```
import numpy as np
c1_tot = 0
c_rand_tot = 0
c_min_tot = 0
trials = 100000
coins = np.zeros((1000,10))

for k in range(trials):
    for i in range(1000):
        for j in range(10):
            result = np.random.rand(1)
            if (result[0] > 0.5): # heads
                coins[i][j] = 1
            else: # tails
                coins[i][j] = 0

    heads_per_coin = np.sum(coins, axis = 1)
    c1_tot += float(heads_per_coin[0]) / 10
    c_rand_tot += float(heads_per_coin[np.random.randint(1000)]) / 10
    c_min_tot += float(heads_per_coin[np.argmin(heads_per_coin)]) / 10

v1 = c1_tot / trials
v_rand = c_rand_tot / trials
v_min = c_min_tot / trials
print "v1 " + str(v1) + " v_rand " + str(v_rand) + " v_min " + str(v_min)
```

```
import numpy as np
import random
import matplotlib.pyplot as plt
runs = 1000
d = 2
def prob5():
    error = 0
    points = 100
    for q in range(runs):
        xs = np.random.uniform(-1,1, (points, d))
        xs = np.c_[np.ones(points), xs]
        target_points = np.random.uniform(-1,1, (2, d))
        ys = np.zeros(points)
        for i in range(len(xs)):
            det = (xs[i][1] - target_points[0][0]) * (target_points[1][1] -
target_points[0][1]) - (xs[i][2] - target_points[0][1]) * (target_points[1][0] -
target_points[0][0])
            if det > 0:
                # on one side
                ys[i] = 1
            else:
                # on the other side
                ys[i] = -1
        processed = np.linalg.pinv(xs)
        weight = np.dot(processed, ys)
        # use weight to classify points
        newys = np.sign(np.dot(np.transpose(weight), np.transpose(xs)))

        for i in range(len(ys)):
            if (newys[i] != ys[i]):
                error += 1
    print float(error) / (runs * points)
```

```
def prob6():
    error = 0
    points = 1000
    for q in range(runs):
        xs = np.random.uniform(-1,1, (points, d))
        xs = np.c_[np.ones(points), xs]

        out_sample = np.random.uniform(-1,1, (points, d))
        out_sample = np.c_[np.ones(points), out_sample]

        target_points = np.random.uniform(-1,1, (2, d))
        ys = np.zeros(points)
        for i in range(len(xs)):
            det = (xs[i][1] - target_points[0][0]) * (target_points[1][1] -
target_points[0][1]) - (xs[i][2] - target_points[0][1]) * (target_points[1][0] -
target_points[0][0])
            if det > 0:
                # on one side
                ys[i] = 1
            else:
                # on the other side
                ys[i] = -1
        processed = np.linalg.pinv(xs)
        weight = np.dot(processed, ys)
        # use weight to classify points
        newys = np.sign(np.dot(np.transpose(weight), np.transpose(xs)))
        outys = np.sign(np.dot(np.transpose(weight), np.transpose(out_sample)))

        for i in range(len(ys)):
            if (outys[i] != ys[i]):
                error += 1
    print float(error) / (runs * points)


def prob7():
    error = 0
    points = 10
    total_iterations = 0
    for q in range(runs):
        xs = np.random.uniform(-1,1, (points, d))
        xs = np.c_[np.ones(points), xs]
        target_points = np.random.uniform(-1,1, (2, d))
        ys = np.zeros(points)
        for i in range(len(xs)):
            det = (xs[i][1] - target_points[0][0]) * (target_points[1][1] -
target_points[0][1]) - (xs[i][2] - target_points[0][1]) * (target_points[1][0] -
target_points[0][0])
            if det > 0:
                # on one side
                ys[i] = 1
            else:
                # on the other side
                ys[i] = -1
        processed = np.linalg.pinv(xs)
        weight = np.dot(processed, ys)

        errors = points
        while(errors != 0):
            total_iterations += 1
            i = np.random.randint(points)

            out = np.sign(np.dot(np.transpose(weight), np.transpose(xs)))
            if (np.array_equal(out, ys)):
                break
            else:
                weight += xs[i]
            errors -= 1
    print float(total_iterations) / runs
```

```
import numpy as np
import random
import matplotlib.pyplot as plt

runs = 1000
d = 2

def prob8_9_10():
    error = 0
    points = 1000
    for q in range(runs):
        xs = np.random.uniform(-1,1, (points, d))
        xs = np.c_[np.ones(points), xs]
        out_sample = np.random.uniform(-1,1, (points, d))
        out_sample = np.c_[np.ones(points), out_sample]

        target_points = np.random.uniform(-1,1, (2, d))
        ys = np.zeros(points)
        for i in range(len(xs)):
            det = (xs[i][1] - target_points[0][0]) * (target_points[1][1] -
target_points[0][1]) - (xs[i][2] - target_points[0][1]) * (target_points[1][0] -
target_points[0][0])
            if det > 0:
                # on one side
                ys[i] = 1
            else:
                # on the other side
                ys[i] = -1

        processed = np.linalg.pinv(xs)
        #weight = np.dot(processed, ys)
        weight = np.array([1, xs[0][0], xs[0][1]])
        weight2 = np.array([1, xs[0][0], xs[0][1], xs[0][0] * xs[0][1], xs[0][0] **
2, xs[0][1] ** 2])
        # use weight to classify points
        newys = np.sign(np.dot(np.transpose(weight), np.transpose(xs)))
        outys = np.sign(np.dot(np.transpose(weight), np.transpose(out_sample)))
        noises = np.random.choice(range(points), points / 10, replace = False)
        for noise in noises:
            ys[noise] *= -1

        for i in range(len(ys)):
            if (outys[i] != ys[i]):
                error += 1
    print float(error) / (runs * points)
    print weight2
```