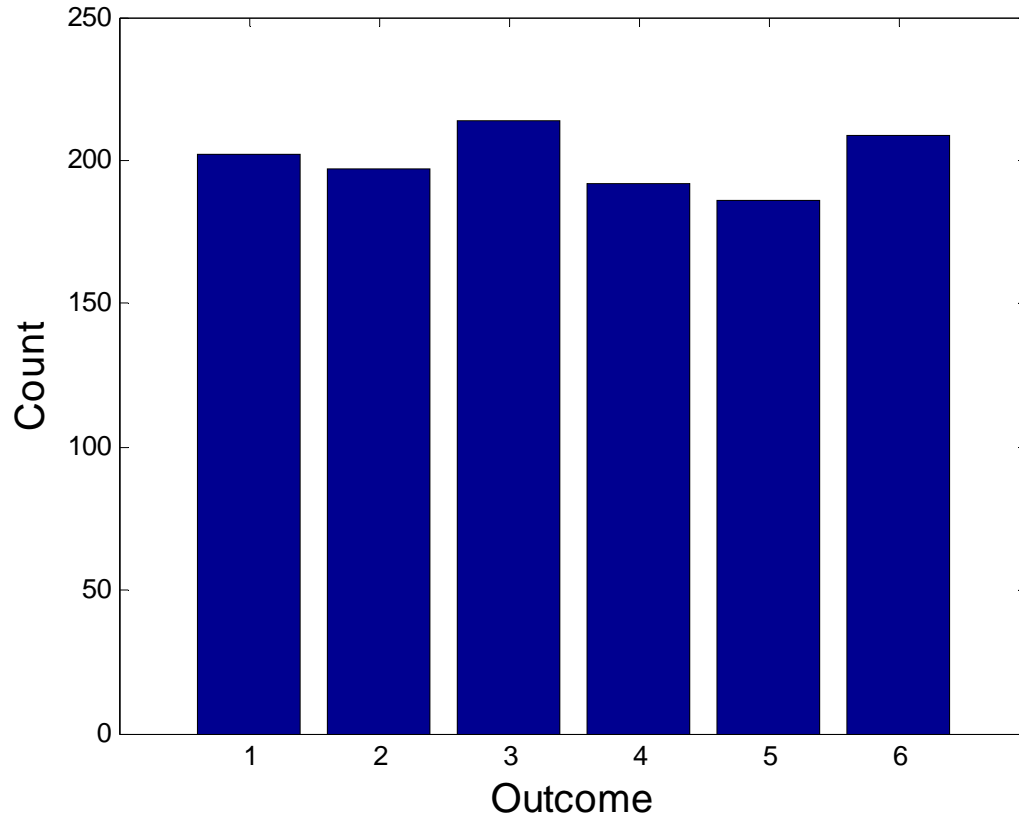


- Previous Lecture:
 - Executing a user-defined function
 - Function scope

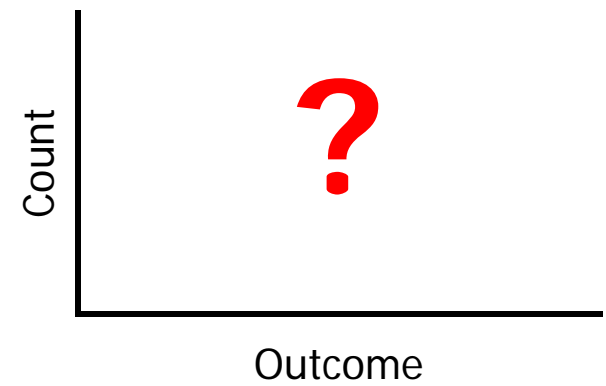
- Today's Lecture:
 - Subfunction
 - 1-d array—vector
 - Probability and random numbers
 - Simulation using random numbers, vectors

- Announcement:
 - Project 3 due Monday 10/5 at 11pm

Outcomes from 1200 rolls of a fair die

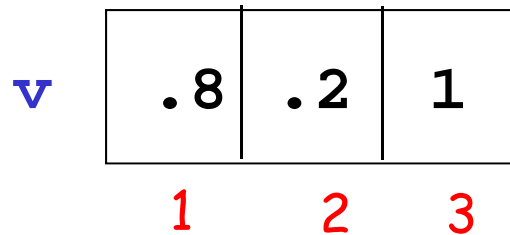


Roll 2 fair dice



1-d array: **vector**

- An array is a **named** collection of **like** data organized into rows or columns
- A 1-d array is a row or a column, called a **vector**
- An **index** identifies the **position** of a value in a vector



Here are a few different ways to create a vector

```
count= zeros(1,6)
```

count

0	0	0	0	0	0
---	---	---	---	---	---

Similar functions: `ones`, `rand`

```
a= linspace(10,30,5)
```

a

10	15	20	25	30
----	----	----	----	----

```
b= 7:-2:0
```

b

7	5	3	1
---	---	---	---

```
c= [3 7 2 1]
```

c

3	7	2	1
---	---	---	---

```
d= [3; 7; 2]
```

d

3
7
2

```
e= d'
```

e

3	7	2
---	---	---

Start with drawing a single line segment

```
a= 0; % x-coord of pt 1
```

```
b= 1; % y-coord of pt 1
```

```
c= 5; % x-coord of pt 2
```

```
d= 3; % y-coord of pt 2
```

```
plot([a c], [b d], '-*')
```

x-values
(a vector)

y-values
(a vector)

Line/marker
format

Making an x-y plot

```
a= [0 4 3 8]; % x-coords
```

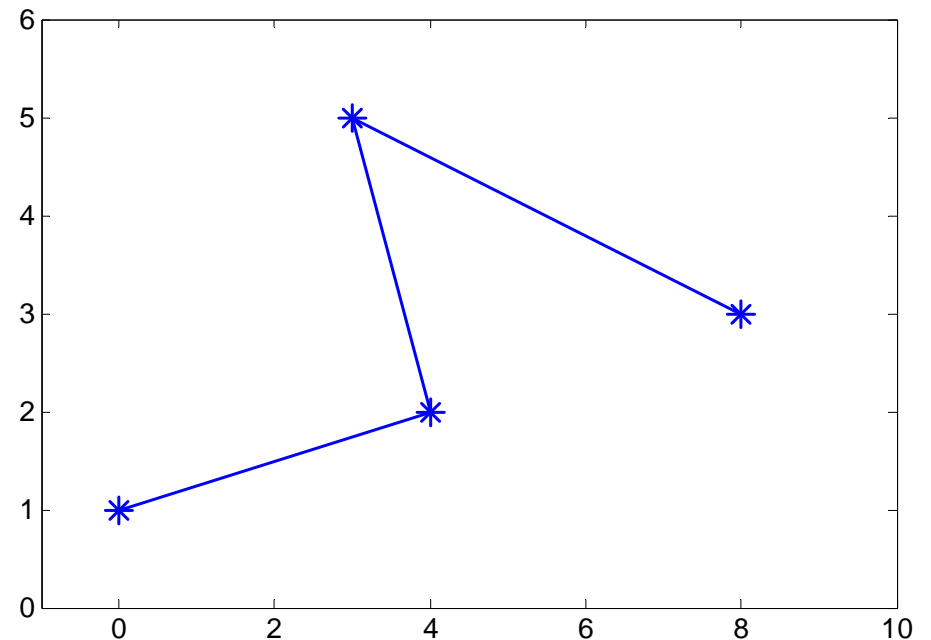
```
b= [1 2 5 3]; % y-coords
```

```
plot(a, b, '-*')
```

x-values
(a vector)

y-values
(a vector)

Line/marker
format



Making an x-y plot with multiple graphs (lines)

```
a= [0 4 5 8];
```

```
b= [1 2 5 3];
```

```
f= [0 4 6 8 10];
```

```
g= [2 2 6 4 3];
```

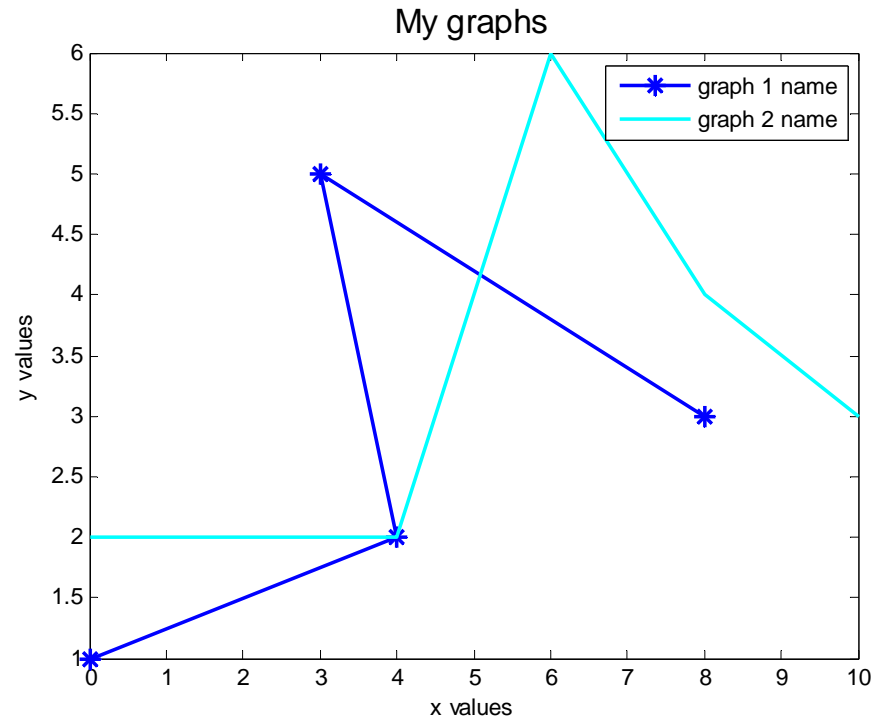
```
plot(a,b,'-*',f,g,'c')
```

```
legend('graph 1 name', 'graph 2 name')
```

```
xlabel('x values')
```

```
ylabel('y values')
```

```
title('My graphs', 'FontSize',14)
```



Array index starts at 1

x	5	.4	.91	-4	-1	7
	1	2	3	4	5	6

Let k be the index of vector x , then

- k must be a positive integer
- $1 \leq k \leq \text{length}(x)$
- To access the k^{th} element: $x(k)$

Accessing values in a vector

score	93	92	87	0	90	82
	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>

Given the vector **score** ...

Accessing values in a vector

score	93	99	87	80	85	82
	1	2	3	4	5	6

Given the vector **score** ...

```
score(4) = 80;
```

```
score(5) = (score(4) + score(5)) / 2;
```

```
k = 1;
```

```
score(k+1) = 99;
```

See `plotComparison2.m`

Example

- Write a program fragment that calculates the **cumulative sums** of a given vector \mathbf{v} .
- The cumulative sums should be stored in a vector of the same length as \mathbf{v} .

1, 3, 5, 0 \mathbf{v}

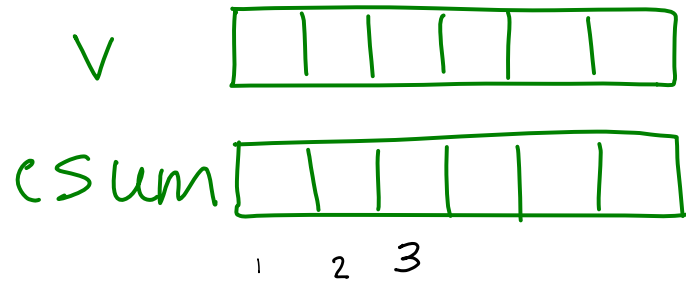
1, 4, 9, 9 cumulative sums of \mathbf{v}

v

--	--	--	--	--	--

esum

--	--	--	--	--	--



$$csum(k) = csum(k-1) + v(k)$$

$$csum(3) = v(1) + v(2) + v(3)$$

$$csum(4) = \underbrace{v(1) + v(2) + v(3)}_{csum(3)} + v(4)$$

$csum(3)$

$$csum(1) = v(1);$$

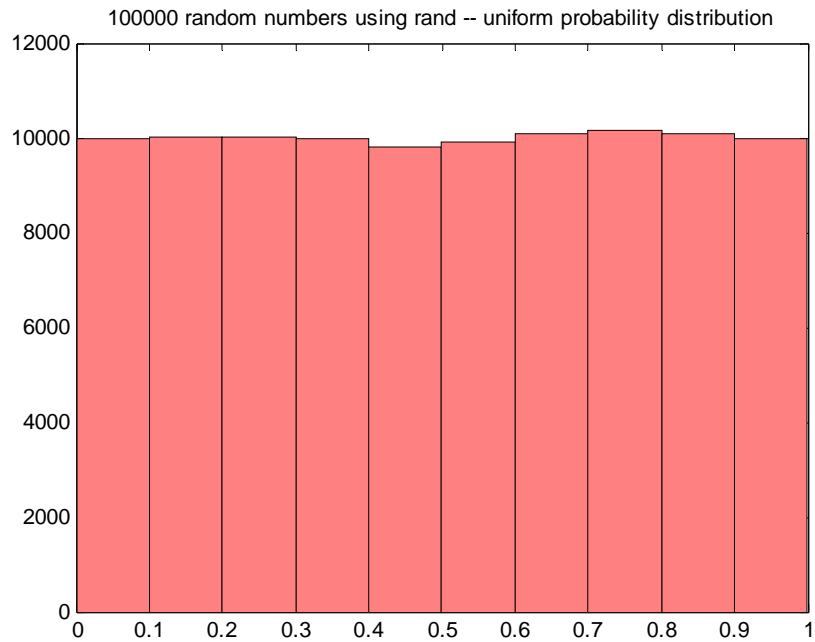
for $k = 2 : \text{length}(v)$

$$csum(k) = csum(k-1) + v(k);$$

end

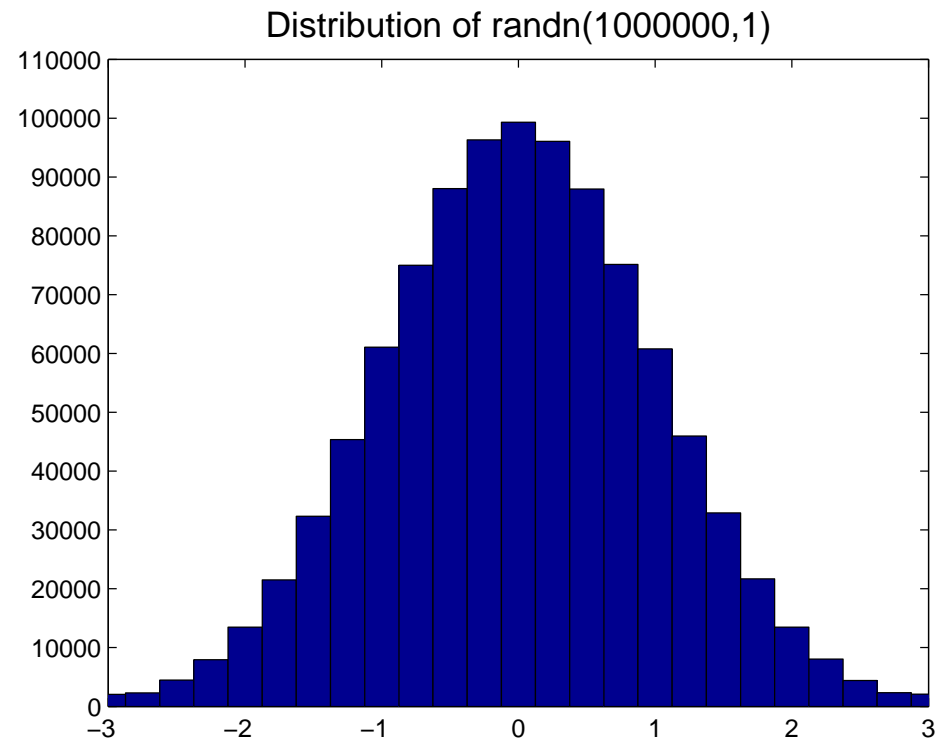
Random numbers

- *Pseudorandom* numbers in programming
- Function `rand(...)` generates random real numbers in the interval (0,1). All numbers in the interval (0,1) are equally likely to occur—**uniform** probability distribution.
- Examples:
 - `rand(1)` one random # in (0,1)
 - `6*rand(1)` one random # in (0,6)
 - `6*rand(1)+1` one random # in (1,7)



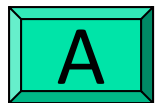
Uniform probability
distribution in $(0,1)$
rand

Normal distribution with
zero mean and unit
standard deviation
randn

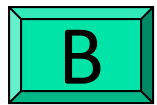


Simulate a fair 6-sided die

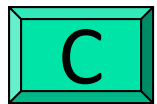
Which expression(s) below will give a random *integer* in [1..6] with equal likelihood?



`round(rand*6)`

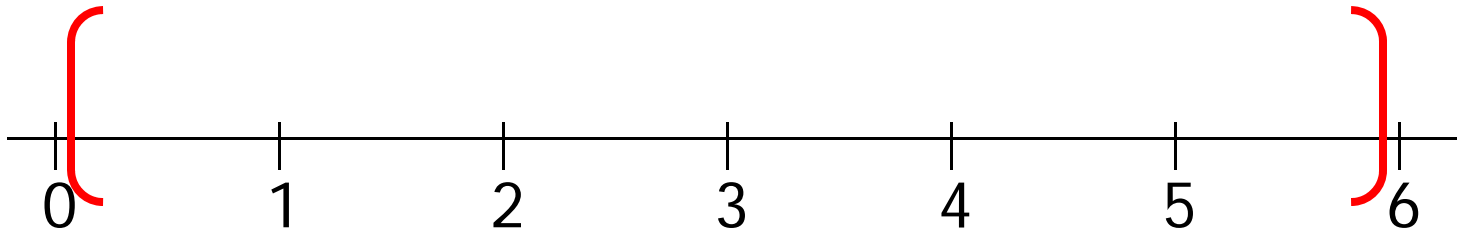


`ceil(rand*6)`

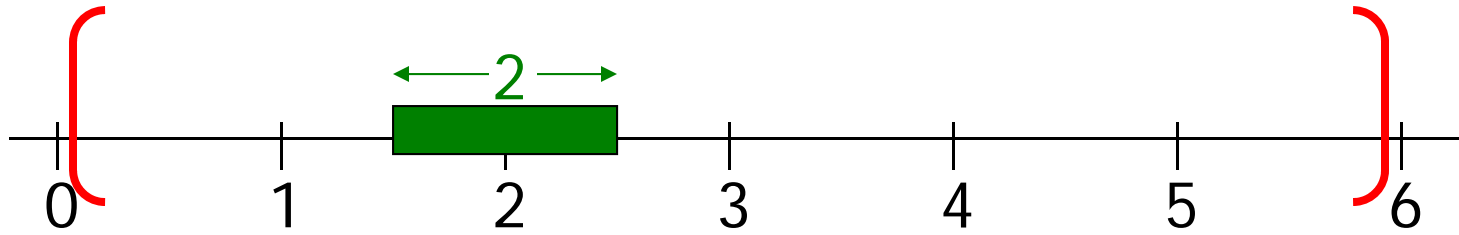


Both expressions above

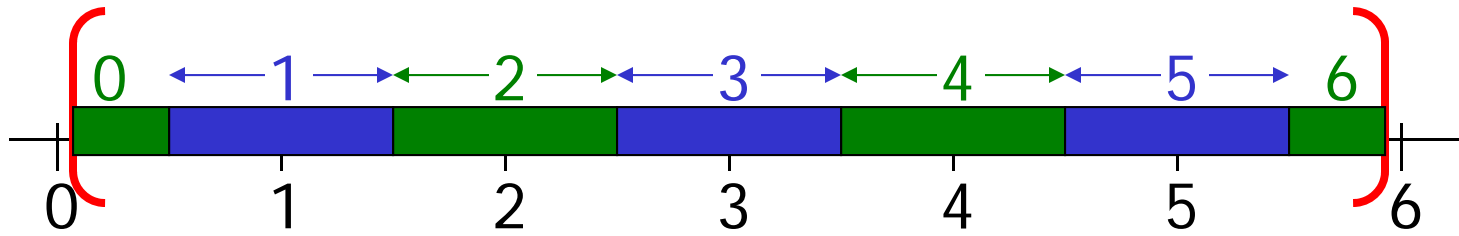
`(rand*6)`



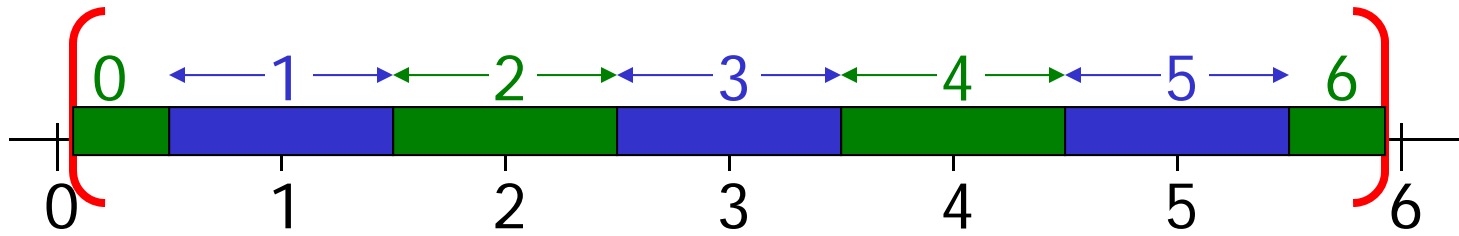
`round(rand*6)`



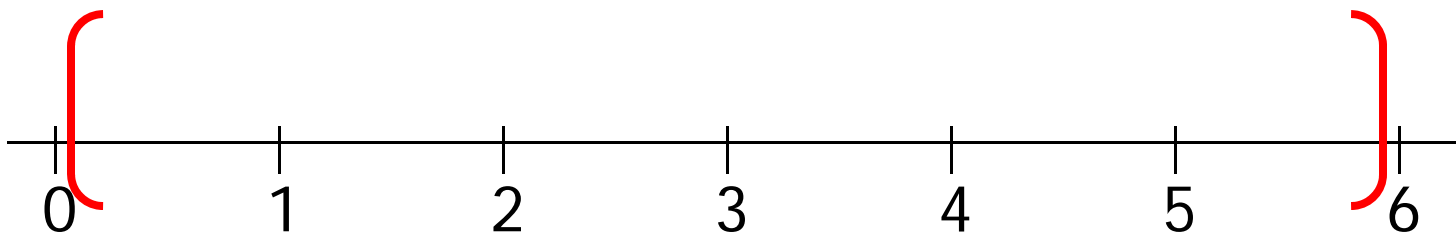
`round(rand*6)`



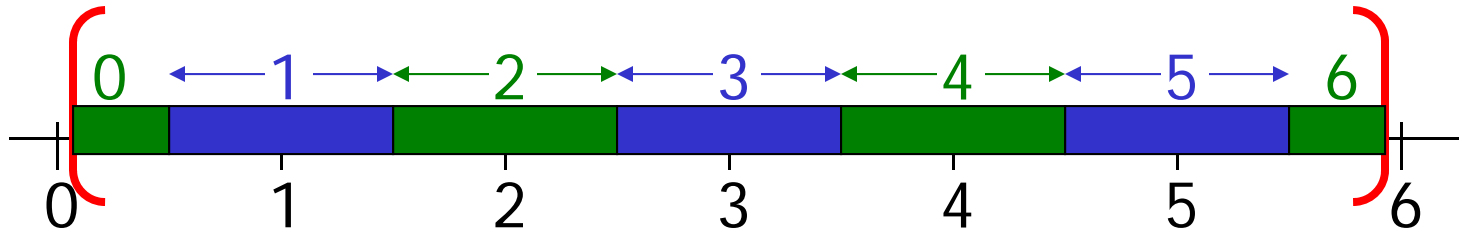
`round(rand*6)`



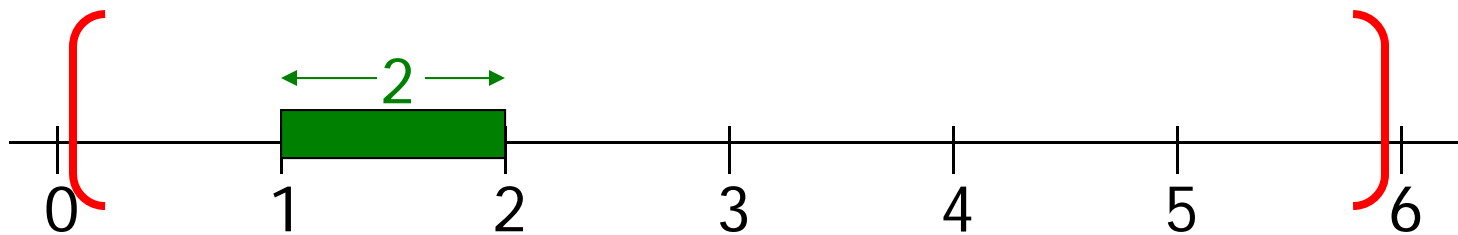
`(rand*6)`



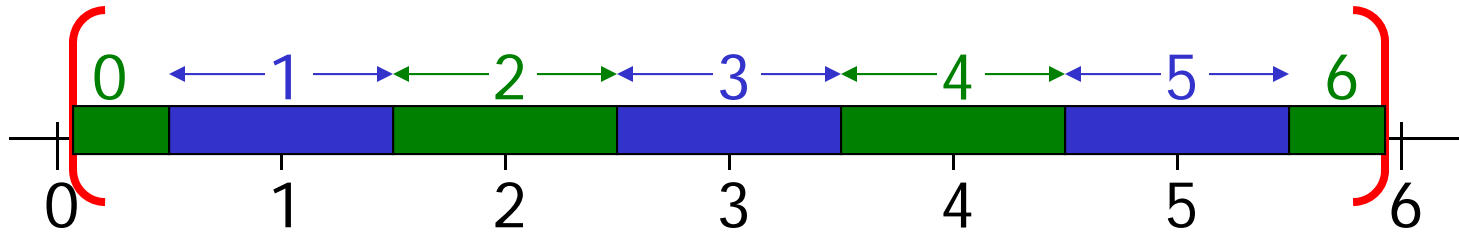
`round(rand*6)`



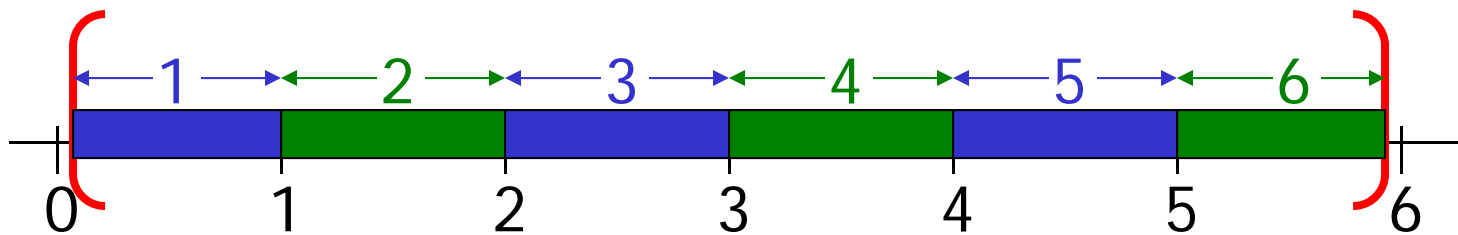
`ceil(rand*6)`



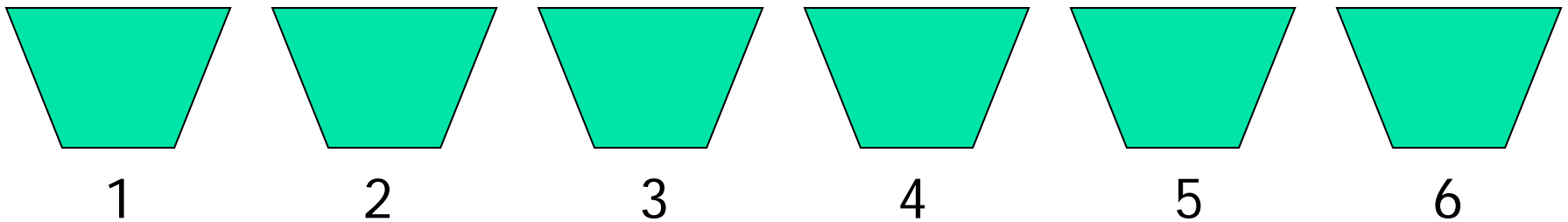
`round(rand*6)`



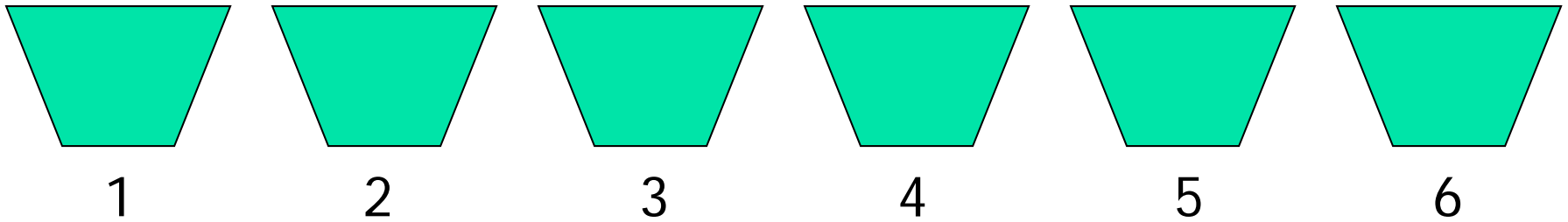
`ceil(rand*6)`



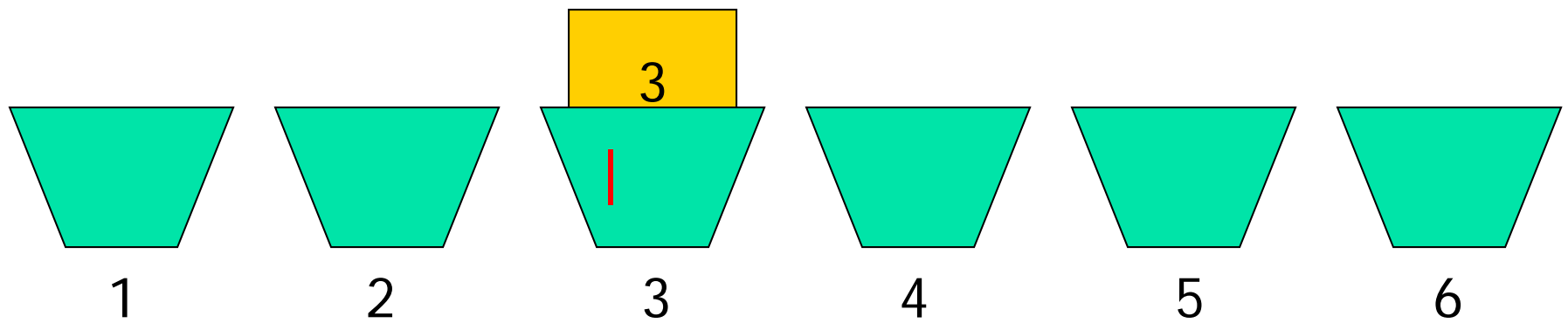
Possible outcomes from rolling a fair 6-sided die



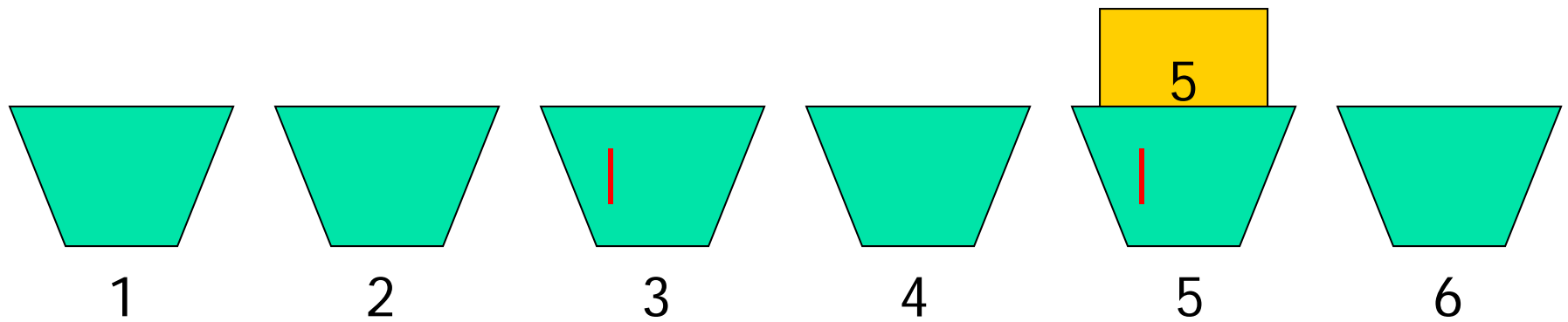
Simulation



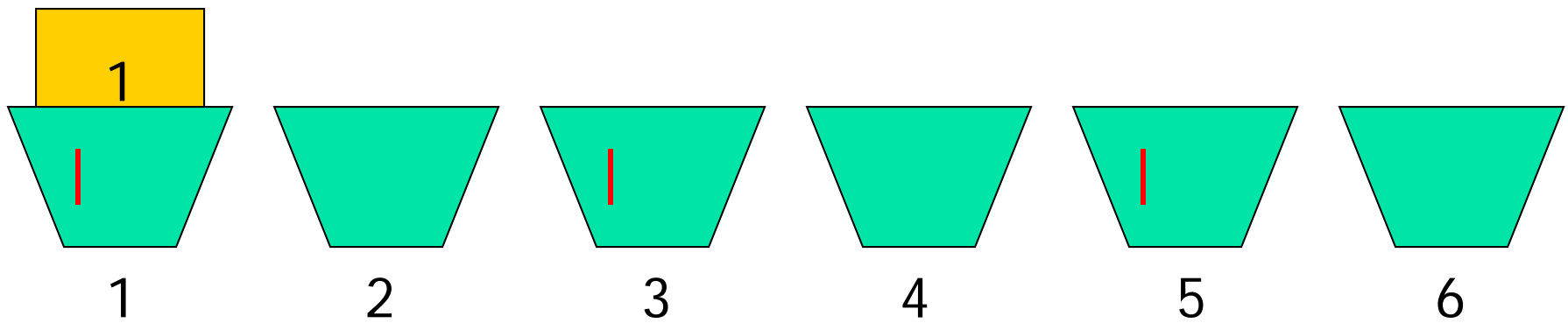
Simulation



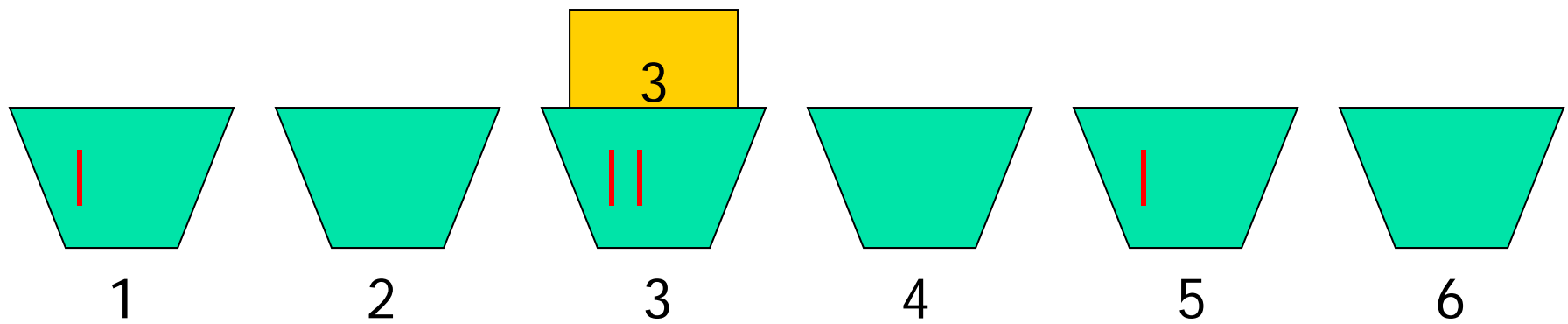
Simulation



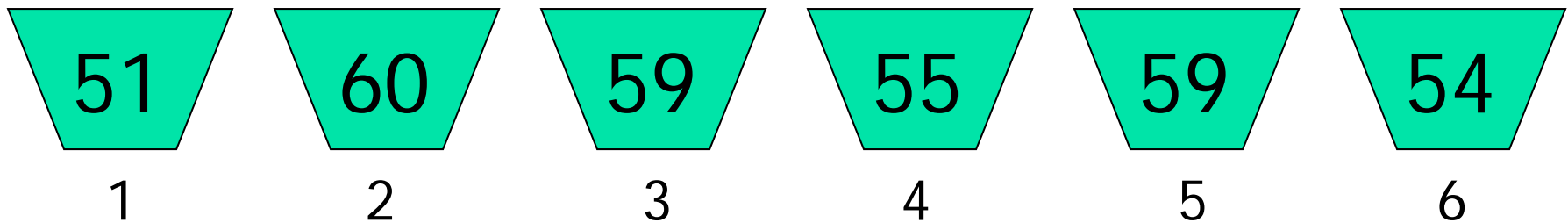
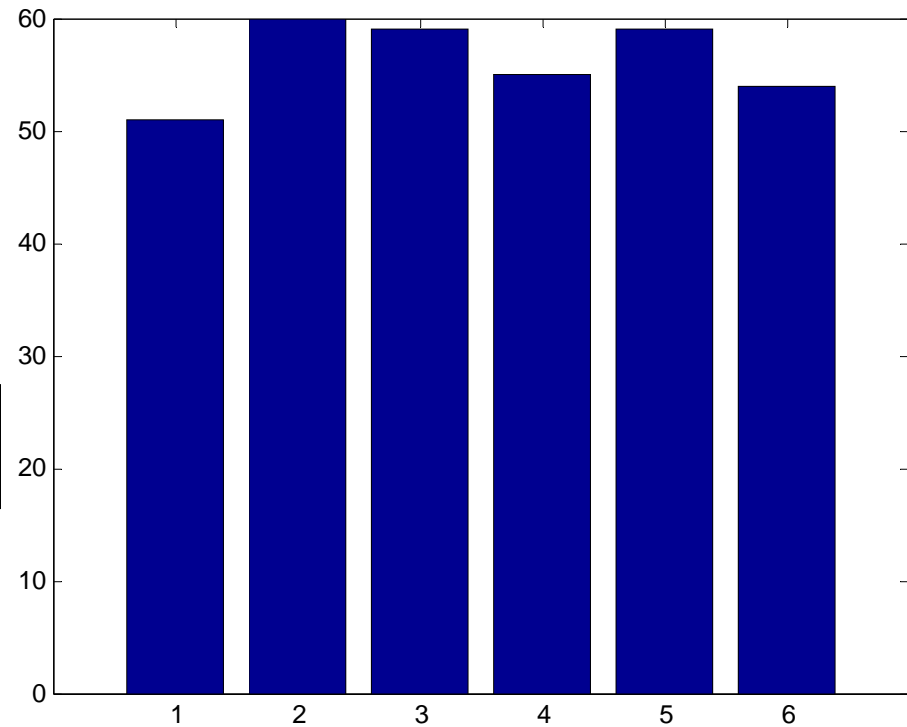
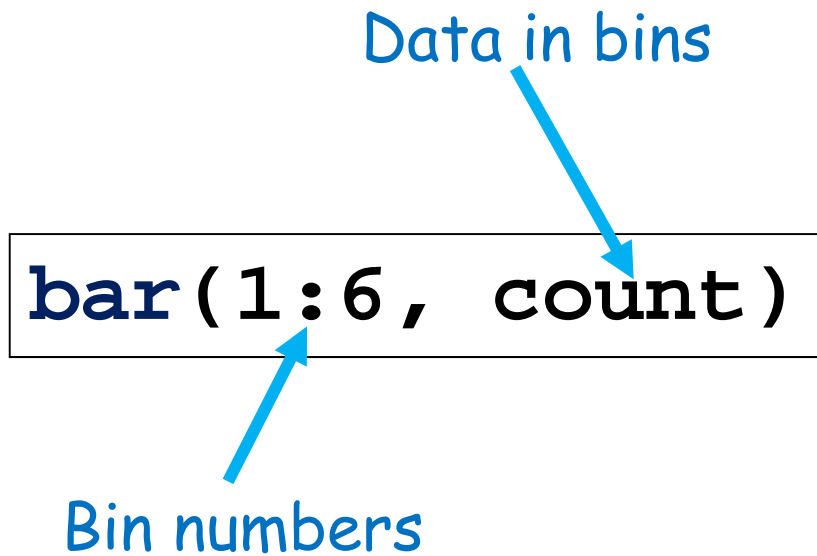
Simulation



Simulation



Simulation result



Keep tally on repeated rolls of a fair die

Repeat the following:

`% roll the die`

`% increment correct "bin"`