Speichern Sie den verketteten String in der Variablen `euroFilename` (❶) und übergeben Sie dann den ursprünglichen Dateinamen aus `amerFilename` und den neuen aus `euroFilename` an die Funktion `shutil.move()`, um die Datei umzubenennen (❸).

Der Aufruf von `shutil.move()` ist vorläufig auskommentiert; stattdessen werden die umzubenennenden und die neuen Dateinamen ausgegeben (❷). Auf diese Weise können Sie zunächst überprüfen, ob die Umbenennungsaktion auch tatsächlich richtig erfolgt. Nach diesem Test können Sie das Kommentarzeichen vor dem Aufruf von `shutil.move()` entfernen und das Programm erneut ausführen, um die Dateien tatsächlich umzubenennen.

Vorschläge für ähnliche Programme

Es gibt noch viele weitere Gründe, um größere Mengen von Dateien umzubenennen:

- Um dem Dateinamen ein Präfix voranzustellen, beispielsweise *spam_*, sodass *eggs.txt* in *spam_eggs.txt* geändert wird
- Um europäische Datumsangaben in Dateinamen in amerikanische umzuwandeln
- Um Nullen aus Dateinamen wie *spam0042.txt* zu entfernen

Projekt: Einen Ordner in einer ZIP-Datei sichern

Nehmen wir an, Sie haben alle Dateien eines Projekts im Ordner *C:\AlsPythonBook* gespeichert. Da Sie all die Arbeit, die Sie in das Projekt investiert haben, nur ungern verlieren möchten, wollen Sie »Momentaufnahmen« dieses Ordners in Form von ZIP-Dateien anlegen. Da Sie außerdem unterschiedliche Versionen aufbewahren möchten, sollen die Dateinamen dieser ZIP-Dateien eine laufende Nummer aufweisen, also etwa *AlsPythonBook_1.zip*, *AlsPythonBook_2.zip* usw. Das könnten Sie zwar auch manuell machen, allerdings ist das eine eher lästige Aufgabe. Außerdem besteht die Gefahr, dass Sie die ZIP-Dateien versehentlich falsch nummerieren. Es wäre viel einfacher, wenn ein Programm diese langweilige Arbeit für Sie übernehmen würde.

Öffnen Sie für dieses Projekt ein neues Dateieditorfenster und speichern Sie das Programm als *backupToZip.py*.

Schritt 1: Den Namen der ZIP-Datei bestimmen

In diesem Programm schreiben wir den Code in die Funktion `backupToZip()`, um die Komprimierung auch ganz einfach in anderen Programmen zur Verfügung

stellen zu können. Am Ende des Programms wird die Funktion aufgerufen, um die Sicherung vorzunehmen. Schreiben Sie zunächst folgenden Code:

```
#! python3
# backupToZip.py - Kopiert einen Ordner und seinen gesamten Inhalt in eine
# ZIP-Datei mit laufender Nummerierung

import zipfile, os   ❶

def backupToZip(folder):
    # Sichert den ganzen Inhalt von "folder" in einer ZIP-Datei

    folder = os.path.abspath(folder) # "folder" muss ein absoluter Pfad sein

    # Ermittelt aufgrund der bereits vorhandenen Dateinamen den Namen für die
    # aktuelle Datei
    number = 1   ❷
    while True:   ❸
        zipFilename = os.path.basename(folder) + '_' + str(number) + '.zip'
        if not os.path.exists(zipFilename):
            break
        number = number + 1

    # TODO: ZIP-Datei erstellen   ❹

    # TODO: Den Verzeichnisbaum durchlaufen und alle Dateien in allen Ordnern
    # komprimieren
    print('Done.')

backupToZip('C:\\delicious')
```

Als Erstes schreiben Sie als Grundgerüst die Shebang-Zeile (`#!`), eine Beschreibung, was das Programm macht, und die Befehle für den Import der Module `zipfile` und `os` (❶).

Anschließend definieren Sie die Funktion `backToZip()`. Ihr einziger Parameter ist der String `folder`, der den Pfad zu dem zu komprimierenden Ordner angibt. Die Funktion ermittelt den Dateinamen für die neue ZIP-Datei, erstellt diese, durchläuft den in `folder` angegebenen Ordner und fügt alle darin enthaltenen Unterordner und Dateien zu der ZIP-Datei hinzu. Für die anderen Schritte, die noch zu erledigen sind, geben Sie vorläufig `TODO`-Kommentare als Gedächtnisstütze im Code an (❹).

Für die Benennung der ZIP-Datei greift die Funktion auf den Grundnamen des in `folder` angegebenen absoluten Pfads zurück. Soll beispielsweise der Ordner *C:\delicious* gesichert werden, lautet der Name der ZIP-Datei *delicious_N.zip*, wobei *N* bei der ersten Ausführung des Programms *1* ist, bei der zweiten Ausführung *2* usw.

Den Wert von *N* ermitteln Sie, indem Sie prüfen, ob *delicious_1.zip* bereits vorhanden ist; wenn ja, müssen Sie nachsehen, ob es auch *delicious_2.zip* schon gibt, usw. Für *N* wird die Variable `number` verwendet (❷). In der Schleife, die `os.path.exists()` aufruft, um zu prüfen, ob die Dateinamen schon vorhanden sind, wird diese Variable ständig inkrementiert (❸). Bei dem ersten noch nicht vorhandenen Dateinamen wird die Schleife mit `break` abgebrochen. Damit haben wir den Dateinamen für das neue ZIP-Archiv gefunden.

Schritt 2: Die neue ZIP-Datei erstellen

Als Nächstes geht es daran, die ZIP-Datei tatsächlich anzulegen. Das Programm muss jetzt wie folgt aussehen:

```
#! python3
# backupToZip.py - Kopiert einen Ordner und seinen gesamten Inhalt in eine
# ZIP-Datei mit laufender Nummerierung

-- schnipp --

    while True:
        zipFilename = os.path.basename(folder) + '_' + str(number) + '.zip'
        if not os.path.exists(zipFilename):
            break
        number = number + 1

    # Erstellt die ZIP-Datei.
    print(f'Creating {zipFilename}...')
    backupZip = zipfile.ZipFile(zipFilename, 'w')   ❶

    # TODO: Den Verzeichnisbaum durchlaufen und alle Dateien in allen Ordnern
    # komprimieren
    print('Done.')

backupToZip('C:\\delicious')
```

Da der vorgesehene Name für die neue ZIP-Datei jetzt in der Variablen `zipFilename` gespeichert ist, können Sie `zipfile.ZipFile()` aufrufen, um das Archiv tatsächlich anzulegen (❶). Als zweites Argument müssen Sie dabei `'w'` übergeben, damit die ZIP-Datei im Schreibmodus geöffnet wird.

Schritt 3: Den Verzeichnisbaum durchlaufen und Inhalte zur ZIP-Datei hinzufügen

Nun müssen wir die Funktion `os.walk()` verwenden, um sämtliche Dateien in dem Ordner und allen seinen Unterordnern zu durchlaufen. Ergänzen Sie das Programm wie folgt:

```
#! python3
# backupToZip.py - Kopiert einen Ordner und seinen gesamten Inhalt in eine
# ZIP-Datei mit laufender Nummerierung

-- schnipp --

    # Durchläuft den Verzeichnisbaum und komprimiert alle Dateien in allen
    # Ordnern
    for foldername, subfolders, filenames in os.walk(folder):   ❶
        print(f'Adding files in {foldername}...')
        # Fügt den aktuellen Ordner zur ZIP-Datei hinzu
        backupZip.write(foldername)   ❷
        # Fügt alle Dateien in diesem Ordner zur ZIP-Datei hinzu

        for filename in filenames:   ❸
            newBase = os.path.basename(folder) + '_'
            if filename.startswith(newBase) and filename.endswith('.zip')
                continue # Keine Sicherung der ZIP-Sicherungsdateien!
            backupZip.write(os.path.join(foldername, filename))
    backupZip.close()
    print('Done.')

backupToZip('C:\\delicious')
```

Wenn Sie die Funktion `os.walk()` in einer `for`-Schleife verwenden (❶), gibt sie in jeder Iteration den Namen des aktuellen Ordners sowie die Namen der darin enthaltenen Dateien und Unterordner zurück.

In der `for`-Schleife wird der laufende Ordner zu der ZIP-Datei hinzugefügt (❷). Die verschachtelte `for`-Schleife geht alle Dateinamen in der Liste `filenames` durch (❸). Alle auf diese Weise gefundenen Dateien werden der ZIP-Datei hinzugefügt. Die einzigen Ausnahmen bilden zuvor angelegte ZIP-Archive.

Bei der ersten Ausführung gibt dieses Programm Folgendes aus:

```
Creating delicious_1.zip...
Adding files in C:\delicious...
Adding files in C:\delicious\cats...
Adding files in C:\delicious\waffles...
Adding files in C:\delicious\walnut...
Adding files in C:\delicious\walnut\waffles...
Done.
```