SELECTED SOLUTIONS

CHAPTER 3

Exercise Set 3.1.1

1. yes
2. no
3. domain is $(-\infty, \infty)$; range is $[-4, \infty)$
4. $f(-2) = -16$
5. $g(-x) = 1 + 3x$
6. $f(0) \cdot g(0) = 0$
7. $g(x + h) = 1 - 3x - 3h$
8. $f(-x) - g(-x) = -2x^3 - 1 - 3x$
9. $-g(x) = -1 + 3x$
10. no

Exercise Set 3.1.2

1. polynomial; 4th degree (quartic). Standard form: $8x^4 + 47x^3 + 11x^2 - 31$
2. binomial, 3rd degree (cubic). Already in standard form.
3. not a polynomial
4. binomial, 3rd degree (cubic). Standard form: $3x^3 + \sqrt{12}x^2$
5. monomial; 1st degree (linear). Already in standard form.

Exercise Set 3.1.3

1. -5
2. -4

3. -2

4. 0

5. -4

6. -2

7. 6

Exercise Set 3.1.4

1. Write a program that asks the user to input a string of integers separated by commas. Convert to a list of integers and output the list.

```
numbers = input('integers? ')
num_list = [int(n) for n in numbers.split(',')]
print(num_list)
```

2. Modify the program so that it handles float numbers in addition to integers.

```
numbers = input('real numbers? ')
num_list = [float(n) for n in numbers.split(',')]
print(num_list)
```

3. Write a program that asks the user to input a list of dogs, separated by forward slashes. Convert to a list of strings and output the list.

```
dogs = input('dogs? ')
dog_list = [d for d in dogs.split('/')]
print(dog_list)
```

Exercise Set 3.1.5

1. A. 1, 0, -1, 1

   B. -1, 1 ,0, 0, 0, 1

   C. 1, 0, 0, 0, 0

2. change **print**(poly) to **pprint**(poly)

3. A. 3*x**4 + 11*x**3 + 19*x + 4

   B. 5 - 6*x**3

   C. x**6

4. No, sympify() does not always produce a polynomial in standard form, as we saw in #3B above.

5. Modify the program so that the user is also asked to input a value **a** to be substituted into the polynomial.

Append or insert the following lines:

```
a = int(input('a? '))
value = poly.subs({x:a})
print('f(', a, ')=', value)
```

Exercise Set 3.1.6

1. Modify program **input_polynomial** so that it asks for a second list of coefficients and outputs a second polynomial.

Append the following lines:

```
co_str2 = input('coefficients of 2nd polynomial? ')
coeff2 = [int(n) for n in co_str2.split(',')]
poly2 = num_list_to_poly(coeff2)
print(poly2)
```

2. Write a program that asks the user to input the coefficients of a polynomial, and also a constant. Multiply the polynomial by that constant.

```
from sympy import *
x = Symbol('x')

def num_list_to_poly(coeff):
   :
   :

# main program
co_str = input('Coefficients? ')
coeff = [int(n) for n in co_str.split(',')]
a = int(input('Constant? '))
coeff2 = [a*n for n in coeff]
poly = num_list_to_poly(coeff2)
print(poly)
```

Exercise Set 3.2.1

1. $-2x^2 - 3x - 2$

2. $10x^2 - 2x + 1$

3. $6x^2 + 2x + 15$

4. $7x^3 - 2x^2 + 3x + 4$

5. $-4a^3 - 2a + 9$

6. $3a^4 - 12a^2 + 33a$

7. $-56x^7 + 8x^6 + 36x^2$

8. $49u^2 + 14uv + v^2$

9. $6x^3 - 22x^2 + 41x - 35$

10. $x^4 + 3x^3 + 5x^2 - 5x - 8$

11. $2a^4 - a^3 - 20a^2 - 29a - 12$

12. $x^3 + 6x^2 + 11x + 6$

13. $2x^3 + 7x^2 - 7x - 12$

Exercise Set 3.2.2

1. $107x6 + 103x^5 - 25x + 12$

2. Add the following lines to the end of program **mathpoly**:

```
diff = expand(polynomial1-polynomial2)
product = expand(polynomial1*polynomial2)
print('Difference:',diff)
print('Product:', product)
```

3. Write a program that asks the user to enter expressions representing the length, width and height of a rectangular solid ("box"). Calculate and output expressions for the volume and surface area.

```
from sympy import *
import poly_fcn              # file containing functions num_string_to_list
                             # and num_list_to_poly

# main program
x=Symbol('x')
length = input('length? ')
length_coeff_list = [int(n) for n in length.split(',')]
L = poly_fcn.num_list_to_poly(length_coeff_list)

width = input('width? ')
width_coeff_list = = [int(n) for n in width.split(',')]
W = poly_fcn.num_list_to_poly(width_coeff_list)

height = input('height? ')
```

```
height_coeff_list = = [int(n) for n in height.split(',')]
H = poly_fcn.num_list_to_poly(height_coeff_list)

volume = expand(L*W*H)
print('volume: ',volume)
SA = expand(2*L*W + 2*L*H + 2*W*H)
print('surface area: ',SA)
```

Exercise Set 3.3.1

1. $\frac{2x^3}{3}$
2. $\frac{2}{3x}$
3. $11x^2 + 2x - 3 + \frac{4}{x}$
4. $\frac{5x^2}{2} - \frac{3x}{2} + 1 - \frac{1}{2x}$
5. $9x^3 + 6x^2 - 4 + \frac{5}{x}$
6. $3x^2 - 6x + \frac{1}{2} - \frac{1}{6x}$

Exercise Set 3.3.2

1. Quotient: $2x^2 - 7x + 18$; Remainder: -31
   $2x^3 - 3x^2 + 4x + 5 = (x + 2)(2x^2 - 7x + 18) + 31$

2. Quotient: $8x^2 - 2x - 3$; Remainder: 0
   $8x^3 - 10x^2 - x + 3 = (x - 1)(8x^2 - 2x - 3)$

3. Quotient: $x + 2$; Remainder: 0
   $x^3 + 3x^2 - 4x - 12 = (x^2 + x + 6)(x + 2)$

4. Quotient: $2x^2 - 5x + 6$; Remainder: -17
   $4x^3 - 8x^2 + 7x - 11 = (2x + 1)(2x^2 - 5x + 6) - 17$

5. Quotient: $3x - 2$; Remainder: $3x + 4$
   $3x^3 - 2x^2 + 5 = (x^2 - 1)(3x - 2) + (3x + 3)$

Exercise Set 3.3.3

1. x + 2

2. The result is not expressed using quotient and remainder. The result is the same as the original division problem: $\frac{x^3-2x^2-4}{x-3}$.

3. No, the program is not useful for finding the quotient and remainder if there is a remainder.

4. Output

Quotient:
$x^2 + x + 3$

Remainder:
5

5. Modify the program to output the result as *dividend = (divisor)(quotient) + remainder.*

```
from sympy import *
# user-defined functions
import poly_fcn as pf                    # Use an alias (pf) for poly_fcn

x = Symbol('x')

coeff_str1 = input("Dividend? ")
coeff_list1 = [int(n) for n in coeff_str1.split(',')]
dividend = pf.num_list_to_poly(coeff_list1)

coeff_str2 = input("Divisor? ")
coeff_list2 = = [int(n) for n in coeff_str2.split(',')]
divisor = pf.num_list_to_poly(coeff_list2)

q,r = div(dividend,divisor)

print(dividend,"= (",divisor,")(",q,")+",r)
```

Exercise Set 3.3.4

1. Place the **poly_to_string**() function in the function library **poly_fcn.py**. Then use function **poly_to_string()** in program **dividepoly.py.**

```
# built-in modules
from sympy import *
```

```
# user-defined functions
import poly_fcn as pf                    # will include functions in file poly_fcn.py

x = Symbol('x')

coeff_str1 = input("Dividend? ")
coeff_list1 = = [int(n) for n in coeff_str1.split(',')]
dividend = pf.num_list_to_poly(coeff_list1)



coeff_str2 = input("Divisor? ")
coeff_list2 = = [int(n) for n in coeff_str2.split(',')]
divisor = pf.num_list_to_poly(coeff_list2)

q.r = div(dividend,divisor)
dividend = pf.poly_to_string(dividend)
divisor = pf.poly_to_string(dividend)

q = pf.poly_to_string(q)
r = pf.poly_to_string(r)
result = dividend + " = (" + divisor +  ")("+q+")"

if int(r) != 0:
   result += "+ "+r

print(result)
```

Exercise Set 3.4.1

1. -31
2. 0
3. 0
4. -9
5. 0
6. Yes because the remainder is 0.
7. No because the remainder is not 0.

Exercise Set 3.4.2

1. -54

2. 886

3. 3494

4. Write a program that asks the user to input a polynomial and a value **a**. Use polynomial division (not **.subs()**) to evaluate the polynomial at that value.

```
from sympy import *
poly = input('polynomial? ')
a = int(input('a? '))
divisor= 'x-'+str(a)
q,r = div(poly,divisor)
print('f(', a, ')=', r)
```

5. Write a program that asks the user to input a dividend and divisor. Based on the remainder, identify if the divisor is a factor of the dividend.

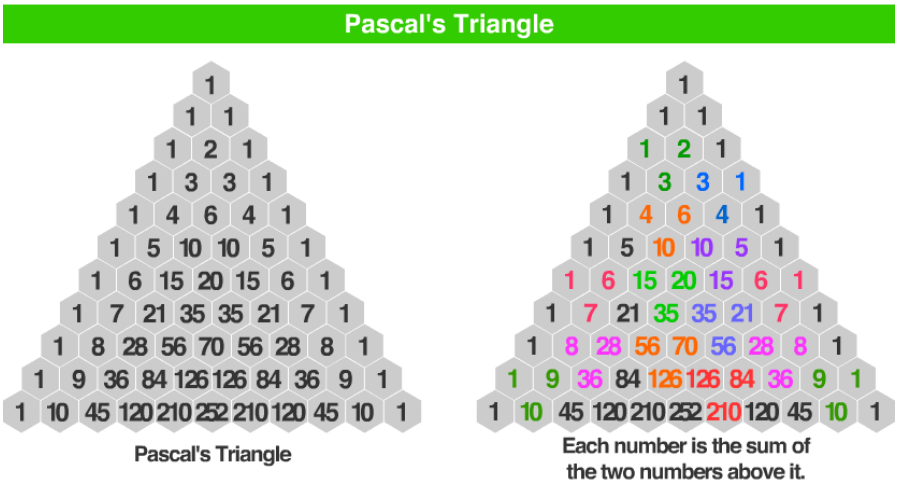
```
from sympy import *
poly = input('dividend? ')
divisor = input('divisor? ')
q,r = div(poly,divisor)
if r==0:
   print(divisor, 'is a factor of', poly)
else:
   print(divisor, 'is not a factor of', poly)
```

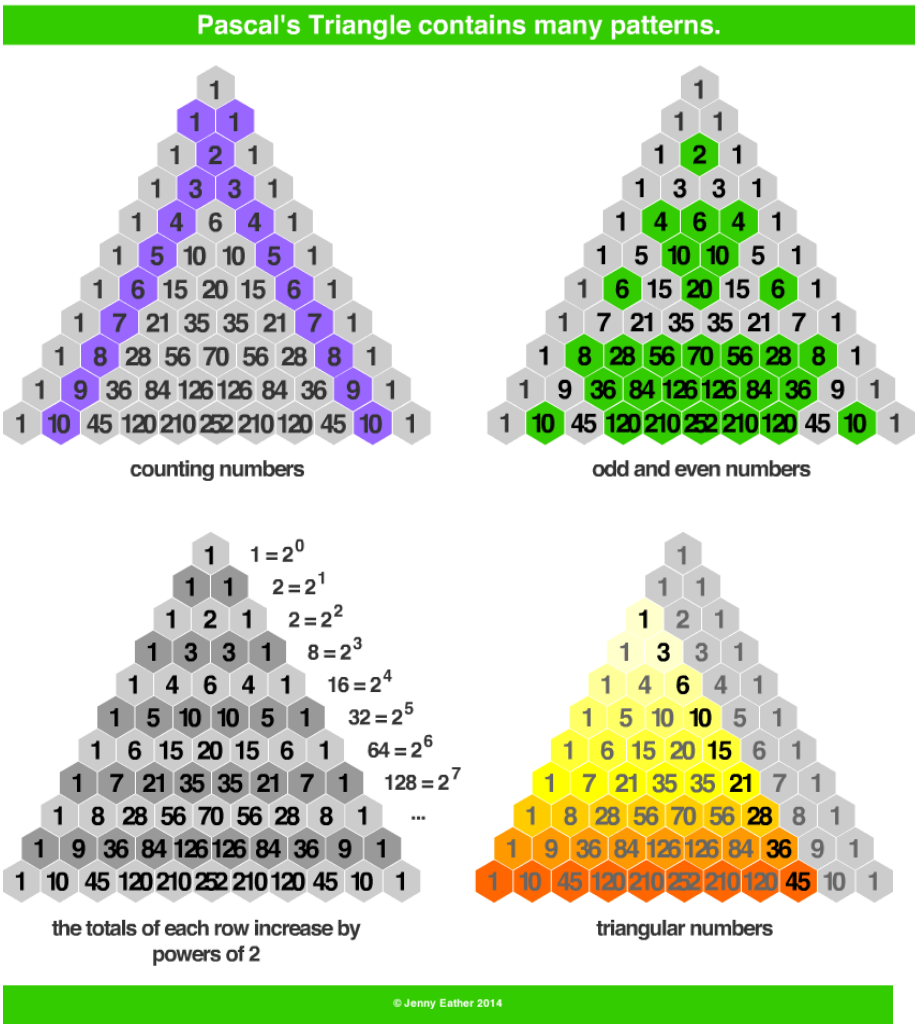
Exercise Set 3.5.1

1. 1, 7, 21, 35, 35, 21, 7, 1

   1, 8, 28, 56, 70, 56, 28, 8, 1

2. Possible Answers:





Exercise Set 3.5.2

1. $\binom{6}{0}$ $\binom{6}{1}$ $\binom{6}{2}$ $\binom{6}{3}$ $\binom{6}{4}$ $\binom{6}{5}$ $\binom{6}{6}$
2. $\binom{n}{0} = \binom{n}{n}$ (because they both equal 1)
3. $\binom{n}{1} = \binom{n}{n-1}$ (because they both equal the row number. (Assuming the first row is numbered 0.)
4. $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$

Exercise Set 3.5.3

1. Fill in the yellow highlighted code to calculate and output the coefficients of $(a + b)^n$.

```
coeffs = ""
for k in range(n+1):
   coeffs += str(n_C_k(n,k))+" "
print(coeffs)
```

2. n = 15:

1 15 105 455 1365 3003 5005 6435 6535 5005 3003 1365 455 105 15 1

Exercise Set 3.5.4

1. $a^3 + 3a^2b + 3ab^2 + b^3$

2. $x^5 + 5x^4 + 10x^3 + 10x^2 + 5x + 1$

3. $u^4 + 8x^3v + 24u^2v^2 + 32uv^3 + 16v^4$

4. $16a^4 - 32a^3 + 24a^2 - 8a + 1$

5. $64y^6$

6. $8640x^3$

7. Modify program **binomial_exp** so that it outputs the complete binomial expansion of $(x+1)^n$.

```
import math
import poly_fcn as pf

# user-defined function
def n_C_k(n,k):
   return int((math.factorial(n))/((math.factorial(n-k))*(math.factorial(k))))

# main program
n = int(input("n? "))
print("(x+1)^",n,"=")

coeffs = [ ]
for k in range(n+1):
   a = n_C_k(n,k)
   coeffs.append(a)

poly_list = pf.num_list_to_poly(coeffs)
print(pf.poly_to_string(poly_list))
```

Exercise Set 3.6.1

1. (-2, 0), (4, 0)

2. (-4, 0), $(\frac{3}{2}, 0)$

3. (2, 0), (4, 0)

4. (3, 0)

5. $\left(\frac{-5+\sqrt{21}}{2}, 0\right), \left(\frac{-5-\sqrt{21}}{2}, 0\right)$

6. (7, 0), (-7, 0)

7. Write a program that asks the user to input the coefficients a and b of linear function f(x) = ax + b. Calculate and output the zero of the function.

```
a = int(input('a? '))
b = int(input('b? '))
x = -b/a
print(x)
```

8. Write a program that asks the user to input the coefficients a, b, and c of a quadratic function $f(x) = ax^2 + bx + c$. Calculate and output the solutions using the quadratic formula. Express answers as decimals rounded to 2 decimal places.

```
from math import sqrt
a = int(input('a? '))
b = int(input('b? '))
c = int(input('c? '))
x1 = -b/(2*a) + sqrt(b**2 - 4*a*c)/(2*a)
x2 = -b/(2*a) - sqrt(b**2 - 4*a*c)/(2*a)
print(x1,x2)
```

9.

```
import poly_fcn as pf
v = input('h,k? ')
p = input('x,y? ')
vertex = [int(n) for n in v.split(',')]
point = [int(n) for n in p.split(',')]

h,k = vertex[0],vertex[1]
x,y = point[0],point[1]
a = (y-k)/(x-h)**2
quad = str(a)+'*(x-'+str(h)+')**2+'+str(k)
quad = quad.replace('--','+')
```

```
quad = quad.replace('+-','-')
print(quad)
```

Exercise Set 3.6.2

1. Fill in the yellow highlighted code to output the original polynomial, its factorization, and zeros.

```
print("\n")
pprint(poly)
print("Factorization:")
pprint(factor(poly)
print("Zeros:")
pprint(zeros)
```

2. $(3x - 4)(x - 5)(x + 17)$; [4/3, 5, -17]

3. $x^2 - 2$; $-\sqrt{2}, \sqrt{2}$

4. $(x - 12)(x - 8)^2 (x + 11)$; -11, 8, 12

5. Modify the program so that it can solve any polynomial equation in the form p(x)=q(x) where p and q are polynomial functions.

```
from sympy import *
import poly_fcn as pf

x = Symbol('x')
poly_str_L_: = input("Coefficients of Left Side? ")
poly_list_L = [int(n) for n in poly_str_L.split(',')]

poly_L = pf.num_list_to_poly(poly_list_L)

poly_str_R_: = input("Coefficients of Right Side? ")
poly_list_R = [int(n) for n in poly_str_R.split(',')]
poly_R = pf.num_list_to_poly(poly_list_R)

print("\nEquation:")
out_string = pf.poly_to_string(poly_L) + " = " + pf.poly_to_string(poly_R)
print(outstring)

poly = poly_L – poly_R
zeros = solve(poly)
print("\nZeros:")
pprint(zeros)
```

6. -3, 1, 7/2

7. -4, -2, 1/3, 2, 7

8. -2, -1/3, 11, 15

Exercise Set 3.7.1

1. 43.7

2. (2,88)

3. Using (1,84) and (5,65): y = -4.75x + 88.75 which is relatively close to the linear regression model y = -4.3x + 86.7.

4. y = .704x + 119.026

5. Write a program that calculates $\sum x,\ \sum y,\ \sum xy, \sum x^2\ and\ (\sum x)^2$ for the following ordered pairs: (1, 3), (2, 6), (3, 11), (4, 12).

```
x = [1, 2, 3, 4]
y = [3, 6, 11, 12]
xy = [ ]
x2 = [ ]
for k in range(4):
   xy.append(x[k]*y[k])
   x2.append(x[k]**2)
sumx = sum(x)
sumy = sum(y)
sumxy = sum(xy)
sumx2 = sum(x2)
sumx_2 = sumx**2
print('Sum of x:',sumx)
print('Sum of y:',sumy)
print('Sum of xy:',sumxy)
print('Sum of x^2:',sumx2)
print('(Sum of x)^2:', sumx_2)
```

Exercise Set 3.7.2

1. Write a program **linreg** to perform linear regression on lists x and y.

```
x = [1,1,2,2,2,3,3,3,4,5]
y = [84,78,88,74,79,75,72,69,71,65]
N = len(x)
xy = [ ]
```

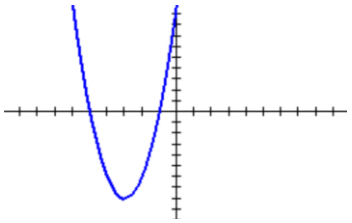
```
x2 = [ ]
for k in range(N):
   xy.append(x[k]*y[k])
   x2.append(x[k]**2)
sumx = sum(x)
sumy = sum(y)
sumxy = sum(xy)
sumx2 = sum(x2)
sumx_2 = sumx**2
m_numerator = N*sumxy - sumx*sumy
m_denominator = N*sumx2 - sumx_2
m = round(m_numerator/m_denominator,2)
b = round((sumy - m*sumx)/N,2)
print('y =',m,'x+',b)
```

2. Modify the program you wrote in question 1 by replacing the assignment statements with input statements.

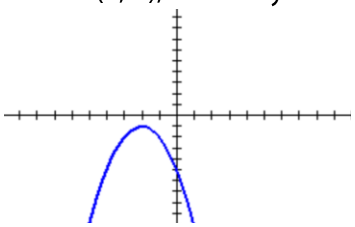
```
import poly_fcn as pf
x_str = input('x list, separated by commas? ')
y_str = input('y list, separated by commas? ')
x = pf.num_string_to_list(x_str)
y = pf.num_string_to_list(y_str)
N = len(x)
xy = [ ]
x2 = [ ]
for k in range(N):
   xy.append(x[k]*y[k])
   x2.append(x[k]**2)
sumx = sum(x)
sumy = sum(y)
sumxy = sum(xy)
sumx2 = sum(x2)
sumx_2 = sumx**2
m_numerator = N*sumxy - sumx*sumy
m_denominator = N*sumx2 - sumx_2
m = round(m_numerator/m_denominator,2)
b = round((sumy - m*sumx)/N,2)
print('y =',m,'x+',b)
```

Exercise Set 3.8.1

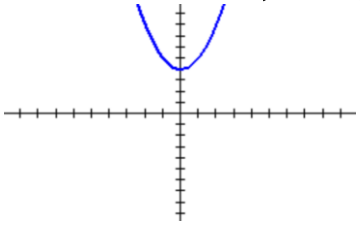
1. $y = 2(x+3)^2 - 8$
   Vertex: (-3,-8); Axis of Symmetry: x = -3; y-int: 10; x-int: 5, -1; face up



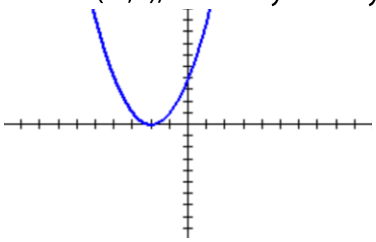
2. $y = -(x-2)^2 - 1$
   Vertex: (2,-1); Axis of Symmetry: x = 2; y-int: -5; x-int: none; face down



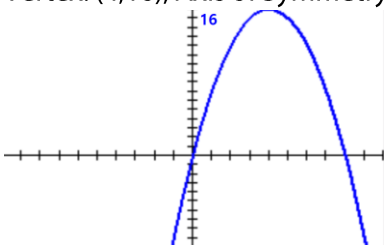
3. $y = x^2 + 4$
   Vertex: (0,4); Axis of Symmetry: x = 0; y-int: 4; x-int: none; face up



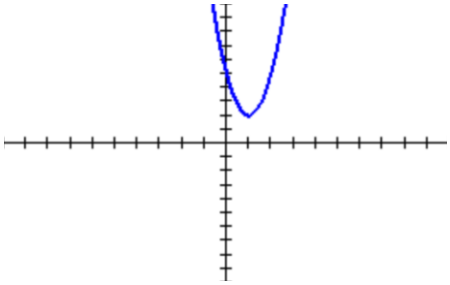
4. $y = x^2 + 4x + 4$
   Vertex: (-2,0); Axis of Symmetry: x = -2; y-int: 4; x-int: -2; face up



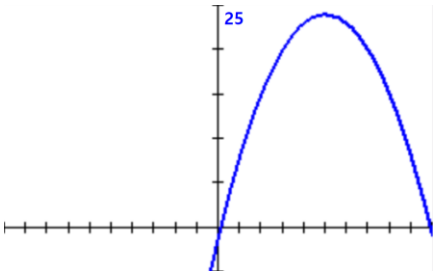
5. $y = -x^2 + 8x$
   Vertex: (4,16); Axis of Symmetry: x = 4; y-int: 0; x-int: 0,8; face down



6. $y = 3x^2 - 6x + 5$
   Vertex: (1,2); Axis of Symmetry: x = 1; y-int: 5; x-int: none; face up



7. $y = -x^2 + 10x - 1$
   Vertex: (5,24); Axis of Symmetry: x = 5; y-int: -1; x-int: 0.1, 9.9; face down



8. Write a program to calculate the x-and y-intercepts of a quadratic function in the form $y = a(x - h)^2 + k$.

```
from sympy import *
x = Symbol('x')
a = int(input('a? '))
h = int(input('h? '))
k = int(input('k? '))
quad = str(a)+'*(x-'+str(h)+')**2 +'+str(k)
quad = quad.replace('--','+')
quad = quad.replace('+-','-')
quad=sympify(quad)
x_int = solve(quad)
print('x-int:',x_int)
y_int = quad.subs({x:0})
print('y-int:',y_int)
```

Exercise Set 3.8.2

1. Vertex: (-6,2); Axis of Symmetry: x = -6; x-intercepts: none; y-intercept: (0,38); face up

2. Vertex: $\left(-\frac{3}{2}, -\frac{9}{2}\right)$; Axis of Symmetry: $x = -\frac{3}{2}$; x-intercepts: (0,0), (-3,0); y-intercept: (0,0); face up

3. Vertex: (3,4); Axis of Symmetry: x = 3; x-intercepts: (1,0), (5,0); y-intercept: (0,-5); face down

4. A. s(2) = 192 ft.
   B. s(t) = 0 when t = 4sec.

5. A. about 156 ft.

   B. about 78 ft.

   C. about 313 ft.

6. A. $1400 per item

   B. $2,940,000 total revenue

7. Write a program that does the following: (1) Asks user to input a, b, and c. (2) Calculates and outputs the vertex, axis of symmetry, y-intercept, and direction of graph.

```
from sympy import *
x = Symbol('x')
a = int(input('a? '))
b = int(input('b? '))
c = int(input('c? '))
quad = str(a)+'*x**2+'+str(b)+'*x+'+str(c)
quad=sympify(quad)
vx = -b/(2*a)
vy = quad.subs({x:vx})
print('vertex: (', vx, ',', vy, ')')
print('axis of symmetry: x=',vx)
y_int = quad.subs({x:0})
print('y-int:',y_int)
if a>0:
   print('face up')
else:
   print('face down')
```

Exercise Set 3.8.3

1. 4

   [-15, 7, 0, 11, -5]

2. Modify the program you wrote in Exercise Set 3.8.2 #4 so that it accepts a polynomial input rather than the coefficients. Check to make sure the degree is 2.

```
from sympy import *
x = Symbol('x')
quad = input('quadratic? ')
quad=sympify(quad)
deg = degree(quad)
if deg==2:
```

```
    a,b,c = poly(quad).all_coeffs()
    vx = -b/(2*a)
    vy = quad.subs({x:vx})
    print('vertex: (', vx, ',', vy, ')')
    print('axis of symmetry: x=',vx)
    y_int = quad.subs({x:0})
    print('y-intercept:',y_int)
    if a>0:
        print('face up')
    if a<0:
        print('face down')
else:
    print('That was not a quadratic function')
```

Exercise Set 3.8.4

1. A. $R(p) = -6p^2 + 600p$

   B. The graph is a face-down parabola with y-intercept at 0. The p-intercepts are 0 and 100. The domain is [0, 100].

   C. p = $50 maximizes the revenue.

   D. The maximum revenue is $15,000.

   E. The quantity is 300.

2. A . $R(x) = (-1/3)x^2 + 100x$

   B. The graph is a face-down parabola with y-intercept at 0. The x-intercepts are 0 and 300. The domain is [0, 300].

   C. A quantity of 150 yields the maximum revenue.

   D. The maximum revenue at quantity 150 is $7500.

   E. The company should charge $50 to maximize revenue.

3. A. Revenue $R(x) = x(15 - \frac{x}{1000}) = 15x - \frac{x^2}{1000}$

   Costs $C(x) = 10{,}000 + 8x$

   B. Profit $P(x) = R(x) - C(x) = -\frac{x^2}{1000} + 7x - 10000$

   C. quantity to maximize profit is x = 3500

   D. max profit at x = 3500 is $2250

   E. price to maximize profit = $11.5

F. Break-Even point is at either 2000 or 5000.

4. Write a program that asks user to input demand function x(p). Calculate and output the Revenue function, domain, quantity to maximize revenue, maximum revenue, and price to maximize revenue.

```
from sympy import *
p = Symbol('p')
demand = input('demand function x= ')
d = sympify(demand)
R = expand(p*d)
print('R(p)=',R)
domain = solve(R)
print('domain =',domain)
a,b,c = poly(R).all_coeffs()
p_max = -b/(2*a)
R_max = R.subs({p:p_max})
x_max = d.subs({p:p_max})
print('quantity to maximize revenue is',x_max)
print('maximum revenue is $', R_max)
print('price to maximize revenue is $',p_max)
```

Exercise Set 3.9.1

1. Already factored; degree = 3; y-int 24; x-int -3,2,4; max TP = 2

2. $f(x) = (x + 9)(x – 9)$; degree = 2; y-int -81; x-int ± 9; max TP = 1

3. $f(x) = -3(x + 5)(x – 7)$; degree = 2; y-int 105; x-int -5,7; max TP = 1

4. $f(x) = (x + 2)(x – 2)(x + 1)(x – 1)$; degree = 5; y-int 0; x-int $\pm 1, \pm 2$; max TP = 4

5. $y = (x + 1)(x – 1)(x – 5)$; degree = 3; y-int 5; x-int $\pm 1, 5$; max TP = 2

6. Already factored; degree = 3; y-int -6; x-int $2, -\frac{3}{2}$; max TP = 2

7. $y = (x + 4)(x^2 – 4x + 16)$; degree = 3; y-int 64; x-int -4; max TP = 2

8. $f(x) = (2x – 5)(4x^2 + 10x + 25)$; degree = 3; y-int -125; x-int $\frac{5}{2}$; max TP = 2

9. Already factored; degree = 4; y-int 16; x-int none; max TP = 3

10. An nth-degree function can have up to n x-intercepts and exactly one y-intercept.

11. Every odd degree function has at least one x-intercept.

12. Write a program that asks the user to input a factored polynomial. Output the expanded polynomial, degree, coefficients, and maximum number of turning points.

```
from sympy import *
x=Symbol('x')
```

```
p = input('factored polynomial? ')
p = sympify(p)
p = expand(p)
d = degree(p)
print('expanded polynomial:',p)
print('degree:',d)
coeffs = poly(p).all_coeffs()
print('coefficients:',coeffs)
print('max # of turning points:',d-1)
```

Exercise Set 3.9.2

1. Local max: 0; local min: -1.9; abs min: -4; min degree = 4

2. Abs max: 11.5; local max: 0; local min: -10.8; min degree = 4

3. Abs min: -22.2; local max: 6.3; local min: -2; abs max: 11; min degree = 4

4. A graph with two endpoints (called a closed interval) will always have at least one absolute maximum and one absolute minimum. (This is called the Extreme Value Theorem which you will learn about in Calculus.)

5. Yes

6. Yes

7. Write a program that calculates and outputs the absolute max and absolute min of a set of y-values.

```
y_input = input("list of y-values: ")
y_values = [int(n) for n in y_input.split(‘,’)]
abs_max = max(y_values)
abs_min = min(y_values)
print("absolute max is", abs_max)
print("absolute min is", abs_min)
```