**Lernnachweis zu Kompetenz B2G:**

**Kompetenz:** Ich kann Funktionen als Objekte behandeln und diese in Variablen speichern und weitergeben.

**Lernnachweis: Behandlung von Funktionen als Objekte mit Flask-Anwendung**

In dieser Lernsequenz habe ich Funktionen als Objekte behandelt, in Variablen gespeichert und weitergegeben. Eine Flask-Anwendung wurde erstellt, um diese Funktionalität zu demonstrieren.

**Funktionalität im Code:** Der Code demonstriert die Behandlung von Funktionen als Objekte in Python. Die Funktion greet wird als Objekt gespeichert und durch die Funktion execute\_function mit einem Parameter aufgerufen. Die Flask-Anwendung stellt eine Route bereit, um diese Funktionalität aufzurufen. Beim Zugriff auf /execute\_greet wird die gespeicherte Funktion mit dem Namen "John" aufgerufen, und das Ergebnis wird als Response zurückgegeben. Dies verdeutlicht die Flexibilität von Funktionen als Objekten, die dynamisch in Variablen gespeichert und weitergegeben werden können.

**Umsetzung im Code mit Flask:**

**Ein Bild, das Text, Screenshot, Software, Betriebssystem enthält.

Automatisch generierte Beschreibung**

**Reflexion:** Die Umsetzung zeigt, wie Funktionen als Objekte behandelt, in Variablen gespeichert und weitergegeben werden können. Die Flask-Anwendung illustriert die Ausführung einer Funktion durch Aufruf über eine definierte Route.

**Lernnachweis zu Kompetenz B2F:**

**Kompetenz:** Ich kann Funktionen als Argumente für andere Funktionen verwenden und dadurch höherwertige Funktionen erstellen.

**Lernnachweis: Verwendung von Funktionen als Argumente mit Flask-Anwendung**

In dieser Lernsequenz habe ich Funktionen als Argumente für andere Funktionen verwendet, um höherwertige Funktionen zu erstellen. Eine Flask-Anwendung wurde erstellt, um dies zu demonstrieren.

Die Flask-Anwendung ruft die Funktion apply\_operation auf, die wiederum andere Funktionen wie add und multiply mit den Zahlen 5 und 3 aufruft. Dies ermöglicht die dynamische Anwendung verschiedener Operationen und veranschaulicht die Flexibilität von Funktionen als höherwertige Konzepte.

**Umsetzung im Code mit Flask:**

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

Automatisch generierte Beschreibung

**Reflexion:** Die Umsetzung illustriert die Verwendung von Funktionen als Argumente für andere Funktionen. Die Flask-Anwendung bietet eine Route, um die höherwertige Funktion apply\_operation aufzurufen. Durch den Zugriff auf /calculate werden die Funktionen add und multiply mit den Zahlen 5 und 3 aufgerufen, und die Ergebnisse werden als Response zurückgegeben. Dies verdeutlicht die Flexibilität und Vielseitigkeit beim Erstellen höherwertiger Funktionen durch die Verwendung von Funktionen als Argumente.

**Lernnachweis zu Kompetenz B2E:**

**Kompetenz:** Ich kann Funktionen als Objekte und Argumente verwenden, um komplexe Aufgaben zu lösen (Anwenden von Closures).

**Lernnachweis: Verwendung von Funktionen als Objekte und Argumente mit Closures in Flask-Anwendung**

In dieser Lernsequenz habe ich Funktionen als Objekte und Argumente verwendet, um komplexe Aufgaben mit Closures zu lösen. Eine Flask-Anwendung wurde erstellt, um dies zu demonstrieren.

**Umsetzung im Code mit Flask:**

Ein Bild, das Text, Screenshot enthält.

Automatisch generierte Beschreibung

**Reflexion:**

Die Umsetzung zeigt die Anwendung von Closures in Python. Die Flask-Anwendung ermöglicht den Zugriff auf die Route /greet\_with\_closure, wodurch zwei Closure-Funktionen erstellt und aufgerufen werden. Diese Closures, greet\_with\_hello und greet\_with\_hi, behalten den Wert des Präfixes aus der äußeren Funktion outer\_function bei. Dadurch können komplexe Aufgaben gelöst werden, indem Funktionen als Objekte und Argumente verwendet werden.