SELECTED SOLUTIONS

CHAPTER 2

Exercise Set 2.1.1

1. { }, finite

2. {2, 3, 5, 7, 11, …} infinite

3. {6, 12, 18, 24, 30, …} infinite

4. { } finite

Exercise Set 2.1.2

1. Yes

2. No

3. No

4. {1, 2, 3, 4, 6, 8, 9, 12, 15, 18, 21, 24}

5. {3, 6, 12, 24}

6. {3, 6}

7. Yes

8. No

9. Yes

10. Yes

11. the set of positive integers other than the positive integer factors of 24

12. {9, 12, 15, 18, 21, 24}

13. {2, 4, 8, 12, 24 }

Exercise Set 2.1.3

1. B.issubset(A); True

2. A.issubset(B); False

3. A.intersection(C); {2}

4. A.difference(B); { 2, 4, 6, 8, 12, 14, 16, 18, 22, 24, 26, 28}

5. A.intersection(B); {10, 20, 30}

6. B.intersection(C); {} or Ø

7. B.difference(A); {} or Ø

8. A.intersection(B.union(C)); {2, 10, 20, 30}

9. (A.intersection(B.union(C)); {2, 3, 5, 7, 10, 11, 13, 17, 19, 20, 23, 29, 30}

Exercise Set 2.1.4

1. 16

2. 0

3. 26

4. 1

5. 39

6. 12

7. 48

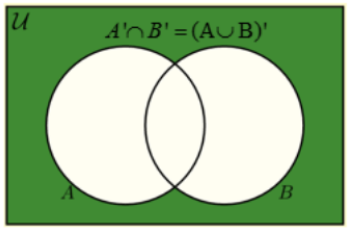
Exercise Set 2.1.5

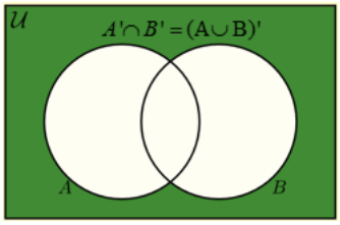
1. 52

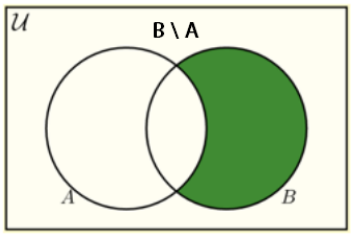
2. 28

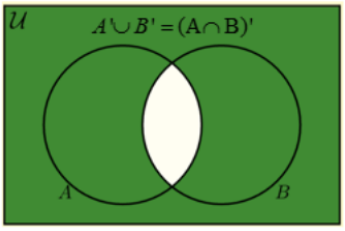
3. $n(A_1) + n(A_2) + \ldots + n(A_n) - n(A_1 \cap A_2 \cap \ldots \cap A_n)$

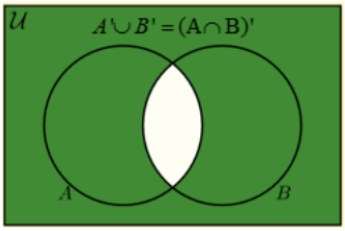
Exercise Set 2.1.6

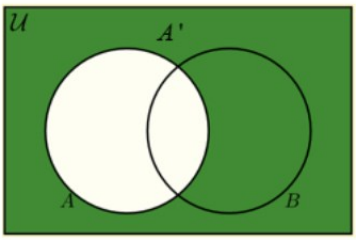
1. 

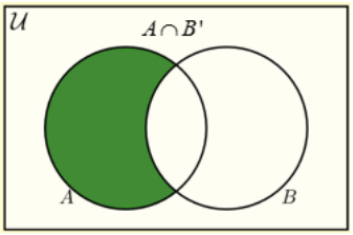
2. 

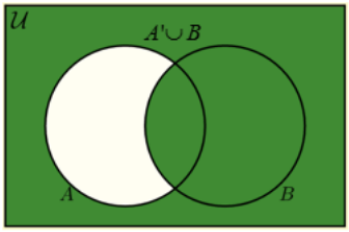
3. 

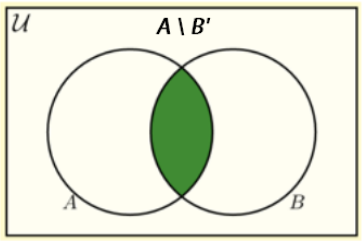
4. 

5. 

6. 

7. 

8. 

9. 

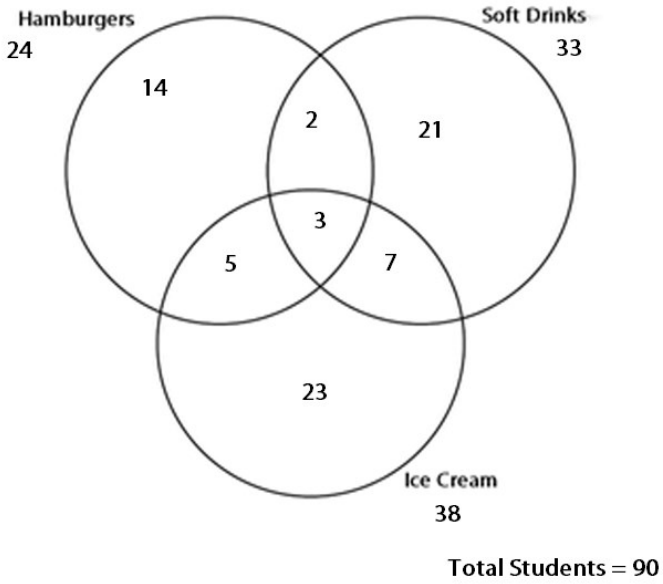
10. A \ B = A ∩ B'

Exercise Set 2.1.7

1. 19

2. 4

3. 15



Exercise Set 2.1.8

1. **print**("n(A\B) =",A – A_and_B)
2. **print**("n(B\A) =",B – A_and_B)

Exercise Set 2.1.9

1. Modify Program 2.1.2 (either version) so that a counter n counts how many times you output a line.

```
crust = ["thick","thin","flatbread"]
size = ["small","medium","large"]
topping = ["pepperoni","mushroom","onion","sausage","bacon"]
n = 0
for c in range(0,3):
   for s in range(0,3):
      for t in range(0,5):
         print(crust[c],size[s],topping[t])
         n +=1
print("Total Pizzas:",n)
```

2. 45 pizzas.

3. 90 pizzas.

```
crust = ["thick","thin","flatbread"]
size = ["small","medium","large"]
topping = ["pepperoni","mushroom","onion","sausage","bacon"]
cheese = ["regular","extra"]
n = 0
for c in range(0,3):
   for s in range(0,3):
      for t in range(0,5):
         for ch in range(0,2):
            print(crust[c],size[s],topping[t],cheese[ch])
            n +=1
print("Total Pizzas:",n)
```

4. The formula is Crusts x Sizes x Toppings x Cheeses

Exercise Set 2.1.10

1. $6 \cdot 10 \cdot 8 \cdot 4 = 1920$

2. $26^3 \cdot 10^3 = 17{,}576{,}000$

3. $26^3 = 17576$

4. $36^6 = 2{,}176{,}782{,}336$ (over 2 billion!)

5. $36 \cdot 35 \cdot 34 \cdot 33 \cdot 32 \cdot 31 = 14{,}022{,}410{,}240$

6. $62^4 = 14{,}776{,}336$

Exercise Set 2.2.1

1. $4 \cdot 3 \cdot 2 \cdot 1 = 24$

2. $4^5 = 1024$

3. $5 \cdot 2 \cdot 3 \cdot 2 = 60$

4. $10 \cdot 9 \cdot 8 \cdot 7 = 5040$

5. $10 \cdot 9 \cdot 8 = 720$

6. $365^3 = 48{,}627{,}125$

7. $52 \cdot 51 = 2652$

8. Write a program that asks the user to input n and k, then outputs ${}_nP_k$.

```
from math import factorial
n = int(input('n? '))
k = int(input('k? '))
if n>0 and k>=0 and n>=k:
```

```
    nPk = factorial(n)/factorial(n-k)
    print("n P k =",int(nPk))
else:
    print("n must be positive, k must be nonnegative, and n>=k")
```

Exercise Set 2.3.1

1. $52^5 = 380,204,032$

2. ${}_{52}C_5 = 2,598,960$

3. ${}_{52}P_5 = 311,875,200$

4. ${}_{40}C_3 \cdot {}_{15}C_3 = 4,495,400$

5. ${}_{10}C_2 \cdot {}_{15}C_2 = 4725$

6. ${}_{16}C_{12} = 1820$

7. ${}_{7}C_2 = 21$; the formula is ${}_{n}C_2$.

8. Modify your program in Exercise Set 2.2.1 #8 to calculate a combination instead of a permutation.

```
from math import factorial
n = int(input('n? '))
k = int(input('k? '))
if n>0 and k>=0 and n>=k:
    nPk = factorial(n)/factorial(n-k)
    nCk = nPk/factorial(k)
    print("n C k =",int(nCk))
else:
    print("n must be positive, k must be nonnegative, and n>=k")
```

Exercise Set 2.3.2

1. ${}_5C_3 = 10$ combinations

2. ${}_6C_3 = 20$ combinations

2. ${}_5P_3 = 60$ permutations. The following are actual words that can be played in Scrabble:

ARE, ART, ATE, EAR, EAT, ERA, RAT, SAT, SEA, SET, TAR, TEA, plus one could argue for some lesser known words and acronyms.

Exercise Set 2.4.1

1. The probability of selecting a weekday (Mon, Tue, Wed, Thu or Fri) is $\frac{5}{7}$.

2. The probability of selecting a month with exactly 30 days (Sep, Apr, Jun, Nov) is $\frac{4}{12} = \frac{1}{3}$.

3. The probability of selecting a Final Four team is $\frac{4}{68} = \frac{1}{17}$.

4. IndexError occurs ("list index out of range") when the number 5 is generated. That's because the list indices are 0 … 4. Fix the **randint** statement:

```
day = randint(0,4)
```

5. Generate 2 dice rolls.

```
from random import randint
ready = input("Press any key to roll the die ")
die_1 = randint(1,6)
die_2 = randint(1,6)
print(die_1,die_2)
```

6. Month of the year

```
from random import randint
month = ['Jan','feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec']
ready = input("Press any key to choose a month ")
m = randint(0,11)
print(month[m])
```

7. Five lottery numbers between 1 and 49.

```
from random import randint
ready = input("Press any key to choose lottery numbers ")

for i in range(0,5):
    print(randint(1,49))
```

8. Random card

```
from random import randint
ready = input("Press any key to deal a card ")

suit = ['Heart','Diamonds','Clubs','Spades']
value = ['Ace',2,3,4,5,6,7,8,9,10,'Jack','Queen','King']
print('You were dealt',value[randint(0,12)],'of',suit[randint(0,3)])
```

9. Two random cards

```
from random import randint
ready = input("Press any key to deal 2 cards ")

suit = ['Heart','Diamonds','Clubs','Spades']
value = ['Ace',2,3,4,5,6,7,8,9,10,'Jack','Queen','King']
print("You were dealt:")
for i in range(1,3):
    print(value[randint(0,12)],'of',suit[randint(0,3)])
```

Exercise Set 2.4.2

1. Modify Program 2.4.3 so that you are to guess a number between 1 and 10 rather than between 1 and 5.

```
from random import randint
num = randint(1,10)
guess = int(input("Guess a number between 1 and 10: "))
if guess == num:
   print("Correct!")
else:
   print("Sorry, the number was",num)
```

2. Modify Program 2.4.3 so that if the user guesses a number outside the range (less than 1 or greater than 10), they will get an error message and the program will end.

```
from random import randint
num = randint(1,10)
guess = int(input("Guess a number between 1 and 10: "))
if guess<1 or guess>10:
   print("You are a bad guesser")
elif guess == num:
   print("Correct!")
else:
   print("Sorry, the number was",num)
```

3. Modify the program in Exercise 1 by allowing the user to keep guessing until they get it right.

```
from random import randint
num = randint(1,10)
guess=0
while guess != num:
   guess = int(input("Guess a number between 1 and 10: "))
   if guess<1 or guess>10:
      print("You are a bad guesser")
   elif guess > num:
      print("Too high")
   elif guess < num:
      print("Too low")
   else:
      print("Correct!  The number was",num)
```

Exercise Set 2.4.3

1.A. The value of card_name[0] is 'Ace'

B. The value of card_vallue[8] is 9

C. The variable type of card_name[0] is str

D. The variable type of card_value[8] is int

2. If card1 is assigned random number 6 and card2 is assigned random number 4,the output is:

You are dealt

7 , 5

Your total is 12

Exercise Set 2.4.4

1. Append to program:

```
:
:
if total>21:
   print("You busted")
elif total==21:
   print("You win!")
else:
   print("Good job")
```

2. Append to program:

```
:
:
# dealer hand
dcard1 =randint(0,12)
dcard2 = randint(0,12)
print("\nDealer\'s hand:")
print(card_name[dcard1],",",card_name[dcard2])
dtotal = card_value[dcard1] + card_value[dcard2]
print("Dealer\'s total is",dtotal)
```

3. Append to program:

```
:
:
# who won?
print("\n")                          # print blank line
if total>21:
   print("You busted, dealer wins")
else:  # player scored less than or equal to 21
   if dtotal>21:
      print("Dealer busted, you win!")
   else:
      if total>dtotal:
         print("Player wins!")
```

```
    elif dtotal>total:
        print("Dealer wins!")
    else:
        print("Dealer and player pushed")
```

4. Immediately after outputting player's total, insert:

```
if total==21:                          # possible Blackjack
    if card1==0 or card2==0:           # if one of the cards is an Ace
        if card2==0:                   # if the second card is the Ace
            card1,card2=card2,card1    # swap cards so 1st card is the Ace
        # check that other card is a Jack, Queen or King
        if card2 in [10,11,12]:
            print("Blackjack!")
```

Exercise Set 2.4.5

1. Modify Program 2.4.4 v3 so that the player can request more than one extra card.

Changes shown in blue.

```
from random import randint

# assign point values to each type of card

card_name=['Ace',2,3,4,5,6,7,8,9,10,'Jack','Queen','King']
card_value=[11, 2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 10, 10]
total = 0                          # initialize the total points
card = [ ]                         # this list will hold the random cards

card.append(randint(0,12))
card.append(randint(0,12))

print("You are dealt")
print(card_name[card[0]],",",card_name[card[1]])
total += card_value[card[0]] + card_value[card[1]]          # increment total points
hit_me = input("Do you want another card? Y/N ")
k = 1                                                       # k is index of card list
while hit_me=="Y":
    card.append(randint(0,12))
    k += 1
    print("You are dealt",card_name[card[k]])
    total += card_value[card[k]]
    hit_me = input("Do you want another card? Y/N ")


print("\nYour total is",total)
:
```

Exercise Set 2.4.6

1. Modify the program you wrote in Exercise Set 2.4.1 #4 (five lottery numbers between 1 and 49) so that they are unique.

```
from random import *
numbers = list(range(1,49))
shuffle(numbers)
for n in range(1,6):
   lucky_num = numbers.pop()
   print(lucky_num)
```

2. Write a program that generates and outputs a batting order for a team of nine baseball players.

```
from random import *
names = ['Axelrod','Barry','Curtis','Damon','Edward','Fahim','Gary','Hiro','Ignacio']
players = list(range(0,8))
shuffle(players)
for n in range(1,9):
   next_guy = players.pop()
   print(names[next_guy])
```

Exercise Set 2.5.1

1. The experiment is spinning the wheel.

2. The sample space is {1, 2, 5, 10, 20}.

3. P(1) = 11/24, P(2) = 7/24, P(5) = 3/24, P(10) = 2/24, P(20) = 1/24.

Exercise Set 2.5.2

1. $P(5) = \frac{3}{24} = \frac{1}{8}$

2. Write a program that generates a winning team based on the given probabilities

```
from random import randint
team_name = ['Honey Badgers','Yetis','Tasmanian Devils','T-Rex\'s']
team_index = [0,0,1,1,1,1,2,2,2,3,3,3]
select = input("Press any key to select a winning team ")
winning_team = randint(0,11)
index = team_index[winning_team]
print("\nWinning team is",team_name[index])
```

Exercise Set 2.5.3

1. 85:15 or 17:3

2. 1:7

3. 25% = 250

4. 65% → 65:35 or 13:7

5. $\frac{1}{301}$

6. $\frac{1}{2000001}$

Exercise Set 2.5.4

1. wheel[10] = 1

2. They lost because wheel[10] = 1 but they bet on 10. They lost 50.

3. They won because wheel[23] = 20 and they bet on 20. They won 20*50 = 1000.

4. **print**("You won $",wager*bet)

5. **print**("You lost $",wager)

6. Modify the program you wrote in Exercise Set 2.5.2 #1 so that it asks the user for the amount of their bet, which team they're betting on, and then output how much they won or lost.

```
from random import randint
team_name = ['Honey Badgers','Yetis','Tasmanian Devils','T-Rex\'s']
odds = [6,3,4,4]
team_index = [0,0,1,1,1,1,2,2,2,3,3,3]
wager = int(input("What is your wager? "))
print("Which team do you want:")
for n in range(4):
   print(n+1,"-",team_name[n])
choice = int(input("? "))
bet = choice-1
team_index = [0,0,1,1,1,1,2,2,2,3,3,3]

# choose random winning team
winning_team = randint(0,11)
index = team_index[winning_team]
print("\nWinning team is",team_name[index])
if bet==winning_team:
   print("You won $",wager*odds[index])
else:
   print("You lost $",wager)
```

Exercise Set 2.5.5

1. Modify **bigwheel.py** to keep track of the user's money as specified in the problem.

```
from random import randint
wheel = [1,1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,5,5,5,10,10,20]
spin = "Y"                              # initialize the spin
money = int(input("How much money do you have? "))
game_over=False                         # initialize game_over flag
while spin=="Y" and not game_over:
    bet = 0                             # initialize the bet amount
    while bet not in [1,2,5,10,20]:
        bet = int(input("What number are you betting on? "))

    wager = int(input("How much is your wager? "))
    while wager>money:
        print("You only have $",money)
        wager=int(input("What is your wager? "))

    winning_number = randint(0,23)
    print("Winning number is",wheel[winning_number],"!")
    if bet==wheel[winning_number]:
        print("You won $",wager*bet)
        money += wager*bet
    else:
        print("You lost $",wager)
        money -= wager
    print("You now have $",money)
    if money>0:
        spin=input("Spin again (Y/N)? ")
    else:
        print("Game over")
        game_over=True
```

Exercise Set 2.6.1

1. $\frac{4}{36} = \frac{1}{9}$
2. $\frac{11}{36}$
3. 7
4. $\frac{1}{2}$
5. $\frac{8}{36} = \frac{2}{9}$
6. $\frac{4}{36} = \frac{1}{9}$
7. $\frac{6}{36} = \frac{1}{6}$
8. $\frac{8}{36} = \frac{2}{9}$
9. Write a program that simulates and outputs the rolling of two 6-sided dice.

```
from random import randint
```

```
die1 = randint(1,6)
die2 = randint(1,6)
sum =die1+die2
print("You rolled",die1,"and",die2)
if sum in [7,11]:
   print("PASS")
if sum in [2,3,12]:
   print("DON\'T PASS")
```

Exercise Set 2.6.2

1. $\frac{21}{36} = \frac{7}{12}$
2. $\frac{7}{36}$
3. $\frac{24}{36} = \frac{2}{3}$
4. $\frac{16}{52} = \frac{4}{13}$
5. $\frac{24}{52} = \frac{6}{13}$

Exercise Set 2.6.3

1. A. independent B. not independent C. not independent D. not mutually exclusive E. mutually exclusive F independent
2. (.8)(.8) = .64
3. $\left(\frac{1}{2}\right)^4 = \frac{1}{16}$
4. If the order doesn't matter, the probability is $\frac{1}{4}$. If the order is important (Oldest is a girl, then 3 boys), the probability is $\frac{1}{16}$.
5. A. $\frac{1}{69C5} = \frac{1}{11238513}$

   B. $\frac{1}{69C5} \cdot \frac{1}{26} = \frac{1}{292201338}$
6. A. $\frac{10}{20} = \frac{1}{2}$

   B. $\frac{14}{20} = \frac{7}{10}$

   C. $\frac{17}{20}$

   D. $\frac{10}{20} \cdot \frac{10}{20} = \frac{1}{4}$

   E. $\frac{10}{20} \cdot \frac{9}{19} = \frac{9}{38}$

7. Write a program that asks the user to input how many children they would like to have (or already have). Then calculate and output the probability of all girls.

```
from fractions import Fraction
n = int(input("How many children? "))
probG = Fraction(1,2**n)
print("The probability of all girls is",probG)
```

Exercise Set 2.6.4

1. $1 - \frac{1}{16} = \frac{15}{16}$

2. $1 - \frac{7}{20} = \frac{13}{20}$

3. Add the following line to the program:

```
print("The probability of at least one boy is",1-probG)
```

Exercise Set 2.6.5

1. Replace match = True with match = next_bd

2. Modify Program 2.6.1 so that it outputs all the matches.

```
from random import randint
birthdays =[ ]
match = [ ]
for n in range(1,24):
   next_bd = randint(1,365)
   if next_bd in birthdays:
      match.append(next_bd)
   birthdays.append(next_bd)

print(birthdays)
print("match found: ",match)
```

3. $1 - \frac{365P_{30}}{365^{30}} \approx .71$

4. A. Write a program that asks the user to input how many people are in a room, and then calculate and output the probability that at least two of them will have the same birthday.

```
from fractions import Fraction
from math import factorial
k = int(input("How many people are in the room? "))
nPk = Fraction(factorial(365),factorial(365-k)).limit_denominator()
prob = 1-Fraction(nPk,365**k).limit_denominator()
print("Probability of 2 people with same birthday is ",prob)
print("which is about",round(float(prob),3))
```

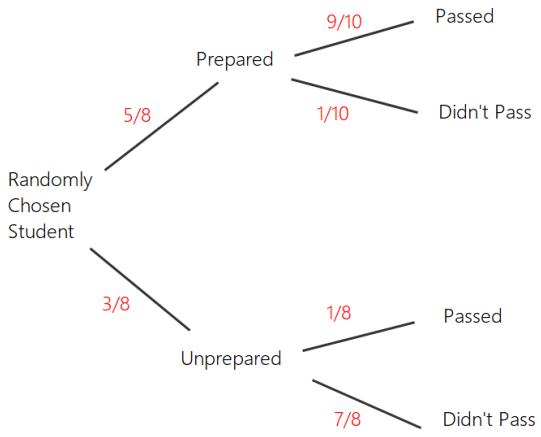
B. The largest value is 365.

Exercise Set 2.7.1

1. 8/9
2. 9/11
3. 1/10
4. 4/5
5. ½
6. 1/13
7. 1/13
8. ½
9. ¼
10. 3/16
11. 3/8

Exercise Set 2.7.2

1.



2. 9/10
3. (5/8)(9/10) = 9/16
4. 1/8
5. (5/8)(9/10) + (3/8)(1/8) = 39/64

6. (5/8)(1/10) + (3/8)(7/8) = 25/64

7. Yes, 39/64 + 25/64 = 1

8. 101/1000

9. 179/631

10. 31/37

11. 271/382

Exercise Set 2.7.3

1. Calculate and output P(A ∩ B).

```
pI = Fraction(nI,nU)                    # probability of intersection
print('P(Baroque art)=',pI)
```

2. Calculate and output P(A | B) and P(B | A).

```
pAB = Fraction(pI,pB)                   # probability of A given B
print('P(art|Baroque)=',pAB)
pBA = Fraction(pI,pA)                   # probability of B given A
print('P(Baroque|art)=',pBA)
```

Exercise Set 2.7.4

1. sentence.**split**(“,”) splits the **string** sentence, using the comma as delimiter, into a **list** of **strings**.

2. **input**() always returns a **string** type.

3. **set**(alphabet) returns a **set** with the same elements (**strings**) as the **list** alphabet.

4. An error is returned because you can’t convert a non-numeric **string** into an **integer**. For example, you can’t do a = int(“A”).

5. An **integer** type is returned.

6. Code to input a set of integers:

```
input_str = input(“Enter set of integers separated by commas: “)
input_list = input_str.split(“,”)
size = len(input_list)
for k in range(0,size):
   input_list[k] = int(input_list[k])
input_set = set(input_list)
```

Exercise Set 2.7.5

1. Yes it can be moved out of the function and no it doesn't matter whether it comes before or after the function definition.

2. Write the program shown in the diagram that prints out a random weekday. Modify the program so that it prints out ten weekdays instead on one weekday.

```
from random import randint
def rand_day():
   days  = ['Mon','Tue','Wed','Thu','Fri']
   num = randint(0,4)
   return days[num]

# main program
for n in range(10):
   day = rand_day()
   print(day)
```

3. Write a program that makes use of a function that squares the integer parameter. Ask the user to input a list of x-values, then call the function to calculate and output each ordered pair.

```
def square(n):
   nsquared = n**2
   ordered_pair = "("+str(n)+","+str(nsquared)+")"
   print(ordered_pair)

# main program
domain = input("Domain? ")
x_list = domain.split(",")
x_list = [int(x_value) for x_value in x_list]
for x_value in x_list:
   square(x_value)
```

4. Write a program that makes use of a function that randomly generates a month and day. (Keep in mind January has 31 days, February has 28 days, etc.) Ask the user how many random birthdays they want generated. Call the function to generate and output each birthday.

```
from random import *

def month_day():
    months=['', 'Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec']
    days = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 ]
    m = randint(1,12)
    d = randint(1,days[m])
    print(months[m], d)
```

```
# main program
b = int(input('How many random birthdays? '))
for n in range(b):
   month_day()
```

Exercise Set 2.7.6

1. Fill in the yellow highlighted code to calculate and output P(A), P(B), P(A ∩ B), P(A U B), p(A | B) and p(B | A).

```
:
:
pA = Fraction(nSetA,nA_U_B)
pB = Fraction(nSetB,nA_U_B)
pA_and_B = Fraction(nA_and_B,nA_U_B)
pAUB = pA + pB – pA_and_B
pA_given_B = Fraction(pA_and_B,pB)
pB_given_A = Fraction(pA_and_B,pA)

print("P(A)=",pA)
print ("P(B)=",pB)
print ("P(A and B)=",pA_and_B)
print("P(A U B)=",pAUB)
print ("P(A | B)=",pA_given_B)
print ("P(B | A)=",pB_given_A)
```

2. P(A U B) = 1 because A U B comprises all the elements.

3. No.

4. Write a function **num_string_to_set()** that converts a comma-delimited string (such as "45,36,18,92,36") into a set of integers with no repeats.

```
def num_string_to_set(a_string):
   a_list = a_string.split(",")
   n_list = [int(a_value) for a_value in a_list]
   n_set = set(n_list)
   return(n_set)

# main program
list_of_num = input("Enter list of numbers: ")
print(num_string_to_set(list_of_num))
```

5. Write a function **num_string_to_frac** that converts a comma-delimited string into a set of fractions.

```
from fractions import Fraction

def num_string_to_frac(a_string):
   a_list = a_string.split(",")
   f_list = [Fraction(a_value) for a_value in a_list]
```

```
    f_set = set(f_list)
    return(f_set)

# main program
list_of_frac = input("Enter list of fractions: ")
frac_list = num_string_to_frac(list_of_frac)
for frac in frac_list:
    print(frac)
```

Exercise Set 2.7.7

1. Output: **INSIDE FUNCTION x = 3 y = 4**

Even though x and y are defined in the main program, their values can still be accessed within the function.

2. NameError

a and b are local variables within the function. Their values cannot be accessed within the main program.

3. Output: **INSIDE FUNCTION x = 5 y = 12**

This refers to local variables defined within the function, not the x and y defined in the main program.

4. Output: **MAIN PROGRAM x = 3 y = 4**

These are local variables in the main program. Their values were unaffected by the change in values to the variables with the same name within the function.

Exercise Set 2.8.1

1. $\frac{3}{5}$ or 60%

2. 30%

3. Probably 100% although Orange County, CA voted for a Democrat in 2016 for the first time since 1936.

4. Theoretical probability is more reliable, as the anecdote in question 3 above demonstrates.

Exercise Set 2.8.2

1. $\frac{1}{9}$

2. The probability is about 8.6%.

3. The probability is 3π/50, or about 19%.