HAW Kiel
Hochschule für Angewandte Wissenschaften Kiel
Kiel University of Applied Sciences

BI141 - Grundlagen funktionaler Programmierung
BI141 - Introduction to functional programming

General information	
Module Code	BI141
Unique Identifier	GrundlFunktP-01-BA-M
Module Leader(s)	Oenings, Hendrik (hendrik.oenings@haw-kiel.de) Angres, Julius (julius.angres@haw-kiel.de)
Lecturer(s)	Angres, Julius (julius.angres@haw-kiel.de)
Offered in Semester	Wintersemester 2024/25
Module duration	1 Semester
Occurrence frequency	Regular
Module occurrence	In der Regel im Wintersemester
Language	Deutsch
Recommended for international students	No
Can be attended with different study programme	Yes

Curricular relevance (according to examination regulations)
Study Subject: B.Eng. - E - Elektrotechnik (PO 2017, V3) Module type: Wahlmodul Semester: 5
Study Subject: B.Eng. - E - Elektrotechnik (PO 2023, V4) Module type: Wahlmodul Semester: 5
Study Subject: B.Eng. - Ming - Medieningenieur/-in (PO 2018, V1 + PO 2021, V2) Module type: Wahlmodul Semester: 5
Study Subject: B.Eng. - Wing - Wirtschaftsingenieurwesen - Elektrotechnik (PO 2017, V1) Module type: Wahlmodul Semester: 5
Study Subject: B.Eng. - Wing - Wirtschaftsingenieurwesen - Elektrotechnik (PO 2023, V2) Module type: Wahlmodul Semester: 5
Study Subject: B.Sc. - INF - Informatik (PO 2021,V1) Module type: Wahlmodul Semester: 5
Study Subject: B.Sc. - INI - Informationstechnologie (PO 2017, V1) Module type: Wahlmodul Semester: 5
Study Subject: B.Sc. - WINF - Wirtschaftsinformatik (6 Sem.) Module type: Wahlmodul Semester: 5
Study Subject: B.Sc. - WINF 7 Sem. - Wirtschaftsinformatik (7 Sem.) Module type: Wahlmodul Semester: 5

Qualification outcome
Areas of Competence: Knowledge and Understanding; Use, application and generation of knowledge; Communication and cooperation; Scientific self-understanding / professionalism.

Die Studierenden - kennen die Unterschiede zwischen imperativer und deklarativer Programmierung - kennen die Vorteile deklarativer Programmierung gegenüber imperativer Programmierung - können grundlegende Programme in einer funktionalen Programmiersprache erstellen - wissen, wie klassische imperative Programmierkonzepte (Datenstrukturen, Schleifen) in funktionalen bzw. funktionallogischen Programmiersprachen abgebildet werden können - können Ein- und Ausgabeoperationen umsetzen und verstehen, wie der Verzicht auf Seiteneffekte funktioniert - verstehen das Konzept der "Lazy Evaluation" und können Algorithmen unter Ausnutzung dieses Konzepts kompakt angeben - kennen die Grundlagen logischer Programmierung und das Konzept des Nichtdeterminismus in der Programmierung
Die Studierenden - können in Anwendungen der multiparadigmatischen Programmiersprache Python Elemente der funktionalen Programmierung erkennen und zielgerichtet zur Verbesserung der Codequalität einsetzen - können Algorithmen unabhängig von der konkreten Implementierung und Auswertung verstehen und beschreiben
Die Studierenden - lernen, Aufgaben in kleinen Gruppen zu bearbeiten - können sich über Programme, Konzepte und Lösungen austauschen und diese vor einer Gruppe präsentieren - können Programme übersichtlich und kompakt darstellen und dokumentieren
Die Studierenden - können Programmiersprachen mit funktionalen Elementen erlernen und anwenden - können Programmiersprachen mit logischen Elementen erlernen und anwenden - sind in der Lage, Aufgabenstellungen sowohl selbstständig als auch im Team umzusetzen

Content information

Content	Abgrenzung Programmierparadigmen: deklarativ vs. imperativ; Funktionale Programmierung; Motivation Vergleich zwischen "strict evaluation" und "lazy evaluation"; Vorteile von "lazy evaluation" Vergleich zwischen imperativen und funktionalen Umsetzungen bekannter Algorithmen (z.B. Quicksort) Programmiersprache(n) Haskell / Curry: - Geschichte, Verbreitung, Compiler/Laufzeitumgebungen (GHC, GHCi) - Funktionen (Definition, Rekursive Funktionen, Funktionen höherer Ordnung, Lambdaausdrücke, Komposition) - Datentypen (vordefinierte Typen, Deklaration, Polymorphie, Typklassen) - Datenstrukturen (Listen, Bäume, unendliche Datenstrukturen) - Ein-/Ausgabe und Behandlung von Seiteneffekten - Nichtdeterministische Funktionen - Logische Constraints Programmiersprache Python: - Nutzung der Funktionen map, filter, reduce - Nutzung von Lambdaausdrücken und Funktionen höherer Ordnung - Vermeidung von Seiteneffekten, Wiederverwendbarkeit
Literature	wird in der Veranstaltung bekanntgegeben

Teaching formats of the courses

Teaching format	SWS
Übung	2

Lehrvortrag	2

Workload	
Number of SWS	4 SWS
Credits	5,00 Credits
Contact hours	48 Hours
Self study	102 Hours

Module Examination	
Examination prerequisites according to exam regulations	None
BI141 - Klausur	Method of Examination: Klausur Duration: 60 Minutes Weighting: 100% wird angerechnet gem. § 11 Absatz 2 PVO: No Graded: Yes

Miscellaneous	
Recommended Prerequisites	- Grundlegendes Verständnis imperativer Programmiersprachen - Grundkenntnisse der informatischen Logik - Verständnis von Algorithmen und Rekursion - Verständnis von Datenstrukturen (Bäume, verkettete Listen) - Grundkenntnisse in der Programmiersprache Python