Modul	Kapitel	Thema	Lernziel
Datenverarbeitung	Einführung in die Informatik	Informatik Definition	Die Studierenden können den Begriff Informatik definieren und sind in der Lage, den Begriff der Modellbildung zu beschreiben.
		Historie	Die Studierenden kennen die Ursprünge der Informatik und können die unterschiedlichen Fachgebiete voneinander abgrenzen.
		Entwicklung von Rechenmaschinen	Die Studierenden kennen das Computer-Konzept nach heutigem Muster und können die wesentlichen Eigenschaften beschreiben.
		Prinzipieller Aufbau von Computern	Die Studierenden können Rechenanlagen nach Ihrer Funktionsweise unterscheiden.
		Digitalrechner	Die Studierenden können die Eigenschaften von Digitalrechnern benennen.
		EVA-Prinzip	Die Studierenden können das EVA-Prinzip beschreiben.
	Zahlensysteme	Dezimalsystem	Die Studierenden kennen die Darstellung von Zahlen im Dezimalsystem.
		Dualsystem	Die Studierenden kennen die Darstellung von Zahlen im Dualsystem und die Interpretation von Bytes als Zahlen.
		Dualsystem Zahlenumwandlung	Die Studierenden können Zahlen vom Dezimalsystem ins Dualsystem umwandeln.
		Oktalsystem	Die Studierenden kennen die Darstellung von Zahlen im Oktalsystem.
		Oktalsystem Zahlenumwandlung	Die Studierenden können Zahlen vom Dezimalsystem ins Oktalsystem umwandeln.
		Hexadezimalsystem	Die Studierenden kennen die Darstellung von Zahlen im Hexadezimalsystem und den Einsatzzweck sinngemäß demonstrieren.
		Hexadezimalsystem Zahlenumwandlung	Die Studierenden können Zahlen vom Dezimalsystem ins Hexadezimalsystem umwandeln.
		Darstellung von ganzen und negativen Zahlen	Die Studierenden kennen die Zahlendarstellung im Zweierkomplement und können damit negative Ganzzahlen darstellen.
		Umwandlung von Zahlen in verschiedene Darstellungssysteme	Die Studierenden können Methoden zur Umwandlung von Zahlen in verschieden Zahlensystemen benennen und anwenden.
	Binäre Arithmetik	Rechenregeln für Binärzahlen	Die Studierenden können die Rechenregeln für Binärzahlen nennen.
		Logische Operatoren	Die Studierenden kennen die logischen Operatoren und können sie verwenden.
		Binäre Addition, Subtraktion und das Zweierkomplement	Die Studierenden kennen die Rechenregeln für die binäre Addition und die binäre Subtraktion sowie das Zweierkomplement und können sie anwenden.
		Binäre Multiplikation und Division	Die Studierenden kennen die Rechenregeln für die binäre Multiplikation und die binäre Division und können sie anwenden.
		Gleitkommazahlen	Die Studierenden kennen den Aufbau von Gleitkommazahlen und können damit dezimale Gleitkommazahlen darstellen.
		Binäre Gleitkommazahlen	Die Studierenden kennen den Aufbau von binären Gleitkommazahlen und können dezimale Gleitkommazahlen in binäre Gleitkommazahlen umwandeln.
		Rechnen mit Gleitkommazahlen	Die Studierenden können Gleitkommazahlen in den mathematischen Grundrechenarten Addition, Subtraktion, Multiplikation und Division rechnen.
		Größeneinheiten	Die Studierenden kennen die Größeneinheiten und deren Abkürzung. Sie können ihre binäre bzw. dezimale Größe angeben und zueinander bewerten.
	Nachricht	Nachricht Definition	Die Studierenden können den Begriff Nachricht definieren und den Weg der Nachrichtenübermittlung darstellen.
	Interpretation	Interpretation und Information	Die Studierenden bestimmen den Begriff Interpretation im Kontext von Information aus einer Nachricht.
	Daten	Daten Definition	Die Studierenden können den Begriff Daten definieren.
		Merkmale	Die Studierenden können Merkmale von Daten angeben und zueinander abgrenzen.
		Darstellung	Die Studierenden erkennen den Nutzen der Darstellung von Daten und können Darstellungen in verschiedene Kontexte zuordnen.
		Metadaten	Die Studierenden können den Begriff Metadaten definieren und Beispiele konstruieren.
		Inkompatibilität	Die Studierenden erkennen typische Probleme im Umgang mit Daten und können sie dem Begriff Inkompatibilität zuordnen.
		Interoperabilität	Die Studierenden erkennen typische Probleme im Umgang mit Daten und können sie dem Begriff Interoperabilität zuordnen.
		Typische Probleme	Die Studierenden können typische Probleme in der Datenverarbeitung klassifizieren und Beispiele konstruieren.
	Information	Information Definition	Die Studierenden können den Begriff Informationsgehalt erklären und den Zusammenhang zwischen Daten und Informationen erklären.
	Codierung	Codierung Definition	Die Studierenden können den Begriff Codierung definieren und erklären.
		Binärcodierung	Die Studierenden können den Begriff Binärcodierung erklären, die Ziele benennen, in der Datenübertragung identifizieren und vom Begriff der Verschlüsselung abgrenzen.
		Wortlänge	Die Studierenden können den Begriff der mittleren Wortlänge beschreiben.
		Entropie	Die Studierenden können den Begriff Entropie beschreiben und mit der mittleren Wortlänge in Beziehung setzen.
		Redundanz	Die Studierenden können den Begriff Code-Redundanz beschreiben, seinen Nutzen diskutieren und von der Quellen-Redundanz abgrenzen.
		BCD-Codierung	Die Studierenden können die Verwendung der BCD-Codierung angeben und Zahlen vom Dezimalsystem in eine BCD-codierte Zahl umwandeln.
		ASCII-Code	Die Studierenden kennen den Aufbau des ASCII-Code und können wichtige Sonderzeichen benennen.
		Unicode	Die Studierenden kennen den Aufbau des Unicode und können seine unterschiedliche Umsetzung mittels UTF zueinander vergleichen.
		Blockcodes und Huffman-Codierung	Die Studierenden können die Merkmale und den Zweck von Blockcodes beschreiben und mit der Huffman-Codierung vergleichen.
		Kompression	Die Studierenden können den Zweck der Verwendung von Kompression und seine Eigenschaften verlustfrei und verlustbehaftet gegenüberstellen.

Modul	Kapitel	Thema	Lernziel
Rechner und Netze	Rechnerarchitekturen	Evolution der Rechnerarten	
		Heutige Rechnerarten	
		Historie und Trends	
		Grundlegende Bestandteile und Ihre Erfinder	
	Die von-Neumann-Architektur	Komponenten eines von-Neumann-Rechners	
		Bus-System	
		Operationsprinzip und die Harvard-Architektur	
		Speichermodell	
		Der von-Neumann-Flaschenhals	
	Der typische Desktop-PC und Laptop	Hardware- und Software-Komponenten	
	Befehlssatz	Befehlssatz Definition	
		Mikroprogramme und CISC	
		Reduced Instruction Set Computer: RISC	
		Abwärtskompatibilität	
	Klassifikation nach Flynn	Single Instruction Single Data (SISD)	
		Single Instruction Multiple Data (SIMD)	
		Multiple Instruction Single Data (MISD)	
		Multiple Instruction Multiple Data (MIMD)	
		Hardwareseitiges Multithreading	
		Multi-Core und Many-Core Architekturen	
	Parallelität innerhalb einer Befehlssequenz	Fließbandverarbeitung: Optimierte Befehlsausführung	
		Superskalare Mikroprozessoren	
	Parallelität in Daten nutzen	Vektorprozessoren	
	Parallele Ausführung mehrerer Befehlssequenzen	Simultanes Multi-Threading innerhalb einer CPU	
		Multi-Core-CPU	
		Multiprozessor-Systeme	
		Multicomputer-Systeme	
	Speicherhierarchie	Speichertechnologien: Register, Cache und Hauptspeicher	
		Caching	
		Festplatten	
		Flash-Speicher und Solid State Disks	
	Evolution der Rechner	Moore's Law	
		TPU	
		Leistungskennzahlen von CPUs	
	Ein- und Ausgabe	Interrupts und DMA	
	Verbindungsstrukturen	Gemeinsamer Bus	
		Zugriffsprotokolle für Busse und gemeinsame Speicher	
		Punkt-Zu-Punkt Verbindungen: PCI-Express und SATA	
	Rechnernetze	Aufbau von Rechnernetzen	
		Kanal und Übertragungsmedium	
		Bandbreite und Übertragungsrate	
		Reichweite	
		Netzwerklatenz	
		Infrastrukturen	
		Das OSI-Schichtenmodell der Datenkommunikation	
	Bitübertragungsschicht	Kupfer, Glasfaser und der freie Raum	
		Leitungscodierung	
		Fehlererkennung: Kanalcodierung	
		Modulation, Multiplexing und Multiple Access	
		Netztopologien	
	Technologien der Sicherungsschicht	Netze im Nahbereich (PAN und WPAN)	
		Lokale Netze: LAN und WLAN	
		Leitungsvermittlung (Circuit Switching)	
		Paketvermittlung (Packet Switching)	
	Netzwerk- und Transportschicht: TCP/IP und das Internet	Überblick über das Internet	
		IP: Internet Protocol	
		IPv6	
		ICMP: Internet Control Message Protocol	
		TCP: Transmission Control Protocol	
		TLS und SSL: Transport Level Security	
		UDP: User Datagram Protocol	
	Anwendungsschicht: Von DNS bis HTTP und URIs	DNS: Domain Name System	
		HTTP und HTTPS: Hypertext Transfer Protocol	
		URI: Uniform Resource Identifier	

Modul	Kapitel	Thema	Lernziel
Programmieren	Einführung in die Programmierung	Programmiersprachen	Die Studierenden kennen eine gemeinsame Eigenschaft, die alle Programmiersprachen charakterisiert.
		Syntax Definition	Die Studierenden können den Begriff Syntax definieren, vom Begriff Semantik unterscheiden und Beispiele konstruieren.
		Semantik Definition	Die Studierenden können den Begriff Semantik definieren, erklären und vom Begriff Syntax unterscheiden.
		Pragmatik Definition	Die Studierenden können den Begriff Pragmatik definieren und Beispiel konstruieren.
	Einführung in die Programmiersprache Python	Programmieren	Die Studierenden können das Grundprinzip der Programmierung beschreiben und darstellen.
		Programmiersprachen	Die Studierenden können die grundlegenden Bestandteile imperativer Programmiersprachen wiedergeben. Sie können zwischen imperativen, visuellen und deklarativen Programmiersprachen unterscheiden und jeweils Beispiele benennen.
		Programmierumgebungen und Auszeichnungssprache	Die Studierenden benutzen hauptsächlich JupyterLab für die Programmierung von Jupyter Notebooks. Sie wenden Markdown für die Gliederung und Formatierung von Texten im Jupyter Notebook an. Die Studierenden benutzen Visual Studio Code als integrierte Entwicklungsumgebung von Python Programmen. Sie können zwischen der Verwendung von JupyterLab und VS Code abwägen.
		Anweisungen	Die Studierenden kennen die Bestandteile von Anweisungen und sind in der Lage selbst Anweisungen zu erstellen.
		Compiler	Die Studierenden können die Aufgaben eines Compilers beschreiben und von einem Interpreter unterscheiden.
		Interpreter	Die Studierenden können die Aufgaben eines Interpreters beschreiben und von einem Compiler unterscheiden.
		Python	Die Studierenden können die charakteristischen Merkmale von Python benennen.
		Python Code ausführen	Die Studierenden können Python Code in der Kommandozeile, in der interaktiven Konsole und im interaktiven Notizbuch (Jupyter Notebook) ausführen.
		Aufbau von Programmen	Die Studierenden können die Vorteile von strukturierten Programmen benennen und erklären. Sie kennen die allgemeinen Grundsätze für die äußere Form eines Programmcodes.
		Erstes Programm schreiben	Die Studierenden können ein Programm in Python schreiben.
		Fehler (Syntax und Laufzeit)	Die Studierenden kennen mögliche Fehlerquellen und können Syntax-, Laufzeit- und Logik-Fehler unterscheiden.
		Variablen und Literale	Die Studierenden verwenden Variablen zielgerichtet und zweckmäßig. Sie kennen die Rolle von Literalen in Programmiersprachen und können Beispiele konstruieren.
		Elementare Datentypen und deren Konvertierung	Die Studierenden kennen die Eigenschaften und den Einsatzzweck von Datentypen. Sie können die elementaren Datentypen in der Programmiersprache Python benennen und in Beispielen darstellen. Die Studierenden wandeln zielgerichtet Datentypen in andere um und erkennen dabei mögliche Probleme.
		Kommentare und Konventionen	Die Studierenden wissen, wozu Kommentare verwendet werden und gestalten damit Programme.
		Ein- und Ausgabefunktionen	Die Studierenden kennen und verwenden die Ein- und Ausgabefunktionen in Python.
		Operatoren und Ausdrücke	Die Studierenden kennen und verwenden die mathematischen und logischen Operatoren in Python.
		Verarbeitung von Zeichenketten	Die Studierenden können Zeichenketten mit built-in Python-Methoden verarbeiten.
		Module und Pakete einbinden	Die Studierenden können die Merkmale und den Zweck von Modulen und Paketen benennen. Sie können Module und Pakete selbst erstellen und anderen in Programmen einbinden.
		Zahlen und Mathe (Ganzzahlen und Gleitkommazahlen)	Die Studierenden kennen die grundlegenden Zahlentypen im Rechner. Sie verstehen den Aufbau einer Gleitkommazahl. Sie können den Begriff Mantisse erklären. Die Studierenden kennen die Grenzen der Gleitkommadarstellung. Sie können Mathematische Operatoren in Programmen verwenden.
		Boolesche Ausdrücke	Die Studierenden kennen logische Operatoren und können diese verwenden, um bedingte Anweisungen zu kombinieren. Sie kennen Vergleichs-Operatoren und können die "Wahrheit" eines Objekts bestimmen.
		Entscheidungslogik (if, elif und else)	Die Studierenden können den Ablauf von Anweisungen zergliedern und entscheiden. Die Studierenden konstruieren bedingte Anweisungen.
		Schleifen (for und while)	Die Studierenden können for- und while-Schleifen unterscheiden. Sie können einschätzen, welche für eine Problemstellung die geeignetere ist und diese in Python umzusetzen.
		Schleifen-Steuerung (break, continue und pass)	Die Studierenden können den Ablauf von Schleifen gestalten.
		Funktionen	Die Studierenden kennen Funktionen und deren Einsatzweck. Sie können Funktionen konstruieren und im Programmcode ausführen.
		Anonyme Funktionen	Die Studierenden kennen anonyme Funktionen und deren Einsatzweck. Sie können anonyme Funktionen konstruieren und im Programmcode ausführen.
		Scope und Lifetime von Variablen	Die Studierenden können die Sichtbarkeit von Variablen bestimmen und im Programm umsetzen.
		Datenstrukturen (List, Tuple, String, Set, Dictionary)	Die Studierenden kennen die unstrukturierten und die strukturierten Datentypen in Python. Sie kennen jeweils die Merkmale und die Funktionen zum Erstellen, zum Errechnen der Anzahl an Elementen, dem Zugriff auf ein Element sowie der Wertzuweisung. Sie verwenden die Datenstrukturen und ihre Funktionen in eigenen Programmen.
		Dateien und Verzeichnisse	Die Studierenden wissen, wie der Zugriff auf Dateien in Python erfolgt. Sie können Dateien in Programmen lesend und schreibend verarbeiten.
		Ausnahmen und Fehlerbehandlung	Die Studierenden wissen, dass in der Programmausführung Fehler auftreten können. Sie können Ausnahmen von Fehlern unterscheiden und gezielte Ausnahmebehandlungen im Programm anwenden.
	Vom Problem zum Programm	Paradigmen	Die Studierenden verstehen die unterschiedliche Betrachtung auf Problemstellungen aus der Realwelt und evaluieren die Programmier-Prinzipien für einen Lösungsansatz.
		Teile-und Herrsche-Prinzip	Die Studierenden können das Prinzip eines Teile-und-Herrsche-Verfahrens erklären. Sie können erklären, unter welchen Umständen Teile-und-Herrsche sich mit Rekursion verbinden lässt. Sie können Beispiele für Teile-und-Herrsche-Algorithmen nennen und beschreiben.
		Versuch-und Irrtum-Prinzip	Die Studierenden können das algorithmische Prinzip Versuch-und-Irrtum erklären. Sie können dazu Beispiele benennen und beschreiben.
		Gierige Algorithmen	Die Studierenden kennen das Greedy-Prinzip und können es erklären. Die Studierenden wissen, wie sich in unterschiedlichen Kontexten der "größte Gewinn" bzw. der "größte Nutzen" beschreiben lässt. Die Studierenden können erklären, warum ein Greedy-Ansatz nicht für jedes Problem eine optimale Lösung finden kann. Sie können beschreiben, warum es sinnvoll sein kann, Greedy zusammen mit Versuch-und-Irrtum zu nutzen.
		Heuristiken	Die Studierenden können das Grundprinzip einer Heuristik beschreiben. Sie können Beispiele angeben, die sich mit einer Heuristik lösen lassen.
	Algorithmen	Begriffserklärung	Die Studierenden kennen die Anforderungen an die Beschreibung von Algorithmen. Sie können den Begriff "Finitheit" beschreiben.
		Schema zur Beschreibung von Algorithmen	Die Studierenden können die grundlegenden Elemente einer Algorithmen Beschreibung benennen und beschreiben.
		Schleifen und Bedingungen	Die Studierenden kennen die Fragen, die sie zur Erstellung von Schleifen beantworten müssen. Sie können eine wiederholte Tätigkeit präzise als Schleife formulieren.
		Anforderungen an Algorithmen	Die Studierenden können sprachlich präzise zwischen Bedingungen und Schleifen unterscheiden. Sie können einfache Probleme mit Bedingungen und Schleifen formulieren.
		Rekursion und Backtracking	Die Studierenden kennen das Grundprinzip der Rekursion und können Beispiele für rekursive Verfahren benennen und beschreiben. Sie kennen das Grundprinzip von Backtracking und Beispiele benennen und beschreiben.
	Suchen und Sortieren	Sequenzielle Suche	Die Studierenden kennen den Begriff des Suchens. Sie können die Funktionsweise der sequentiellen Suche erklären.
		Binäre Suche	Die Studierenden können den Aufbau und die Eigenschaften eines binären Suchbaums erklären. Sie können für eine gegebene Menge von Werten und eine gegebene Ordnungsrelation einen binären Suchbaum erstellen. Sie können das Problem benennen, das durch eine ungeeignete Einfüge Reihenfolge beim Aufbau eines Suchbaums entsteht. Sie können ein rekursives Verfahren zur Suche in einem binären Suchbaum erstellen (Algorithmen-Skizze). Die Studierenden können ein rekursives Verfahren zur sortierten Ausgabe der Werte in einem Suchbaum erstellen.
		Sortierverfahren	Die Studierenden kennen die Zeitkomplexitäten verschiedener Sortierfahren und können die Algorithmen benennen.
	Komplexität	O-Kalkül (Landau-Notation)	Die Studierenden können den Sinn der Untersuchung der Komplexität von Verfahren erklären. Sie kennen die mathematischen Rechenregeln bei der Ermittlung der Komplexität eines Verfahrens. Die Studierenden können die Rechenregeln auf einen Ausdruck im O-Kalkül anwenden. Die Studierenden wissen, wie sie die Komplexität einer Schleife bestimmen können. Sie können die Komplexität einer Bedingung ermitteln. Die Studierenden bestimmen die Komplexität eines Programmfragments.
		Vergleich der relativen Laufzeiten von Sortierverfahren	Die Studierenden können die relativen Laufzeiten von Sortierverfahren wiedergeben.
	Umsetzung einfacher Algorithmen in Programme	Strukturierung von Programmen	Die Studierenden kennen den Aufbau von Programmen. Sie konstruieren kleine Programme in Python.
	Funktionale Programmierung	lambda, map, filter, reduce	Die Studierenden kennen den Begriff der funktionalen Programmierung. Sie können die Funktionen lambda, map, filter und reduce auf Problemstellungen anwenden.
		List-Comprehension	Die Studierenden erkennen Einsatzmöglichkeiten zur Verwendung von List-Comprehension. Sie wenden sie zielgerichtet und zweckmäßig an, um ihren Programmcode eleganten und effizient zu schreiben, der leicht zu lesen und zu debuggen ist.
		Packing und Unpacking	Die Studierenden erkennen Einsatzmöglichkeiten zur Verwendung von Packing und Unpacking. Sie wenden die Funktion an, um ihren Programmcode effizienter zu gestalten.
		zip	Die Studierenden erkennen Einsatzmöglichkeiten der Funktion zip. Sie wenden die Funktion an, um ihren Programmcode effizienter zu gestalten.
	Elementare Programmiertechniken	Beispiele für Funktionale Programmierung	Die Studierenden können funktionale Programme erkennen. Sie konstruieren selbst kleine Beispielprogramme.
	Code Bugs finden und beheben	Einsatz der integrierten Entwicklungsumgebung	Die Studierenden lernen die integrierte Entwicklungsumgebung Visual Studio Code kennen.
		Debugging	Die Studierenden erkennen die Einsatzmöglichkeiten von Debugging. Sie wenden das Verfahren an, um ihren Programmcode zu analysieren und zu verbessern. Die Studierenden können Debugging in JupyterLab und Visual Studio Code anwenden.