1. Напишите код, моделирующий выпадение поля в рулетке (с учетом поля зеро). # my solution ###

In [ ]:
```
import numpy as np

for i in range(0, 10):
    a = input()
    x = np.random.uniform(0, 36, 1)
    if 1 <= x <= 18 and x != 0:
        print(int(x), 'Черное')
    elif 18 < x <= 36 and x != 0:
        print(int(x), 'Красное')
    else:
        print(int(x), 'Зеро')
```

2.1)Напишите код, проверяющий любую из теорем сложения или умножения вероятности на примере рулетки или подбрасывания монетки. # my solution ###

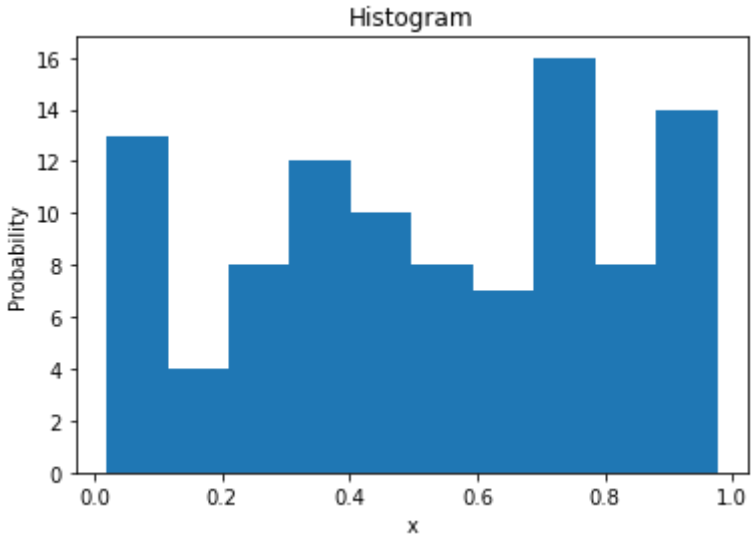
In [ ]:
```
import numpy as np
k, m, z = 0, 0, 0
n = 100

for i in range(0, n):
    a = input()
    x = np.random.uniform(0, 36, 1)
    if 1 <= x <= 18 and x != 0:
#         print(int(x), 'Черное')
        k = k + 1
    elif 18 < x <= 36 and x != 0:
#         print(int(x), 'Красное')
        m = m + 1
    else:
#         print(int(x), 'Зеро')
        z = z + 1
print(f"Сумма вероятности выпадения 'красное', 'черное', и 'зеро' = {k}/{n} ,
      f"{z / n} = {k / n + k / n + z / n}")
```

2.2)Сгенерируйте десять выборок случайных чисел x0, ..., x9 и постройте гистограмму распределения случайной суммы x0+x1+ ...+ x9 # my solution ###

In [58]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
x = np.random.rand(100)
num_bins = 10
n, bins, patches = plt.hist(x, num_bins)
plt.xlabel('x')
plt.ylabel('Probability')
plt.title('Histogram')
```

Out[58]:
```
Text(0.5, 1.0, 'Histogram')
```



3.1)Дополните код Монте-Карло последовательности независимых испытаний расчетом соответствующих вероятностей (через биномиальное распределение) и сравните результаты. # my solution ###

In [59]:
```
import numpy as np
import itertools
import math
k, n = 0, 10000
a = np.random.randint(0, 2, n)
b = np.random.randint(0, 2, n)
c = np.random.randint(0, 2, n)
d = np.random.randint(0, 2, n)
x = a + b + c + d
for i in range(0, n):
    if x[i] == 2:
        k = k + 1
#print(a, b, c, d)
#print(x)
print(k, n, k/n)
```

```
3741 10000 0.3741
```

In [ ]:
```
#Биномиальное распределение
```

In [60]:
```
import math

n = 10
k = 3
p = 0.5

c_n_k = math.factorial(n) / (math.factorial(k) * math.factorial(n - k))
probability = c_n_k * (p ** k) * ((1 - p) ** (n - k))

print(c_n_k, probability)
```

```
120.0 0.1171875
```

3.2) Повторите расчеты биномиальных коэффициентов и вероятностей k успехов в последовательности из n независимых испытаний, взяв другие значения n и k.

In [61]:
```
import math

n = 100
k = 35
p = 0.5

c_n_k = math.factorial(n) / (math.factorial(k) * math.factorial(n - k))
probability = c_n_k * (p ** k) * ((1 - p) ** (n - k))
```

```
print(c_n_k, probability)
```

```
1.095067153187963e+27 0.0008638556657416528
```

4. Из урока по комбинаторике повторите расчеты, сгенерировав возможные варианты перестановок для других значений n и k

In [62]:
```
for p in itertools.product("01",repeat=3):
  print(''.join(p))
```

```
000
001
010
011
100
101
110
111
```

In [65]:
```
for p in itertools.permutations("01234",2):
    print(''.join(str(x) for x in p))
```

```
01
02
03
04
10
12
13
14
20
21
23
24
30
31
32
34
40
41
42
43
```

In [ ]:
```
for p in itertools.combinations("012345",3):
    print(''.join(p))
```

In [ ]:
```
for p in itertools.product("012345",repeat=3):
    print(''.join(p))
```

5. Дополните код расчетом коэффициента корреляции x и y по формуле

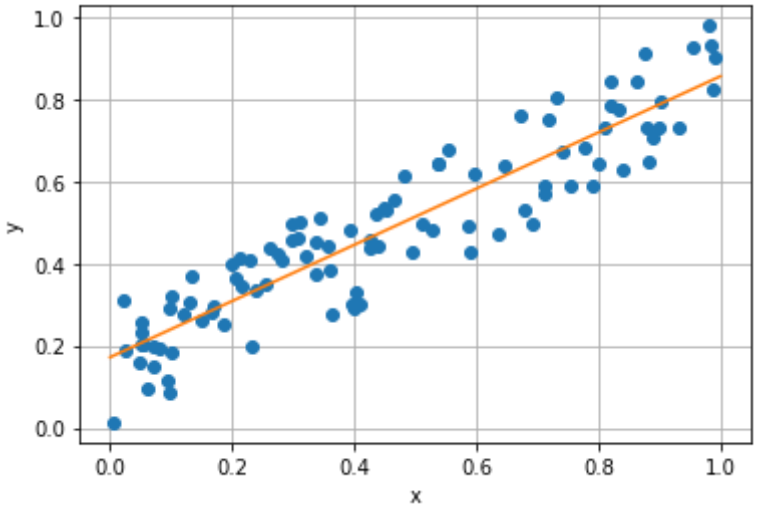
In [70]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
n = 100
r = 0.7
x = np.random.rand(n)
y = r*x + (1 - r)*np.random.rand(n)
plt.plot(x, y, 'o')
plt.xlabel('x')
plt.ylabel('y')
plt.grid(True)

a = (np.sum(x)*np.sum(y) - n*np.sum(x*y))/(np.sum(x)*np.sum(x) - n*np.sum(x*x
b = (np.sum(y) - a*np.sum(x))/n

# coff_corr = np.sum((x-(np.sum(x)/n))*(y-(np.sum(y)/n))/
```

```
#                     np.sqrt((np.sum((x-(np.sum(x)/n))**2))*(np.sum((y-(np.su
# эта конструкция ломает всю программу, не нашел почему, но ошибка возникает
A = np.vstack([x, np.ones(len(x))]).T
a1, b1 = np.linalg.lstsq(A, y)[0]
print(a, b)
print(a1, b1)
# print(coff_corr)
plt.plot([0, 1], [b, a + b])
plt.show()
```

```
<ipython-input-70-c844e0032397>:20: FutureWarning: `rcond` parameter will cha
nge to the default of machine precision times ``max(M, N)`` where M and N are
the input matrix dimensions.
To use the future default and silence this warning we advise to pass `rcond=N
one`, to keep using the old, explicitly pass `rcond=-1`.
  a1, b1 = np.linalg.lstsq(A, y)[0]
0.6831228600868746 0.17187509540888626
0.6831228600868746 0.17187509540888646
```



In [ ]: