Lernnachweis zu Kompetenz B3G:

**Kompetenz**:

B3G: Ich kann einfache Lambda-Ausdrücke schreiben, die eine einzelne Operation durchführen, z.B. das Quadrieren einer Zahl oder das Konvertieren eines Strings in Großbuchstaben.

**Lernnachweis**:

Während meines Studiums im Modul 323 habe ich mich mit der Fähigkeit auseinandergesetzt, einfache Lambda-Ausdrücke zu schreiben, die einzelne Operationen durchführen. Hier werde ich diese Kompetenz anhand von Beispielen demonstrieren, wie das Quadrieren einer Zahl oder das Konvertieren eines Strings in Großbuchstaben.

**Beispiel 1: Lambda-Ausdruck zum Quadrieren einer Zahl**

quadriere = **lambda** x: x\*\***2**

# Verwendung des Lambda-Ausdrucks

ergebnis = quadriere(**5**)

**print**(ergebnis) # Ausgabe: 25

**Beispiel 2: Lambda-Ausdruck zum Konvertieren eines Strings in Großbuchstaben**

grossbuchstaben = **lambda** text: text.upper()

# Verwendung des Lambda-Ausdrucks

ergebnis = grossbuchstaben("hello")

**print**(ergebnis) # Ausgabe: "HELLO"

**Für Flask Applikation**:

**http://localhost:5000/transformiere** mit den Parametern operation (z.B., quadriere oder grossbuchstaben) und wert (z.B., 5)

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

**Lernprozess und Reflexion**:

Lambda-Ausdrücke bieten eine kompakte Möglichkeit, anonyme Funktionen zu erstellen, insbesondere wenn die Funktion nur eine einzige Operation ausführt. Die Verwendung von Lambda-Ausdrücken kann den Code lesbarer machen, besonders wenn sie in Kombination mit Funktionen wie map, filter oder sorted verwendet werden.

**Fazit**:

Während meines Lernprozesses habe ich festgestellt, dass Lambda-Ausdrücke besonders nützlich sind, wenn kurze, prägnante Funktionen benötigt werden und es nicht notwendig ist, eine vollständige benannte Funktion zu erstellen.

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

Lernnachweis zu Kompetenz B3F:

**Kompetenz**:

B3F: Ich kann Lambda-Ausdrücke schreiben, die mehrere Argumente verarbeiten können.

**Lernnachweis**:

Während meines Studiums im Modul 323 habe ich mich mit der Fähigkeit auseinandergesetzt, Lambda-Ausdrücke zu schreiben, die mehrere Argumente verarbeiten können. Hier werde ich diese Kompetenz anhand von Beispielen demonstrieren.

**Beispiel 1: Lambda-Ausdruck für die Addition zweier Zahlen**

addiere = **lambda** x, y: x + y

# Verwendung des Lambda-Ausdrucks

ergebnis = addiere(**3**, **5**)

**print**(ergebnis) # Ausgabe: 8

**Beispiel 2: Lambda-Ausdruck für die Konkatenation zweier Strings**

konkateniere = **lambda** str1, str2: str1 + str2

# Verwendung des Lambda-Ausdrucks

ergebnis = konkateniere("Hallo, ", "Welt!")

**print**(ergebnis) # Ausgabe: "Hallo, Welt!"

**Für Flask Applikation**:

**http://localhost:5000/transformiere** mit den Parametern operation (z.B., multipliziere oder verkette\_mit\_trennzeichen), wert1 und wert2.

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

**Lernprozess und Reflexion**:

Die Verwendung von Lambda-Ausdrücken mit mehreren Argumenten ermöglicht eine kompakte Darstellung von Funktionen, die mehrere Eingaben verarbeiten. Dies ist besonders nützlich, wenn kurze und prägnante Funktionen benötigt werden.

**Fazit**:

Während meines Lernprozesses habe ich festgestellt, dass Lambda-Ausdrücke mit mehreren Argumenten in Situationen, in denen sie angemessen sind, den Code vereinfachen können. Sie sind besonders praktisch, wenn die Funktion kurz ist und an Ort und Stelle verwendet wird, z.B. in Funktionen wie map oder filter.

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

Lernnachweis zu Kompetenz B3E:

**Kompetenz**:

B3E: Ich kann Lambda-Ausdrücke verwenden, um den Programmfluss zu steuern, z.B. durch Sortieren von Listen basierend auf benutzerdefinierten Kriterien.

**Lernnachweis**:

Während meines Studiums im Modul 323 habe ich mich mit der Fähigkeit auseinandergesetzt, Lambda-Ausdrücke zu verwenden, um den Programmfluss zu steuern, insbesondere durch Sortieren von Listen basierend auf benutzerdefinierten Kriterien. Hier werde ich diese Kompetenz anhand eines Beispiels demonstrieren.

**Beispiel: Sortieren einer Liste von Wörtern nach ihrer Länge**

woerter = ["Apfel", "Banane", "Erdbeere", "Kiwi", "Orange"]

# Verwendung eines Lambda-Ausdrucks für die Sortierung nach Länge

sortierte\_woerter = sorted(woerter, key=**lambda** x: len(x))

**print**(sortierte\_woerter)

# Ausgabe: ['Kiwi', 'Apfel', 'Banane', 'Orange', 'Erdbeere']

**Für Flask Applikation**:

**http://localhost:5000/sortiere\_benutzer** mit dem Parameter kriterium (z.B., nach\_name oder nach\_alter).

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

**Lernprozess und Reflexion:**

Die Verwendung von Lambda-Ausdrücken zur Steuerung des Programmflusses bietet eine kompakte Möglichkeit, benutzerdefinierte Kriterien für bestimmte Aktionen anzugeben, wie zum Beispiel das Sortieren von Daten.

**Fazit:**

Während meines Lernprozesses habe ich festgestellt, dass diese Technik besonders nützlich ist, wenn wir eine schnelle und temporäre Funktion benötigen, die nur an einer Stelle im Code verwendet wird, wie zum Beispiel bei der Sortierung.

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