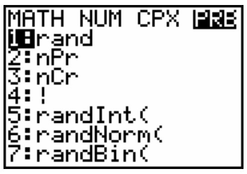
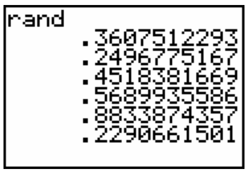
# 7 Simulation

## 7.1 Random numbers

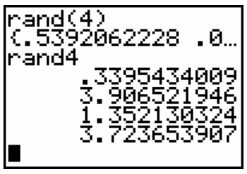
The command **rand** generates the following random numbers:

| | | |
|---|---|---|
| **rand** | → | an arbitrary number *x* between 0 and 1 ( $0 < x < 1$ ) |
| **rand(4)** | → | a list of 4 arbitrary numbers between 0 en 1 |
| **rand4** | → | an arbitrary number *x* between 0 en 4 ( $0 < x < 4$ ) |
| **A+(B-A)rand** | → | an arbitrary number *x* between *A* en *B* ( $A < x < B$ ) |

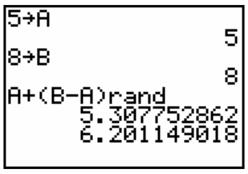




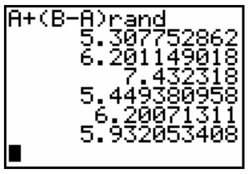
```
rand
        .3607512293
        .2496775167
        .4518381669
        .5689935586
        .8833874357
        .2290661501
```



```
rand(4)
{.5392062228 .0…
rand4
        .3395434009
        3.906521946
        1.352130324
        3.723653907
```



```
5→A
                  5
8→B
                  8
A+(B-A)rand
        5.307752862
        6.201149018
```

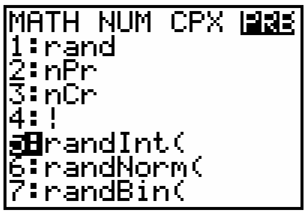


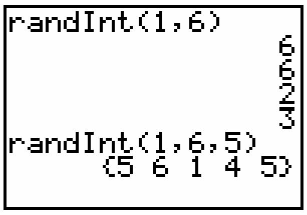
```
A+(B-A)rand
        5.307752862
        6.201149018
           7.432318
        5.449380958
         6.20071311
        5.932053408
```

The command **rand** starts generating at random numbers from a seed. The standard value of the seed is 0. Starting from a specific seed you will always get the same sequence of random numbers. So if you to generate at random numbers, by several students at the same time, it can be useful to let each student first choose an arbitrary value for the seed **rand**: e.g. **144→rand**.

**randInt** generates at random integers as follows:

| | | |
|---|---|---|
| **randInt(1,6)** | → | an arbitrary integer between 1 and 6 (1 and 6 included) |
| **randInt(1,6,5)** | → | a list of 5 arbitrary integers between 1 and 6 |





```
randInt(1,6)
              6
              6
              2
              3
randInt(1,6,5)
    {5 6 1 4 5}
```

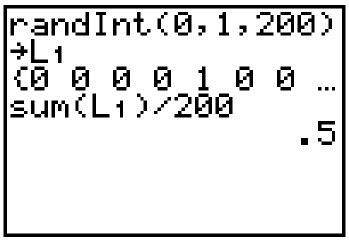
## 7.2 Coin tosses

To simulate the tossing of a coin we encode heads by **1** and tails by **0**.

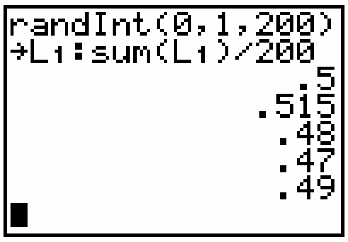
We simulate two hundred coin tosses and we will save the results in list **L₁** as follows:

**randInt(0,1,200)→L₁**.

We can count the number of heads by the **sum** command. Dividing this result by 200 gives us the relative frequency of the event heads.



```
randInt(0,1,200)
→L₁
{0 0 0 0 1 0 0 …
sum(L₁)/200
             .5
```

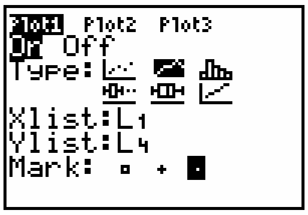


```
randInt(0,1,200)
→L₁:sum(L₁)/200
             .5
           .515
            .48
            .47
            .49
```

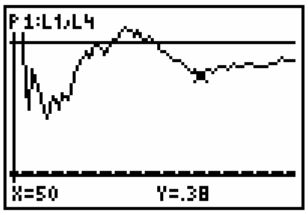
With the following list we will create a visualization of the previous simulation:

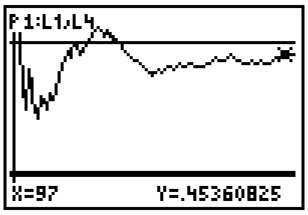
```
L1 = seq(X,X,1,200)
L2 = randInt(0,1,200)
L3 = cumSum(L2)
L4 = L3/L1
Y1  = 1/2
```

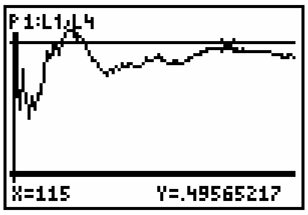
The following screens show that the relative frequency of the event heads in the long run approximate ½, the theoretical probability of the event heads.



```
Plot1 Plot2 Plot3
On Off
Type:
Xlist:L1
Ylist:L4
Mark: ▫ + ·
```





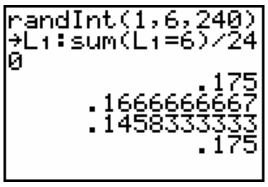


Notice that more tosses don’t automatically cause a better approximation. If we continue tossing the relative frequency will get as close to ½ as we want. But no one can tell us how many tosses we have to do.

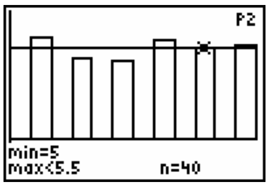
## 7.3 Throwing dice

We will simulate 240 throws of a dice by **randInt(1,6,240)→L1** and calculate the relative frequency of the events six spots as follows: **sum(L1=6)/240**.

The following histogram is a visualisation of Bernoulli’s law of large numbers.



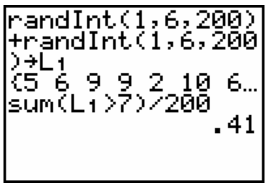
```
randInt(1,6,240)
→L1:sum(L1=6)/24
0
                .175
       .1666666667
       .1458333333
                .175
```



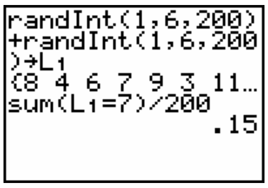
**randInt(1,6)+randInt(1,6)** simulates the throwing of two dice.

By repeatedly executing this command you can approximate the probability of the events *the sum of the point is less than, equal to or more than seven*.

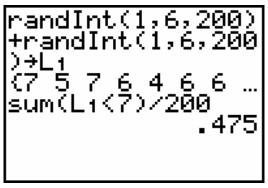
| Dice 1 | Dice 2 | Dice 1 | Dice 2 | Dice 1 | Dice 2 | Dice 1 | Dice 2 | Dice 1 | Dice 2 | Dice 1 | Dice 2 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 1 | 3 | 1 | 4 | 1 | 5 | 1 | 6 | 1 |
| 1 | 2 | 2 | 2 | 3 | 2 | 4 | 2 | 5 | 2 | 6 | 2 |
| 1 | 3 | 2 | 3 | 3 | 3 | 4 | 3 | 5 | 3 | 6 | 3 |
| 1 | 4 | 2 | 4 | 3 | 4 | 4 | 4 | 5 | 4 | 6 | 4 |
| 1 | 5 | 2 | 5 | 3 | 5 | 4 | 5 | 5 | 5 | 6 | 5 |
| 1 | 6 | 2 | 6 | 3 | 6 | 4 | 6 | 5 | 6 | 6 | 6 |



```
randInt(1,6,200)
+randInt(1,6,200
)→L1
{5 6 9 9 2 10 6…
sum(L1>7)/200
                 .41
```



```
randInt(1,6,200)
+randInt(1,6,200
)→L1
{8 4 6 7 9 3 11…
sum(L1=7)/200
                 .15
```



```
randInt(1,6,200)
+randInt(1,6,200
)→L1
{7 5 7 6 4 6 6 …
sum(L1<7)/200
                .475
```