Kompetenzübersicht

Kompetenzband:	HZ	Grundlagen	Fortgeschritten	Erweitert
Unterschiede zwischen funktionaler Programmierung und anderen Programmierparadigmen aufzeigen.	1	AG1: Ich kann die Eigenschaften von Funktionen beschreiben (z.Bsp. pure function) und den Unterschied zu anderen Programmier-Strukturen erläutern (z.Bsp. zu Prozedur). • LU01a - Deklarative vs. Imperative Programmierung • LU01b - Strukturierte Programmierung • LU01c - Funktionale Programmierung • LU02a - Grundkonzepte der funktionalen Programmierung • LU02b - Pure Functions	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) • LU02c - Immutable Values	AE1: Ich kann aufzeigen wie Probleme in den verschiedenen Konzepten (OO, prozedural und funktional) gelöst werden und diese miteinander vergleichen. • LU02d - By Value und By Reference in Python • LU02e - By Value und By Reference bei Dataclasses und Objekten in Python
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 Desing der deklarativen Programmierung transferieren.
Funktionale Programmierung umsetzen	2	CG1: Ich kann ein Algorithmus erklären • LU01d - Trace Table	CF1: Ich kann Algorithmen in funktionale Teilstücke aufteilen	CE1: Ich kann Funktionen in zusammenhängende Algorithmen implementieren.
		CG2: Ich kann Funktionen als Objekte behandeln und diese in Variablen speichern und weitergeben. • LU03d - First-Class Functions	CF2: Ich kann Funktionen als Argumente für andere Funktionen verwenden und dadurch höherwertige Funktionen erstellen. • LU03d - First-Class Functions	CE2: Ich kann Funktionen als Objekte und Argumente verwenden, um komplexe Aufgaben zu lösen und den Code sauberer und effizienter zu gestalten. • LU03d - First-Class Functions

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. • LU04b - Die lambda-Funktion in Python	CF3: Ich kann Lambda-Ausdrücke schreiben, die mehrere Argumente verarbeiten können. • LU04b - Die lambda-Funktion in Python	CE3: Ich kann Lambda-Ausdrücke verwenden, um den Programmfluss zu steuern, z.B. durch Sortieren von Listen basierend auf benutzerdefinierten Kriterien. • LU04b - Die lambda-Funktion in Python
		CG4: Ich kann die Funktionen Map, Filter und Reduce einzeln auf Listen anwenden. • LU04a - Ternärer Bedingungsoperator in Python • LU04d - Die map-Funktion in Python • LU04e - Die filter-Funktion in Python • LU04f - Die reduce-Funktion in Python	CF4: 1. Ich kann Map, Filter und Reduce kombiniert verwenden, um Daten zu verarbeiten und zu manipulieren, die komplexere Transformationen erfordern. • LU04a - Ternärer Bedingungsoperator in Python • LU04d - Die map-Funktion in Python • LU04e - Die filter-Funktion in Python • LU04f - Die reduce-Funktion in Python	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. • LU04a - Ternärer Bedingungsoperator in Python • LU04d - Die map-Funktion in Python • LU04e - Die filter-Funktion in Python • LU04f - Die reduce-Funktion in Python
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 Leis-tung von Code aufzählen.	DF2:Ich kann vorgegebene Massnah-men zur Verbesserung der Leistung von Code umsetzen.	DE2: Ich kann effiziente Algorithmen, Techniken oder Datenstrukturen auswählen und einsetzen, um die Leistung von Code zu ver-bessern.