> <li>• <a href="#">LU02a - Grundkonzepte der funktionalen Programmierung</a></li> <li>• <a href="#">LU02b - Pure Functions</a></li> </ul>	<p>AF1: Ich kann das Konzept von *immutable values* erläutern und dazu Beispiele anwenden. Somit kann ich dieses Konzept funktionaler Programmierung im Unterschied zu anderen Programmiersprachen erklären (z.Bsp. im Vergleich zu referenzierten Objekten)</p> <ul style="list-style-type: none"> <li>• <a href="#">LU02c - Immutable Values</a></li> </ul>	<p>AE1: Ich kann aufzeigen wie Probleme in den verschiedenen Konzepten (OO, prozedural und funktional) gelöst werden und diese miteinander vergleichen.</p> <ul style="list-style-type: none"> <li>• <a href="#">LU02d - By Value und By Reference in Python</a></li> <li>• <a href="#">LU02e - By Value und By Reference bei Dataclasses und Objekten in Python</a></li> </ul>
Anforderungen und Design beschreiben	1	BG1: Ich kann den Unterschied zwischen Anforderungen der imperativen Programmierung (definierte Folge von Handlungsanweisungen) und der deklarativen Programmierung (Beschreibung des Endzustandes) erklären.	BF1: Ich kann den Endzustand als Anforderung im Sinne der deklarativen Programmierung beschreiben. (Das gewünschte Ergebnis wird beschrieben statt die Arbeitsschritte.)	BE1: Ich kann Anforderungen aus der imperativen Programmierung in Anforderungen der deklarativen Programmierung transferieren. („klar definierte Abfolge“ transformieren zu „Endergebnis beschreiben“)
		BG2: Ich kann Elemente des Functional Design erklären. (zBsp. Immutable data types, model, solution, domain of interest, constructors, composable operators)	BF2: Ich kann für eine Problemstellung ein Functional-Design entwerfen und dabei die Elemente des Functional Designs anwenden.	BE2: Ich kann ein Design einer imperativen Programmierung in ein Design der deklarativen Programmierung transferieren.
Funktionale Programmierung umsetzen	2	<p>CG1: Ich kann ein Algorithmus erklären</p> <ul style="list-style-type: none"> <li>• <a href="#">LU01d - Trace Table</a></li> </ul>	CF1: Ich kann Algorithmen in funktionale Teilstücke aufteilen	CE1: Ich kann Funktionen in zusammenhängende Algorithmen implementieren.
		<p>CG2: Ich kann Funktionen als Objekte behandeln und diese in Variablen speichern und weitergeben.</p> <ul style="list-style-type: none"> <li>• <a href="#">LU03d - First-Class Functions</a></li> </ul>	<p>CF2: Ich kann Funktionen als Argumente für andere Funktionen verwenden und dadurch höherwertige Funktionen erstellen.</p> <ul style="list-style-type: none"> <li>• <a href="#">LU03d - First-Class Functions</a></li> </ul>	<p>CE2: Ich kann Funktionen als Objekte und Argumente verwenden, um komplexe Aufgaben zu lösen und den Code sauberer und effizienter zu gestalten.</p> <ul style="list-style-type: none"> <li>• <a href="#">LU03d - First-Class Functions</a></li> </ul>

Kompetenzband:	HZ	Grundlagen	Fortgeschritten	Erweitert
		CG3: Ich kann einfache Lambda-Ausdrücke schreiben, die eine einzelne Operation durchführen, z.B. das Quadrieren einer Zahl oder das Konvertieren eines Strings in Großbuchstaben.	CF3: Ich kann Lambda-Ausdrücke schreiben, die mehrere Argumente verarbeiten können.	CE3: Ich kann Lambda-Ausdrücke verwenden, um den Programmfluss zu steuern, z.B. durch Sortieren von Listen basierend auf benutzerdefinierten Kriterien.
		CG4: Ich kann die Funktionen Map, Filter und Reduce einzeln auf Listen anwenden.	CF4: 1. Ich kann Map, Filter und Reduce kombiniert verwenden, um Daten zu verarbeiten und zu manipulieren, die komplexere Transformationen erfordern.	CE4: 1. Ich kann Map, Filter und Reduce verwenden, um komplexe Datenverarbeitungsaufgaben zu lösen, wie z.B. die Aggregation von Daten oder die Transformation von Datenstrukturen.
Refactoring und bestehenden Code optimieren	3,4	DG1: Ich kann einige Refactoring-Techniken aufzählen, die einen Code lesbarer und verständlicher machen.	DF1: Ich kann mit Refactoring-Techniken einen Code lesbarer und verständlicher machen.	DE1: Ich kann die Auswirkungen des Refactorings auf das Verhalten des Codes einschätzen und sicherstellen, dass das Refactoring keine unerwünschten Nebeneffekte hat.
		DG2: Ich kann allgemeine Massnahmen zur Verbesserung der Leistung von Code aufzählen.	DF2: Ich kann vorgegebene Massnahmen zur Verbesserung der Leistung von Code umsetzen.	DE2: Ich kann effiziente Algorithmen, Techniken oder Datenstrukturen auswählen und einsetzen, um die Leistung von Code zu verbessern.

From:

<https://wiki.bzz.ch/> - BZZ - Modulwiki

Permanent link:

<https://wiki.bzz.ch/modul/m323/kompetenzuebersicht>Last update: **2025/11/17 13:36**