Kapitel 3: Starta programmering på riktigt

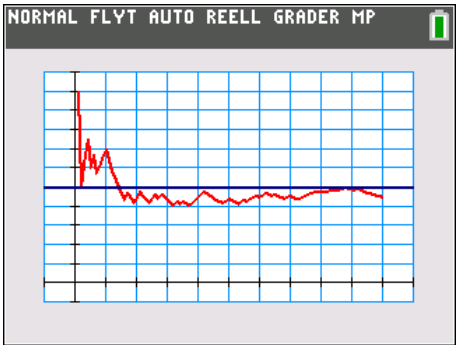
Övning 1: Funktioner, villkor och loopar

I den tredje övningen i kapitel 3 ska vi göra flera simuleringar som illustrerar *Stora talens lag* i ett praktiskt sammanhang. Vi behöver då en slumpgenerator, därför måste biblioteket för slumptal importeras.

Syfte:

- Använda en funktion
- Implementera den avgränsade loopen For i flera exempel
- Använda villkorssatser

I statistik och sannolikhetsteori är *Stora talens lag* en sats som beskriver resultatet av att upprepa samma experiment ett stort antal gånger. Denna lag anger att om samma experiment, som ska vara oberoende av varandra, upprepas ett stort antal gånger måste genomsnittet av resultaten av försöken ligga nära det förväntade värdet. Resultatet kommer också närmare det förväntade värdet i takt med att antalet försök ökar.



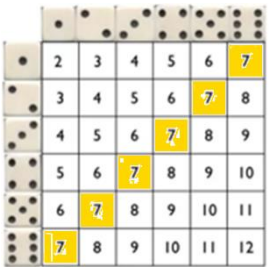
Figuren ovan visar att vid slantsingling så stabiliseras den relativa frekvensen för krona (eller klave) efter många kast vid ett värde omkring 0,5, vilket är vad man kan förvänta sig.

Exempel: Ett enkelt exempel på lagen är att kasta tärningar. Ett kast innebär sex olika händelser med lika stora sannolikheter. Det förväntade värdet (förkortas ofta EV för *Exepected value*) av tärningshändelsen är:

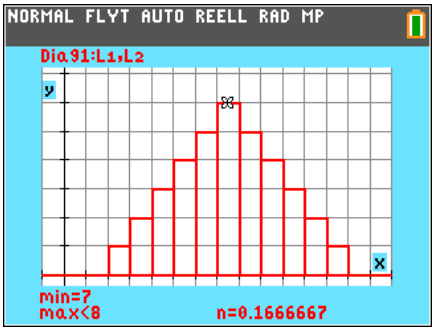
$$\frac{1+2+3+4+5+6}{6}=3{,}5$$

Om vi bara kastar en tärning några få gånger kan genomsnittet av de erhållna resultaten vara långt ifrån det förväntade värdet. Låt oss säga att du kastade en tärning tre gånger och resultatet blev 3, 6, 6. Genomsnittet av resultaten blir då 5. Enligt lagen om stora tal, så blir det genomsnittliga värdet om vi kastar tärningarna ett stort antal gånger att vara närmare det förväntade värdet på 3,5.

I denna övning ska vi göra en del experiment med fyra tärningar och titta närmare på sannolikheten experimentellt att få en viss totalsumma.



Kastar du en tärning så är ju sannolikheten 1/6 för varje utfall 1 till och med 6. Kastar du två tärningar så ser utfallen för summan ut som i diagrammet till höger. Man ser direkt att summan 7 kan man få på 6 sätt: 1+6, 2+5, 3+4, 4+3, 5+2, 6+1. Sannolikheten blir alltså 6/36. Man kan också räkna ut maxsumman som 2 gånger 3,5.



För summan 6 och 8 blir sannolikheten 5/36, för summan 5 och 9 blir den 4/36 osv. Se diagrammet i marginalen som visar de olika sannolikheterna.

Nog med grundläggande teori om sannolikheter. Nu ska vi se hur vi kan skriva skript som simulera kast med 4 tärningar.

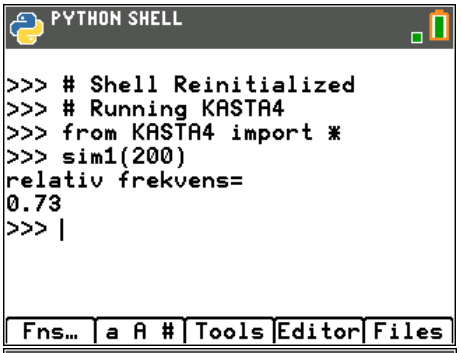
Simulering

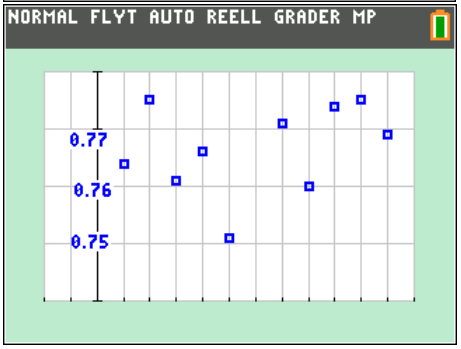
Vi ska nu skriva ett program som utför ett antal simuleringar av kast med fyra tärningar. Vi ska sedan beräkna summan av prickarna vid varje kast och titta på när summan är mindre eller lika med 18. Om summan är <=18 så ökar värdet på ett räkneverk med 1. Satsen räkneverk=räkneverk+1 ökar värdet.

- Då vi ska arbeta med slumptal importerar vi först biblioteksmodulen **random import**. Tryck f1 (Fns…) och välj sedan i menyn Modul och sedan 2:random.
- Vi definierar sedan funktionen sim1(n) där n är antalet upprepningar av försöket. Tryck f1 (Fns…) och välj func och sedan 1 def function().
- Sedan skriver vi in startvärdet 0 för vårt räkneverk.
- Sedan kommer for-satsen som vi använder för att (upprepa en del av programmet n antal gånger. Vi skriver alltså här sim1(100) om vi vill köra 100 gånger.
- Nu kommer if-satsen där vi undersöker om totala antalet prickar är ≤ 16. Om det är sant så räknas räkneverket upp.
- Till sist beräknas kvoten räkneverk/n som är andelen kast där total-summan blir ≤ 16.

För att köra programmet så trycker du Run och på tangenten [vars] och väljer sedan sim1() och skriver in värdet på *n*, t.ex. 200. Sedan trycker du på [enter]. Kör några gånger med samma värde och se hur det beräknade värdet varierar. Tryck på [▲] för att snabbt göra många körningar.

Du kan göra körningen ett stort antal gånger, t. ex. 2000. Kör några gånger. Diagrammet visar de värden vi fick när vi gjorde detta 10 gånger. Vi får värden mellan 0,751 och 0,775 ca.





Lärarkommentar: Diskutera med eleverna skillnaden mellan att skriva

randint(1,6)+randint(1,6)+randint(1,6)+randint(1,6)<=16

som vi gjorde i skriptet och

randint(1,6)*4<=16 eller **randint(4,24)**

Be dem att skriva in enligt ovan och sedan tolka resultatet.

Exakt beräkning

När vi gjorde ett antal simuleringar fick vi ett medelvärde på ca 0,76. Nu ska vi skriva ett ganska tufft program som gör exakta beräkningar. Det åstadkommer vi genom fyra for-satser (en för varje tärningskast). Det som händer är att programmet går igenom alla kombinationer och tittar vilka som ger 16 prickar eller mindre. Om man skulle göra detta utan någon hjälp av räknare eller dator skulle det ta väldigt lång tid.

```
EDITOR: KASTA4
PROGRAM LINE 0021
# rv=räkneverk
def beräkna():
  rv=0
  for i in range(1,7):
    for j in range(1,7):
      for k in range(1,7):
        for l in range(1,7):
          if i+j+k+l<=16:
            rv=rv+1
  return rv
Fns… | a A # | Tools | Run | Files
```

```
PYTHON SHELL
>>> # Shell Reinitialized
>>> # Running KASTA4
>>> from KASTA4 import *
>>> beräkna()
986
>>>
Fns… | a A # | Tools | Editor | Files
```

Vi kör nu programmet en gång och ser vilket resultat vi får. Sammanlagt finns det 986 kombinationer som ger en totalsumma som är 16 eller mindre. Totalt finns det $6^4 = 1296$ kombinationer. Detta ger att sannolikheten att få 16 eller mindre är

$$\frac{986}{1296} \approx 0{,}7608..$$

Cirka tre av fyra gånger så får du alltså i genomsnitt en poängsumma som är 16 eller mindre.

Tabellen nedan visar sannolikheterna för att få totalsumman 4-24. Lägg märke till symmetrin för antalet kombinationer omkring 14 i tabellen. Pröva nu också andra totalsummor och se att det stämmer med tabellen.

	A total	B komb	C kum_komb	D sannolik…	E kum_sann
1	4	1	1	0.000772	0.000772
2	5	4	5	0.003086	0.003858
3	6	10	15	0.007716	0.011574
4	7	20	35	0.015432	0.027006
5	8	35	70	0.027006	0.054012
6	9	56	126	0.04321	0.097222
7	10	80	206	0.061728	0.158951
8	11	104	310	0.080247	0.239198
9	12	125	435	0.096451	0.335648
10	13	140	575	0.108025	0.443673
11	14	146	721	0.112654	0.556327
12	15	140	861	0.108025	0.664352
13	16	125	986	0.096451	0.760802
14	17	104	1090	0.080247	0.841049
15	18	80	1170	0.061728	0.902778
16	19	56	1226	0.04321	0.945988
17	20	35	1261	0.027006	0.972994
18	21	20	1281	0.015432	0.988426
19	22	10	1291	0.007716	0.996142
20	23	4	1295	0.003086	0.999228
21	24	1	1296	0.000772	1.

C21 =c20+b21

Fördjupning:

Om vi plottar totalsumma mot sannolikheten för 4 respektive 6 kast så ser det ut så här. Vi ser tydligt att fördelningen alltmer närmar sig en *normalfördelning* när antalet tärningar ökar.