Lernnachweis zu Kompetenz B1G:

**Kompetenz**:

B1G: Ich kann ein Algorithmus erklären.

**Lernnachweis**:

Während des Moduls 323 habe ich mich mit der Entwicklung und Erklärung von Algorithmen auseinandergesetzt. Hier werde ich einen einfachen Algorithmus zur Berechnung der Fibonacci-Folge erklären.

**Algorithmus: Fibonacci-Folge berechnen**

Die Fibonacci-Folge ist eine Zahlenfolge, bei der jede Zahl die Summe der beiden vorherigen Zahlen ist. Der Algorithmus dazu kann rekursiv implementiert werden:

**def** **fibonacci**(n):

**if** n <= **1**:

**return** n

**else**:

**return** fibonacci(n-**1**) + fibonacci(n-**2**)

**Für Flaks Applikation**:

[**http://localhost:5000/fibonacci/10**](http://localhost:5000/fibonacci/10) **bis zur Position 10 zu sehen.**

**Lernprozess und Reflexion**:

Die Entwicklung dieses Algorithmus half mir, die Struktur von rekursiven Algorithmen zu verstehen. Die Fibonacci-Folge ist ein einfaches Beispiel, das den Konzepten von Rekursion und dynamischer Programmierung nahekommt.

**Fazit**:

Der Prozess der schrittweisen Addition und der Selbstaufrufe in der Rekursion waren anfangs herausfordernd, aber durch Übung und das Anwenden in verschiedenen Kontexten wurde mir klar, wie mächtig rekursive Algorithmen sein können.

**Info**: Vollständiger Code im Flask Projekt (Github Repo).

Lernnachweis zu Kompetenz B1F:

**Kompetenz**:

B1F: Ich kann Algorithmen in funktionale Teilstücke aufteilen.

**Lernnachweis**:

Während des Moduls 323 habe ich mich intensiv mit der Fähigkeit beschäftigt, Algorithmen in funktionale Teilstücke zu zerlegen. Hier werde ich diese Kompetenz anhand eines einfachen Sortieralgorithmus demonstrieren: den Bubble Sort.

**Algorithmus: Bubble Sort**

Bubble Sort ist ein einfacher Sortieralgorithmus, bei den benachbarten Elementen verglichen und getauscht werden, wenn sie in der falschen Reihenfolge sind. Dieser Algorithmus kann in funktionale Teilstücke zerlegt werden.

**Funktion für den Vergleich zweier Elemente**:

**def** **compare\_elements**(a, b):

**return** a > b

**Funktion für den Tausch zweier Elemente**:

**def** **swap\_elements**(arr, i, j):

arr[i], arr[j] = arr[j], arr[i]

**Hauptfunktion des Bubble Sort mit funktionalen Teilstücken**:

**def** **bubble\_sort**(arr, compare\_func, swap\_func):

n = len(arr)

**for** i **in** range(n):

**for** j **in** range(**0**, n-i-**1**):

**if** compare\_func(arr[j], arr[j+**1**]):

swap\_func(arr, j, j+**1**)

**Für Flaks Applikation**:

**http://localhost:5000/sort/5,2,8,1,7 besuchst du, um die sortierten Zahlen zu sehen.**

**Lernprozess und Reflexion**:

Die Aufteilung des Bubble-Sort-Algorithmus in funktionale Teilstücke ermöglicht eine bessere Lesbarkeit und Wartbarkeit des Codes. Die Funktionen compare\_elements und swap\_elements sind unabhängige, wiederverwendbare Bausteine, die nicht nur im Bubble Sort, sondern auch in anderen Algorithmen verwendet werden können.

**Fazit**:

Diese Modularität erleichtert die Fehlerbehebung und das Verständnis des Codes. Es hilft auch, den Algorithmus leichter zu testen und zu optimieren, da jede Funktion einzeln überprüft werden kann.

**Info**: Vollständiger Code im Flask Projekt (Github Repo).

Lernnachweis zu Kompetenz B1E:

**Kompetenz**:

B1E: Ich kann Funktionen in zusammenhängende Algorithmen implementieren.

**Lernnachweis**:

Während des Moduls 323 habe ich mich intensiv mit der Fähigkeit auseinandergesetzt, Funktionen in zusammenhängende Algorithmen zu implementieren. Hier werde ich diese Kompetenz anhand eines Algorithmus zur Berechnung des Durchschnitts einer Liste von Zahlen demonstrieren.

**Algorithmus: Durchschnitt berechnen**

Der Durchschnitt einer Liste von Zahlen wird berechnet, indem alle Zahlen addiert und durch die Anzahl der Zahlen geteilt werden. Hier ist die Implementierung in Python:

**def** **berechne\_durchschnitt**(zahlen):

summe = sum(zahlen)

anzahl = len(zahlen)

**if** anzahl == **0**:

**return** **0**

**else**:

durchschnitt = summe / anzahl

**return** durchschnitt

**Für Flask Applikation**:

**http://localhost:5000/durchschnitt mit dem Parameter zahlen (z.B., zahlen=1,2,3,4,5**

**Wichtig**: Curl oder Postman und sende eine POST-Anfrage.

**Lernprozess und Reflexion**:

Die Implementierung von Funktionen in zusammenhängende Algorithmen ermöglicht es, komplexe Aufgaben in überschaubare Teile zu zerlegen. Die Funktion berechne\_durchschnitt ist hier ein einfaches Beispiel. Sie greift auf die vordefinierte Funktion sum und len zu, um die Summe und die Anzahl der Zahlen zu berechnen.

**Fazit**:

Die Verwendung von Funktionen macht den Code lesbarer und erleichtert das Debugging sowie die Wartung. Es ermöglicht auch die Wiederverwendung von Code, da die gleichen Funktionen in verschiedenen Teilen des Algorithmus oder in anderen Algorithmen verwendet werden können.

**Info**: Vollständiger Code im Flask Projekt (Github Repo).