Manipulierter Würfel

Dickbauer Y., Moser P., Perner M.

PS Computergestützte Modellierung, WS 2016/17

December 15, 2016

Outline

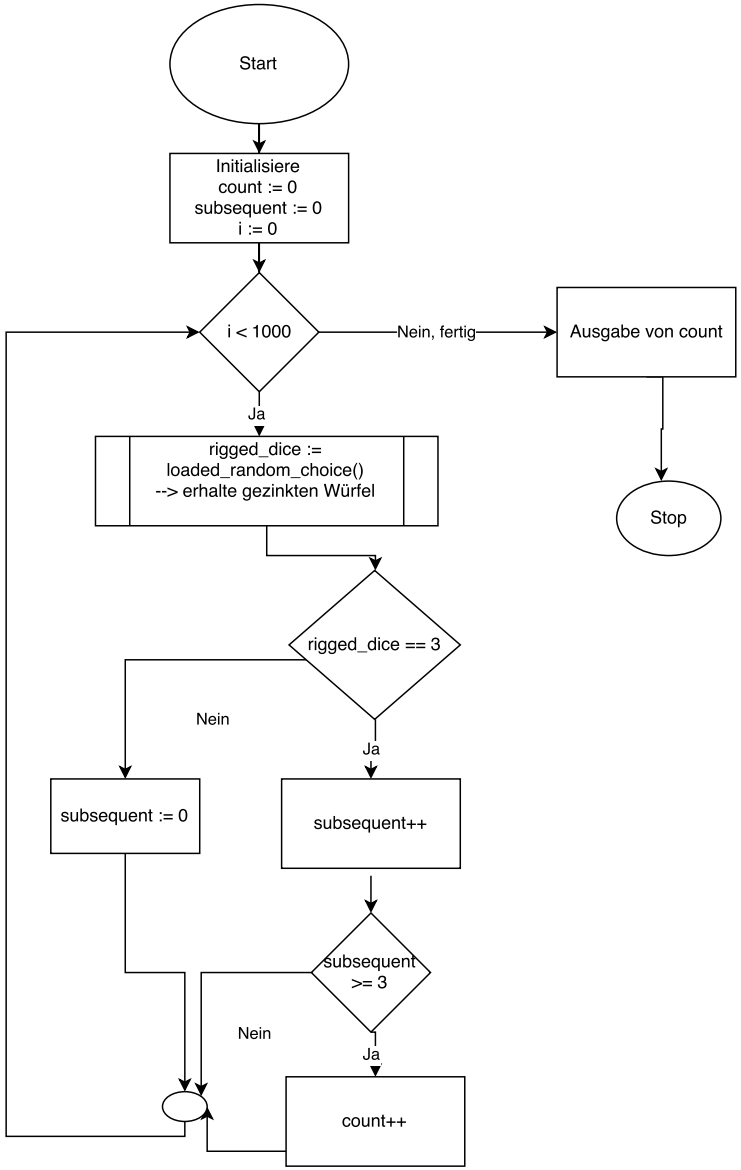
Aufgabenstellung

Ein manipulierter Würfel soll geworfen werden. 1000x würfeln. Es gelten folgende Wahrscheinlichkeiten

Augenzahl	1	2	3	4	5	6
Wahrscheinlichkeit	1/10	1/20	1/5	1/10	1/2	1/20

- Eingabe: -
- Output: Wie oft kommt es vor, dass 3x hintereinander die Zahl "3" gewürfelt wird (3333 = zweimal 3333).

Flow Chart


Start
Initialisiere
count := 0
subsequent := 0
i := 0
i < 1000
Nein, fertig
Ausgabe von count
Stop
Ja
rigged_dice :=
loaded_random_choice()
--> erhalte gezinkten Würfel
rigged_dice == 3
Nein
Ja
subsequent := 0
subsequent++
subsequent
>= 3
Nein
Ja
count++

Main Funktion - Programmeinstieg

```
1   RIGGED_DICE_PROBS = (1/10, 1/20, 1/5, 1/10, 1/2, 1/20)
2   NUMBER_OF_THROWS = 1000
3
4   def main():
5       count = 0 #result
6       subsequent = 0 #how often did we see it at the actual position
7       for i in range(NUMBER_OF_THROWS):
8           rigged_dice = loaded_random_choice(RIGGED_DICE_PROBS) + 1
9           if rigged_dice == CHECK_DICE:
10              # we've got one more
11              subsequent += 1
12              if subsequent >= AMOUNT_OF_TANDEMS:
13                  count += 1
14          else:
15              # that's the wrong dice -> set actual amount of subsequents back to z
16              subsequent = 0
17
18      print('Anzahl␣an␣{}␣mal␣hintereinander␣eine␣{}:␣{}'.format(
19          AMOUNT_OF_TANDEMS, CHECK_DICE, count))
```

Funktion loaded_random_choice(..)

- Diese Funktion verlangt eine WSKL Liste als Eingabeparameter
- Gibt einen Index zurück, welcher 0 bis $|probality\_list| - 1$ sein kann.
- Diese Indizes haben eine gewichtete WSKL, welche jeweils an der Position in der Eingabeliste steht
- Beispiel probility_list := [0.9, 0.1] $\Rightarrow$ mit p=90% wird 0 zurückgegeben, p=10% für 1

```
1  def loaded_random_choice(probability_list):
2      n = len(probability_list)
3      random_number = random.random()
4      cum_p = 0
5      for i in range(n):
6          cum_p += probability_list[i]
7          if cum_p > random_number:
8              return i
9      return None
```

Beispiel anhand fixer Zufallszahlen

- Annahme der Zufallszahlen wie folgt:

iteration	0	1	2	3	4	5-999
ZZ	0.05	0.21	0.20	0.22	0.09	0.09
rigged_dice	1	3	3	3	1	1

i := 0
- rigged_dice != 3 $\Rightarrow$ subsequent = 0, count = 0

i := 1
- rigged_dice == 3 $\Rightarrow$ subsequent = 1, count = 0

i := 2
- rigged_dice == 3 $\Rightarrow$ subsequent = 2, count = 0

Beispiel anhand fixer Zufallszahlen

i := 3

- rigged_dice == 3 $\Rightarrow$ subsequent = 3 $\Rightarrow$ count = 1

i := 4

- rigged_dice != 3 $\Rightarrow$ subsequent = 0, count = 1

Nach 1000 Iteration ist count = 1, also genau 1x 333 hintereinander

Anhang: Modifikation des Source Codes um Demo Beispiel zu erhalten

```
1     # Fuege folgenden Code vor random_number_from_interval() in lib.py ein:
2   ZZ = [0.05, 0.21, 0.20, 0.22] + [0.01]*1000
3   i = -1
4   def my_rand():
5       global i
6       i += 1
7       return ZZ[i]
8   random.random = my_rand
```