Aufgabe 1: Datenstrukturen und Dateioperationen

Ziel: Vertiefe dein Verständnis von Python-Datenstrukturen und lerne, wie man mit Dateien arbeitet.

1. Erstelle ein Wörterbuch, das verschiedene Datentypen speichert, z.B. `name`, `age`, `languages` (eine Liste der Programmiersprachen, die du kennst), und `has_python_experience` (ein Boolean, der angibt, ob du Erfahrung mit Python hast).
2. Schreibe eine Funktion, die dieses Wörterbuch als Argument nimmt und die Informationen in einer formatierten Weise in eine Textdatei schreibt.

Aufgabe 2: Komplexe Listenmanipulation und Funktionen

Ziel: Verbessere deine Fähigkeiten in der Listenmanipulation und im Umgang mit Funktionen durch die Lösung eines realistischeren Problems.

1. Gegeben sei eine Liste von Dictionaries, wobei jedes Dictionary Informationen über einen Studenten enthält (z.B. Name, Matrikelnummer, und Noten in verschiedenen Fächern).
2. Schreibe eine Funktion, die die Durchschnittsnote jedes Studenten berechnet und diese Information in einem neuen Dictionary zusammen mit dem Namen und der Matrikelnummer speichert.
3. Sortiere die Liste der neuen Dictionaries basierend auf der Durchschnittsnote in absteigender Reihenfolge und drucke die Informationen jedes Studenten aus.

Aufgabe 3: Exception Handling und Dateiverarbeitung

Ziel: Lerne, wie man mit Fehlern in Python umgeht und wie man Dateien effizient verarbeitet.

1. Schreibe eine Funktion, die eine Datei einliest. Die Funktion sollte in der Lage sein, mit Situationen umzugehen, in denen die Datei nicht existiert, und eine benutzerdefinierte Fehlermeldung ausgeben.
2. Erweitere die Funktion, indem du die Zeilen der Datei einliest und die Anzahl der Wörter pro Zeile zählst. Gib ein Dictionary zurück, dass die Zeilennummern und die entsprechende Anzahl der Wörter enthält.
3. Füge eine weitere Funktionalität hinzu, die es ermöglicht, nach einem bestimmten Wort in der Datei zu suchen und die Zeilennummern auszugeben, in denen das Wort vorkommt.

Aufgabe 4: Einführung in Numpy und Matplotlib

Ziel: Erkunde grundlegende Funktionen der Bibliotheken Numpy und Matplotlib.

1. Verwende Numpy, um einen Array mit 100 zufälligen Zahlen zu erstellen.
2. Verwende Matplotlib, um ein Histogramm dieser Zahlen zu zeichnen.

Aufgabe 5: Einfache Machine Learning-Aufgabe mit Scikit-learn

Ziel: Mach deine ersten Schritte im Machine Learning durch die Anwendung eines einfachen Klassifikators.

1. Verwende den Iris-Datensatz aus Scikit-learn und Teile ihn in Trainings- und Testdaten auf.
2. Trainiere einen einfachen Klassifikator (z.B. KNeighborsClassifier) mit den Trainingsdaten.
3. Bewerte die Leistung des Klassifikators auf den Testdaten.

Aufgabe 6: Matplotlib Graphen

Ziel: Nutze Matplotlib, um eine ansprechende Visualisierung zu erstellen.

1. Erstelle Daten für einen Sinus- und einen Kosinus-Graphen.
2. Zeichne beide Graphen in dasselbe Diagramm, mit verschiedenen Farben und Legenden.
3. Füge dem Diagramm Achsenbeschriftungen und einen Titel hinzu, um es professioneller zu gestalten.

Aufgabe 7: Grundlagen der Objektorientierten Programmierung (OOP)

Ziel: Erlerne die Grundprinzipien der objektorientierten Programmierung in Python durch das Erstellen und Arbeiten mit Klassen und Objekten.

Teil 1: Klassendefinition und Instanziierung

1. Definiere eine Klasse `Student`, die folgende Eigenschaften (Attribute) speichert: `name`, `student_id`, `grades` (eine Liste von Noten).
2. Füge der Klasse eine Methode `add_grade` hinzu, die eine neue Note zu den bereits vorhandenen Noten hinzufügt.
3. Füge eine Methode `calculate_average` hinzu, die die Durchschnittsnote des Studenten berechnet und zurückgibt.

Teil 2: Vererbung

1. Definiere eine neue Klasse `GraduateStudent`, die von der Klasse `Student` erbt, aber zusätzliche Eigenschaften oder Methoden enthält, z.B. `thesis_topic`.
2. Überschreibe die `calculate_average` Methode für `GraduateStudent`, um eine andere Berechnung der Durchschnittsnote zu ermöglichen, die vielleicht strengere Kriterien berücksichtigt.

Teil 3: Polymorphismus und Abstraktion

1. Erstelle eine Funktion `print_student_details`, die ein `Student`-Objekt als Argument nimmt und dessen Details ausdruckt. Diese Funktion sollte polymorph sein, d.h., sie sollte unterschiedliche Ausgaben erzeugen, je nachdem, ob sie ein `Student`- oder ein `GraduateStudent`-Objekt erhält.
2. Überlege dir eine abstrakte Klasse oder Methode, die in diesem Kontext sinnvoll sein könnte, und implementiere sie. Eine Möglichkeit wäre, eine abstrakte Methode `get_status` zu definieren, die in den abgeleiteten Klassen unterschiedlich implementiert wird, um den Status des Studierenden (z.B. "undergraduate" oder "graduate") zurückzugeben.