Fachhochschule Graubünden
University of Applied Sciences

Bachelorstudienrichtung Artificial Intelligence in Software Engineering

# Modulübersicht

# Modulübersicht Bachelorstudienrichtung Artificial Intelligence in Software Engineering

# Pflichtmodule

## Modulbeschreibung: AISE102 - Artificial intelligence on a human level

**Leitidee**

Despite impressive results in the field of AI, we are still far from human-level AI: current AI approaches are great at solving specific tasks they were programmed for - but when it comes to unforeseen tasks, they fail.
In this module, we analyse where the weaknesses of current AI approaches lie and how they can be addressed. Different problems and approaches to solving them will be discussed and students will be motivated to develop their own solution strategies.
The aim is to create an understanding of what current AI can (not) do and how AI needs to be further developed to support people broadly and reliably.

**Typ**

Wahlpflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

# Modulbeschreibung: AISE201 - Constraint Programming and Optimization

**Leitidee**

Der Kurs "Constraint Programming and Optimization" vermittelt den Studierenden fundierte Kenntnisse und praktische Fähigkeiten im Bereich der Constraint-Programmierung und Optimierungstechniken. Der Schwerpunkt liegt auf der Modellierung und Lösung komplexer Probleme mittels Constraints sowie der Anwendung von Optimierungsalgorithmen zur Bestimmung optimaler Lösungen unter gegebenen Beschränkungen. Die Studierenden lernen, wie sie verschiedene Constraint-Programmierparadigmen und Optimierungsverfahren einsetzen können, um effiziente und effektive Lösungen für reale Anwendungsfälle zu entwickeln. Durch eine Kombination aus theoretischen Vorlesungen, praktischen Übungen und einem abschließenden Projekt erwerben die Studierenden die Fähigkeit, komplexe Optimierungsprobleme zu analysieren, zu modellieren und mithilfe geeigneter Tools und Methoden zu lösen.

**Typ**

Wahlpflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

- **Grundlagen der Constraint-Programmierung verstehen:**
  - Die grundlegenden Konzepte und Prinzipien der Constraint-Programmierung kennen.
  - Unterschiede zwischen Constraint-Programmierung und anderen Programmierparadigmen erkennen.
- **Modellierung von Problemen mit Constraints:**
  - Komplexe Probleme mittels Constraints modellieren.
  - Verschiedene Arten von Constraints (z.B. logische, arithmetische) anwenden und kombinieren.
- **Optimierungsalgorithmen anwenden:**
  1. Unterschiedliche Optimierungsverfahren wie lineare Programmierung, ganzzahlige Programmierung und heuristische Methoden verstehen und einsetzen.
  2. Methoden zur Lösung von Optimierungsproblemen unter Beschränkungen implementieren.
- **Verwendung von Constraint-Programmierwerkzeugen:**
  1. Moderne Constraint-Programmierframeworks und -werkzeuge wie **Google OR-Tools**, **Gurobi** und **Gecode** effektiv nutzen.
  2. Integration dieser Werkzeuge in Softwareentwicklungsprojekte durchführen.
- **Analyse und Bewertung von Lösungen:**
  1. Lösungen auf ihre Optimalität und Effizienz hin bewerten.
  2. Performanz von Constraint- und Optimierungsalgorithmen analysieren und optimieren.
- **Praktische Anwendungen und Best Practices:**

1. Best Practices der Constraint-Programmierung und Optimierung in realen Projekten anwenden.
2. Fallstudien und reale Anwendungsbeispiele analysieren und umsetzen.

– **Projektmanagement und Teamarbeit:**
1. Eigenständige Projekte zur Lösung komplexer Optimierungsprobleme planen, durchführen und dokumentieren.
2. Effektive Zusammenarbeit im Team fördern und Kommunikationsfähigkeiten entwickeln.

# Modulbeschreibung: AISE202 - Funktionale Programmierung

**Leitidee**

Der Kurs "Funktionale Programmierung mit Python" bietet den Studierenden eine fundierte Einführung in die funktionalen Programmierparadigmen und deren Anwendung in der Programmiersprache Python. Der Schwerpunkt liegt auf dem Verständnis und der praktischen Umsetzung funktionaler Konzepte wie unveränderliche Datenstrukturen, höhere Funktionen, rekursive Algorithmen und funktionale Programmiermuster. Durch eine Kombination aus theoretischen Vorlesungen und praxisorientierten Übungen lernen die Studierenden, wie sie funktionale Programmiertechniken effektiv einsetzen können, um sauberen, wartbaren und effizienten Code zu schreiben. Der Kurs zielt darauf ab, die Denkweise funktionaler Programmierung zu vermitteln und deren Vorteile in der Softwareentwicklung aufzuzeigen.

**Typ**

Wahlpflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

Nach Abschluss des Kurses sollen die Studierenden:

- **Grundlagen der funktionalen Programmierung verstehen:**
  - Die Prinzipien und Paradigmen der funktionalen Programmierung kennen.
  - Unterschiede zwischen imperativer und funktionaler Programmierung erkennen.
- **Funktionale Konzepte in Python anwenden:**
  - Höhere Funktionen, Lambda-Ausdrücke, Map, Filter und Reduce verstehen und einsetzen.
  - Unveränderliche Datenstrukturen und deren Nutzen in der Programmierung anwenden.
- **Rekursion und rekursive Algorithmen implementieren:**
  3. Rekursive Ansätze zur Problemlösung entwickeln und optimieren.
  4. Tail-Recursion und ihre Bedeutung in der funktionalen Programmierung verstehen.
- **Funktionale Programmiermuster nutzen:**
  3. Dekoratoren, Closures und Currying in Python einsetzen.
  4. Funktionale Datenverarbeitung mit Generatoren und Iterators durchführen.
- **Funktionale Programmierung zur Verbesserung der Codequalität einsetzen:**
  3. Sauberen, modularen und wartbaren Code durch funktionale Prinzipien schreiben.
  4. Fehleranfälligkeit und Seiteneffekte durch funktionale Techniken minimieren.
- **Erweiterte funktionale Bibliotheken und Tools nutzen:**
  3. Funktionale Programmierbibliotheken wie `functools` und `itertools` effektiv einsetzen.
  4. Externe Bibliotheken zur funktionalen Programmierung kennenlernen und anwenden.

# Modulbeschreibung: AISE203 - Distributed and Reactive Programming

**Leitidee**

Der Kurs "Distributed and Reactive Programming" vermittelt den Studierenden umfassende Kenntnisse und praktische Fähigkeiten zur Entwicklung verteilter und reaktiver Systeme unter Verwendung moderner, in der Industrie weit verbreiteter Frameworks. Der Schwerpunkt liegt auf der Erstellung skalierbarer, fehlertoleranter und reaktiver Anwendungen, die effizient auf Änderungen und Anforderungen reagieren können. Die Studierenden lernen, wie sie Frameworks wie **Apache Kafka** und **Akka** einsetzen können, um robuste und skalierbare Softwarelösungen zu entwerfen und zu implementieren. Zusätzlich werden ergänzende Technologien und Tools behandelt, die in der Praxis zur Unterstützung verteilter und reaktiver Systeme eingesetzt werden. Durch praxisorientierte Übungen, Fallstudien und ein abschließendes Projekt erwerben die Studierenden die Fähigkeit, komplexe verteilte und reaktive Systeme zu entwickeln, die den Anforderungen moderner Softwareanwendungen gerecht werden.

**Typ**

Wahlpflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

Nach erfolgreichem Abschluss dieses Moduls "Distributed and Reactive Programming" werden die Studierenden in der Lage sein:

Nach Abschluss des Kurses sollen die Studierenden:

- **Grundlagen verteilter und reaktiver Systeme verstehen:**
  - Die grundlegenden Prinzipien verteilter Systeme und reaktiver Programmierung kennen.
  - Unterschiede zwischen synchroner und asynchroner Programmierung sowie zwischen verteilter und reaktiver Architektur verstehen.
- **Verteilte Programmierung mit Akka anwenden:**
  - Das **Akka-Framework** zur Implementierung verteilter Anwendungen verstehen und einsetzen.
  - Konzepte wie Aktoren, Clusterbildung und Fehlertoleranzmechanismen in Akka nutzen.
- **Nachrichtenorientierte Middleware mit Apache Kafka verwenden:**
  5. Grundkonzepte von **Apache Kafka** für verteilte Kommunikation und Datenstreaming verstehen.
  6. Kafka-Produzenten und -Konsumenten implementieren sowie Themen und Partitionen effektiv verwalten.
- **Reaktive Programmierung mit RxJava umsetzen:**
  5. **RxJava** zur Implementierung reaktiver Datenströme und asynchroner Verarbeitung nutzen.
  6. Reaktive Programmiermuster anwenden, um effiziente und skalierbare Anwendungen zu entwickeln.

- **Fehlertoleranz und Skalierbarkeit in verteilten Systemen sicherstellen:**
    5. Methoden zur Implementierung von Fehlertoleranz, Lastverteilung und Skalierbarkeit in verteilten Systemen anwenden.
    6. Monitoring und Logging für verteilte Anwendungen einrichten und nutzen.
- **Performance und Skalierbarkeit optimieren:**
    5. Methoden zur Evaluierung und Verbesserung der Performance von fortgeschrittenen KI-Modulen in Softwareanwendungen anwenden.
    6. Skalierbare und effiziente KI-Lösungen für größere Softwareprojekte entwickeln

# Modulbeschreibung: AISE204 - Low Code / No Code

**Leitidee**

Das Modul "Entwicklung von Low-Code/No Code Plattformen" konzentriert sich auf die Prinzipien, Technologien und bestehenden Softwarelösungen zur Entwicklung von Applikationen mit Hilfe von Low-Code und No Code Plattformen. Low-Code/No Code Ansätze ermöglichen es Entwicklern und Nicht-Entwicklern, Anwendungen schnell und effizient zu erstellen, ohne umfangreiche Programmierkenntnisse zu benötigen. Der Kurs vermittelt den Studierenden fundierte Kenntnisse über die Architektur, die grundlegenden Komponenten und die Entwicklungsmethoden solcher Plattformen. Ein wesentlicher Bestandteil der Vorlesung ist die Prozessmodellierung mit BPML (Business Process Modeling Language) Tools und deren theoretisches Verständnis. Die Erkenntnisse aus der Prozessmodellierung werden genutzt, um eine praktische Anwendung im Bereich der Wirtschaftsinformatik mithilfe der Microsoft Low-Code Plattform durchzuführen. Durch eine Kombination aus theoretischen Vorlesungen, praktischen Übungen und einem abschließenden Projekt erwerben die Studierenden die Fähigkeiten, Low-Code/No Code Methoden effektiv in den Softwareentwicklungsprozess zu integrieren und anzuwenden.

**Typ**

Wahlpflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

Nach Abschluss des Kurses sollen die Studierenden:

- **Grundlagen von Low-Code/No Code verstehen:**
  - Die grundlegenden Konzepte und Prinzipien von Low-Code und No Code Plattformen kennen.
  - Unterschiede und Gemeinsamkeiten zwischen traditionellen Softwareentwicklungsansätzen und Low-Code/No Code Methoden erkennen.
- **Architektur und Komponenten von Low-Code/No Code Plattformen beherrschen:**
  - Die Architektur von Low-Code/No Code Plattformen verstehen, einschließlich Frontend, Backend, Datenbanken und Integrationsschnittstellen.
  - Wesentliche Komponenten wie Drag-and-Drop-Editoren, Workflow-Engines und Datenmanagementsysteme analysieren und anwenden.
- **Prozessmodellierung mit BPML Tools anwenden:**
  7. Grundlagen der Prozessmodellierung verstehen und BPML Tools effektiv nutzen.
  8. Geschäftsprozesse modellieren und analysieren, um Anforderungen für die Entwicklung von Anwendungen zu definieren.
- **Technologien und Frameworks zur Entwicklung von Low-Code/No Code Plattformen anwenden:**
  7. Überblick über führende Technologien und Frameworks, die zur Entwicklung von Low-Code/No Code Plattformen eingesetzt werden.

8. Einsatz von Web-Technologien, APIs, Datenbanken und Cloud-Diensten zur Unterstützung der Plattformentwicklung verstehen.

- **Bestehende Low-Code/No Code Plattformen analysieren und bewerten:**
  7. Analyse und Vergleich von bestehenden Plattformen wie Mendix, OutSystems, Microsoft PowerApps und anderen.
  8. Identifikation von Best Practices und Lessons Learned aus erfolgreichen Implementierungen.
- **Methoden und Prinzipien zur Entwicklung eigener Low-Code/No Code Plattformen anwenden:**
  7. Systematische und disziplinierte Methoden zur Planung, Entwicklung und Implementierung einer eigenen Plattform verstehen und anwenden.
  8. Anwendung von Design Patterns und Architekturprinzipien zur Sicherstellung der Skalierbarkeit, Wartbarkeit und Erweiterbarkeit der Plattform.
- **Praktische Fähigkeiten in der Entwicklung von Low-Code/No Code Plattformen entwickeln:**
  3. Praktische Erfahrungen in der Programmierung und Konfiguration von Low-Code/No Code Tools und Komponenten sammeln.
  4. Entwicklung von benutzerfreundlichen Schnittstellen und Automatisierung von Geschäftsprozessen mittels Low-Code/No Code Ansätzen verstehen und implementieren.
- **Sicherheits- und Datenschutzaspekte berücksichtigen:**
  - Sicherheitsherausforderungen und Datenschutzanforderungen bei der Entwicklung und Nutzung von Low-Code/No Code Plattformen erkennen und adressieren.
  - Implementierung von Sicherheitsmaßnahmen und Datenschutzstrategien in der eigenen Plattform durchführen.
- **Praktische Projekte planen und durchführen:**
  - Eigenständige Projekte zur Entwicklung einer eigenen Low-Code/No Code Plattform planen, durchführen und dokumentieren.
  - Effektive Teamarbeit und Kommunikationsfähigkeiten im Entwicklungsprozess fördern.

# Modulbeschreibung: AISE205 – Betriebssysteme und Computernetzwerke

**Leitidee**

Das Modul "Betriebssysteme und Computernetzwerke" fokussiert auf die grundlegenden Prozesse zur Steuerung und Verwaltung der Betriebsmittel eines Rechnersystems sowie die Verteilung dieser Ressourcen an die Benutzer. Rechnernetze ermöglichen den Zusammenschluss eigenständiger Computersysteme zur Kommunikation und gemeinsamen Nutzung von Ressourcen.

**Typ**

Wahlpflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

Nach erfolgreichem Abschluss dieses Moduls "Betriebssysteme und Computernetzwerke" werden die Studierenden in der Lage sein:

Nach Abschluss des Kurses sollen die Studierenden:

- **Grundlagen der Betriebssysteme verstehen:**
  - Die grundlegenden Konzepte und Architektur von Betriebssystemen kennen.
  - Prozesse und Threads sowie deren Verwaltung und Synchronisation verstehen.
  - Speicherverwaltungstechniken und -strategien analysieren können.
  - Dateisysteme und Ein-/Ausgabe-Mechanismen erläutern.
- **Ressourcenverwaltung und -verteilung beherrschen:**
  - Methoden zur effizienten Verteilung von Rechnerressourcen an Benutzer und Anwendungen anwenden.
  - Sicherheits- und Schutzmechanismen zur Ressourcenverwaltung implementieren.
- **Grundlagen der Computernetzwerke verstehen:**
  9. Netzwerktopologien und -architekturen sowie deren Vor- und Nachteile kennen.
  10. Die Funktionsweise von Netzwerkprotokollen wie TCP/IP verstehen.
  11. Routing- und Switching-Methoden sowie deren Implementierung erläutern.
- **Netzwerksicherheit und -verwaltung umsetzen:**
  9. Sicherheitsmechanismen und -protokolle in Netzwerken anwenden.
  10. Netzwerkinfrastrukturen konfigurieren und verwalten können.
- **Praktische Fähigkeiten in Betriebssystemen und Netzwerken entwickeln:**
  9. Betriebssysteme installieren, konfigurieren und administrieren.
  10. Netzwerke planen, aufbauen und troubleshootieren.
  11. Tools und Software zur Netzwerküberwachung und -verwaltung effektiv nutzen.
- **Integration von Betriebssystemen und Netzwerken in komplexe IT-Infrastrukturen verstehen:**
  9. Wechselwirkungen zwischen Betriebssystemen und Netzwerken analysieren.

10. Best Practices für die Integration und Optimierung von Betriebssystemen und Netzwerken anwenden.

# Modulbeschreibung: AISE206 – Semantik von Programmiersprachen

**Leitidee**

Die Vorlesung "Semantik von Programmiersprachen" bietet eine fundierte Einführung in die theoretischen Grundlagen und praktischen Anwendungen der Semantik moderner Programmiersprachen. Studierende lernen die wesentlichen semantischen Modelle kennen, darunter die operative, denotationale und axiomatische Semantik, und verstehen deren Bedeutung für die Entwicklung von Programmiersprachen. Ein besonderer Schwerpunkt liegt auf der Typentheorie und deren Rolle bei der Gestaltung sicherer und effizienter Sprachen. Durch die Anwendung semantischer Konzepte in der Programmiersprache Python sowie die Einführung in funktionale Programmiertechniken in Python wird die Verbindung zwischen Theorie und Praxis deutlich. Der Kurs behandelt zentrale Themen wie Kontrollstrukturen, Speicherverwaltung, Nebenläufigkeit und Fehlerbehandlung, wobei die semantischen Prinzipien ausschließlich in Python analysiert und angewendet werden.

**Typ**

Pflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

Nach Abschluss des Kurses sollen die Studierenden:

- **Grundlagen der Programmiersprachensemantik verstehen:**
  - Die grundlegenden Konzepte und Modelle der Programmiersprachensemantik kennenlernen.
  - Unterschiede und Gemeinsamkeiten zwischen operativer, denotationaler und axiomatischer Semantik erkennen und verstehen.
- **Semantische Modelle anwenden:**
  - Operative, denotationale und axiomatische Semantik in praktischen Beispielen anwenden.
  - Die Bedeutung dieser Modelle für die Programmiersprachenentwicklung und -analyse verstehen.
- **Typentheorie und ihre Anwendung:**
  12. Die Grundlagen der Typentheorie verstehen und ihre Rolle bei der Gestaltung sicherer und effizienter Programmiersprachen erkennen.
  13. Typensysteme in Python analysieren und anwenden, um die Sicherheit und Effizienz von Programmen zu verbessern.
- **Funktionale Programmiertechniken in Python nutzen:**
  11. Einführung in funktionale Programmierung und deren Konzepte in Python verstehen.
  12. Funktionale Programmiertechniken anwenden, um sauberen und effizienten Code zu schreiben.
- **Semantische Analyse von Python-Programmen durchführen:**

12. Kontrollstrukturen, Speicherverwaltung, Nebenläufigkeit und Fehlerbehandlung in Python semantisch analysieren.
13. Semantische Unterschiede und Gemeinsamkeiten innerhalb der Python-Sprache identifizieren und verstehen.

– **Sicherheits- und Effizienzaspekte in der Programmiersprache Python berücksichtigen:**
    11. Semantische Prinzipien zur Verbesserung der Sicherheit und Effizienz von Python-Programmen anwenden.
    12. Best Practices zur Vermeidung von häufigen semantischen Fehlern und Sicherheitslücken in Python entwickeln.
– **Praktische Projekte planen und durchführen:**
    5. Eigenständige Projekte zur semantischen Analyse und Optimierung von Python-Programmen planen, durchführen und dokumentieren.
    6. Effektive Teamarbeit und Kommunikationsfähigkeiten im Entwicklungsprozess fördern.

# Modulbeschreibung: AISE207 – Softwaretechnik I

**Leitidee**

Der Kurs *"Softwaretechnik I"* legt den Grundstein für die Ausbildung von Softwareingenieuren, indem er die wesentlichen Prinzipien und Praktiken der Softwareentwicklung vermittelt. Die Studierenden lernen, wie man Anforderungen systematisch erfasst und dokumentiert, um eine solide Basis für den gesamten Entwicklungsprozess zu schaffen. Durch den Fokus auf Softwareentwurf und Implementierung lernen die Studierenden, qualitativ hochwertige Software zu entwickeln, die den Bedürfnissen der Benutzer entspricht. Die Einführung in agile Methoden fördert die Teamarbeit und ermöglicht den Studierenden, effektiv in dynamischen Umgebungen zu arbeiten. Durch praktische Projekte wird das erlernte Wissen direkt angewendet, was zu einem tiefen Verständnis der Softwareentwicklung führt.

**Typ**

Pflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

Nach erfolgreichem Abschluss dieses Moduls "Software Technik I" werden die Studierenden in der Lage sein:

- Verstehen der Grundlagen der Softwaretechnik und der Softwareentwicklungsprozesse.
- Fähigkeit zur Anforderungsanalyse und Erstellung von Lasten- und Pflichtenheften.
- Anwendung von Softwareentwurfsprinzipien und Erstellung von UML-Diagrammen.
- Implementierung von Best Practices in der Programmierung, einschließlich Test-Driven Development (TDD).
- Durchführung und Automatisierung von Softwaretests sowie effektive Teamarbeit mit agilen Methoden.

# Modulbeschreibung: AISE208 – Softwaretechnik II

**Leitidee**

*Softwaretechnik II* baut auf den Grundlagen des ersten Kurses auf und vertieft das Wissen über fortgeschrittene Architekturen und Technologien. Die Studierenden werden befähigt, komplexe Softwarelösungen zu entwerfen und zu implementieren, indem sie moderne Entwurfsmuster und Prinzipien anwenden. Der Kurs fördert das Verständnis für die Herausforderungen verteilter Systeme und die Bedeutung von Softwarequalität in der Entwicklung. Durch die Kombination von Projektmanagement und praktischen Anwendungen erwerben die Studierenden umfassende Fähigkeiten, die sie in realen Softwareprojekten anwenden können.

**Typ**

Pflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

Nach erfolgreichem Abschluss dieses Moduls "Software Technik II" werden die Studierenden in der Lage sein:

- **Verständnis fortgeschrittener Architekturen**
  - Die Studierenden können verschiedene Architekturen (z. B. Microservices, Event-driven Architecture) beschreiben und deren Vor- und Nachteile in der Praxis erläutern.
- **Anwendung von Entwurfsmustern**
  - Die Studierenden sind in der Lage, fortgeschrittene Design Patterns in Softwareprojekten anzuwenden und ihre Implementierung zu erklären.
- **Entwicklung verteilter Systeme**
  14. Die Studierenden verstehen die Herausforderungen der Entwicklung verteilter Systeme und können Multithreading und Synchronisation effektiv nutzen.
- **Sicherstellung von Softwarequalität**
  13. Die Studierenden können Metriken zur Softwarequalität anwenden und Werkzeuge zur Qualitätssicherung (z. B. SonarQube) nutzen.
- **Projektmanagement und Dokumentation**
  14. Die Studierenden sind in der Lage, Projektpläne zu erstellen, Ressourcen zu managen und DevOps-Praktiken anzuwenden.
- **Praktische Anwendung fortgeschrittener Konzepte**
  13. Die Studierenden können komplexe Softwareanwendungen unter Anwendung fortgeschrittener Techniken und Prinzipien in einem Team entwickeln.
- **Reflexion und Präsentation**
  7. Die Studierenden entwickeln Fähigkeiten zur kritischen Reflexion ihrer Projekterfahrungen und zur effektiven Präsentation ihrer Ergebnisse vor einem Publikum.

# Modulbeschreibung: AISE209 – Compilerbau

**Leitidee**

Die Vorlesung Compilerbau enthält alles, was Sie für die Implementierung einer vollwertigen, effizienten Skriptsprache benötigen. Sie lernen sowohl High-Level-Konzepte rund um Parsing und Semantik als auch Details wie Bytecode-Darstellung und Garbage Collection. Der Kurs entwickelt einen eigenen Sprache und die Zugehörigen Interpreter und Run-time Umgebungen.

**Typ**

Pflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

Nach erfolgreichem Abschluss dieses Moduls "Compilerbau" werden die Studierenden in der Lage sein:

- **Einführung in Interpreter-Design und Entwicklung der Lox Sprache**
- **Lexikalische Analyse**
- **Syntaxanalyse**
- **Interpreter-Implementierung (Tree-Walking)**
- **Erweiterung der Sprache Lox**
- **Bytecode-Interpreter und virtuelle Maschinen**
- **Praktische Projekte und Übungen**
- Fokus des Kurses:Der Kurs orientiert sich eng am Aufbau und den Beispielen aus *Crafting Interpreters*.
- Es wird besonderes Augenmerk auf die schrittweise Implementierung eines Interpreters gelegt, sowohl als Tree-Walking-Interpreter in Java als auch als Bytecode-Interpreter in C.
- Theoretische Inhalte werden durch praktische Übungen und Implementierungsprojekte ergänzt, die das Verständnis der Konzepte vertiefen.

# Modulbeschreibung: AISE210 – Softwaremanagement

**Leitidee**

"Software Management" vermittelt den Studierenden praxisorientierte Kenntnisse und Fähigkeiten im Bereich der Verwaltung und Leitung von Softwareentwicklungsprojekten. Der Fokus liegt auf den wesentlichen Aspekten des Softwareprojektmanagements, einschließlich Planung, Organisation, Führung, Kontrolle und Abschluss von Projekten. Durch die Kombination von theoretischen Grundlagen und praktischen Übungen lernen die Studierenden, wie sie Softwareprojekte effizient und erfolgreich steuern können. Der Kurs deckt verschiedene Methoden und Techniken ab, darunter agile Ansätze, klassische Projektmanagementmethoden, Risikomanagement, Teamführung und Kommunikation mit Stakeholdern. Ziel ist es, die Studierenden darauf vorzubereiten, in realen Projektszenarien Verantwortung zu übernehmen und qualitativ hochwertige Softwarelösungen termingerecht und innerhalb des Budgets zu liefern.

**Typ**

Pflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

Nach erfolgreichem Abschluss dieses Moduls "Softwaremanagement" werden die Studierenden in der Lage sein:

- **Grundlagen des Softwareprojektmanagements verstehen:**
  - Die wesentlichen Konzepte und Prinzipien des Projektmanagements im Kontext der Softwareentwicklung kennen.
  - Den Unterschied zwischen agilen und klassischen Projektmanagementmethoden verstehen.
- **Projektplanung und -organisation beherrschen:**
  - Effektive Techniken zur Projektplanung und -organisation anwenden, einschließlich Zeit- und Ressourcenmanagement.
  - Meilensteine, Zeitpläne und Budgets erstellen und verwalten.
- **Teamführung und Kommunikation entwickeln:**
  15. Führungsstile und -techniken kennenlernen und anwenden, um ein effektives Projektteam zu leiten.
  16. Effektive Kommunikationsstrategien entwickeln, um mit Teammitgliedern und Stakeholdern zu interagieren.
- **Risikomanagement durchführen:**
  14. Methoden zur Identifikation, Bewertung und Minderung von Projektrisiken anwenden.
  15. Strategien entwickeln, um unvorhergesehene Herausforderungen im Projektverlauf zu bewältigen.
- **Qualitätsmanagement implementieren:**
  15. Qualitätsstandards und -prozesse definieren und in Softwareprojekten umsetzen.

16. Methoden zur Sicherstellung der Softwarequalität während des gesamten Entwicklungszyklus anwenden.

– **Agile Methoden und Tools nutzen:**
  14. Agile Projektmanagementmethoden wie Scrum und Kanban verstehen und implementieren.
  15. Werkzeuge zur Unterstützung agiler Prozesse effektiv einsetzen.
– **Projektabschluss und -bewertung durchführen:**
  8. Methoden zur erfolgreichen Beendigung von Projekten kennen und anwenden.
  9. Projektergebnisse bewerten und Lessons Learned dokumentieren.

# Modulbeschreibung: AISE211 – Requirements Engineering

**Leitidee**

Der Kurs "Software Requirements Engineering" vermittelt den Studierenden praxisorientierte Methoden und Techniken zur Erfassung, Analyse, Spezifikation und Verwaltung von Softwareanforderungen. Ziel ist es, die Studierenden darauf vorzubereiten, effektive und nachvollziehbare Anforderungen zu entwickeln, die als Grundlage für den gesamten Softwareentwicklungsprozess dienen. Durch eine Kombination aus theoretischem Wissen und praktischen Übungen lernen die Studierenden, wie sie Anforderungen systematisch erheben, dokumentieren und validieren können. Der Kurs legt besonderen Wert auf die Anwendung von Best Practices und den Einsatz unterstützender Werkzeuge, um die Qualität und Effizienz im Requirements Engineering zu steigern. Zudem werden Methoden zur Kommunikation und Zusammenarbeit mit Stakeholdern behandelt, um eine erfolgreiche Umsetzung der Anforderungen sicherzustellen. Abschließend arbeiten die Studierenden an einem praxisnahen Projekt, in dem sie die erlernten Konzepte anwenden und vertiefen.

**Typ**

Pflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

Nach erfolgreichem Abschluss dieses Moduls "Requirements Engineering" werden die Studierenden in der Lage sein:

- **Grundlagen und Prinzipien verstehen:**
  - Die Bedeutung und Ziele des Requirements Engineering im Softwareentwicklungsprozess erkennen.
  - Unterschiedliche Arten von Anforderungen (funktional, nicht-funktional, domänenspezifisch) unterscheiden können.
- **Methoden und Techniken anwenden:**
  - Effektive Techniken zur Anforderungserhebung wie Interviews, Workshops und Umfragen einsetzen.
  - Anforderungen systematisch analysieren, priorisieren und dokumentieren.
- **Werkzeuge nutzen:**
  17. Unterstützende Tools für das Requirements Engineering einsetzen, z.B. für die Modellierung und Verwaltung von Anforderungen.
  18. Einsatz von Software zur Versionskontrolle und Nachverfolgung von Anforderungen verstehen.
- **Kommunikation und Zusammenarbeit fördern:**
  16. Effektive Kommunikation mit Stakeholdern zur Klärung und Validierung von Anforderungen gestalten.
  17. Zusammenarbeit in interdisziplinären Teams zur Erarbeitung und Umsetzung von Anforderungen koordinieren.

- **Qualitätssicherung betreiben:**
  17. Methoden zur Überprüfung und Validierung von Anforderungen anwenden, um deren Qualität sicherzustellen.
  18. Risiken im Requirements Engineering identifizieren und geeignete Gegenmaßnahmen entwickeln.
- **Praxisprojekte durchführen:**
  16. Ein eigenständiges Projekt im Bereich Requirements Engineering planen, durchführen und präsentieren.
  17. Erlernte Methoden und Techniken in einem praxisnahen Kontext anwenden und reflektieren.

# Modulbeschreibung: AISE401 Requirements Engineering

**Leitidee**

Das Modul "Requirements Engineering" beschäftigt sich mit den Techniken, Methoden und Prozessen zur Erfassung, Analyse, Spezifikation und Verifikation von Anforderungen für Softwareprodukte und -systeme. Es hebt die entscheidende Rolle von Anforderungen im Softwareentwicklungsprozess hervor und zeigt, wie qualitativ hochwertige Anforderungen zur Entwicklung erfolgreicher Softwareprodukte beitragen können.

**Typ**

Pflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

- Verständnis der Bedeutung von Requirements Engineering im Softwareentwicklungsprozess.
- Erlernen der Techniken zur Ermittlung, Spezifikation und Validierung von Anforderungen.
- Entwicklung von Fähigkeiten zur effektiven Kommunikation und Dokumentation von Anforderungen zwischen verschiedenen Stakeholdern.
- Reflexion über die Herausforderungen und Best Practices im Requirements Engineering.

# Modulbeschreibung: AISE501 - AI in SE I

**Leitidee**

Der Kurs "AI in Software Engineering I" fokussiert sich auf die Integration und Anwendung von Large Language Models (LLMs) sowie verwandten Techniken im Softwareentwicklungsprozess. Zusätzlich werden wesentliche Software Engineering Prinzipien, behandelt, um die Qualität und Wartbarkeit von Softwareprojekten zu gewährleisten. Studierende erwerben ein tiefgehendes Verständnis der Funktionsweise von LLMs und lernen, wie fortschrittliche KI-Methoden eingesetzt werden können, um innovative Lösungen für komplexe Probleme im Kontext der Softwareentwicklung zu entwickeln. Der Kurs verbindet theoretische Grundlagen der Künstlichen Intelligenz mit praktischen Anwendungen und etablierten Software Engineering Praktiken, indem die Studierenden eigene AI-gestützte Tools und Features für Code-Editoren entwickeln. Durch praxisorientierte Projekte und Übungen wird die Fähigkeit gefördert, LLMs effektiv zu nutzen und in bestehende Softwareentwicklungsmethoden zu integrieren, während gleichzeitig hochwertige, saubere und wartbare Codebasen erstellt werden.

**Typ**

Pflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

Nach erfolgreichem Abschluss dieses Moduls "AI in Software Engineering" werden die Studierenden in der Lage sein:

- **Grundlagen von LLMs verstehen:**
  - Die Architektur und Funktionsweise von Large Language Models wie GPT-4 kennen.
  - Die Prinzipien von Retrieval-Augmented Generation (RAG)
  - Finetuning and Anwendungen auf Code Erzeugung und Fehleranalyse
- **LLM-Techniken anwenden:**
  - Methoden zur Anpassung und Feinabstimmung von LLMs für spezifische Softwareentwicklungsaufgaben einsetzen
  - Techniken wie Prompt Engineering und Few-Shot Learning effektiv nutzen.
- **Software Engineering Prinzipien integrieren:**
  19. Die Prinzipien von Clean Code verstehen und anwenden, um qualitativ hochwertigen, lesbaren und wartbaren Code zu schreiben.
  20. Designprinzipien und Best Practices im Software Engineering in die Entwicklung von AI-gestützten Tools integrieren.
- **AI-gestützte Tools entwickeln:**
  18. Funktionen wie intelligente Autovervollständigung, Code-Generierung und Fehlererkennung in eigenen Code-Editoren implementieren.
  19. Kontext bezogene Methoden zur Verbesserung der Code-Analyse und -Generierung nutzen.
- **Eigene Projekte realisieren:**
  19. Ein eigenes Projekt zur Entwicklung eines AI-gestützten Code-Editors planen, durchführen und dokumentieren.

20. Best Practices der Softwareentwicklung und KI-Integration anwenden, um einen funktionsfähigen und effizienten Code-Editor zu erstellen.

– **Performance und Benutzerfreundlichkeit optimieren:**
  18. Methoden zur Evaluierung und Verbesserung der Performance von LLM-Modulen in Code-Editoren anwenden.
  19. Benutzerfreundliche Schnittstellen und intuitive Interaktionen gestalten, die den Programmierprozess unterstützen.
– **Ethik und Verantwortung berücksichtigen:**
  10. Ethische Aspekte und gesellschaftliche Implikationen des Einsatzes von LLMs im Software-Engineering reflektieren.
  11. Verantwortungsbewusste und nachhaltige KI-Lösungen entwickeln, die den Anforderungen der Entwickler gerecht werden.

# Modulbeschreibung: AISE502 - AI in SE II

**Leitidee**

Das Modul "AI in Software Engineering II" baut auf den Grundlagen des ersten Teils auf und vertieft das Verständnis für die Anwendung fortgeschrittener Künstlicher Intelligenz (KI) Techniken im Softwareentwicklungsprozess. Der Fokus liegt auf der Verbesserung der Codequalität durch KI-gestützte Methoden, der Entwicklung intelligenter AI Agents sowie der Anwendung komplexer Prompt Engineering Techniken wie Chain of Thoughts und Graphs of Thoughts. Zusätzlich werden wesentliche Aspekte der Softwarearchitektur behandelt, um die Integration von KI in robuste und skalierbare Softwarelösungen zu gewährleisten. Studierende lernen, wie sie fortschrittliche KI-Modelle nutzen können, um effizienteren, saubereren und robusteren Code zu schreiben sowie komplexe Entwicklungsaufgaben zu automatisieren und zu optimieren. Durch praxisorientierte Projekte und intensive Übungen erwerben die Studierenden die Fähigkeiten, komplexe AI-gestützte Softwarelösungen zu entwerfen, zu implementieren und zu evaluieren, die den gesamten Softwareentwicklungszyklus unterstützen und verbessern.

**Typ**

Pflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

- **Fortgeschrittene KI-Techniken verstehen:**
  - Die Konzepte und Anwendungen von AI Agents im Software Engineering kennen.
  - Fortgeschrittene Prompt Engineering Techniken wie Chain of Thoughts und Graphs of Thoughts verstehen und anwenden können.
- **Komplexe AI-gestützte Software-Tools entwickeln:**
  - Methoden zur automatisierten Code-Optimierung und -Refactoring mit KI einsetzen.
  - Tools zur statischen Codeanalyse und Fehlerbehebung mithilfe von KI entwickeln und nutzen.
- **Softwarearchitektur integrieren:**
  21. Grundlegende und fortgeschrittene Prinzipien der Softwarearchitektur verstehen und anwenden.
  22. Architekturmuster für die Integration von KI in Softwarelösungen kennenlernen und implementieren, insbesondere Microservices.
  23. Best Practices für die Gestaltung skalierbarer und wartbarer KI-gestützter Systeme anwenden.
- **Intelligente AI Agents entwickeln:**
  20. AI Agents konzipieren und implementieren, die spezifische Aufgaben im Softwareentwicklungsprozess autonom ausführen.
  21. Interaktive und adaptive AI Agents erstellen, die mit Entwicklern kommunizieren und kooperieren können.
- **Komplexe Prompt Engineering Techniken anwenden:**
  21. Chain of Thoughts zur Verbesserung der Problemlösungsfähigkeiten von LLMs einsetzen.

22. Graphs of Thoughts zur Strukturierung und Visualisierung von Denkprozessen in KI-Modellen nutzen.

- **Fortgeschrittene Projekte realisieren:**
  20. Komplexe AI-gestützte Softwarelösungen planen, entwickeln und dokumentieren.
  21. Best Practices der Softwareentwicklung und KI-Integration anwenden, um hochqualitative und wartbare Softwareprojekte zu erstellen.
- **Performance und Skalierbarkeit optimieren:**
  12. Methoden zur Evaluierung und Verbesserung der Performance von fortgeschrittenen KI-Modulen in Softwareanwendungen anwenden.
  13. Skalierbare und effiziente KI-Lösungen für größere Softwareprojekte entwickeln.
- **Ethik und Verantwortung weiter vertiefen:**
  - Vertiefte ethische Überlegungen und gesellschaftliche Implikationen des Einsatzes von fortgeschrittener KI im Software Engineering reflektieren.
  - Verantwortungsbewusste und nachhaltige KI-Lösungen entwickeln, die ethischen Standards entsprechen.

# Modulbeschreibung: AISE503 - AI Methods in Software Testing

**Leitidee**

Das Modul "AI Methods in Software Testing" fokussiert sich auf das Testen von Software durch AI-gestützte Methoden, basierend auf den bewährten Prinzipien, Werkzeugen und Ideen des Test-driven Software Development (TDD). AI-Methoden tragen dazu bei, die Qualität von Softwareprodukten zu verbessern, die Testeffizienz zu steigern und gleichzeitig die Kosten sowie den Zeitaufwand für Softwaretests zu reduzieren. Der Kurs vermittelt den Studierenden fundierte Kenntnisse über den Einsatz von Künstlicher Intelligenz im Softwaretestprozess, einschließlich der Automatisierung der Testfallgenerierung, der intelligenten Auswertung von Testergebnissen, der automatisierten Codekontrolle und der Integration von AI-gestützten Testmethoden in CI/CD-Frameworks. Durch eine Kombination aus theoretischen Vorlesungen, praktischen Übungen und einem abschließenden Projekt erwerben die Studierenden die Fähigkeit, AI-Methoden effektiv in den TDD-Zyklus zu integrieren und somit die Softwarequalität nachhaltig zu erhöhen.

**Typ**

Wahlpflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

Nach Abschluss des Kurses sollen die Studierenden:

- **Grundlagen des Test-driven Software Development (TDD) verstehen:**
  - Die grundlegenden Konzepte und Prinzipien des TDD kennen.
  - Unterschiede und Gemeinsamkeiten zwischen TDD und anderen Testmethoden erkennen.
- **AI-Methoden im Kontext von TDD anwenden:**
  - Verschiedene AI-Techniken wie maschinelles Lernen, Deep Learning und natürliche Sprachverarbeitung im TDD-Prozess einsetzen.
  - Automatisierung von Testfällen und Testskripten mittels AI-Methoden implementieren.
- **AI-gestützte Testfallgenerierung durchführen:**
  24. Methoden zur automatischen Generierung von Testfällen mittels AI entwickeln und anwenden.
  25. Einsatz von AI zur Verbesserung der Testabdeckung und -effizienz.
- **Intelligente Auswertung von Testergebnissen:**
  22. Methoden zur automatischen Fehlererkennung und -vorhersage mittels AI entwickeln und anwenden.
  23. Analyse von Fehlerdaten zur Optimierung des Testprozesses durchführen.
- **Automatisierte Codekontrolle mit AI implementieren:**
  23. AI-Methoden zur Überprüfung und Verbesserung von Codequalität und -sicherheit einsetzen.
  24. Integration von AI-gestützter Codeanalyse in den TDD-Zyklus durchführen.
- **Integration von AI in CI/CD-Frameworks:**

22. Methoden zur nahtlosen Integration von AI-gestützten Testmethoden in Continuous Integration und Continuous Deployment (CI/CD) Pipelines kennen und anwenden.
23. Nutzung von Tools in Kombination mit AI-gestützten Testwerkzeugen.

- **Best Practices für AI-gestütztes Softwaretesten anwenden:**
  14. Best Practices zur nahtlosen Integration von AI in den TDD-Zyklus kennen und umsetzen.
  15. Design Patterns und Architekturprinzipien für die Integration von AI nutzen.
- **Sicherheits- und Datenschutzaspekte berücksichtigen:**
  - Sicherheitsherausforderungen und Datenschutzanforderungen bei der Nutzung von AI im Testprozess erkennen und adressieren.
  - Techniken zur Sicherstellung der Datensicherheit und Compliance in AI-gestützten Testsystemen anwenden.
- **Praktische Projekte planen und durchführen:**
  - Eigenständige Projekte zur Implementierung von AI-Methoden im TDD-Prozess planen, durchführen und dokumentieren.
  - Effektive Teamarbeit und Kommunikationsfähigkeiten im Entwicklungsprozess fördern.

# Modulbeschreibung: AISE504 - AI in Software Design and Engineering

**Leitidee**

Das Modul "AI in Software Design und Engineering" konzentriert sich auf den Einsatz von Künstlicher Intelligenz (KI) und Machine Learning (ML) Techniken zur Unterstützung und Optimierung von Softwaredesign- und Engineering-Prozessen. Der Kurs verfolgt einen systematischen, disziplinierten, strukturierten und quantifizierten Ansatz im Konzeptionsprozess zur Implementierung von Softwaresystemen. Dabei liegt der Fokus auf dem Einsatz von AI-Tools, die den Softwaredesign- und Architekturprozess unterstützen, um die Effizienz, Qualität und Wartbarkeit von Softwaresystemen nachhaltig zu verbessern. Durch eine Kombination aus theoretischen Vorlesungen, praktischen Übungen und einem abschließenden Projekt erwerben die Studierenden die Fähigkeiten, AI-Methoden effektiv in den Softwaredesign- und Engineering-Prozess zu integrieren.

**Typ**

Wahlpflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

- **Grundlagen von AI und ML im Software Design verstehen:**
  - Die grundlegenden Konzepte und Prinzipien von AI und ML im Kontext des Softwaredesigns und -engineerings kennen.
  - Unterschiede und Gemeinsamkeiten zwischen traditionellen Softwareentwicklungsansätzen und AI-gestützten Methoden erkennen.
- **Systematische und strukturierte Designprozesse entwickeln:**
  - Methoden zur systematischen und disziplinierten Nutzung von AI im Konzeptionsprozess von Softwaresystemen verstehen und anwenden.
  - Quantitative Metriken zur Bewertung und Verbesserung von Designentscheidungen durch AI nutzen.
- **AI-Methoden im Softwaredesign anwenden:**
  26. Verschiedene AI-Techniken wie maschinelles Lernen, Deep Learning und natürliche Sprachverarbeitung (NLP) im Softwaredesign einsetzen.
  27. Einsatz von AI-Tools zur Automatisierung und Optimierung von Designprozessen implementieren.
- **Graduelle Verbesserungsansätze implementieren:**
  24. Techniken zur kontinuierlichen Verbesserung von Softwaresystemen mittels AI-gestützter Analysen und Optimierungen entwickeln.
  25. Einsatz von AI zur Identifikation und Implementierung von Verbesserungsmöglichkeiten im Entwicklungszyklus.
- **AI-gestützte Designwerkzeuge und Methoden effektiv nutzen:**
  25. Moderne AI-gestützte Designwerkzeuge und Methoden wie **UML-Assistenten, Requirements Engineering Tools, Architektur-Analyse-Tools** kennenlernen und einsetzen.

26. Integration dieser Werkzeuge in den Softwareentwicklungsprozess durchführen.

- **Analyse und Bewertung von Designlösungen:**
  24. Lösungen auf ihre Effektivität, Effizienz und Qualität hin bewerten.
  25. Performanz von AI-gestützten Designmethoden analysieren und optimieren.
- **Sicherheits- und Datenschutzaspekte berücksichtigen:**
  16. Sicherheitsherausforderungen und Datenschutzanforderungen bei der Nutzung von AI im Softwaredesign erkennen und adressieren.
  17. Techniken zur Sicherstellung der Datensicherheit und Compliance in AI-gestützten Designprozessen anwenden.
- **Praktische Projekte planen und durchführen:**
  - Eigenständige Projekte zur Implementierung von AI-Methoden im Softwaredesign planen, durchführen und dokumentieren.
  - Effektive Teamarbeit und Kommunikationsfähigkeiten im Entwicklungsprozess fördern.

# Modulbeschreibung: AISE505 - Security in Artificial Intelligence

**Leitidee**

Das Modul "Security in Artificial Intelligence" befasst sich mit den Sicherheits- und Datenschutzherausforderungen, die mit der Implementierung und Nutzung künstlicher Intelligenz (KI) und Large Language Models (LLMs) in verschiedenen Domänen verbunden sind. Die Studierenden lernen sowohl die Bedrohungslandschaft kennen, die KI- und LLM-Systeme betrifft, als auch die Techniken und Best Practices, um diese Systeme zu schützen und robust zu gestalten. Der Kurs legt besonderen Wert auf einen systematischen, disziplinierten und strukturierten Ansatz zur Identifikation, Bewertung und Abwehr von Sicherheitsrisiken in KI-Systemen. Durch theoretische Vorlesungen, praktische Übungen und ein abschließendes Projekt erwerben die Studierenden die Fähigkeiten, Sicherheitsstrategien effektiv in den Entwicklungs- und Betriebsprozess von KI-Systemen zu integrieren.

**Typ**

Wahlpflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

Nach Abschluss des Kurses sollen die Studierenden:

Nach Abschluss des Kurses sollen die Studierenden:

- **Verständnis der Bedrohungslandschaft für KI- und LLM-Systeme:**
  - Die spezifischen Bedrohungen und Schwachstellen von KI- und LLM-Systemen identifizieren.
  - Unterschiede zwischen traditionellen IT-Systemen und KI-Systemen hinsichtlich Sicherheitsanforderungen verstehen.
- **Kenntnis von Sicherheits- und Datenschutzherausforderungen in KI:**
  - Datenschutzprobleme im Zusammenhang mit der Nutzung von KI und LLMs erkennen und analysieren.
  - Ethische Implikationen und gesellschaftliche Auswirkungen von Sicherheitslücken in KI-Systemen reflektieren.
- **Analyse und Abwehr von Jail-Break-Angriffen auf LLMs:**
  28. Verständnis der Mechanismen und Techniken hinter Jail-Breaks von LLMs.
  29. Strategien zur Erkennung, Abwehr und Prävention von Jail-Break-Angriffen entwickeln.
- **Anwendung von Sicherheitstechniken und Best Practices:**
  26. Techniken zur Absicherung von KI-Modellen gegen Angriffe wie Adversarial Attacks, Datenvergiftung und Modell-Inversion verstehen und anwenden.
  27. Best Practices zur Sicherstellung der Robustheit und Vertrauenswürdigkeit von KI-Systemen implementieren.
- **Integration von Sicherheitsmaßnahmen in den KI-Entwicklungszyklus:**

27. Sicherheitsaspekte systematisch in den gesamten Entwicklungsprozess von KI-Systemen integrieren.
28. Methoden zur kontinuierlichen Überwachung und Verbesserung der Sicherheit von KI-Anwendungen anwenden.

- **Verwendung von Sicherheitstools und Frameworks für KI:**
  26. Moderne Sicherheitstools und Frameworks, die speziell für KI-Systeme entwickelt wurden, kennenlernen und nutzen.
  27. Sicherheitsbewertungen und Audits von KI-Systemen durchführen.
- **Entwicklung von Strategien zur Risikominimierung:**
  18. Strategien zur Identifikation, Bewertung und Minimierung von Sicherheitsrisiken in KI-Systemen entwickeln.
  19. Notfallpläne und Reaktionsstrategien für Sicherheitsvorfälle in KI-Anwendungen erstellen.
- **Praktische Fähigkeiten in der Sicherung von KI-Systemen:**
  - Praktische Erfahrungen in der Implementierung von Sicherheitsmaßnahmen in realen KI-Projekten sammeln.
  - Sicherheitslücken in bestehenden KI-Anwendungen identifizieren und beheben.
- **Ethische und rechtliche Aspekte der KI-Sicherheit berücksichtigen:**
  - Rechtliche Rahmenbedingungen und Compliance-Anforderungen für den Einsatz von KI-Systemen verstehen.
  - Ethische Richtlinien und Standards für die Entwicklung und Nutzung von sicheren KI-Systemen anwenden.

# Modulbeschreibung: AISE506 - Software Repository Mining and Data Analytics

**Leitidee**

Das Modul "Software Repository Mining and Data Analytics" befasst sich mit der Extraktion und Analyse von Software Repositorien. Durch die Analyse können Erkenntnisse genutzt werden, um Softwaresysteme, Projekte oder Software Engineering Praktiken auf Schwachstellen geprüft und optimiert werden.

**Typ**

Wahlpflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

Nach Abschluss des Kurses sollen die Studierenden:

- **Grundlagen des Software Repository Mining verstehen:**
  - Die Struktur und den Inhalt von Software-Repositorien kennenlernen.
  - Die Bedeutung und den Nutzen der Analyse von Software-Repositorien für die Softwareentwicklung und -wartung erkennen.
- **Techniken der Datenextraktion und -aufbereitung anwenden:**
  - Methoden zur Extraktion von Daten aus verschiedenen Software-Repositorien verstehen und anwenden.
  - Datenbereinigung und -vorbereitung für die Analyse durchführen.
- **Klassische Information Retrieval-Verfahren beherrschen:**
  30. Grundlagen und Techniken des Information Retrieval verstehen.
  31. Suchalgorithmen und Ranking-Methoden implementieren und anwenden.
- **Semantische Suchen und LLM-basierte Q&A einsetzen:**
  28. Semantische Suchtechniken zur Verbesserung der Suchgenauigkeit nutzen.
  29. Large Language Models (LLM) für Frage-Antwort-Systeme in der Repository-Analyse einsetzen.
- **Retrieval-Augmented Generation (RAG) und erweiterte RAG-Techniken nutzen:**
  29. Die Konzepte und Anwendungen von RAG verstehen.
  30. Erweiterte RAG-Techniken zur Generierung relevanter Inhalte aus Repository-Daten anwenden.
- **Code-Analysetools effektiv einsetzen:**
  28. Verwendung und Konfiguration von Code-Analysetools wie SonarQube zur Identifikation von Codequalität und Sicherheitslücken.
  29. Analyseergebnisse interpretieren und Maßnahmen zur Verbesserung ableiten.
- **Datenanalyse und Visualisierungstechniken anwenden:**
  20. Statistische Methoden und Data-Analytics-Techniken zur Analyse von Repository-Daten nutzen.
  21. Ergebnisse mittels Visualisierungstools anschaulich darstellen.

- **Best Practices und ethische Aspekte der Repository-Analyse berücksichtigen:**
  - Best Practices für die Durchführung von Repository Mining und Datenanalysen anwenden.
  - Ethische und datenschutzrechtliche Aspekte bei der Analyse von Software-Repositorien verstehen und berücksichtigen.
- **Praktische Projekte planen und durchführen:**
  - Eigenständige Projekte zur Analyse von Software-Repositorien planen, durchführen und dokumentieren.
  - Teamarbeit und effektive Kommunikation im Entwicklungsprozess fördern.

# Modulbeschreibung: AISE507 - Integration of AI in Conventional Software

**Leitidee**

Der Kurs "Integration of AI in Conventional Software" untersucht, wie Künstliche Intelligenz (KI) nahtlos in herkömmliche Softwareinfrastrukturen und -systeme integriert werden kann. Die Studierenden lernen sowohl die technischen Herausforderungen kennen, die mit einer solchen Integration einhergehen, als auch die besten Praktiken, um die Vorteile von KI in bestehenden Softwarelösungen voll auszuschöpfen. Der Kurs deckt verschiedene Aspekte der KI-Integration ab, einschließlich Datenmanagement, API-Entwicklung, Skalierbarkeit, Sicherheit und ethische Überlegungen. Besonderer Fokus liegt auf der Integration von **AI-Agents** sowie dem Umgang mit stochastischen Eigenschaften von LLMs, die variable Antworten und Laufzeiten aufweisen. Durch eine Kombination aus theoretischen Vorlesungen, praktischen Übungen und einem abschließenden Projekt erwerben die Studierenden die Fähigkeit, KI-Komponenten effektiv in bestehende Softwarearchitekturen zu integrieren und so die Funktionalität und Effizienz der Systeme zu verbessern.

**Typ**

Wahlpflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

- **Grundlagen der KI-Integration verstehen:**
  - Die grundlegenden Konzepte und Prinzipien der KI-Integration in konventionelle Software kennen.
  - Unterschiede und Gemeinsamkeiten zwischen herkömmlicher Softwareentwicklung und KI-gestützter Entwicklung erkennen.
- **Technische Herausforderungen meistern:**
  - Technische Hürden bei der Integration von KI, wie Datenmanagement, Modellbereitstellung und Systemkompatibilität, identifizieren und lösen.
  - Methoden zur Sicherstellung der Skalierbarkeit und Performance von integrierten KI-Systemen anwenden.
- **Best Practices für die KI-Integration anwenden:**
  32. Best Practices zur nahtlosen Integration von KI in bestehende Softwarearchitekturen kennen und umsetzen.
  33. Design Patterns und Architekturprinzipien für die Integration von KI nutzen.
- **Werkzeuge und Frameworks effektiv einsetzen:**
  30. Moderne Tools und Frameworks wie  für die Bereitstellung und Verwaltung von KI-Modellen nutzen.
  31. APIs und Microservices zur Integration von KI-Funktionalitäten in bestehende Systeme entwickeln.
- **Datenmanagement und -vorbereitung optimieren:**

31. Methoden zur effektiven Datenvorbereitung und -verwaltung für KI-Anwendungen verstehen und anwenden.
32. Datenpipelines und ETL-Prozesse zur Unterstützung der KI-Integration implementieren.

- **Sicherheits- und Datenschutzaspekte berücksichtigen:**
  30. Sicherheitsherausforderungen und Datenschutzanforderungen bei der Integration von KI in Softwarelösungen erkennen und adressieren.
  31. Techniken zur Sicherstellung der Datensicherheit und Compliance in KI-gestützten Systemen anwenden.
- **AI-Agents entwickeln und integrieren:**
  22. Konzepte und Architektur von **AI-Agents** verstehen und deren Einsatzmöglichkeiten in konventionellen Softwarelösungen identifizieren.
  23. Methoden zur Entwicklung und Integration autonomer und interaktiver AI-Agents in bestehende Systeme anwenden.
- **Stochastische AI-Ergbenisse handhaben:**
  - Die Natur stochastischer Methoden in der KI verstehen, einschließlich variabler Antworten und Laufzeiten.
  - Strategien zur Handhabung und Optimierung von Systemen entwickeln, die auf stochastischen AI-Methoden basieren.
- **Ethische und gesellschaftliche Implikationen reflektieren:**
  - Ethische Überlegungen und gesellschaftliche Auswirkungen der KI-Integration in Softwarelösungen verstehen.
  - Verantwortungsbewusste und nachhaltige KI-Anwendungen entwickeln.
- **Praktische Projekte planen und durchführen:**
  - Eigenständige Projekte zur Integration von KI in bestehende Softwarelösungen planen, durchführen und dokumentieren.
  - Effektive Teamarbeit und Kommunikationsfähigkeiten im Entwicklungsprozess fördern.

# Modulbeschreibung: AISE508 – Programmiersprachen

**Leitidee**

Der Kurs *"Programmiersprachen"* bietet den Studierenden eine vertiefte Auseinandersetzung mit verschiedenen Programmiersprachen und deren spezifischen Paradigmen. Der Fokus liegt auf der praktischen Programmierung in Java und C, wobei die Studierenden die erlernten Konzepte aus vorhergehenden Kursen anwenden und vertiefen. In diesem Kurs werden die Studierenden mit der Syntax, Semantik und den typischen Anwendungsszenarien dieser Programmiersprachen vertraut gemacht. Dabei wird ein besonderer Wert auf die Verknüpfung der praktischen Programmierung mit den semantischen Konzepten gelegt, um ein umfassendes Verständnis für die Bedeutung der Semantik in der Programmierung zu entwickeln.

**Typ**

Pflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

Nach erfolgreichem Abschluss dieses Moduls "Programmiersprachen" werden die Studierenden in der Lage sein:

Nach Abschluss des Kurses sind die Studierenden in der Lage:

- Die Syntax und die grundlegenden Konzepte von Java und C zu verstehen und anzuwenden.
- Objektorientierte Programmierung in Java zu implementieren und die semantischen Unterschiede zu imperativen Ansätzen in C zu erkennen.
- Die Speicherverwaltung in C zu verstehen und sicher mit Pointern und dynamischen Datenstrukturen umzugehen.
- Die Konzepte und Praktiken aus Python auf Java und C zu übertragen und die Unterschiede in der Implementierung und Semantik zu analysieren.
- Anwendungsprojekte in Java und C zu entwickeln und dabei semantische Überlegungen zu integrieren.
- Die Bedeutung von Best Practices in der Programmierung zu erkennen und in ihrer eigenen Programmierpraxis anzuwenden.

# Modulbeschreibung: CDS106 - Machine Learning

**Leitidee**

Maschinelles Lernen wird immer wichtiger für Praxis und Wissenschaft. Dieser Kurs vermittelt grundlegende Prinzipien und Methoden des maschinellen Lernens.

**Typ**

Pflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

Nach erfolgreicher Teilnahme am Modul sind die Studierenden in der Lage,

- Supervised und Unsupervised Machine Learning Modelle gezielt auszuwählen, trainieren, evaluieren und die Resultate interpretieren
- die zugrundeliegenden mathematischen Modelle zu verstehen und dessen Einfluss auf die Daten zu erklären
- die erlernten Modelle in der Praxis mit realen Daten einzusetzen
- die Erkenntnisse zu formulieren, dokumentieren und kommunizieren

# Modulbeschreibung: CDS108 - Deep Learning

**Leitidee**

Machine und Deep Learning Modelle sind in aller Munde und werden gerne in vielen Projekten angewendet. Oft werden die Optimierungs- und Umsetzungsschritte oft vergessen. Jedes Modell lernt nur so gut wie die Eingabe vom Programmierer, oder Data Scientist. In diesem Kurs lernen die Studierende die wichtigen Schritte von Machine und Deep Learning Modelle an Hand von Praxisdaten aufzubauen. Sie werden sehen, dass die üblich benutzten Kost Funktionen nicht in jedem Fall anwendbar sind und erweiterte Funktionen genutzt werden müssen.

**Typ**

Pflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

Nach erfolgreicher Teilnahme am Modul sind die Studierenden in der Lage,

- Tensoren zu bestimmen.
- die Kost Funktion im Bezug auf Ihr Modell zu wählen.
- die passende Verbindungsfunktion anzuwenden.
- einfache und rekurrente Neuronale Netze zu bestimmen.
- ein Klassifizierungs-, Regressions- und Zeitreihenmodell umzusetzen.
- die Kost Funktion zu erweitern und in Machine Learning Modellen anzuwenden.

# Modulbeschreibung: CDS109 - Natural Language Processing

**Leitidee**

Ein beträchtlicher Anteil der täglich entstehenden Daten ist in Form von Text. So existieren an der Schnittstelle zwischen Mensch und Maschine zahlreiche Anwendungsgebiete. Die Studierenden lernen mittels Natural Language Processing (NLP), Techniken und Methoden zur maschinellen Verarbeitung natürlicher Sprache. Das gelernte wird in einem Praxisprojekt umgesetzt.

**Typ**

Pflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

Nach erfolgreicher Teilnahme am Modul sind die Studierenden in der Lage,

- Strukturen aus natürlichsprachlichen Daten zu extrahieren
- regelbasierte und lernbasierte Ansätze und Methoden aus dem Bereich NLP zu kennen, verstehen und einzusetzen
- NLP Komponenten und Systeme zu evaluieren
- den aktuellen Stand der Forschung in diesem Fachgebiet zu kennen
- aktuelle Limitierungen und Herausforderungen aufzuzeigen
- grundlegende Konzepte des Informationssuchverhalten zu kennen

# Modulbeschreibung: CDS112 - Big Data

**Leitidee**

Die weltweit verfügbare Datenmenge wächst rasant. Neue Methoden für die Verarbeitung von Big Data sind notwendig. Die Studierenden lernen in diesem Modul grosse Datenmengen aus unterschiedlichen Quellen und in unterschiedlichen Formaten zu verarbeiten. Dies mit dem Ziel einen Mehrwert aus den Daten zu erzeugen.

**Typ**

Wahlpflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

Nach erfolgreicher Teilnahme am Modul sind die Studierenden in der Lage,

- Big Data zu definieren und einzuordnen
- Big Data Konzepte und Architekturen zu erkennen, verstehen und gezielt einzusetzen
- Verteilte Speichersysteme und Rechenumgebungen für einfache Aufgabenstellungen einzusetzen
- Konzepte und Technologien für die Streaming-Datenverarbeitung verstehen und anwenden

# Modulbeschreibung: CDS115 - NoSQL-Datenbanken

**Leitidee**

Dieser Kurs vermittelt die grundlegenden Konzepte verteilter Datenbanksysteme. Die Studierenden erwerben die Kompetenz einen geeigneten Datenbanktyp für datenzentrierte Anwendungen zu evaluieren, ein Datenbankdesign zu erstellen, sowie komplexe Abfragen zu formulieren.

**Typ**

Wahlpflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

Nach erfolgreicher Teilnahme am Modul sind die Studierenden in der Lage,

- die Architektur von verteilten Datenbanken zu kennen
- einen geeigneten Datenbanktyp zu evaluieren
- ein Datenbankdesign für verteilte Datenbanken zu erzeugen
- komplexe Abfragen (DML, DDL, DCL, TCL) in einer Datenbanksprache zu formulieren

# Modulbeschreibung: CDS117 - Reinforcement Learning

**Leitidee**

Ergänzend zu Supervised und Unsupervised Learning ist das Reinforcement Learning Paradigma eine vielversprechende Methode. Insbesondere bei der Entscheidungsfindung autonomer Systeme im Bereich Robotik, Game oder auch im Gesundheitswesen. Mit dem Paradigma des Reinforcement Learnings lernt ein Agent eigenständig eine Strategie, um die erhaltenen Belohnungen zu maximieren. In dieser Vorlesung werden die Grundlagen des Reinforcement Learnings vermittelt.

**Typ**

Wahlpflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

Nach erfolgreicher Teilnahme am Modul sind die Studierenden in der Lage,

- die Anwendungen des Reinforcement Learnings zu kennen
- geeignete Reinforcement Learning Methoden einzusetzen
- Reinforcement Learning Modelle zu evaluieren

# Modulbeschreibung: CDS121 - Recommender Systems

**Leitidee**

Empfehlungssysteme sind weit verbreitet mit Anwendungen im Tourismus (e.g. Airbnb), im Bereich Filme (e.g. Netflix) und Musik (e.g. Spotify) In diesem Modul erwerben die Studierenden die Kompetenz ein Empfehlungssysteme zu entwerfen, umzusetzen und zu evaluieren.

**Typ**

Wahlpflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

Nach erfolgreicher Teilnahme am Modul sind die Studierenden in der Lage,

- Anwendungen von Empfehlungssystemen zu nennen
- ein Empfehlungssystem nach dem Ansatz des content-based oder collaborative Filtering Ansatzes zu entwerfen
- ein Empfehlungssystem zu entwickeln
- die Performance eines Empfehlungssystems zu evalulieren
- Die Bedeutung der user experience im Kontext von Empfehlungssystemen verstehen
- Den Einfluss von Empfehlungssysteme  auf Individuen und Gesellschaft kritisch zu reflektieren

# Modulbeschreibung: CDS122 - Time Series Analysis

**Leitidee**

Das Modul "Time Series Analysis" konzentriert sich auf die Techniken und Methoden zur Untersuchung zeitlicher Datenfolgen. Die Studierenden lernen, Muster, Trends und Anomalien in zeitlichen Daten zu identifizieren, Vorhersagen zu treffen und fundierte Entscheidungen basierend auf der Analyse zeitlicher Daten zu treffen.

**Typ**

Wahlpflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

- **Verständnis** der Grundprinzipien und Herausforderungen bei der Analyse von Zeitreihendaten haben.
- **Techniken und Modelle** zur Analyse, Vorhersage und Interpretation von Zeitreihen effektiv anwenden können.
- **Vorverarbeitung und Transformation** von Zeitreihendaten durchführen können, um sie für Analysen zu optimieren.
- **Ergebnisse der Zeitreihenanalyse** in einem klaren und verständlichen Format präsentieren und kommunizieren können.
- **Praktische Erfahrung** in der Anwendung von Zeitreihenanalyse-Techniken auf realen Daten haben.

# Modulbeschreibung: CDS123 - Large Language Models

**Leitidee**

Das Modul "Large Language Models" bietet eine tiefgehende Untersuchung von großskaligen Sprachmodellen, wie sie in vielen modernen Anwendungen der künstlichen Intelligenz eingesetzt werden. Die Studierenden werden die Architektur, das Training und die Einsatzmöglichkeiten solcher Modelle erforschen und ihre potenziellen Auswirkungen auf Industrie, Forschung und Gesellschaft bewerten.

**Typ**

Wahlpflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

- **Kenntnisse** über die Kernkonzepte, Technologien und Architekturen großer Sprachmodelle besitzen.
- **Praktische Fähigkeiten** entwickelt haben, um große Sprachmodelle für verschiedene Aufgaben einzusetzen und anzupassen.
- **Ethik und Verantwortung** in den Vordergrund ihrer Arbeit mit großen Sprachmodellen stellen und die damit verbundenen Herausforderungen kritisch bewerten können.
- **Innovative Lösungen** entwickeln können, die die Möglichkeiten großer Sprachmodelle nutzen.
- **Aktuelle Forschung und Entwicklungen** im Bereich großer Sprachmodelle kritisch analysieren und bewerten können.

# Modulbeschreibung: CDS206 - Cloud Computing

**Leitidee**

Im Rahmen dieser Vorlesung werden Cloud Computing Konzepte vermittelt. Die Studierenden lernen die dazugehörigen Technologien einzusetzen.

**Typ**

Wahlpflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

Nach erfolgreicher Teilnahme am Modul sind die Studierenden in der Lage,

- Anforderungen und Lösungsansätze in Bezug auf die Elastizität und Skalierbarkeit einer Applikation zu formulieren
- Cloud-basierte Lösungsansätze zu entwickeln
- eine Anwendung zu virtualisieren

# Modulbeschreibung: CDS207 - Cryptography und Security

**Leitidee**

Die Studierenden erwerben grundlegende Kenntnisse in Cryptography und Security. Sie lernen die Sicherheit von Anwendungen und Infrastrukturen zu beurteilen und gezielt Sicherheitsmassnahmen einzusetzen.

**Typ**

Wahlpflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

Nach erfolgreicher Teilnahme am Modul sind die Studierenden in der Lage,

- die Sicherheit von Anwendungen und Infrastrukturen in Bezug auf Data, Web und Cloud Security zu beurteilen und entsprechende Massnahmen vorzuschlagen
- geeignete Sicherheitsmechanismen einzusetzen in ausgewählten Bereichen einzusetzen
- die grundlegenden Konzepte der Kryptographie zu verstehen und erklären
- die Sicherheit von Netzwerk-Protokollen und Systemen zu analysieren

# Modulbeschreibung: CDS208 - Frontend-Entwicklung

**Leitidee**

Das Web ist die Plattform für anspruchsvolle und interaktive Anwendungen. In vielen Bereichen werden traditionelle Fat- und Rich-Client-Anwendungen durch Web-Anwendungen ersetzt. In diesem Modul lernen lernen die Studierenden die Entwicklung einer moderner Full-Stack-Web-Anwendung.

**Typ**

Wahlpflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

Nach erfolgreicher Teilnahme am Modul sind die Studierenden in der Lage,

- die Architektur einer Full-Stack-Web-Anwendung zu skizzieren
- mit Web-Technologien eine Web-Anwendung zu entwickeln

# Modulbeschreibung: CDS211 - Systemnahe Programmierung

**Leitidee**

Programmiersprachen wie Python bieten einen guten Zugang zur Welt der Softwareentwicklung und werden von Beginn des Studiums an gelernt. Wir werfen aber einen Blick auf das, was unter der Decke passiert, wenn ich mein Python-Programm starte, auf Ebene des Betriebssystems und auf der Hardware. Mit diesem Wissen können wir effizientere Programme schreiben und auch Software für sogenannte Embedded-Systeme entwickeln, also Systeme mit beschränkten Ressource (Mikrocontroller) und hohen Anforderungen an deren Funktion (zum Beispiel zeitlicher Natur: Echtzeit). Dies machen wir ganz praktisch an unserem Rechner aber auch mit eigener Hardware auf Basis Arduino/Raspberry Pi.

**Typ**

Wahlpflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

Nach erfolgreicher Teilnahme am Modul sind die Studierenden in der Lage,

- ... wichtige Komponenten eines digitalen Rechnersystems und Betriebssystems zu benennen und deren Funktion zu erläutern
- ... zu wissen was bei der Ausführung ihrer Programme im Hintergrund passiert, auf Ebene des Betriebssystems und auf Ebene der Hardware
- ... können dieses Wissen nutzen, um Software für Embedded Systems mit und ohne Betriebssystem zu entwickeln
- ... haben anhand von Beispielhardware (Mikrocontroller ohne Betriebssystem, Raspberry Pi mit Linux) Applikationen für diese entwickelt.
- ... können Software in der Programmiersprache C / C++ entwickeln und können Assemblersprache (x86) grundsätzlich verstehen
- ... kennen Tools zum Compilieren, Linken und Analysieren von Programmen

# Modulbeschreibung: CDS212 - IT Development and Operations (DevOps)

**Leitidee**

Der Betrieb von komplexen IT-Infrastrukturen ist anspruchsvoll. Die Zusammenarbeit von Entwicklungs- und Betriebs-Teams während dem kompletten Lebenszyklus einer Software essenziell für Unternehmen und herausfordernd für Mitarbeiter. In diesem Modul erwerben die Studierenden praxisorientiert, ausgewählte IT-Operations Grundlagen. Ausgewählte IT Development and Operations (DevOps) Methoden werden vermittelt und angewandt.

**Typ**

Wahlpflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

Nach erfolgreicher Teilnahme am Modul sind die Studierenden in der Lage

- IT-Infrastrukturen zu betreiben
- Datenbanken und Anwendungen zu betreiben
- DevOps Methoden anzuwenden
- Monitoring
- Automatisierung von Prozessen

# Modulbeschreibung: CDS401 - Mathematik I

**Leitidee**

Erste Grundlagen der Analysis, linearen Algebra und Stochastik

**Typ**

Pflichtmodul

**Umfang**

6 ECTS-Punkte

**Lernergebnisse**

Siehe Kursbeschreibungen unten.

# Modulbeschreibung: CDS402 - Mathematik II

**Leitidee**

Aufbauend auf dem Modul Mathematik I wird eine Auswahl an Themenbereichen aus der Analysis und linearen Algebra vermittelt. Der Fokus liegt hierbei auf Konzepten und Methoden, die ein hohes Transferpotential im Bereich der rechnergestützten Datenwissenschaften haben. Neben den fachlichen Kompetenzen werden auch logisches Denken, rationales Argumentieren und faktenbasiertes Entscheiden geschult.

**Typ**

Pflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

Die Studierenden:

**Analysis**

- können Integrale aufstellen und berechnen, auch hinsichtlich praktischer Problemstellungen aus Alltag und Technik.
- können die Methoden der partiellen Integration und Substitution zur Berechnung von Integralen anwenden.
- können mit uneigentlichen Integralen umgehen.
- können Linien-, Flächen- und Volumenintegrale aufstellen und berechnen.
- können mit Funktionen von mehreren Variablen umgehen bzw. diese für konkrete Beispiele nutzen und diese sowohl integrieren als auch differenzieren (partielle Differentiation).
- kennen Skalar- und Vektorfelder und können mit diesen mathematisch umgehen bzw. diese interpretieren.

**Lineare Algebra**

- beherrschen die Algebra der komplexen Zahlen und können diese in arithmetischer, trigonometrischer und exponentieller Form darstellen.
- können die Lösungen einer Potenzgleichung in den komplexen Zahlen bestimmen.
- kennen die elementaren Matrixoperationen und können diese auf konkrete Beispiele anwenden.
- können Matrizen invertieren und diagonalisieren, Determinanten von Matri-zen bestimmen, Eigenwerte und -vektoren von Matrizen bestimmen.
- können lineare Abbildungen mittels Matrizen beschreiben, deren Kern und Bild bestimmen und deren Eigenschaften beurteilen.
- können mit Hilfe von Skalarprodukten Längen, Flächen und Volumina bestimmen.
- kennen die Vektorraumstruktur und können diese auf verschiedene Räume anwenden.

# Modulbeschreibung: CDS406 - Mathematik III

**Leitidee**

Grundlagen der Stochastik und der Differentialgleichungen

**Typ**

Pflichtmodul

**Umfang**

6 ECTS-Punkte

**Lernergebnisse**

Die Studierenden:

**Stochastik**

- kennen die grundlegenden Begriffe, Methoden und Hilfsmittel der Kombinatorik und können diese anwenden.
- kennen die Definition von Zufallsexperiment, Ereignissen und die Wahrscheinlichkeitsaxiome und können diese anwenden.
- können die Regeln der Wahrscheinlichkeitsrechnung anwenden.
- kennen verschiedene diskrete und stetige Wahrscheinlichkeitsverteilungen und können diese anwenden.
- kennen die Grundbegriffe der Statistik und können diese anwenden.
- können Daten statistisch anhand von Kennwerten und Masszahlen beschreiben und Zusammenhänge mit Hilfe von Korrelation und Regression untersuchen.

**Differentialgleichungen**

- können gewöhnliche Differentialgleichungen (DGL) 1. und 2. Ordnung klassifizieren und diskutieren.
- können die wichtigsten einfachen analytischen Methoden anwenden, um DGL 1. und 2. Ordnung zu lösen.
- können freie, gedämpfte und angeregte harmonische Schwingungen mit Hilfe von DGL und deren Lösungen beschreiben.
- können periodische und nicht-periodische Signale mit Hilfe von Fourier-Entwicklungen darstellen.
- können die Fourier-Transformation auf (nicht-) periodische Signale anwenden.
- können partielle Differentialgleichungen (PDGL) 1. und 2. Ordnung klassifizieren und diskutieren.
- können die wichtigsten einfachen analytischen Methoden anwenden, um PDGL 1. und 2. Ordnung zu lösen.
- kennen ausgewählte PDGL und deren Eigenschaften.
- beherrschen die grundlegende Syntax von Python und können numerische Berechnungen und Visualisierungen des Moduls mit Hilfe von Python durchführen.

# Modulbeschreibung: CDS407 - Agiles Projektmanagement und Nachhaltigkeit

**Leitidee**

Die Studierenden kennen die agilen Projektmanagement Methoden und können als Team Mitglied oder in einer Schlüsselrolle in einem agilen Projekt mitarbeiten.

Die Studierenden kennen und verstehen die Grundlagen und wichtigsten Konzepte der nachhaltigen Entwicklung. Interesse und Lust am Thema sollen geweckt und vermittelt werden. Sie sollen motiviert werden, nachhaltige Entwicklung zu einem Grundwert ihres beruflichen und persönlichen Handelns zu machen.

**Typ**

Pflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

Nach erfolgreicher Teilnahme am Modul sind die Studierenden in der Lage,

- eine geeignete Projektmanagement Methode für ein Projekt auszuwählen
- die Unterschiede zwischen unterschiedlichen Projektmanagement Methoden mit den Vorteilen und Nachteilen zu kennen
- Agile Projektmanagement Methoden wie Scrum oder Kanban in einem Projekt einzusetzen
- Anforderungen an ein Projekt zu formulieren und schätzen
- Nachhaltige Entwicklung im ökonomisch-ökologisch-sozialen Kontext und kennen die zentralen Treiber und Herausforderungen.
- Die Umsetzungsansätze der auf staatlicher Ebene und in Unternehmen.
- Die Bedeutung einer effizienten Nutzung von Ressourcen.
- Wie Unternehmen ihre wirtschaftlichen Handlungen und Zielsetzungen auf die Rahmenbedingungen (Umwelt und Interessengruppen) ausrichten.
- Den Bezug verschiedener Module aus ihren Studiengängen zum Thema Nachhaltigkeit.

# Modulbeschreibung: CDS408 - Innovationsmanagement und Design Thinking

**Leitidee**

Innovation ist die zielgerichtete Durchsetzung neuer Problemlösungen, die in technologischer, wirtschaftlicher, ökologischer oder sozialer Hinsicht einen wahrnehmbaren Nutzen stiftet.

Mit Design Thinking wird ein Ansatz vermittelt, der zum Lösen von Problemen und zur Entwicklung neuer Ideen führen kann.

Innovation und auch der Design Thinking Ansatz sind gerade im Umfeld der rechnergestützten Datenwissenschaften elementar.

**Typ**

Pflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

Nach erfolgreicher Teilnahme am Modul sind die Studierenden in der Lage,

- die wichtigsten Begrifflichkeiten und Grundlagen sowie die Erfolgsfaktoren des Innovationsmanagements zu benennen
- die Ziele, Aufgaben und Prozesse des Innovationsmanagements zu erläutern
- Modelle und Prozesse in Gruppen anhand eines frei zu bestimmenden Unternehmens anzuwenden
- die Bedeutung von Design Thinking zu erklären,
- den Prozess und die wichtigsten Tools zu beschreiben,
- die Methodologie und Tools an einer Challenge anzuwenden.

# Modulbeschreibung: CDS409 - First Certificate in English B2

**Leitidee**

Die Studierenden verstehen und nutzen die englische Sprache als wichtiges Medium der internationalen Verständigung. Sie kommunizieren erfolgreich auf Englisch in alltäglichen Situationen in ihrem beruflichen und persönlichen Umfeld. Sie setzen sich mit wirtschaftlichen, beruflichen, gesellschaftlichen und kulturellen Themen auseinander und reagieren situativ angemessen. Sie werden auch für kulturell bedingte Unterschiede im Kommunikationsverhalten sensibilisiert.

**Typ**

Pflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

Nach erfolgreicher Teilnahme am Modul sind die Studierenden in der Lage, auf dem Niveau B2 des GERS:

- die Hauptinhalte komplexer Texte zu konkreten und abstrakten Themen zu verstehen; im eigenen Spezialgebiet auch Fachdiskussionen zu verstehen.
- sich so spontan und fliessend zu verständigen, dass ein normales Gespräch mit Muttersprachlern ohne grössere Anstrengung auf beiden Seiten gut möglich ist.
- sich zu einem breiten Themenspektrum klar und detailliert auszudrücken, einen Standpunkt zu einer aktuellen Frage zu erläutern und die Vor- und Nachteile verschiedener Möglichkeiten anzugeben.

# Modulbeschreibung: CDS410 - Applied English for Computational and Data Scientists

**Leitidee**

Das Ziel dieses Moduls ist die pragmatische Anwendung der englischen Sprache in Kontexten, wie sie für Computational und Data Scientists typisch sind. Die Studierenden erwerben zudem relevante sprachliche Fähigkeiten und Fertigkeiten für den Umgang mit englischsprachigen Quellen im Studium an der FHGR.

**Typ**

Pflichtmodul

**Umfang**

4 ECTS-Punkte

**Lernergebnisse**

Nach erfolgreicher Teilnahme am Modul sind die Studierenden in der Lage, auf dem Niveau C1 des GERS:

- Texte und andere Quellen aus verschiedenen CDS-Anwendungsgebieten zu erfassen, sie zu analysieren und angemessen auf sie zu reagieren
- Englisch aus verschiedenen Quellen wie Fachartikeln, Fachdiskussionen, Präsentationen, Audio- und Videoaufzeichnungen zu verstehen und wirksam zu reagieren
- Informationen in verschiedenen CDS-Kontexten aus Studium und Beruf angemessen bzw. adressatengerecht in schriftlicher und mündlicher Form zu kommunizieren
- englische Standardausdrücke aus dem Bereich CDS zu erkennen, zu verstehen und anzuwenden
- sprachliche Unterschiede zwischen englischsprachigen und deutschsprachigen akademischen Texten zu erkennen und bei der Interpretation zu berücksichtigen
- Sitten und Gebräuche englischsprachiger Kulturen zu kennen und das eigene Verhalten im Geschäftsumfeld entsprechend anzupassen