

Objektorientierung und Funktionale Programmierung

If, else, while

Kristof Van Laerhoven
kvl@eti.uni-siegen.de

Michael Moeller
Michael.moeller@uni-siegen.de

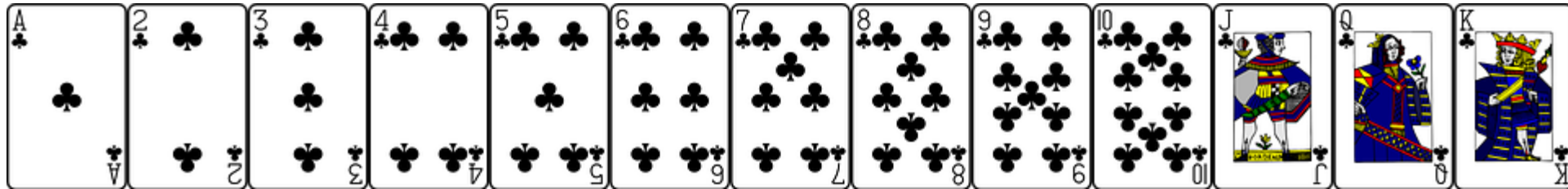
uni-siegen.de



Bislang haben wir einige Datentypen und Manipulationen dieser kennengelernt.

Oft muss man auf unterschiedliche Situationen unterschiedlich reagieren. Als Beispiel wollen wir ein einfaches Spiel Namens Goofspiel (oder Game of Pure Strategy) implementieren.

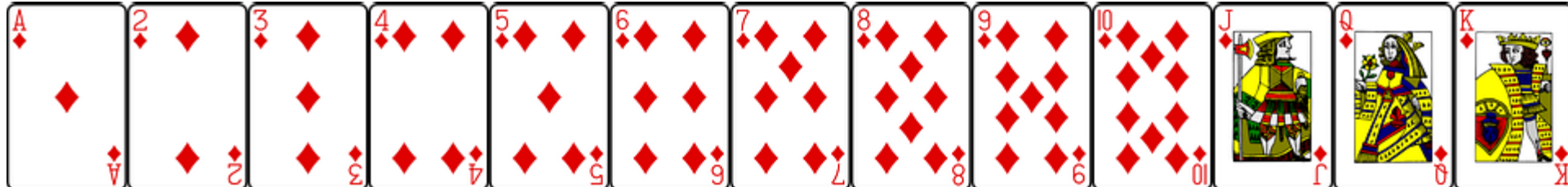
Spieler 1



Preiskarten



Spieler 2



Jeder Spieler erhält eine Farbe eines Kartenspiels. Eine dritte Farbe wird gemischt repräsentiert die Preiskarten, von denen die oberste aufgedeckt wird. Jeder Spieler spielt nun eine Karte. Wer die höhere legt, gewinnt die Preiskarte. Bei Gleichstand bekommt keiner etwas. Wer am Ende die meisten Punkte hat gewinnt.

Was müssen implementieren können?

punkte_spieler_1 = 0

punkte_spieler_2 = 0

- Ziehe eine zufällige Preiskarte
- Spielen einer Karte von Spieler 1 und Spielen einer Karte von Spieler 2
- Wenn die Karte von Spieler 1 einen höheren Wert hat als von Spieler 2
 - Erhöhe punkte_spieler_1 um den Wert der Preiskarte
- Wenn die Karte von Spieler 2 einen höheren Wert hat als von Spieler 1
 - Erhöhe punkte_spieler_2 um den Wert der Preiskarte

Wiederhole obige Abfolge bis keine Karten mehr da sind.

Falls $\text{punkte_spieler_1} > \text{punkte_spieler_2}$

- Spieler 1 hat gewonnen!

Falls $\text{punkte_spieler_1} < \text{punkte_spieler_2}$

- Spieler 2 hat gewonnen!

Falls $\text{punkte_spieler_1} = \text{punkte_spieler_2}$

- Unentschieden!

Fangen wir einmal hinten an

Falls `punkte_spieler_1 > punkte_spieler_2`

➤ Spieler 1 hat gewonnen!

Falls `punkte_spieler_1 < punkte_spieler_2`

➤ Spieler 2 hat gewonnen!

Falls `punkte_spieler_1 = punkte_spieler_2`

➤ Unentschieden!

```
if punkte_spieler_1 > punkte_spieler_2:
    print("Spieler 1 hat gewonnen!")
elif punkte_spieler_1 < punkte_spieler_2:
    print("Spieler 2 hat gewonnen!")
else:
    print('Dieses Spiel war unentschieden!')
```

if (**boolscher_ausdruck**):

*führe diesen Code nur dann aus wenn boolscher_ausdruck = True ist
Es wird alles ausgeführt was eingerückt ist!*

elif (**anderer_boolscher_ausdruck**):

*führe diesen Code nur dann aus wenn
boolscher_ausdruck = False und anderer_boolscher_ausdruck = True ist
Es wird alles ausgeführt was eingerückt ist!*

else:

*führe diesen Code nur dann aus wenn alle if und elif Ausdrücke False waren
Es wird alles ausgeführt was eingerückt ist!*

Einrückungen sind in Python Teil des Programms!
Sie haben einen Einfluss auf die Bedeutung und Funktionsweise des Codes!

Was passiert im ersten und was im zweiten Fall?

```
a = 1
b = 2
c = 3

if a>b:
    print('Der Wert von a ist größer als der von b!')
    if c>b:
        print('Der Wert von c ist größer als der von b')

if a>b:
    print('Der Wert von a ist größer als der von b!')
if c>b:
    print('Der Wert von c ist größer als der von b')
```

Was sind typische Vergleichsoperationen für if-statements?

Operator	Beschreibung	Beispiel	Resultat
==	Sind zwei Werte gleich?	3==3	True
!=	Sind zwei Werte ungleich?	3!=3	False
>	Ist der linke Wert echt größer als der rechte?	3>3	False
<	Ist der linke Wert echt kleiner als der rechte?	3<3	False
>=	Ist der linke Wert größer oder gleich dem rechten?	3>=3	True
<=	Ist der linke Wert kleiner oder gleich dem rechten?	3<=3	True

Zudem gibt es logische Verknüfungen boolscher Variablen

Operator	Beschreibung	Beispiel	Resultat
and	True nur wenn beide Werte True sind.	True and False	False
or	True wenn mindestens einer der Werte True ist.	True or True	True
not	True wenn der Wert False ist.	not False	True

```
punkte_spieler_1 = 0  
punkte_spieler_2 = 0
```

- Ziehe eine zufällige Preiskarte
- Spielen einer Karte von Spieler 1 und Spielen einer Karte von Spieler 2
- Wenn die Karte von Spieler 1 einen höheren Wert hat als von Spieler 2
 - Erhöhe punkte_spieler_1 um den Wert der Preiskarte
- Wenn die Karte von Spieler 2 einen höheren Wert hat als von Spieler 1
 - Erhöhe punkte_spieler_2 um den Wert der Preiskarte

Wiederhole obige Abfolge bis keine Karten mehr da sind.

Wie funktioniert das?

```
if punkte_spieler_1 > punkte_spieler_2:  
    print("Spieler 1 hat gewonnen!")  
elif punkte_spieler_1 < punkte_spieler_2:  
    print("Spieler 2 hat gewonnen!")  
else:  
    print('Dieses Spiel war unentschieden!')
```

while (boolscher_ausdruck):

*wiederhole diesen Code immer wieder, bis boolscher_ausdruck = False ist
Es wird alles ausgeführt was eingerückt ist!*

Wichtig: Es muss sicher gestellt werden, dass **boolscher_ausdruck** irgendwann **False** ist!

Beispiel

```
kriterium_umweiterzumachen = True
while (kriterium_umweiterzumachen):
    print('Hier machen wir irgendetwas!')

    antwort = input('Haben wir genug gemacht? ')

    if (antwort == 'ja') or (antwort == 'Ja'):
        kriterium_umweiterzumachen = False
```

```
Hier machen wir irgendetwas!

Haben wir genug gemacht? nein
Hier machen wir irgendetwas!

Haben wir genug gemacht? nein
Hier machen wir irgendetwas!

Haben wir genug gemacht? ja

In [112]:
```

Zurück zu unserem Spiel

```
punkte_spieler_1 = 0
punkte_spieler_2 = 0
```

```
sind_noch_karten_uebrig = True
while (sind_noch_karten_uebrig):
```

- Ziehe eine zufällige Preiskarte
- Spielen einer Karte von Spieler 1 und Spielen einer Karte von Spieler 2
- Wenn die Karte von Spieler 1 einen höheren Wert hat als von Spieler 2
 - Erhöhe punkte_spieler_1 um den Wert der Preiskarte
- Wenn die Karte von Spieler 2 einen höheren Wert hat als von Spieler 1
 - Erhöhe punkte_spieler_2 um den Wert der Preiskarte

```
## Aktualisiere sind_noch_karten_uebrig
```

```
if punkte_spieler_1 > punkte_spieler_2:
    print("Spieler 1 hat gewonnen!")
elif punkte_spieler_1 < punkte_spieler_2:
    print("Spieler 2 hat gewonnen!")
else:
    print('Dieses Spiel war unentschieden!')
```

Zurück zu unserem Spiel

```
punkte_spieler_1 = 0
punkte_spieler_2 = 0
```

```
sind_noch_karten_uebrig = True
while (sind_noch_karten_uebrig):
```

- Ziehe eine zufällige Preiskarte
- Spielen einer Karte von Spieler 1 und Spielen einer Karte von Spieler 2
- Wenn die Karte von Spieler 1 einen höheren Wert hat als von Spieler 2
 - Erhöhe punkte_spieler_1 um den Wert der Preiskarte
- Wenn die Karte von Spieler 2 einen höheren Wert hat als von Spieler 1
 - Erhöhe punkte_spieler_2 um den Wert der Preiskarte

```
## Aktualisiere sind_noch_karten_uebrig
```

```
if punkte_spieler_1 > punkte_spieler_2:
    print("Spieler 1 hat gewonnen!")
elif punkte_spieler_1 < punkte_spieler_2:
    print("Spieler 2 hat gewonnen!")
else:
    print('Dieses Spiel war unentschieden!')
```

*Diesen Teil
können wir
auch schon!*

Zurück zu unserem Spiel

```
punkte_spieler_1 = 0
punkte_spieler_2 = 0
```

```
sind_noch_karten_uebrig = True
while (sind_noch_karten_uebrig):
```

- Ziehe eine zufällige Preiskarte
- Spielen einer Karte von Spieler 1 und Spielen einer Karte von Spieler 2

```
if wert_karte_1 > wert_karte_2:
    punkte_spieler_1 += wert_der_preiskarte
elif wert_karte_1 < wert_karte_2:
    punkte_spieler_2 += wert_der_preiskarte
```

```
## Aktualisiere sind_noch_karten_uebrig
```

```
if punkte_spieler_1 > punkte_spieler_2:
    print("Spieler 1 hat gewonnen!")
elif punkte_spieler_1 < punkte_spieler_2:
    print("Spieler 2 hat gewonnen!")
else:
    print('Dieses Spiel war unentschieden!')
```

Zurück zu unserem Spiel

```
punkte_spieler_1 = 0  
punkte_spieler_2 = 0
```

```
sind_noch_karten_uebrig = True  
while (sind_noch_karten_uebrig):
```

- Ziehe eine zufällige Preiskarte
- Spielen einer Karte von Spieler 1 und Spielen einer Karte von Spieler 2

Muss aus den verbleibenden Preiskarten eine zufällige auswählen und der Variable wert_der_preiskarte einen Wert zuordnen

```
if wert_karte_1 > wert_karte_2:  
    punkte_spieler_1 += wert_der_preiskarte  
elif wert_karte_1 < wert_karte_2:  
    punkte_spieler_2 += wert_der_preiskarte
```

```
## Aktualisiere sind_noch_karten_uebrig
```

```
if punkte_spieler_1 > punkte_spieler_2:  
    print("Spieler 1 hat gewonnen!")  
elif punkte_spieler_1 < punkte_spieler_2:  
    print("Spieler 2 hat gewonnen!")  
else:  
    print('Dieses Spiel war unentschieden!')
```

Ziehe eine zufällige Preiskarte



Muss aus den verbleibenden Preiskarten eine zufällige auswählen und der Variable wert_der_preiskarte einen Wert zuordnen

Wir brauchen einen neuen Datentyp – er sollte die eine **Sammlung von Werten** beschreiben können!

Eine solche Datenstruktur in Python sind **Listen**!

- Listen werden mit eckigen Klammer geschrieben.
- Sie können Elemente unterschiedlicher Datentypen enthalten.
- Sie sind veränderbar (mutabel)

```
meine_liste = [1,4,5,8]
meine_liste = ['ass', 'zwei', 'drei', 'vier', 'könig']
meine_liste = ['ass', 2, 3.0]
print(type(meine_liste))
print(meine_liste[0])
print(meine_liste[1])
meine_liste[1] = 'Pik Zwei'
print(meine_liste)
```

```
<class 'list'>
ass
2
['ass', 'Pik Zwei', 3.0]
```

Was können wir mit Listen alles machen?

Elemente/Listen mittels Indizes auswählen

```
meine_liste = ['ass', 'zwei', 'drei', 'vier', 'könig']  
print(meine_liste)  
print(meine_liste[-1])  
print(meine_liste[0:3])  
print(meine_liste[0:5:2])
```

```
['ass', 'zwei', 'drei', 'vier', 'könig']  
könig  
['ass', 'zwei', 'drei']  
['ass', 'drei', 'könig']
```

Einen Eintrag anhängen

```
meine_liste = ['ass', 'zwei', 'drei', 'vier', 'könig']  
meine_liste.append('dame')  
print(meine_liste)
```

```
['ass', 'zwei', 'drei', 'vier', 'könig', 'dame']
```

Einen Eintrag einfügen

```
meine_liste = ['ass', 'zwei', 'drei', 'vier', 'könig']  
meine_liste.insert(1, 'dame')  
print(meine_liste)
```

```
['ass', 'dame', 'zwei', 'drei', 'vier', 'könig']
```

Einen Eintrag löschen

```
meine_liste = ['ass', 'zwei', 'drei', 'vier', 'könig']  
meine_liste.remove('drei')  
print(meine_liste)
```

```
['ass', 'zwei', 'vier', 'könig']
```

Was können wir mit Listen alles machen?

Ein Element an einem bestimmten Index zurückgeben und dies löschen

```
meine_liste = ['ass', 'zwei', 'drei', 'vier', 'könig']  
print(meine_liste.pop(2))  
print(meine_liste)
```

```
drei  
['ass', 'zwei', 'vier', 'könig']
```

`.pop()` ohne index nimmt immer das letzte Element der Liste!

Was wir herausfinden wollten:

Ziehe eine zufällige Preiskarte



Muss aus den verbleibenden Preiskarten eine zufällige auswählen und der Variable wert_der_preiskarte einen Wert zuordnen

Einfachste Realisierung: Mische die Preiskarten am Anfang, dann ziehe immer die oberste Karte mit `.pop()`!

Import Befehl

Wie kann man eine Liste mischen?

Wir können ein Python *Modul* für Zufallsoperationen *verfügbar machen*

Das Verfügbarmachen von installierten Modulen funktioniert über den Befehl **import**

Mit **help("modules")** kann man alle verfügbaren Module anzeigen.

Die meisten Module, die man gerne benutzen möchte aber noch nicht installiert sind, kann man im miniconda Terminal mittels **conda install modulname** herunterladen/installieren.

Wir benötigen hier erstmal nur das Modul **random**, aber die Vielzahl an frei verfügbaren Modulen (und ganzen Frameworks für Fortgeschrittene Anwendungen wie das maschinelle Lernen) machen Python äußerst beliebt!

Module definieren außerdem einen *Namensraum (namespace)* – alle Funktionen einem Modul werden über Modulname.funktionsname angesprochen.

```
preiskarten = [1,2,3,4,5,6,7,8,9,10]
print(preiskarten)
import random
random.shuffle(preiskarten)
print(preiskarten)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[5, 4, 9, 6, 1, 7, 8, 10, 3, 2]
```

Für unser Problem:



Zurück zu unserem Spiel

```
import random
punkte_spieler_1 = 0
punkte_spieler_2 = 0
preiskarten = [1,2,3,4,5,6,7,8,9,10]
random.shuffle(preiskarten)
sind_noch_karten_uebrig = True

while (sind_noch_karten_uebrig):
    wert_der_preiskarte = preiskarten.pop()
```

- Spielen einer Karte von Spieler 1 und Spielen einer Karte von Spieler 2

```
if wert_karte_1 > wert_karte_2:
    punkte_spieler_1 += wert_der_preiskarte
elif wert_karte_1 < wert_karte_2:
    punkte_spieler_2 += wert_der_preiskarte
```

```
sind_noch_karten_uebrig = len(preiskarten)>0

if punkte_spieler_1 > punkte_spieler_2:
    print("Spieler 1 hat gewonnen!")
elif punkte_spieler_1 < punkte_spieler_2:
    print("Spieler 2 hat gewonnen!")
else:
    print('Dieses Spiel war unentschieden!')
```

*Schöne Python-Vereinfachung:
while (Liste):
führt diesen Code aus,
solange Liste nicht leer ist*

Zurück zu unserem Spiel

```
import random
punkte_spieler_1 = 0
punkte_spieler_2 = 0
preiskarten = [1,2,3,4,5,6,7,8,9,10]
random.shuffle(preiskarten)

while (preiskarten):
    wert_der_preiskarte = preiskarten.pop()
```

- Spielen einer Karte von Spieler 1 und Spielen einer Karte von Spieler 2

```
if wert_karte_1 > wert_karte_2:
    punkte_spieler_1 += wert_der_preiskarte
elif wert_karte_1 < wert_karte_2:
    punkte_spieler_2 += wert_der_preiskarte
```

```
if punkte_spieler_1 > punkte_spieler_2:
    print("Spieler 1 hat gewonnen!")
elif punkte_spieler_1 < punkte_spieler_2:
    print("Spieler 2 hat gewonnen!")
else:
    print('Dieses Spiel war unentschieden!')
```

Um die Eingabe des Spiels zu vereinfachen, werden wir nicht zwei Menschen sondern zwei Computerstrategien gegeneinander antreten lassen:

- *Der eine Spieler spielt immer eine zufällige Karte,*
- *der andere immer die gleiche Karte wie die Preiskarte.*

Das fertige Spiel – mit Logs

```
2 import random
3 punkte_spieler_1 = 0
4 punkte_spieler_2 = 0
5 preiskarten = [1,2,3,4,5]
6 karten_spieler_1 = [1,2,3,4,5]
7 karten_spieler_2 = [1,2,3,4,5]
8
9 random.shuffle(preiskarten)
10 random.shuffle(karten_spieler_1)
11
12 while (preiskarten):
13     wert_der_preiskarte = preiskarten.pop()
14     print('\nDer Wert der Preiskarte ist ', wert_der_preiskarte)
15     wert_karte_1 = karten_spieler_1.pop()
16     print('Spieler 1 spielt eine ', wert_karte_1)
17     wert_karte_2 = wert_der_preiskarte
18     karten_spieler_2.remove(wert_karte_2)
19     print('Spieler 2 spielt eine ', wert_karte_2)
20
21     if wert_karte_1 > wert_karte_2:
22         punkte_spieler_1 += wert_der_preiskarte
23         print('Spieler 1 gewinnt die Preiskarte. Er hat nun ', punkte_spieler_1, ' Punkte.')
24     elif wert_karte_1 < wert_karte_2:
25         punkte_spieler_2 += wert_der_preiskarte
26         print('Spieler 2 gewinnt die Preiskarte. Er hat nun ', punkte_spieler_2, ' Punkte.')
27     else:
28         print('Gleichstand - keiner bekommt die Preiskarte!')
29
30 print('\n\nFinaler Spielstand:')
31 print('Spieler 1 hat ', punkte_spieler_1, 'Punkte')
32 print('Spieler 2 hat ', punkte_spieler_2, 'Punkte \n')
33
34 if punkte_spieler_1 > punkte_spieler_2:
35     print("Spieler 1 hat gewonnen!")
36 elif punkte_spieler_1 < punkte_spieler_2:
37     print("Spieler 2 hat gewonnen!")
38 else:
39     print('Dieses Spiel war unentschieden!')
```

Help Variable Explorer Plots Files

Console 1/A X

Der Wert der Preiskarte ist 5
Spieler 1 spielt eine 3
Spieler 2 spielt eine 5
Spieler 2 gewinnt die Preiskarte. Er hat nun 5 Punkte.

Der Wert der Preiskarte ist 4
Spieler 1 spielt eine 4
Spieler 2 spielt eine 4
Gleichstand - keiner bekommt die Preiskarte!

Der Wert der Preiskarte ist 1
Spieler 1 spielt eine 2
Spieler 2 spielt eine 1
Spieler 1 gewinnt die Preiskarte. Er hat nun 1 Punkte.

Der Wert der Preiskarte ist 3
Spieler 1 spielt eine 5
Spieler 2 spielt eine 3
Spieler 1 gewinnt die Preiskarte. Er hat nun 4 Punkte.

Der Wert der Preiskarte ist 2
Spieler 1 spielt eine 1
Spieler 2 spielt eine 2
Spieler 2 gewinnt die Preiskarte. Er hat nun 7 Punkte.

Finaler Spielstand:
Spieler 1 hat 4 Punkte
Spieler 2 hat 7 Punkte

Spieler 2 hat gewonnen!

In [154]:

Die Wiederholung von Spielzügen haben wir mittels einer **while**-Schleife umgesetzt. Die zweite gängige Art dies zu tun sind sogenannte **for**-Schleifen.

Die anderen Programmiersprachen ähnlichste Art eine **for**-Schleifen zu verwenden, ist ein Index über eine **range**

```
for i in range(5):  
    print(i)
```

```
0  
1  
2  
3  
4
```

Im Allgemeinen hat **range** einen Startpunkt, einen Endpunkt, und eine Schrittweite

```
for i in range(1,10,3):  
    print(i)
```

```
1  
4  
7
```

Python können **for**-Schleifen aber auch noch allgemeiner, nämlich für beliebige **iterierbare Objekte** verwendet werden. Unter den uns bekannten Objekten, sind dies z.B. Listen:

```
meine_liste = ['hallo', 4, [1,2,3]]  
for element in meine_liste:  
    print(element)
```

```
hallo  
4  
[1, 2, 3]
```

Mir **enumerate** kann zusätzlich auf den Index zugegriffen werden:

```
meine_liste = ['hallo', 4, [1,2,3]]  
for index, element in enumerate(meine_liste):  
    print('An index {} steht {}'.format(index, element))
```

```
An index 0 steht hallo  
An index 1 steht 4  
An index 2 steht [1, 2, 3]
```

Was passiert als nächstes?

Wir besprechen mögliche Fallstrike und verschönern den Code!

Insbesondere isolieren wir bestimmte Teile in *Funktionen*!

Mögliche Übungen:

- Ändern sie das Programm so, dass es eine **for**-Schleife statt einer **while**-Schleife verwendet.
- Ändern sie das Programm so, dass es ein Spiel von ihnen gegen den Computer wird. Geben sie die noch verfügbaren Karten in der Hand aus (mittels **print**) und nutzen sie die Funktion **input** um eine Karte auswählen zu lassen. Überprüfen sie, ob diese auch wirklich in der Hand existiert, z.B. indem sie in einer **while**-Schleife solange nach einen input fragen, bis eine Karte aus der Hand ausgewählt wurde!
- Gestalten sie die ‚künstliche Intelligenz‘ ihres Computergegners interessanter: Er könnte z.B. mit 70% Wahrscheinlichkeit die verbleibende Karte verwenden dessen Wertigkeit der Preiskarte am nächsten ist und mit 30% Wahrscheinlichkeit eine völlig zufällige Karte nehmen. Hierfür werden sie zusätzliche Befehle nachschauen müssen.

Vergleich if, elif, else

Vergleichsoperatoren == != > < >= <=

Verknüpfung/Ändern boolescher Werte and or not

While-Schleifen while

Listen erzeugen mit [..., ..., ...] und Einträge ändern

Manipulation von Listen append insert remove

Element in einer Liste zurückgeben und löschen pop

Liste aller verfügbaren Module anzeigen help("Modules")

Modul importieren import

Modul random

Zufälliges Permutieren einer Liste random.shuffle

a+=b als Kurzform für a=a+b

For-Schleifen über iterierbare Objekte for in

Zahlen von a bis b in Schritten c range(a, b, c)

Auflistung mit Indices enumerate