```
'''
Created on Feb 14,2011
@author: ola
'''

import random

def toCardString(card):
    rankStrings = ["ace","two","three","four","five","six","seven",
                   "eight","nine","ten","jack","queen","king"]
    suitStrings = ["spades", "hearts", "diamonds","clubs"]
    return rankStrings[card[0]] + " of " + suitStrings[card[1]]

def get_rank_counts(hand):
    '''
    returns list of how often each rank 1-13 occurs in a hand
    e.g., list[3] = # occurrences of a 3, list[11] = # occurrences of jack
    '''
    counts = [0]*13
    for card in hand:
        rank = card[0]
        counts[rank] += 1
    return counts

def is_pair(hand):

    ranks = get_rank_counts(hand)
    if ranks.count(2) == 1 and ranks.count(1) == 3:
        return True
    return False

def get_deck():
    '''
    create and return an unshuffled deck of cards
    '''
    d = []
    for rank in range(0,13):
        for suit in range(0,4):
            d.append([rank,suit])
    return d

def get_hand(deck):
    random.shuffle(deck)
    return deck[0:5]

def print_hand(hand):
    print '[',
    for card in hand:
        print toCardString(card),',',
    print ']'


def deal_demo():
    deck = get_deck()
    print_hand(deck)
    print_hand(get_hand(deck))
    print_hand(get_hand(deck))

def simulate(n):
    '''
    Simulate dealing n poker hands
    '''
    deck = get_deck()
    pc = 0
    for i in range(0,n):
        hand = get_hand(deck)
        if is_pair(hand):
            pc += 1
    print "# hands = %d, # pairs = %d, prob = %.3f" % (n,pc, 1.0*pc/n)

if __name__ == "__main__":
    simulate(500)
```



```
'''
Created on Feb 13, 2011
@author: ola
'''

import random

_DIESIDES = 6

def dice():
    '''
    Returns a random roll of two 'fair' die
    '''
    a = random.randint(1,_DIESIDES)
    b = random.randint(1,_DIESIDES)
    return a+b

def generate_rolls(n):
    '''
    Return list of n simulated dice rolls
    '''
    st = [dice() for i in range(0,n)]
    return st

def get_bar(length):
    '''
    Return String/bar for graphing based on max-bar = 60
    '''
    return '*'*int(length*60)


def print_stats(rolls):
    '''
    Print statistics for rolls, a list of simulated dice rolls
    rolls is a list of int values representing dice rolls
    '''
    counters = [0]*(2*_DIESIDES+1)
    for roll in rolls:
        counters[roll] += 1
    mx = max(counters)
    for i in range(2,2*_DIESIDES+1):
        print "%3d %s %5d" % (i,get_bar(counters[i]*1.0/mx),counters[i])

def main():
    st = generate_rolls(100000)
    print_stats(st)

if __name__ == "__main__":
    main()

"""
One run of program shown below:

 2 ********** 2784
 3 ******************** 5472
 4 ****************************** 8353
 5 **************************************** 10959
 6 ***************************************************** 14002
 7 ************************************************************ 16515
 8 ***************************************************** 13973
 9 **************************************** 11174
10 ****************************** 8327
11 ******************** 5693
12 ********** 2748
"""
```

```
'''
Created on Feb 13, 2011
@author: ola
'''

import random

def get_tosses(count):
    '''
    Toss simulated coin count (int) times
    Returns list of random coin tosses, 'T' or 'H'
    length of returned list is count
    '''
    tosses = []
    for i in range(0,count):
        if random.random() < 0.5:
            tosses.append("H")
        else:
            tosses.append("T")
    return tosses

def get_tosses2(count):
    tosses = ["T" if random.random() < 0.5 else "H" for i in range(0,count)]
    return tosses

def stats(tosses):
    '''
    Prints statistics for tosses, a list of
    coin tosses, each element of tosses either 'H' or 'T'
    '''
    tails = tosses.count("T")
    print "#tails = %d, # heads = %d" % (tails, len(tosses)-tails)
    runs = []
    rc = 0
    for toss in tosses:
        if toss == 'T':
            rc += 1
        else:
            if rc != 0:
                runs.append(rc)
            rc = 0
    runs.append(rc)
    maxt = max(runs)
    counts = [0]*(maxt+1)
    for t in runs:
        counts[t] += 1
    print "\nTail Runs"
    print "%s\t%s" % ("length", "count")
    for t in range(1,len(counts)):
        print "%3d\t%d" % (t,counts[t])

def simulate():
    stats(get_tosses(10000))

if __name__ == "__main__":
    simulate()

'''
Output of one run shown below

#tails = 4995, # heads = 5005

Tail Runs
length    count
  1   1208
  2   656
  3   307
  4   152
  5   75
  6   35
  7   24
  8   10
  9   5
 10   2
 11   2
 12   1
 13   0
 14   1
'''
```