Send More Money

Diese Aufgabe wird gern als Geschichte erzählt:

> Ein amerikanischer Schüler macht eine Weltreise. Als er gerade Europa besichtigt, geht ihm das Geld aus. Er schickt eine Postkarte nach Hause, auf der zu lesen ist:
>
> ```
>   SEND
> + MORE
> ------
>  MONEY
> ```
>
> Wenn jeder Buchstabe für eine Ziffer steht (jeder für eine andere) – welchen Geldbetrag wünscht sich der Student von seinen Eltern?

Wir wollen eine Strategie entwickeln, wie wir derartige Zahlenrätsel mithilfe eines Computers lösen können.

Zuerst denkt man wohl an rohe Gewalt: jeder Buchstabe steht für eine Ziffer von 0 bis 9. Wir probieren einfach alles durch! Wir benötigen die Werte der Buchstaben s,e,n,d,m,o,r und y:

```
def money():
    for s in range(10):
        for e in range(10):
            for n in range(10):
                for d in range(10):
                    for m in range(10):
                        for o in range(10):
                            for r in range(10):
                                for y in range(10):
                                    # Loesung testen
```

Bevor wir das jedoch ausprobieren, drei kurze Überlegungen:

1.) 8 Variable, von denen jede 10 Werte annehmen kann, macht insgesamt 10^8 bzw. 100 Millionen Möglichkeiten. Das könnte lange dauern! Fügen wir einfach beim Testen in eine Schleife einen `print`-Befehl ein. Dann sehen wir, ob sich was tut und können im Notfall das Programm mittels <strg>C unterbrechen.
2.) Wir haben gelernt, dass der Befehl `range` eine Liste erzeugt. Und das ist wörtlich zu nehmen. Das bedeutet aber, dass die innerste Schleife für y 10 Millionen mal eine Liste **neu erzeugt**. Das ist sicher nicht rationell. Bereiten wir diese Listen besser zuerst vor! Und damit können wir auch noch
3.) die Einschränkung formulieren, dass die Zahlen nicht mit einer Null beginnen sollen. S und M dürfen nur von 1 bis 9 gehen.

Wir gelangen zu

```
ranges = range(1,10)
rangee = range(0,10)
rangen = range(0,10)
ranged = range(0,10)
rangem = range(1,10)
rangeo = range(0,10)
ranger = range(0,10)
rangey = range(0,10)

def money():
    for s in ranges:
        print s
        for e in rangee:
            for n in rangen:
                for d in ranged:
                    for m in rangem:
                        for o in rangeo:
                            for r in ranger:
                                for y in rangey:
                                    # Loesung testen
```

Das könnten wir testen, wenn wir statt der Kommentarzeile den Platzhalterbefehl `pass` eintragen. Auf meinem Computer (Pentium 133) dauert das etwas über 3 Minuten. Nicht sehr ermunternd, speziell wenn wir daran denken, dass a noch gar nicht überprüft wird, ob die Lösung stimmt!

Weitere Ideen:

1.) Bei kurzer Betrachtung zeigt sich, dass M nur den Wert 1 haben kann (es entsteht ja als Übertrag bei der Addition von zwei einstelligen Zahlen, eventuell noch +1). Das reduziert uns die Rechenzeit auf 10%, da `rangem = [1]` übrigbleibt.
2.) Wir haben noch gar nicht berücksichtigt, dass unterschiedliche Buchstaben für unterschiedliche Ziffern stehen. Wie kann man das lösen?

Wenn Werte einander ausschließen, kann man praktischerweise eine Liste mitführen, in der verzeichnet steht, welche Ziffern momentan in Verwendung stehen. Ich nenne diese Liste `used`. Wenn `used` etwa [1,0,0,1,1,0,0,0,0,0] ist, dann soll das bedeuten, dass 0, 3 und 4 belegt sind.

Hilft uns das bezüglich Rechenzeit? Sehr viel! Statt in einer Schleife und in allen ihren Unterschleifen mit einer bereits in Verwendung stehenden Ziffer sinnlos zu arbeiten, erzwingen wir sofort die Weiterarbeit mit der nächsten Ziffer. Dies leistet der Befehl `continue`. Er bricht die Bearbeitung des Schleifenkörpers ab und setzt mit dem nächsten Wert neu fort. (Beachte den Unterschied zu `break`!)
Jede Teilschleife wird also zu

```
wenn Ziffer bereits vergeben: continue
belege die Ziffer
arbeite
gib die Ziffer wieder frei
```

Und wie testen wir, ob die aktuelle Kombination der Variablen passt? Wir berechnen einfach den Wert der Zahlen SEND MORE und MONEY aus den Stellenwerten der Ziffern.

Und damit können wir die Aufgabe endlich lösen.

```
ranges = range(1,10)
rangee = range(0,10)
rangen = range(0,10)
ranged = range(0,10)
rangem = [1]
rangeo = range(0,10)
ranger = range(0,10)
rangey = range(0,10)

def money():
    used = [0,0,0,0,0,0,0,0,0,0]
    for s in ranges:
        print s
        used[s]=1
        for e in rangee:
            if used[e]: continue
            used[e]=1
            for n in rangen:
                if used[n]: continue
                used[n]=1
                for d in ranged:
                    if used[d]: continue
                    used[d]=1
                    for m in rangem:
                        if used[m]: continue
                        used[m]=1
                        for o in rangeo:
                            if used[o]: continue
                            used[o]=1
                            for r in ranger:
                                if used[r]: continue
                                used[r]=1
                                for y in rangey:
                                    if used[y]: continue

                                    z1 = 1000*s+100*e+10*n+d
                                    z2 = 1000*m+100*o+10*r+e
                                    z3 = 10000*m+1000*o+100*n+10*e+y
                                    if z1+z2==z3:
                                        print s,e,n,d,'+',m,o,r,e,'=',m,o,n,e,y

                                used[r]=0
                            used[o]=0
                        used[m]=0
                    used[d]=0
                used[n]=0
            used[e]=0
        used[s]=0
```

Das sieht sogar optisch interessant aus!

Wo bleibt die Intelligenz?

Unser Programm arbeitet eigentlich ziemlich dumm, indem es alle Möglichkeiten stur durchprobiert. Hätten wir 10 verschiedene Buchstaben statt der obigen 8, würde das Programm auch noch 100 mal so lang brauchen.
Beschleunigung erhalten wir aber nur, wenn wir als Gegenleistung ein wenig unser Gehirn einsetzen. Denn alles, was wir uns logisch über die gesuchten Ziffern überlegen, können wir ins Programm einfließen lassen und damit die Menge der zu durchsuchenden Kombinationen einschränken, so wie wir es mit dem M bereits durchgeführt haben.

Untersuchen wir etwa die letzte Stelle der drei Zahlen: hier muss D+E=Y sein, oder mit Übertrag D+E=Y+10. Das könnte man am Beginn der y-Schleife sofort testen und bei Nichterfüllung mit `continue` weitermachen. Bringt aber nicht allzu viel. Viel cleverer ist es, die Gleichungen als Berechnungsformel für Y anzusehen. Wir sparen dann die Schleife für Y ganz ein und senken die Rechenzeit auf ein Zehntel!

```
y = d+e
if y>=10: y -= 10
```

Genauso könnte man nun, wenn man eine Variable für den Übertrag einführt, die vorletzte Stelle berechnen statt erraten...

So kann man das Programm schrittweise weiter verbessern: je mehr Intelligenz hineinkommt, desto schneller findet es die Lösung! Wer entwickelt das schnellste Programm (Benchmark-Funktionen verwenden!)

Eine **Strategie zur Lösung derartiger Zahlenrätsel** haben wir also kennengelernt.

- Probiere alle Möglichkeiten durch
- Berücksichtige Einschränkungen (keine gleichen Ziffern bei unterschiedlichen Buchstaben)
- Lass ein wenig (oder mehr) Intelligenz aus der Analyse der Aufgabe einfließen

Besuche die Schulbibliothek und blättere in Büchern mit mathematischen Rätseln – Du wirst zahlreiche ähnliche Rätsel finden und lösen können.

Natürlich macht das Lösen ohne Computer **viel mehr Spaß**. Doch zur Kontrolle ist unser Programm gut geeignet, oder als Hilfsmittel (lass Dir nur eine einzige Ziffer zur Kontrolle anzeigen und verrate Dir nicht gleich die ganze Lösung!), oder ganz besonders dann, wenn Du selbst solche Rätsel erfinden möchtest. Dann siehst Du gleich, ob Deine Aufgabe lösbar ist, und wie viele Lösungen es insgesamt tatsächlich gibt!

Bemerkung: Bei Multiplikationen und Divisionen kannst Du zur Beschleunigung die Eigenschaft verwenden, dass Ziffern (eventuell mit Übertrag) Vielfache voneinander sind. Wenn etwa die Aufgabe ....a mal ....b ist ...c heißt, so muss `(a*b)%10==c` erfüllt sein!

Kostproben, aber zuerst ohne Computer zu lösen versuchen (Sonst hat man nur den halben Spaß. Merke: der Weg ist das Ziel)! Einige sind ohne Computer sogar schneller zu lösen als mit...

abb . ac abb cdd aecd	DO+RE = MI FA+SI = LA RE+SI+LA = SOL	ATOM.ATOM ....ATOM	ELEVEN - THREE EIGHT	aab.cde dfe afag ehi afafii	Kann BLACK die Quadratwurzel von BEAUTIFUL sein?

Wenn ZOO^2=TOPAZ ist, welches Wort steht dann für TOP+PAT ?	Das Quadrat von XX ist MMCC. Welches Wort erhält man, wenn man XX durch C teilt und das Ergebnis quadriert?	