In [161]:

```
import numpy as np
import matplotlib.pyplot as plt
import itertools
from mpl_toolkits.mplot3d import Axes3D
%matplotlib inline
```

1. Задание (теорема сложения)

Найти вероятность выпадения 2 или 5 очков при подбрасывании игральной кости, на гранях которой имеются соответственно 1,2,3,4,5 и 6 очков.

$$P(2) = \frac{1}{6}$$
$$P(5) = \frac{1}{6}$$
$$P(2) + P(5) = \frac{2}{6} = \frac{1}{3}$$

In [26]:

```
a = np.random.randint(1, 7, 100000)
len(a[(a == 2) | (a == 5)]) / len(a)
```

Out[26]:

```
0.33473
```

2. Задание (теорема умножения)

Найти вероятность того, что при двух подбрасываниях той же самой игральной кости сначала выпадет 2, а затем 5.

$$P(2) = \frac{1}{6}$$
$$P(5) = \frac{1}{6}$$
$$P(2) \cdot P(5) = \frac{1}{36}$$

In [71]:

```
b = np.random.randint(1, 7, 100000)
# Будет ошибка в случае, если 2ка последняя. Можно сделать проверку на последний э
# но для эксперимента хватит и так
sum(b[np.where(b == 2)[0] + 1] == 5) / len(b)
```

Out[71]:

```
0.02716
```

3. Задание

Найти вероятность выпадения 2 и 5 очков при двух подбрасываниях той же самой игральной игральной кости.

В данном случае количество положительных исходов увеличилось вдвое, так как нас устраивают варианты 5 после 2 и 2 после 5

$$P(2) \cdot P(5) + P(5) \cdot P(2) = \frac{1}{36} + \frac{1}{36} = \frac{1}{18}$$

In [130]:

```
c = np.random.randint(1, 7, 10000)
# Будет ошибка в случае, если 2ка/5ка последняя. Можно сделать проверку на последн
# но для эксперимента хватит и так
x = sum(c[np.where(c == 2)[0] + 1] == 5) / len(c)
y = sum(c[np.where(c == 5)[0] + 1] == 2) / len(c)
print(x,'+', y)
```

```
0.027 + 0.0259
```

4. Задание (Геометрическая вероятность +интервалы)

На отрезке AB длиной 20 см наугад отметили точку C. Какова вероятность, что она находится на расстоянии не более 9 см от точки A и не более 15 см от точки B?

Примем за данное, что попадание в любой участок отрезка равновероятное событие. Тогда, длина отрезка, который отстоит от точки A на 9 ед и от точки B на 15 ед, составит 4 ед. А значит, вероятность попадания в него будет

$$\frac{4}{20} = \frac{1}{5}$$

In [153]:

```
plt.figure(figsize=(10,1))
plt.plot([0,9], [1,1], '|-', color='blue')
plt.plot([20, 5], [1,1], '|-', color='green')
plt.plot([5,9], [1,1], '|-', color='red')


plt.axis('off')
plt.show()
```

5. Задание.

Телефонный номер состоит из 7 цифр. Какова вероятность, что это номер 8882227?

Предположим, что дана 10-чная система счисления. Тогда, количество цифр, возможных в каждом знаке номера будет равно 10. Так как предполагается, что это телефонный номер, то порядок следования цифр имеет значение. Отсюда,

$$A_n^k = \frac{n!}{(n-k)!}$$

$A_{10}^{7} = \frac{10!}{3!} = 10 \cdot 9 \cdot 8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 = 604800$

Значит вероятность будет составлять $\frac{1}{604800}$

In [160]:

```
l = []
for p in itertools.permutations("0123456789",7):
    l.append(''.join(str(x) for x in p))
len(l)
```

Out[160]:

```
604800
```

6. Задание.

Набирая номер телефона, абонент забыл 2 последние цифры, и, помня только то, что эти цифры различны и среди них нет нуля, стал набирать их наудачу. Сколько вариантов ему надо перебрать, чтобы наверняка найти нужный номер? Какова вероятность того, что он угадает номер с первого раза?

Формула для вычисления перестановок (порядок имеет значение) без повторений чисел:

$$A_n^k = \frac{n!}{(n-k)!}$$

В нашем случае количество чисел равно 9 (без 0), нужно найти все комбинации из 2х чисел

$A_{9}^{2} = \frac{9!}{7!} = 8 \cdot 9 = 72$

Вероятность найти с 1го раза - $\frac{1}{72}$

In [158]:

```
l = []
for p in itertools.permutations("123456789",2):
    l.append(''.join(str(x) for x in p))
print(l)
len(l)
```

```
['12', '13', '14', '15', '16', '17', '18', '19', '21', '23', '24', '2
5', '26', '27', '28', '29', '31', '32', '34', '35', '36', '37', '38',
'39', '41', '42', '43', '45', '46', '47', '48', '49', '51', '52', '5
3', '54', '56', '57', '58', '59', '61', '62', '63', '64', '65', '67',
'68', '69', '71', '72', '73', '74', '75', '76', '78', '79', '81', '8
2', '83', '84', '85', '86', '87', '89', '91', '92', '93', '94', '95',
'96', '97', '98']
```

Out[158]:

```
72
```

7. Задание** (необязательное)

Чёрный куб покрасили снаружи белой краской, затем разрезали на 27 одинаковых маленьких кубиков и как попало сложили из них большой куб. С какой вероятностью все грани этого куба будут белыми?

Для того, чтобы получилось 27 одинаковых кубиков нужно, чтобы каждая грань была разделена на 9 равных частей.

Разобьем получившиеся кубики по местам их расположения:

- 8 штук расположены на углах куба, 3 грани будут белыми, 3 черными
- 12 штук будут расположены между ними, на ребрах куба, 2 грани белые, 4 черные
- 6 штук расположены по центру каждой грани, 1 грань белая, 5 черных
- 1 кубик в центре большого куба, 6 черных граней

Сначала узнаем общее количество возможных комбинаций.

Очевидно, что количество мест для кубиков - 27, соответственно, при единственном возможном положении кубика, количество перестановок составило бы 27!

Но, каждый кубик може быть повернут 24-я различными способами, то есть у нас получается 27 раз по 24 варианта или 24^{27}

Итого, общее количество комбинаций - $27! \cdot 24^{27}$

Теперь посчитаем количество устраивающих нас комбинаций:

- Черный кубик должен находиться строго в центре, повернут может быть любым образом $\Rightarrow 1! \cdot 1 \cdot 24 = 24$
- Для кубиков с одной белой гранью возможно 6 размещений, каждый кубик может быть повернут 4-я способами $\Rightarrow 6! \cdot 4^6$
- Для кубиков с двумя белыми гранями возможно 12 размещений, каждый кубик может быть повернут 2-я способами $\Rightarrow 12! \cdot 2^{12}$
- Для кубиков с тремя белыми гранями возможно 8 размещений, каждый кубик может быть повернут 3-я способами $\Rightarrow 8! \cdot 3^8$

Следовательно, вероятность того, что все грани куба будут белыми

$$P = \frac{24 \cdot 6! \cdot 4^6 \cdot 12! \cdot 2^{12} \cdot 8! \cdot 3^8}{27! \cdot 24^{27}}$$

In [188]:

```
a = 24**27
b = np.math.factorial(27)
c = 24*np.math.factorial(6)*(4**6)*(np.math.factorial(12))*(2**12)*(np.math.factor
c / (a * b)
```

Out[188]:

```
1.8298051356415021e-37
```

In [182]:

```
x, y, z = np.indices((8, 8, 8))
cube2 = (x >= 5) & (y >= 5) & (z >= 5)
voxels = cube2
colors = np.empty(voxels.shape, dtype=object)
colors[cube2] = 'green'

fig = plt.figure(figsize = (5,5))
ax = fig.gca(projection='3d')
ax.voxels(voxels, facecolors=colors, edgecolor='k')
plt.axis('off')
plt.show()
```