

Logik-Rätsel

Derartige Denkaufgaben kann man prinzipiell auf zwei Arten lösen:

- 1.) die kluge Art: mittels Nachdenken und Schlussfolgern. Sie ist dem Menschen vorbehalten.
- 2.) die mühsame Art: mittels Durchprobieren aller Möglichkeiten. Das kann der Computer auch – hier braucht's ja keinen Verstand, sondern nur kleinliche Buchführung über alle zu testenden Möglichkeiten.

Wir wollen Logikaufgaben mit Python lösen.

Was ist zu tun:

- 1.) Der Mensch übersetzt die Aufgabe in eine dem Computer vertraute Sprache
- 2.) Der Computer sucht sich alle Möglichkeiten
- 3.) Der Computer testet jede dieser Möglichkeiten und zeigt die richtigen an

Eine derartige Methode wird von Informatikern als **brute force** (rohe Gewalt) bezeichnet. Der Name ist treffend gewählt: statt Hirn zu investieren, vertraut man auf die (dumme) Muskelkraft des Computers – er ist schnell und übersieht nichts.

Vorüberlegungen

- die logischen Funktionen heißen in Python `==`, `!=`, `not`, `and` und `or`. (Nicht mit den bitweisen Operationen `&` und `|` verwechseln!)
- die Umgangssprache nimmt es oft nicht sehr genau mit der Logik (was aber vielleicht einen Hinweis darstellt, dass diese Art von Logik nichts 'natürliches' ist, sondern eine mathematische Entwicklung des antiken Abendlandes).
Ein spezieller Problemfall: Logiker unterscheiden zwischen den Worten 'oder' und 'entweder oder' sehr streng. Der Unterschied besteht darin, dass letztere Version ein Zutreffen von beiden Alternativen verbietet.

Beispiele:

'Wenn ich Pizza oder Cola bekomme, bin ich glücklich.' Kriegt man beides, ist man auch glücklich.

'Er kommt mit dem Flugzeug oder mit dem Schiff aus Amerika zu uns'. Hier gibt's kein 'beides'. Also sollte es heißen 'Er kommt entweder mit dem...'

'Ich will Dich oder keine'.

- Umsetzungstips: 'und' und 'oder' sind wohl klar.
Die Implikation 'wenn A dann B' übersetzt man am besten in *if A: if B: richtig*.
Oder negativ formuliert *if A and not B: falsch*.
Das exclusive 'entweder oder' kann man meistens mit einem `!=` übersetzen.

Krimi oder Western?

Was soll heute im Kino angeschaut werden – der Krimi oder der Western? Die 7 Freunde treffen je eine Aussage. Wir sollen herausfinden, wer was sehen möchte.

Bernd: "Gitte oder Ingrid wollen den Western sehen."

Gitte: "Frank oder Jens schwärmen für den Krimi."

Frank: "Entweder will Bernd oder Ingrid den Western sehen."

Ingrid: "Gitte und Vera sind entweder beide für den Western, oder beide für den Krimi."

Jens: "Entweder entscheiden sich Frank für den Western und Vera für den Krimi, oder es stimmen Ingrid für den Western und Petra für den Krimi."

Petra: "Bernd und Franks Interesse gilt dem Western, oder Ingrid und Jens wollen nicht den Krimi."

Vera: "Petra wählt den Western, oder Bernd und Gitte sehen lieber den Krimi."

Nehmen wir etwa für den Western die Zahl 1, für den Krimi die Zahl 2. Dann können wir mit geschachtelten for-Schleifen alle Möglichkeiten erzeugen.

Beim Testen einer Möglichkeit müssen alle Bedingungen erfüllt sein – wir zählen einfach mit, wie viele der Aussagen zutreffen. Sind es alle 7, haben wir eine Lösung gefunden und zeigen sie an.

(Danach arbeiten wir jedoch weiter – es könnte ja mehrere Lösungen geben!)

Wir erzeugen in *solve* alle Möglichkeiten und rufen dabei *check* auf. Die Variablen sollen den Anfangsbuchstaben der Personen entsprechen.

Version 1

```
def solve():
    print "B G F I J P V"
    for b in [1,2]:
        for g in [1,2]:
            for f in [1,2]:
                for i in [1,2]:
                    for j in [1,2]:
                        for p in [1,2]:
                            for v in [1,2]:
                                check(b,g,f,i,j,p,v)
```

```
def check(b,g,f,i,j,p,v):
    n = 0
    if g==2 or i==2 : n+=1
    if f==1 or j==1 : n+=1
    if b!=i : n+=1
    if g==v : n+=1
    if (f==2 and v==1) != (i==2 and p==1) : n+=1
    if (b==2 and f==2) or (i==2 and j==2) : n+=1
    if (p==2) or (b==1 and g==1) : n+=1

    if n==7: print b,g,f,i,j,p,v
```

n ist der Zähler

Bernd

Gitte

Frank

Ingrid

Jens

Petra

Vera

falls alles richtig war

Eigentlich arbeiten wir sehr unrationell. Auch wenn bereits die erste Bedingung nicht erfüllt sein sollte, arbeiten wir alle 6 weiteren ab. Dabei ist doch schon hier klar, dass diese Kombination gar keine Lösung sein kann.

Verbesserung: statt zu testen und einen Zähler zu erhöhen drehen wir die Tests um: Wir testren auf das **Nichterfülltsein** der Aussage. Falls das zutrifft, arbeiten wir gar nicht mehr weiter und verlassen sofort mit *return* die Funktion. Bei größeren Aufgaben bringt das eine wesentliche Laufzeitreduktion!

Version 2

```
def check(b,g,f,i,j,p,v):
    if not (g==2 or i==2): return
    if not (f==1 or j==1): return
    if not (b!=i): return
    if not (g==v): return
    if not ( (f==2 and v==1) != (i==2 and p==1) ): return
    if not ( (b==2 and f==2) or (i==2 and j==2) ): return
    if not ( (p==2) or (b==1 and g==1) ): return

    print b,g,f,i,j,p,v
```

Ein anderer Ansatz für *solve*: ich möchte im Vorhinein alle Möglichkeiten erzeugen, und dann erst alle durchtesten. (Das empfiehlt sich nur für kleinere Aufgaben, da alle Werte im Speicher gehalten werden müssen. In unserem Fall sind es 2^7 , also nur 128 Stück.) Das klappt mit dem Konzept der 'list comprehension'. Da die erzeugten Möglichkeiten als Tupel übergeben werden, habe ich bei *check* doppelte Klammern: außen die jeder Funktion, innen die des Tupels.

Weiters möchte ich gerne 'sprechendere' Variablennamen haben.

Version 3

```
K = 1
W = 2

def solve():
    print "B G F I J P V"
    l = [(b,g,f,i,j,p,v) for b in [K,W] for g in [K,W] for f in [K,W] \
        for i in [K,W] for j in [K,W] for p in [K,W] \
        for v in [K,W] ]
    for test in l: check(test)

def check((b,g,f,i,j,p,v)):
    if not (g==W or i==W): return
    if not (f==K or j==K): return
    if not (b!=i): return
    if not (g==v): return
    if not ( (f==W and v==K) != (i==W and p==K) ): return
    if not ( (b==W and f==W) or (i==W and j==W) ): return
    if not ( (p==W) or (b==K and g==K) ): return

    print b,g,f,i,j,p,v
```

Weitere Aufgaben:

Schularbeitsnoten

Sieben Schüler machen Angaben über ihre Schularbeitsnoten. Jede Note von 1 bis 4 kommt vor, niemand hat einen 5er bekommen.

1. Wenn Britta einen 2er geschrieben hat, dann hat David keinen 1er.
2. Wenn Christian einen 3er hat, dann hat Erhard einen 2er.
3. Hat Florian keinen 1er, hat Astrid einen 4er.
4. Hat Astrid einen 4er, dann hat Gisela keinen 1er.
5. Hat David einen 1er, dann hat Erhard keinen 2er.
6. Wenn Britta einen 3er hat, bekam Erhard keinen 1er.
7. Hat Gisela einen 3er, dann hat Britta keinen 1er.
8. Hat David keinen 4er, dann hat weder Astrid noch Christian einen 2er.
9. Britta hat eine andere Note als Erhard und Astrid eine andere als Christian.
10. David und Florian haben die gleiche Note, Gisela und Christian wurden ebenfalls gleich benotet.

Was ist hier neu:

Nun gibt es für jede Person 4 verschiedene Möglichkeiten. Weiters müssen wir die Tatsache, dass jede Note von 1 bis 4 vorkam, berücksichtigen.

Bestechungsversuche

Sechs Beamte sollten von der Mafia bestochen werden. Folgende Aussagen wurden getätigt:

1. Der Kreisdirektor sollte Bargeld bekommen, oder der Regierungsrat eine Segeljacht.
2. Dem Kriminalkommissar wollte man zu einem Grundstück verhelfen, oder dem Richter zu Wertpapieren.
3. Wenn dem Kriminalkommissar ein Auto angeboten wurde, hat man dem Staatsanwalt eine Segeljacht offeriert.
4. Wollte man den Polizeipräsidenten mit einem Wohnmobil bestechen, dann versuchte man es beim Regierungsrat mit einem Grundstück.
5. Wenn der Regierungsrat nicht das Auto bekommen sollte, war für den Staatsanwalt weder die Segeljacht noch das Wohnmobil vorgesehen.
6. Der Kriminalkommissar sollte weder das Grundstück noch das Wohnmobil erhalten.
7. Das Auto war weder für den Polizeipräsidenten noch für den Staatsanwalt bestimmt.

Wer sollte womit bestochen werden?

Was ist hier neu:

Jedem Element der Personenmenge ist genau ein Element der Geschenkemenge zuzuordnen. Wir müssen die mehrfache Vergabe eines Geschenks unmöglich machen. Es läuft also auf eine Permutation hinaus. Eine Lösungsstrategie wäre das tatsächliche Bestimmen der Permutationen, eine andere die Kennzeichnung bereits vergebener Gegenstände (wie wir es bei den Zahlenrätseln gemacht haben).

Wahre und falsche Aussagen

Welche Aussagen sind wahr, und welche sind falsch?

- A: Die Aussagen C und E sind wahr.
- B: Die Aussagen D und H sind falsch.
- C: Die Aussagen G oder H sind wahr.
- D: B und G sind beide wahr oder beide falsch.
- E: Genau eine der Aussagen D und G ist wahr.
- F: D ist wahr, oder H ist falsch.
- G: Aussage C ist wahr.
- H: Von den Aussagen B, D und G sind genau zwei falsch.

Was ist hier neu:

Von jeder Aussage müssen wir zwei Versionen berücksichtigen: dass sie wahr ist, und dass sie falsch ist.

z.B.

```
if a==1:
    if c==1 and e==1: richtig
else:
    if not (c==1 and e==1): richtig
```

Frühstücksbuffet

Über das Frühstücksbuffet im Hotel konnte man hören:

- Martin: "Wenn Wurst oder Tee fehlten, dann war auch kein Kaffee da."
- Norbert: "Wenn keine Semmeln da waren, gab's weder Tee noch Fruchtsaft."
- Robert: "Wenn Käse vorhanden war, gab es auch genug Marmelade."
- Stefan: "Wenn die Butter fehlte, war jedenfalls Kaffee erhältlich."
- Tobias: "Gab es keinen Käse, dann wartete man auch vergeblich auf Wurst."
- Uwe: "Wenn man Fruchtsaft bekam, musste man auf Marmelade verzichten."
- Volker: "Wenn Kaffee fehlte, dann war auch keine Butter da."

Das hört sich schlimm an – war es wirklich so schrecklich im Hotel?

Was ist hier neu:

Die if-Bedingungen sind hier etwas komplexer.

Pfusch am Bau

Am Neubau sind Schäden aufgetreten. Um die Schuldfrage zu klären, werden die Beteiligten befragt. Nach gegenseitigen Schuldzuweisungen geben offenbar widersprüchliche Angaben.

Architekt: "Der Fliesenleger lügt immer, aber der Maurer nie."

Dachdecker: "Der Architekt und der Tischler lügen, sobald sie den Mund aufmachen."

Elektriker: "Entweder sagt der Maler oder der Zimmermann stets die Wahrheit."

Fliesenleger: "Der Maurer lügt immer oder der Zimmermann nie."

Glaser: "Entweder lügen Elektriker und Dachdecker nie oder beide immer."

Installateur: "Der Zimmermann lügt ständig und der Glaser nie."

Maler: "Entweder ist der Tischler oder der Installateur ein notorischer Lügner."

Maurer: "Entweder sagen Architekt und Fliesenleger stets die Wahrheit oder beide nie."

Tischler: "Wenn ich immer bei der Wahrheit bleibe, dann tut das auch der Installateur."

Zimmermann: "Der Maurer lügt, aber der Elektriker ist ehrlich."

Wer lügt, und wer sagt die Wahrheit?

Zeugenaussagen

Nach dem Banküberfall sagten Zeugen, die als gute Beobachter bekannt sind, folgendes aus:

1. Wenn der Täter keine Glatze hatte und mit einem PKW geflüchtet ist, dann war er nicht tätowiert.
2. Wenn er einen Overall trug, war er glatzköpfig.
3. Wenn er groß und glatzköpfig war, flüchtete er mit einem PKW.
4. Wenn er Dialekt sprach und einen Trainingsanzug trug, dann hatte er eine Tätowierung und einen Vollbart.
5. Wenn er das Geld in einem Koffer weggeschleppt hatte, war er klein oder trug einen Tennisanzug.
6. Wenn er groß war und das Geld in die Tasche gestopft hatte, trug er einen Tennisanzug und hatte volle Haare.
7. Wenn er klein war, sprach er Dialekt und machte sich mit einem Fahrrad aus dem Staub.
8. Wenn er tätowiert war, dann war er bartlos.
9. Wenn er einen Trainingsanzug trug und tätowiert war, dann hatte er das Geld in einen Koffer gepackt.

Der Hilfskommissar macht sich ans Werk und offeriert bald seine Folgerungen:

1. Der Täter war groß
2. Er trug einen Overall
3. Er benutzte einen Koffer für das Geld
4. Er fuhr mit dem Fahrrad davon
5. Er sprach Dialekt
6. Wenn er tätowiert war, hatte er eine Glatze
7. Wenn er nicht tätowiert war, hatte er volle Haare
8. Wenn er volle Haare und einen Vollbart hatte, dann verstaute er das Geld in der Tasche

Welche Folgerungen sind richtig, welche Falsch?

Was ist hier neu:

Wir müssen aus einem 'unvollständigen' Informationsvorrat schöpfen und nur Teilaspekte testen.