Lernnachweis zu Kompetenz B4G:

**Kompetenz**:

B4G: Ich kann die Funktionen Map, Filter und Reduce einzeln auf Listen anwenden.

**Lernnachweis**:

Während meines Studiums im Modul 323 habe ich die Fähigkeit erworben, die Funktionen map, filter und reduce einzeln auf Listen anzuwenden. Hier werde ich jede Funktion separat demonstrieren.

**Beispiel 1: Anwendung von map zur Verdopplung der Elemente einer Liste:**

zahlen = [**1**, **2**, **3**, **4**, **5**]

# Verwendung von map, um jedes Element zu verdoppeln

verdoppelte\_zahlen = list(map(**lambda** x: x \* **2**, zahlen))

**print**(verdoppelte\_zahlen)

# Ausgabe: [2, 4, 6, 8, 10]

**Beispiel 2: Anwendung von filter zur Auswahl gerader Zahlen aus einer Liste:**

zahlen = [**1**, **2**, **3**, **4**, **5**, **6**, **7**, **8**, **9**, **10**]

# Verwendung von filter, um nur gerade Zahlen auszuwählen

gerade\_zahlen = list(filter(**lambda** x: x % **2** == **0**, zahlen))

**print**(gerade\_zahlen)

# Ausgabe: [2, 4, 6, 8, 10]

**Beispiel 3: Anwendung von reduce zur Berechnung der Summe einer Liste:**

**from** **functools** **import** reduce

zahlen = [**1**, **2**, **3**, **4**, **5**]

# Verwendung von reduce, um die Summe der Elemente zu berechnen

summe = reduce(**lambda** x, y: x + y, zahlen)

**print**(summe)

# Ausgabe: 15

**Für Flask Applikation**:

**http://localhost:5000/transformiere\_liste** mit dem Parameter operation (z.B., verdopple, waehle\_gerade\_aus, berechne\_summe)

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

**Lernprozess und Reflexion**:

Die Funktionen map, filter und reduce bieten leistungsstarke Werkzeuge für die Verarbeitung von Listen. Während meines Lernprozesses habe ich festgestellt, dass sie besonders nützlich sind, um Code effizient und lesbar zu gestalten, insbesondere wenn komplexe Iterationen oder Bedingungen erforderlich sind.

**Fazit**:

Map ermöglicht es, eine Funktion auf jedes Element einer Liste anzuwenden, filter ermöglicht es, Elemente basierend auf einer Bedingung auszuwählen, und reduce ermöglicht es, eine Funktion auf kumulierte Weise auf die Elemente einer Liste anzuwenden.

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

Lernnachweis zu Kompetenz B4F:

**Kompetenz**:

B4F: Ich kann Map, Filter und Reduce kombiniert verwenden, um Daten zu verarbeiten und zu manipulieren, die komplexeren Transformationen erfordern.

**Lernnachweis**:

Während meines Studiums im Modul 323 habe ich die Fähigkeit erworben, die Funktionen map, filter und reduce kombiniert zu verwenden, um Daten zu verarbeiten und zu manipulieren, die komplexeren Transformationen erfordern. Hier werde ich diese Kompetenz anhand eines Beispiels demonstrieren.

**Beispiel: Kombinierte Anwendung von map, filter und reduce zur Analyse von Daten**

Angenommen, wir haben eine Liste von Studentenobjekten, wobei jedes Objekt Informationen wie Name, Alter und Noten enthält. Wir möchten den Durchschnitt der Noten für Studenten berechnen, die älter als 20 Jahre sind.

**from** **functools** **import** reduce

studenten = [

{"name": "Alice", "alter": **22**, "noten": [**80**, **85**, **90**]},

{"name": "Bob", "alter": **19**, "noten": [**75**, **88**, **92**]},

{"name": "Charlie", "alter": **25**, "noten": [**90**, **92**, **95**]},

{"name": "David", "alter": **21**, "noten": [**78**, **85**, **88**]},

]

durchschnittsnote = reduce(

**lambda** x, y: x + y,

map(**lambda** student: student["noten"], filter(**lambda** student: student["alter"] > **20**, studenten)),

) / len(studenten)

**print**(durchschnittsnote)

# Ausgabe: 91.0

**Für Flask Applikation**: [**http://localhost:5000/berechne\_durchschnittsnote\_aelter\_20**](http://localhost:5000/berechne_durchschnittsnote_aelter_20)

**Lernprozess und Reflexion**:

Die kombinierte Anwendung von map, filter und reduce bietet eine leistungsstarke Möglichkeit, komplexe Transformationen auf Daten durchzuführen. Während meines Lernprozesses habe ich festgestellt, dass dies besonders nützlich ist, wenn spezifische Bedingungen erfüllt sein müssen, um bestimmte Transformationen durchzuführen.

**Fazit**:

Es ist wichtig zu beachten, dass die Kombination dieser Funktionen zu einer effizienten und lesbaren Datenverarbeitung führen kann, wenn sie korrekt eingesetzt werden.

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

Lernnachweis zu Kompetenz B4E:

**Kompetenz**:

B4E: Ich kann Map, Filter und Reduce verwenden, um komplexe Datenverarbeitungsaufgaben zu lösen, wie z.B. die Aggregation von Daten oder die Transformation von Datenstrukturen.

**Lernnachweis**:

Während meines Studiums im Modul 323 habe ich die Fähigkeit erworben, die Funktionen map, filter und reduce kombiniert zu verwenden, um komplexe Datenverarbeitungsaufgaben zu lösen, wie z.B. die Aggregation von Daten oder die Transformation von Datenstrukturen. Hier werde ich diese Kompetenz anhand eines Beispiels demonstrieren.

**Beispiel: Aggregation von Datenstrukturen mit map, filter und reduce**

Angenommen, wir haben eine Liste von Bestellungen, wobei jede Bestellung ein Dictionary mit Informationen wie Kundenname, Produkt und Menge ist. Wir möchten eine Gesamtsumme der bestellten Mengen für ein bestimmtes Produkt berechnen.

**from** **functools** **import** reduce

bestellungen = [

{"kunde": "Alice", "produkt": "Apfel", "menge": **5**},

{"kunde": "Bob", "produkt": "Banane", "menge": **3**},

{"kunde": "Charlie", "produkt": "Apfel", "menge": **2**},

{"kunde": "David", "produkt": "Orange", "menge": **4**},

{"kunde": "Alice", "produkt": "Banane", "menge": **6**},

]

# Aggregation mit map, filter und reduce

gesamtmenge\_apfel = reduce(

**lambda** x, y: x + y,

map(**lambda** bestellung: bestellung["menge"], filter(**lambda** bestellung: bestellung["produkt"] == "Apfel", bestellungen)),

)

**print**(gesamtmenge\_apfel)

**Für Flask Applikation**: [**http://localhost:5000/gesamtmenge\_produkt?produkt=Apfel**](http://localhost:5000/gesamtmenge_produkt?produkt=Apfel) **(um Apfel zu sehen)**

**Lernprozess und Reflexion**:

Die Anwendung von map, filter und reduce in Kombination ermöglicht eine flexible und leistungsstarke Datenverarbeitung. Während meines Lernprozesses habe ich erkannt, dass diese Funktionen in der Lage sind, komplexe Aufgaben auf eine kompakte und lesbarere Weise zu lösen.

**Fazit**:

Es ist wichtig zu beachten, dass die Verwendung dieser Funktionen in komplexen Szenarien einen klaren Überblick über die Datenstruktur erfordert und dass die Funktionen aufeinander abgestimmt sind, um das gewünschte Ergebnis zu erzielen.

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