Python3 Aufgaben

Inhaltsverzeichnis

Vorbemerkungen . . . 1
Aufgabe 1: Erkennen von IPv4 Adressen . . . 2
  Schlüsselwörter . . . 3
Aufgabe 1.1: Schreiben einer Funktion . . . 3
Lösung 1: . . . 3
  Vorbemerkung . . . 3
  Teillösungen . . . 4
Aufgabe 2: Versuch und Irrtum . . . 9
Lösung 2: . . . 9

Author: jz
eMail: jz@mgeg.de
Version: 1.0.0

Vorbemerkungen

Die folgenden Aufgaben orientieren sich an Fragestellungen wie sie in der Praxis so, oder so ähnlich vorkommen.
Bei einigen Aufgaben werden mehrere Fragen gestellt die nach Schwierigkeitsgrad gestaffelt sind. Ziel soll nicht sein alle Fragen richtig zu beantworten. Die schwierigeren Fragen werden für diejenigen Teilnehmer gestellt, denen die Beantwortung der ersten Fragen leichter fallen.
Für die Lösung der Fragen sind viele Lösungswege denkbar. Einige sind eleganter als andere. Entscheidend ist es aber überhaupt ein Lösung zu finden. Bitte beschränken Sie sich bei der Programmierung der Lösungen auf die bisher behandelten Schlüsselwörter/Konstrukte. Diese sind im unteren Teil einer Aufgabe aufgeführt.
Bitte verwenden Sie für die Lösung der Aufgaben nicht Google oder andere Suchmaschinen sondern versuchen Sie selber eine Lösung zu erarbeiten.
Wenn es Probleme gibt die Sie nicht lösen können, untersuchen Sie Ihr Script auf folgende Punkte:

- sind die Einrückungen korrekt (verwenden Sie ausschliesslich vier Leerzeichen zum Einrücken)
- lesen Sie die Fehlermeldung des Interpreters **genau**, die Fehlermeldungen zeigen Ihnen in der Regel die fehlerhafte Zeile sowie die Art des Problems an.
- folgen Sie dem Vier Augen Prinzip; Lassen Sie Ihren Nachbarn über Ihren Code drüber schauen. Häufig erkennt ein Aussenstehender Fehler schneller weil er nicht *betriebsbind* ist.
- Jeden Fehler den Sie selber finden verschafft Ihnen ein Erfolgserlebnis, trotzdem sollten Sie nich zögern um nach Unterstützung zu fragen.

- starten Sie Python3 interacktiv und probieren Sie einzelne Konstrukte in der Python3 Konsole aus

Wer Rechtschreibfehler findet darf sie behalten

Aufgabe 1: Erkennen von IPv4 Adressen

Sie erhalten eine CSV Datei in der IPv4 Adressen eingetragen sind.
Lesen Sie die Datei mit Python3 ein. Das Program soll nun für jede Zeile die unten aufgeführten Fragen beantworten.

1. Handelt es sich bei der aktuellen Zeile um eine IPv4 Host Adresse?
2. Handelt es sich bei der aktuellen Zeile um eine IPv4 Netzmaske?
3. Handelt es sich bei der aktuellen Zeile um eine IPv4 Adresse mit einer Netzmaske in Kurzschreibweise?
4. Handelt es sich bei der aktuellen Zeile um eine IPv4 Broadcast Adresse?
5. Handelt es sich bei der aktuellen Zeile um eine private IPv4 Adresse?
6. Handelt es sich bei der aktuellen Zeile um eine IPv6 Adresse?
7. Handelt es sich bei der aktuellen Zeile um einen gültigen Wert?

Die Aufgaben sind nach Schwierigkeitsgrad gestellt. Schwerere Aufgaben folgen den leichteren Aufgaben.
In der CSV sind mehrere Spalten enthalten. In der ersten Zeile steht eine Beschreibung der einzelnen Spalten.

network_stuff.csv

```
address;private;network;broadcast;netmask;hosts;ipv6;bin
138.201.41.13;;;;;;;
192.168.12.22;;;;;;;
some garbage;;;;;;;
172.32.9.31;;;;;;;
8.8.8.8;;;;;;;
192.168.23.0/24;;;;;;;
13.13.13.15/28;;;;;;;
10.0.12.23/12;;;;;;;
192.168.0.8/29;;;;;;;
7.7.7.7;;;;;;;
192.168.23.63/30;;;;;;;
255.255.255.240;;;;;;;
0.0.0.0;;;;;;;
fe80::f486:739b:e9e5:23f6/64;;;;;;;
127.0.1.53;;;;;;;
225.255.255.240/28;;;;;;;
192.168.178.21/24;;;;;;;
192.168.123.255;;;;;;;
```

Schlüsselwörter

- **Kommentare**
- **einfache Variablen**
- **Listen**
- **Dictionaries**
- **print**
- **formatierte Strings** (print(f'{chars}{zeichenkette}{chars}'))
- **if, elif, else**
- **for** oder **while**
- **string.split()**
- **open with**
- **Vergleiche** (< > == >=)
- **Funktionen** (def)
- **Substrings** (string[2:5])
- **input('Wieviel Geld wollen Sie heute ausgeben: ')**
- **list.append(result)**

Aufgabe 1.1: Schreiben einer Funktion

Das obige Programm erkennt zuverlässig IPv4 Adressen. Diese Funktionalität ist für Sie wertvoll. Lagern Sie die obige Funktionalität daher in eine Funktion aus. Die Funktion soll als Argument die IP-Adresse erhalten. Die Funktion soll eine der folgenden Antworten geben:

- öffentliche IPv4 Adresse
- private IPv4 Adresse
- Netzwerkadresse
- Broadcast
- Broadcast
- Netzmaske
- IPv6 Adresse
- etwas Müll

Lösung 1:

Vorbemerkung

In der Realität sind die Dinge häufiger deutlich komplexer als es zunächst den Anschein hat. Bei der

Lösung der Aufgabe geht es nicht um eine 100%'tige Lösung, dafür gibt es viel zu viele Sonderfälle zu beachten. In der Praxis würde man zur Lösung dieser Aufgabe auf Bibliotheken zurückgreifen. Je nach Qualität einer Bibliothek sind diese Sonderfälle dann berücksichtigt und der Program Code wäre deutlich einfacher.
Die im weiteren vorgestellte Lösung verwendet die bisher gelernten Schlüsselwörter. Auch hier ist zu beachten das die Lösung keineswegs die beste (geschickteste) Lösung darstellt. Python selber hat eine große Vielzahl von Konstukten um Aufgaben sehr effizient zu lösen. Die Anzahl der Lösungsmöglichkeiten ist schier unendlich, es geht hier um eine einfach nachzuvollziehende Lösung.

Zur Lösung der Aufgabe ist es hilfreich die Aufgabe zunächst in kleinere Aufgaben zu unterteilen. So könnten Sie sich z.B. folgende Liste erstellen um die Aufgabe in kleinere Aufgaben zu zerlegen:

Teillösungen

- lese eine CSV Datei zeilenweise ein
- nehme jede Zeile und extrahiere die erste Spalte
- untersuche die erste Spalte auf ungültige Zeichen und breche den aktuellen Schleifendurchlauf bei ungültigen Zeichen ab
- Die erste Spalte muss aus vier Dezimalzahlen bestehen die durch drei Punkte voneinander getrennt sind.
- Die letzte Zahl kann auch einen / als Trennzeichen zwischen IP und Netzmaske in Kurzschreibweise enthalten
- Alle Bestandteile können jetzt in Ganze Dezimalzahlen (Integers) umgewandelt werden.
- Sollte die jeweilige Stelle der IP Adresse nicht im Bereich zwischen 0 bis 255 liegen, kann die Bearbeitung der aktuellen Zeile abgebrochen werden.

Bis hierhin ist der erste große Teil der Aufgabe gelöst. Jetzt kommen wir zur Lösung der gegestellten Aufgaben. Im folgenden wird die Aufgabe 1 gelöst, die Lösung der weiteren Fragen erfolgt analog.

eine mögliche Lösung des ersten Teils

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-

file = '/home/tn/bin/network_stuff.csv'

first_line = True
# Datei einlesen und die erste Zeile ueberspringen
with open(file, 'r') as fh:
    for line in fh.readlines():
        if first_line is True:
            first_line = False
            continue
        # erste Spalte
        rows = line.split(';')
```

```
        pattern = rows[0]
        octett = pattern.split('.')

        # Es kann nur dann eine gültige IP sein, wenn es genau vier Oktetts hat
        lenght = 0
        for i in octett:
            lenght = lenght + 1
        if len != 4:
            continue

        # haben wir eine Netzmaske in Kurzschreibweise
        items = octett[3].split('/')
        if items > 1:
            netmask = items[1]
            ipaddress = [octett[0], octett[1], octett[2], items[0]]
            # naechste Zeile wenn die Netzmaske keine Zahl ist
            try:
                int(netmask)
            except:
                continue
        else:
            netmask = False
            ipaddress = octett

        # enthalten alle otetts der IPadresse nur Integers
        try:
            for octett in ipaddress:
                int(octett)
        except:
            continue

        # liegen die Integers im Bereich zwischen 0 und 255
        next_line = True
        for octett in ipaddress:
            if octett >= 0 and octett < 256:
                next_line = True
            else:
                next_line = False
                break
        # Abbruch der aktuellen Zeile
        if next_line is False:
            continue

        # jetzt koennen die Fragen nach dem Typ der Adresse beantwortet werden

        # ermitteln der Host Adresse
        # einfachste Annahme, keine Netzmaske
        if netmask is False and octett[3] > 0 and octett[3] < 255:
            print(f'{octett[0]}.{octett[1]}.{octett[2]}.{octett[3]} ist eine Host
Adresse')
        else:
```

```
            pass
```

Wann immer Programmcode mehrfach verwendet wird, sollte er in ein Funktion ausgelagert werden

Die erste Aufgabe gilt als gelöst, wenn sich keine Netzmaske ermitteln lässt. Dann kann die IP Adresse als Host Adresse betrachtet werden wenn die erste Stelle der IP Adresse kleiner 255 ist.

In der Realität ist die Frage natürlich etwas komplexer, aber für unsere Zwecke reicht die Lösung aus.

Handelt es sich um eine Host oder um eine Netzadresse

Lösung der Frage ob es sich bei der IP Adresse um eine Netzwerkadresse oder um eine Broadcast Adresse handelt.
Dazu muss eine Netzmaske vorhanden sein. Im folgenden wird jetzt berechnet welche Netzwerkadressen und welche Broadcast Adressen möglich sind. Durch einen Vergleich kann dann ermittelt werden ob es sich um:

1. eine Host Adresse handelt
2. eine Netzwerk Adresse handelt
3. eine Broadcast Adresse handelt

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-


file = '/home/tn/bin/network_stuff.csv'

# Funktion die prueft ob alle Zeichen der Eingabe in einer gegebenen Liste enthalten
sind
# Die Funktion gibt True odder False zurueck
def is_something(pattern, liste):
    for item in pattern:
        schalter = False
        for element in liste:
            if item == element:
                schalter = True
                break
        if schalter is False:
            return False
    return True

# berechnen der Netzwerkadressen und der Broadcasts zu einer Netzmask
def netzwerke():
    counter = 0
    amount_networks = 0
    ipranges = {}
```

```
    for i in range(24, 31):
        amount_networks = 2 ** counter
        counter += 1
        sprung = int(256 / amount_networks)
        ipliste = []
        for i2 in range(0, amount_networks):
            ipliste.append((i2*sprung, (i2+1)*sprung-1))
        ipranges[str(i)] = ipliste
    return ipranges

iprange = netzwerke()
# Datei einlesen und die erste Zeile ueberspringen
first_line = True
with open(file, 'r') as fh:
    for line in fh.readlines():
        # erste Zeile ueberspringen
        if first_line is True:
            first_line = False
            continue
        # die erste Spalte ermitteln
        rows = line.split(';')
        # die erste Spalte in Oktetts zerlegen und in einer Liste speichern
        pattern = rows[0]
        octetts = pattern.split('.')

        # ermittelen der Lange der Liste
        l = 0
        for i in octetts:
            l = l + 1

        # es kann nur eine IP Adresse sein wenn es genau vier Oktetts gibt
        if l != 4:
            continue

        # Das letzte Oktett kann eine Netzmaske enthalten
        items = octetts[3].split('/')
        l = 0
        for i in items:
            l = l + 1

        # Wenn die Lange des letzten Oktetts 1 ist, haben wir keine Netzmaske
        if l == 1:
            netmask = ''
        # Wenn die Lange des letzten Oktetts 2 ist, haben wir eine Netzmaske
        elif l == 2:
            octetts[3] = items[0]
            netmask = items[1]
        else:
            continue

        # enhaelt ein Oktett ungueltige Zeichen
```

```
        # (gueltige Zeichen sind die Zahlen 0 bis 9)
        invalid = False
        liste = ['0','1','2','3','4','5','6','7','8','9']
        for octett in octetts:
            result = is_something(octett, liste)
            # Wenn die enthaltenen Zeichen nur Zahlen sind pruefen wir ob das Oktett
            # zwischen 0 und 256 liegt
            if result is not False:
                _octett = int(octett)
                if _octett >= 0 and _octett <= 255:
                    pass
            else:
                # Das Oktett liegt ausserhalb des gueltigen Bereichs
                invalid = True
                break
        if invalid is True:
            break

        # Wenn es eine Netzmaske gibt pruefen wir ob sie gueltig ist (2 bis 30)
        typ = 'host'
        if netmask != '':
            invalid = False
            liste = ['0','1','2','3','4','5','6','7','8','9']
            result = is_something(netmask, liste)
            if result is not False:
                _netmask = int(netmask)
                if _netmask >= 2 and _netmask <= 30:
                    pass
                else:
                    invalid = True
            if invalid is True:
                break

            # Vergleiche ob die aktuelle IP Adresse eine Netzwerkadresse oder
            # Broadcast ist
            result1 = iprange[str(netmask)]
            typ = 'host'
            for network in result1:
                if str(octetts[3]) == str(network[0]):
                    typ = 'netzwerk'
                elif octetts[3] == network[1]:
                    typ = 'broadcast'
                else:
                    typ = 'host'

        # Ergebnis
        print(f'IP: {octetts[0]}.{octetts[1]}.{octetts[2]}.{octetts[3]} -- Netmaske:
{netmask}, Typ: {typ}')
```

Aufgabe 2: Versuch und Irrtum

Wenn Sie in Python3 z.B. auf eine Variable zugreifen wollen die nicht existiert bekommen Sie eine Fehlermeldung und die Ausführung des Programmes wird unterbrochen. Es gibt eine Vielzahl von Situationen in denen ein Fehler in der Programmausführung auftritt die zu einem Abbruch des Programmes führen.
Für diese Fälle kennt Python3 das Konzept der Ausnahmebehandlung. Dabei werden die kritischen Code Teile in einen try Block geschrieben. Die Konstruktion sieht wie folgt aus:

nicht zulässige Division durch 0

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-

zahl = 12
try:
    ergebnis = zahl / 0
except:
    print('Das war wohl nix, aber es geht weiter')
```

Wenn Sie versuchen eine Zeichenkette in einen Integer (Ganzzahl) umzuwandeln, wird Python mit einer Fehlermeldung abbrechen. Schreiben Sie eine *robuste* Funktion zum umwandeln von Zeichenketten in Integers die als Rückgabe die umgewandelte Zahl oder ein False für den Fehlerfall zurückgibt.

Lösung 2:

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-
def get_int(pattern):
    try:
        result = int(pattern)
        return result
    except:
        return False
```