**Lernnachweis zu Kompetenz A1G:**

**Kompetenz:** Ich kann Funktionseigenschaften, wie pure functions, beschreiben und den Unterschied zu anderen Programmierstrukturen, insbesondere zu Prozeduren, erläutern.

**Lernnachweis: Beschreibung von Funktionseigenschaften und Unterschied zu Prozeduren**

In dieser Lernsequenz vertiefte ich mein Verständnis von Funktionen in der Programmierung. Funktionen sind eigenständige Codeeinheiten, die spezifische Aufgaben erfüllen und optional Rückgabewerte liefern. Der Fokus lag auf der "Reinheit" von Funktionen, insbesondere bei "pure functions", die keine Seiteneffekte haben und für gleiche Eingaben stets denselben Ausgabewert liefern.

**Umsetzung im Code:** In der erstellten Flask-Applikation spiegelt sich dieses Konzept in der Funktion add\_numbers wider. Diese Funktion ist als "pure function" gestaltet, da sie nur auf ihre Eingabewerte reagiert und keinen Zustand ändert. Die Flask-Route home ruft die Funktion add\_numbers auf und printet das Ergebnis.

**Flask-App Code Beispiel:**

Ein Bild, das Text, Screenshot, Software, Multimedia-Software enthält.

Automatisch generierte Beschreibung

**Reflexion:** Die intensive Auseinandersetzung mit Funktionen förderte mein Verständnis für modulare Programmierung. Pure functions ermöglichen wiederverwendbaren, vorhersagbaren und klar verständlichen Code. Der Vergleich mit Prozeduren betont die Bedeutung von Funktionen als eigenständige und klare Codeeinheiten. Dies trägt zu einem effizienten und strukturierten Programmierstil bei.

**Lernnachweis zu Kompetenz A1F:**

**Kompetenz:** Ich kann das Konzept von immutable values erläutern und dazu Beispiele anwenden. Somit kann ich dieses Konzept funktionaler Programmierung im Unterschied zu anderen Programmiersprachen erklären, beispielsweise im Vergleich zu referenzierten Objekten.

**Lernnachweis: Erklärung von Immutable Values in der funktionalen Programmierung**

Im Gegensatz zu veränderbaren Werten (mutable values) können unveränderliche Werte nach ihrer Erstellung nicht mehr verändert werden. Dieses Konzept unterscheidet sich grundlegend von anderen Programmierparadigmen, insbesondere von referenzierten Objekten in anderen Programmiersprachen.

**Umsetzung im Code:** In der erstellten Flask-Applikation demonstriert die Funktion modify\_string das Konzept der immutable values. Anstatt den Original-String direkt zu verändern, wird eine modifizierte Kopie zurückgegeben.

**Flask-App Code Beispiel:**

Ein Bild, das Text, Screenshot, Software, Multimedia-Software enthält.

Automatisch generierte Beschreibung

**Reflexion:** Die Arbeit mit immutable values fördert funktionale Programmierkonzepte, indem sie unveränderliche Datenstrukturen bevorzugt. Dies führt zu weniger Seiteneffekten und erleichtert das Verständnis und Debugging von Code. In der entwickelten Flask-Applikation wird das Konzept durch die Funktion modify\_string verdeutlicht, die eine modifizierte Kopie des ursprünglichen Strings zurückgibt, ohne diesen direkt zu verändern.

**Lernnachweis zu Kompetenz A1E:**

**Kompetenz:** A1E: Ich kann aufzeigen, wie Probleme in den verschiedenen Konzepten (OO, prozedural und funktional) gelöst werden und diese miteinander vergleichen.

**Lernnachweis zu A1E:** Während meiner Auseinandersetzung mit dem Modul 323 habe ich die Problemlösung in den Programmierparadigmen objektorientiert (OO), prozedural und funktional intensiv studiert.

**Lernprozess und Reflexion:** Die Vergleichsanalyse dieser Paradigmen ermöglichte mir ein tieferes Verständnis ihrer Stärken und Schwächen. Im prozeduralen Ansatz liegt der Fokus auf Verfahren und Schritten, während das objektorientierte Paradigma die Struktur betont. Das funktionale Paradigma zeichnet sich durch klare Funktionen und Unveränderlichkeit aus.

**Umsetzung im Code:**

Ein Bild, das Text, Screenshot, Software enthält.

Automatisch generierte Beschreibung

**Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung**

**Fazit:**

Die Anwendung dieser Paradigmen in meiner Flask-Applikation half mir, die Vor- und Nachteile in der Praxis zu erkennen. Beispielsweise ermöglichte mir die Nutzung von Klassen und Objekten im OO-Paradigma eine verbesserte Strukturierung und Wartung der Anwendung.