



CONTENTS

1	Images in Python	3
1.1	Adding color	4
1.2	Using an existing image	7
1.3	Faster with NumPy	8
2	Polynomials I: Basics and Operations	11
2.1	Introduction to Polynomials	11
2.2	Adding and Subtracting Polynomials	13
2.2.1	Subtraction	14
2.2.2	Adding Polynomials in Python	15
2.3	Scalar multiplication of polynomials	17
2.4	Multiplying Polynomials	18
2.4.1	Multiplying a monomial and a polynomial	19
2.4.2	Multiplying polynomials	20
2.5	Multiplying Polynomials in Python	22
2.6	Something surprising about lists	25
2.7	Common Polynomial Products	25
2.7.1	Difference of squares	25
2.7.2	Powers of binomials	28
3	Python Lists	31
3.1	Evaluating Polynomials in Python	32
3.2	Walking the list backwards	33
3.3	Plot the polynomial	35

4	Python Classes	39
4.1	Object-Oriented Programming Introduction	39
4.2	Parent classes	41
4.3	Making a Polynomial class	42
5	Polynomials II: Factoring	47
5.1	Factoring Polynomials	47
5.1.1	How to factor polynomials	48
6	Rational Functions	57
6.1	Rational Functions	57
6.2	Partial Fractions	64
7	Polynomials III: Graphing and Interpolation	71
7.1	Graphing Polynomials	71
7.1.1	Leading term in graphing	74
7.2	Interpolating with Polynomials	75
7.2.1	Interpolating polynomials in python	78
A	Answers to Exercises	81
	Index	91

Images in Python

Every photo filter on your phone, every face detector, and every self-driving car's camera does the same basic thing: it treats a picture as a big grid of numbers and does math on them. In this chapter, you will make images from scratch with Python and take real photos apart.

As you saw in the camera chapter, a color image is a grid of pixels, each with a red, green, and blue value from 0 to 255. In Python, we store that as a three-dimensional NumPy array of 8-bit integers (type `uint8`, which holds exactly the values 0 to 255).

If you have an RGB image that is 480 pixels tall and 640 pixels wide, you will need a $480 \times 640 \times 3$ NumPy array.

A separate library, `imageio`, does the file handling. (Install it with `pip3 install imageio` if you need to.) It

- Reads an image file (like JPEG files) and creates a NumPy array.
- Writes a NumPy array to a file in standard image formats

Let us create a simple Python program that creates a file containing an all-black image that is 640 pixels wide and 480 pixels tall. Create a file called `create_image.py`:

```
1 import numpy as np
2 import imageio.v3 as iio
3 import sys
4
5 # Check command-line arguments
6 if len(sys.argv) < 2:
7     print(f"Usage {sys.argv[0]} <outfile>")
8     sys.exit(1)
9
10 # Constants
11 IMAGE_WIDTH = 640
12 IMAGE_HEIGHT = 480
13
14 # Create an array of zeros
15 image = np.zeros((IMAGE_HEIGHT, IMAGE_WIDTH, 3), dtype=np.uint8)
16
17 # Write the array to the file
18 iio.imwrite(sys.argv[1], image)
```

To run this, you supply the name of the file you want to create as a *command-line argument*; Python puts it in `sys.argv[1]`. The extension (like `.png` or `.jpeg`) will tell `imageio` what format you want written. Run it now:

```
1 python3 create_image.py blackness.png
```

Open the image to confirm that it is 640 pixels wide, 480 pixels tall, and completely black.

1.1 Adding color

Worked example. What color is a pixel with the value (255, 255, 0)?

Solution. Full red, full green, and no blue. As you learned in the eye chapter, red and green light together look yellow.

Now, let us walk through the image, pixel-by-pixel, adding some red. We will gradually increase the red from 0 on the left to 255 on the right.

```
1 import numpy as np
2 import imageio.v3 as iio
3 import sys
4
5 # Check command-line arguments
6 if len(sys.argv) < 2:
7     print(f"Usage {sys.argv[0]} <outfile>")
8     sys.exit(1)
9
10 # Constants
11 IMAGE_WIDTH = 640
12 IMAGE_HEIGHT = 480
13
14 # Create an array of zeros
15 image = np.zeros((IMAGE_HEIGHT, IMAGE_WIDTH, 3), dtype=np.uint8)
16
17 for col in range(IMAGE_WIDTH):
18     # Red goes from 0 to 255 (left to right)
19     r = int(col * 255.0 / IMAGE_WIDTH)
20
21     # Update all the pixels in that column
22     for row in range(IMAGE_HEIGHT):
23         # Set the red pixel
24         image[row, col, 0] = r
25
26 # Write the array to the file
27 iio.imwrite(sys.argv[1], image)
28
```

When you run the function to create a new image, it will be a fade from black to red as you move from left to right:



Now, inside the inner loop, update the blue channel so that it goes from zero at the top to 255 at the bottom:

```
1  # Update all the pixels in that column
2  for row in range(IMAGE_HEIGHT):
3
4      # Update the red channel
5      image[row,col,0] = r
6
7      # Blue goes from 0 to 255 (top to bottom)
8      b = int(row * 255.0 / IMAGE_HEIGHT)
9      image[row,col,2] = b
10
11 iio.imwrite(sys.argv[1], image)
```

When you run the program again, you will see the color fades from black to blue as you go down the left side. As you go down the right side, the color fades from red to magenta.

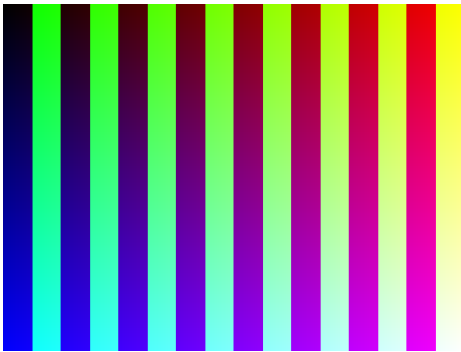


Notice that red and blue with no green looks magenta to your eye.

Next, let us add some stripes of green:

```
1 import numpy as np
2 import imageio.v3 as iio
3 import sys
4
5 # Check command line arguments
6 if len(sys.argv) < 2:
7     print(f"Usage {sys.argv[0]} <outfile>")
8     sys.exit(1)
9
10 # Constants
11 IMAGE_WIDTH = 640
12 IMAGE_HEIGHT = 480
13 STRIPE_WIDTH = 40
14 pattern_width = STRIPE_WIDTH * 2
15
16 # Create an image of all zeros
17 image = np.zeros((IMAGE_HEIGHT, IMAGE_WIDTH, 3), dtype=np.uint8)
18
19 # Step from left to right
20 for col in range(IMAGE_WIDTH):
21
22     # Red goes from 0 to 255 (left to right)
23     r = int(col * 255.0 / IMAGE_WIDTH)
24
25     # Should I add green to this column?
26     should_green = col % pattern_width > STRIPE_WIDTH
27
28     # Update all the pixels in that column
29     for row in range(IMAGE_HEIGHT):
30
31         # Update the red channel
32         image[row,col,0] = r
33
34         # Should I add green to this pixel?
35         if should_green:
36             image[row,col,1] = 255
37
38         # Blue goes from 0 to 255 (top to bottom)
39         b = int(row * 255.0 / IMAGE_HEIGHT)
40         image[row,col,2] = b
41
42 iio.imwrite(sys.argv[1], image)
```

When you run this version, you will see the previous image in every other stripe. In the other half, you will see that green fades to cyan down the left side, and yellow fades to white down the right side.



1.2 Using an existing image

imageio can also be used to read in any common image file format. Let us read in an image and save each of the red, green, and blue channels out as its own image.

Create a new file called `separate_image.py`:

```
1 import numpy as np
2 import imageio.v3 as iio
3 import sys
4 import os
5
6 # Check command line arguments
7 if len(sys.argv) < 2:
8     print(f"Usage {sys.argv[0]} <infile>")
9     sys.exit(1)
10
11 # Read the image
12 path = sys.argv[1]
13 image = iio.imread(path)
14
15 # What is the filename?
16 filename = os.path.basename(path)
17
18 # What is the shape of the array?
19 original_shape = image.shape
20
21 # Log it
22 print(f"Shape of {filename}:{original_shape}")
23
24 # Names of the colors for the filenames
25 colors = ['red', 'green', 'blue']
26
27 # Step through each of the colors
28 for i in range(3):
29
30     # Create a new image
```

```

31 newimage = np.zeros(original_shape, dtype=np.uint8)
32
33 # Copy one channel
34 newimage[:, :, i] = image[:, :, i]
35
36 # Save to a file
37 new_filename = f"{colors[i]}_{filename}"
38 print(f"Writing {new_filename}")
39 iio.imwrite(new_filename, newimage)

```

Now, you can run the program with any common RGB image type:

```

1 python3 separate_image.py dog.jpg

```

This will create three images: red_dog.jpg, green_dog.jpg, and blue_dog.jpg. Look at them side by side. A red sweater is bright in the red image and dark in the others; the sky is brightest in the blue one.

1.3 Faster with NumPy

The loops above visit all 307,200 pixels one at a time, which is slow. NumPy can do the same work on a whole array at once. This one line makes the left-to-right red fade without any loops:

```

1 image[:, :, 0] = np.linspace(0, 255, IMAGE_WIDTH).astype(np.uint8)

```

NumPy copies the row of 640 values down every one of the 480 rows. Real image-processing code is written this way, and it can be hundreds of times faster.

Exercise 1 Grayscale

Write a program that reads a color image and writes a grayscale version. A common formula for how bright a pixel looks is $0.299R + 0.587G + 0.114B$. Why is green weighted most heavily?

Working Space

Answer on Page 81

Exercise 2 White Pixel

What RGB value makes a pixel white?

Working Space

Answer on Page 81

Polynomials I: Basics and Operations

2.1 Introduction to Polynomials

The height of a tossed ball, the area of a garden, the profit of a lemonade stand, the path of a roller coaster, the curve a graphics program draws for the letter S: all of them are described by polynomials. They are the workhorses of algebra, and computers are very good at them. In this chapter, you will learn to add and multiply polynomials by hand and teach Python to do it for you.

Watch Khan Academy's **Polynomials intro** video at <https://youtu.be/Vm7H0VT1Ico>

A *monomial* is the product of a number and a variable raised to a non-negative (but possibly zero) integer power. Here are some examples of monomials:

$$3x^2$$

$$\pi x^2$$

$$7x$$

$$-\frac{2}{3}x^{12}$$

$$-2x^{15}$$

$$(3.33)x^{100}$$

$$3$$

$$0$$

The exponent is called the *degree* of the monomial. For example, $3x^{17}$ has degree 17, $-7x$ has degree 1, and 3.2 has degree 0 (because you can think of it as $(3.2)x^0$).. The exponent cannot be x (or any non-constant variable), as that becomes an exponential function rather than polynomial.¹

The number in the product is called the *coefficient*. Example: $3x^{17}$ has a coefficient of 3, $-2x$ has a coefficient of -2, and $(3.4)x^{1000}$ has a coefficient of 3.4.

A *polynomial* is the sum of one or more monomials. Here are some polynomials:

$$4x^2 + 9x + 3.9$$

$$-2x^{10} + (3.4)x - 45x^{900} - 1$$

$$\pi x^2 + \pi x + \pi$$

¹A quadratic is a polynomial with a maximum degree of 2

3.3

$7x + 2$

$3x^{20}$

We say that each monomial is a *term* of the polynomial.

$x^{-5} + 12$ is *not* a polynomial because the first term has a negative exponent.

$x^2 - 32x^{\frac{1}{2}} + x$ is *not* a polynomial because the second term has a non-integer exponent.

$\frac{x+2}{x^2+x+5}$ is *not* a polynomial because it is not just a sum of monomials.

Exercise 3 Identifying Polynomials

Circle only the polynomials.

Working Space

$$-2x^3 + \frac{1}{2}x + 3.9(4.5)x^2 + \pi x$$

7

$$2x^{-10} + 4x - 1$$

$$x^{\frac{2}{3}}$$

$$3x^{20} + 2x^{19} - 5x^{18}$$

Answer on Page 81

We typically write a polynomial starting at the term with the highest degree and proceed in decreasing order to the term with the lowest degree:

$$2x^9 - 3x^7 + \frac{3}{4}x^3 + x^2 + \pi x - 9.3$$

This is known as the *standard form*. The first term of the standard form is called the *leading term*, and we often call the coefficient of the leading term the *leading coefficient*. We sometimes speak of the degree of the polynomial, which is just the degree of the leading term.

Exercise 4 Standard of a Polynomial

Write $21x^2 - x^3 + \pi - 1000x$ in standard form. What is the degree of this polyno-

Working Space

mial? What is its leading coefficient?

Answer on Page 81

Exercise 5 Evaluate a Polynomial

Let $y = x^3 - 3x^2 + 10x - 12$. What is y when x is 4?

Working Space

Answer on Page 82

We would be remiss in our duties if we did not mention one more thing about polynomials: Mathematicians have defined a polynomial to be a sum of a *finite* number of monomials.

It is certainly possible to have a sum of an infinite number of monomials like this:

$$1 + \frac{1}{2}x + \frac{1}{4}x^2 + \frac{1}{8}x^3 + \frac{1}{16}x^4 + \dots$$

This is an example of an *infinite series*, which we do not consider polynomials. Infinite series are interesting and useful, but we will not discuss them in detail until later in the course.

2.2 Adding and Subtracting Polynomials

Watch Khan Academy's **Adding polynomials** video at <https://youtu.be/ahdKdxsTj8E>

When adding two monomials of the same degree, you sum their coefficients:

$$7x^3 + 4x^3 = 11x^3$$

Using this idea, when adding two polynomials, you convert them into one long polynomial, then simplify by combining terms with the same degree. For example:

$$\begin{aligned}
(10x^3 - 2x + 13) + (-5x^2 + 7x - 12) \\
&= 10x^3 + (-2)x + 13 + (-5)x^2 + 7x + (-12) \\
&= 10x^3 + (-5)x^2 + (-2 + 7)x + (13 - 12) \\
&= 10x^3 - 5x^2 + 5x + 1
\end{aligned}$$

Exercise 6 Adding Polynomials Practice

Add the following polynomials:

Working Space

1. $2x^3 - 5x^2 + 3x - 9$ and $x^3 - 2x^2 - 2x - 9$

2. $3x^5 - 5x^3 + 3x^2 - x - 3$ and $2x^4 - 2x^3 - 2x^2 + x - 9$

Answer on Page 82

Notice that in the second question, the degree 1 term disappears completely: $(-x) + x = 0$

One more tricky thing can happen — sometimes the coefficients do not add nicely. For example:

$$\pi x^2 - 3x^2 = (\pi - 3)x^2$$

That is as far as you can simplify it.

2.2.1 Subtraction

Now, watch Khan Academy's **Subtracting polynomials** at <https://youtu.be/5ZdxnFspyP8>.

When subtracting one polynomial from the other, it is a lot like adding two polynomials. The difference: when making the two polynomials into one long polynomial, we multiply each monomial that is being subtracted by -1. For example:

$$\begin{aligned}
 (2x^2 - 3x + 9) - (5x^2 - 7x + 4) \\
 &= 2x^2 + (-3)x + 9 + (-5)x^2 + 7x + (-4) \\
 &= (2 - 5)x^2 + (-3 + 7)x + (9 - 4) \\
 &= -3x^2 + 4x + 5
 \end{aligned}$$

Exercise 7 Subtracting Polynomials Practice

Subtract the following polynomials:

Working Space

1. $(2x^3 - 5x^2 + 3x - 9) - (x^3 - 2x^2 - 2x - 9)$
2. $(3x^5 - 5x^3 + 3x^2 - x - 3) - (2x^4 - 2x^3 - 2x^2 + x - 9)$

Answer on Page 82

2.2.2 Adding Polynomials in Python

In Python, we represent a polynomial as a list of coefficients. The first entry is the constant term (the coefficient of x^0), the next is the coefficient of x^1 , and so on, so the entry at index i goes with x^i . The last entry is the leading coefficient. So $-5x^3 + 3x^2 - 4x + 9$ becomes $[9, -4, 3, -5]$, and $2x^3 + 4x^2 - 9$ becomes $[-9, 0, 4, 2]$, where the 0 marks the missing x term.

To add the two polynomials, we add the entries with the same index: $[9 + (-9), -4 + 0, 3 + 4, -5 + 2] = [0, -4, 7, -3]$, which is $-3x^3 + 7x^2 - 4x$.

Create a file called `add_polynomials.py`, and type in the following:

```

1 def add_polynomials(a, b):
2     degree_of_result = len(a)
3     result = []
4     for i in range(degree_of_result):
5         coefficient_a = a[i]
```

```

6         coefficient_b = b[i]
7         result.append(coefficient_a + coefficient_b)
8     return result
9
10 polynomial1 = [9.0, -4.0, 3.0, -5.0]
11 polynomial2 = [-9.0, 0.0, 4.0, 2.0]
12 polynomial3 = add_polynomials(polynomial1, polynomial2)
13
14 print('Sum =', polynomial3)

```

Run the program.

Unfortunately, this code only works if the polynomials are the same length. For example, try making polynomial1 have a larger degree than polynomial2:

```

1 # x**4 - 5x**3 + 3x**2 - 4x + 9
2 polynomial1 = [9.0, -4.0, 3.0, -5.0, 1.0]
3
4 # 2x**3 + 4x**2 - 9
5 polynomial2 = [-9.0, 0.0, 4.0, 2.0]
6 polynomial3 = add_polynomials(polynomial1, polynomial2)
7 print('Sum =', polynomial3)

```

See the problem?

Exercise 8 Dealing with polynomials of different degrees

Working Space

Can you fix the function `add_polynomials` to handle polynomials of different degrees?

Here is a hint: In Python, there is a `max` function that returns the largest of the numbers it is passed.

```

1 biggest = max(5,7)

```

Here `biggest` would be set to 7.

Here is another hint: If you have an array `mylist`, `i`, a non-negative integer, is

only a legit index if `i < len(mylist)`.

[Answer on Page 82](#)

2.3 Scalar multiplication of polynomials

If you multiply a polynomial with a number, the distributive property applies:

$$(3.1)(2x^2 + 3x + 1) = (6.2)x^2 + (9.3)x + 3.1$$

(When we are talking about things that are more complicated than a number, we use the word *scalar* to mean “Just a number”. This is the product of a scalar and a polynomial.)

In `add_polynomials.py`, add a function to that multiplies a scalar and a polynomial:

```
1 def scalar_polynomial_multiply(s, pn):
2     result = []
3     for coefficient in pn:
4         result.append(s * coefficient)
5     return result
```

Somewhere near the end of the program, test this function:

```
1 polynomial4 = scalar_polynomial_multiply(5.0, polynomial1)
2 print('Scalar product =', polynomial_to_string(polynomial4))
```

Exercise 9 Subtract polynomials in Python

Now, implement a function that does subtraction using `scalar_polynomial_multiply` and `add_polynomials`.

[Working Space](#)

It should look like this:

```
1 def subtract_polynomial(a, b):
2     ...Your code here...
3
4 polynomial5 = [9.0, -4.0, 3.0, -5.0]
5 polynomial6 = [-9.0, 0.0, 4.0, 2.0, 1.0]
```

```

6 polynomial7 =
  ↪ subtract_polynomial(polynomial5,
  ↪ polynomial6)
7 print('Difference =',
  ↪ polynomial_to_string(polynomial7))

```

Answer on Page 82

2.4 Multiplying Polynomials

Watch Khan Academy's **Multiplying monomials** at <https://www.youtube.com/watch?v=7EplN4Ch98Q>.

To review, when you multiply two monomials, you take the product of their coefficients and the sum of their degrees:

$$(2x^6)(5x^3) = (2)(5)(x^6)(x^3) = 10x^9$$

If you have a product of more than two monomials, multiply *all* the coefficients and sum *all* the exponents:

$$(3x^2)(2x^3)(4x) = (3)(2)(4)(x^2)(x^3)(x^1) = 24x^6$$

Exercise 10 Multiplying monomials

Multiply these monomials:

Working Space

1. $(3x^2)(5x^3)$

2. $(2x)(4x^9)$

3. $(-5.5x^2)(2x^3)$

4. $(\pi)(-2x^5)$

5. $(2x)(3x^2)(5x^7)$

Answer on Page 83

2.4.1 Multiplying a monomial and a polynomial

Watch Khan Academy's **Multiplying monomials by polynomials** at <https://youtu.be/pD2-H15ucNE>. When multiplying a monomial and a polynomial, you use the distributive property. After that, it is just multiplying several pairs of monomials:

$$\begin{aligned}(3x^2)(4x^3 - 2x^2 + 3x - 7) \\&= (3x^2)(4x^3) + (3x^2)(-2x^2) + (3x^2)(3x) + (3x^2)(-7) \\&= 12x^5 - 6x^4 + 9x^3 - 21x^2\end{aligned}$$

Exercise 11 Multiplying a monomial and a polynomial

Multiply:

Working Space

1. $(3x^2)(5x^3 - 2x + 3)$

2. $(2x)(4x^9 - 1)$

3. $(-5.5x^2)(2x^3 + 4x^2 + 6)$

4. $(\pi)(-2x^5 + 3x^4 + x)$

5. $(2x)(3x^2)(5x^7 + 2x)$

Answer on Page 83

2.4.2 Multiplying polynomials

Watch Khan Academy's **Multiplying binomials by polynomials** video at https://youtu.be/D6mivA_8L8U

When you are multiplying two polynomials, you will use the distributive property several times to make it one long polynomial. You will then combine the terms with the same degree. For example,

$$\begin{aligned}(2x^2 - 3)(5x^2 + 2x - 7) &= (2x^2)(5x^2 + 2x - 7) + (-3)(5x^2 + 2x - 7) \\ &= (2x^2)(5x^2) + (2x^2)(2x) + (2x^2)(-7) + (-3)(5x^2) + (-3)(2x) + (-3)(-7) \\ &= 10x^4 + 4x^3 + (-14)x^2 + (-15)x^2 + (-6)x + 21 \\ &= 10x^4 + 4x^3 - 29x^2 - 6x + 21\end{aligned}$$

One common form that you will see is multiplying two binomials together:

$$(2x + 7)(5x + 3) = (2x)(5x + 3) + (7)(5x + 3) = (2x)(5x) + (7)(5x) + (2x)(3) + (7)(3)$$

Notice the product has become the sum of four parts: the firsts, the inners, the outers, and the lasts. People sometimes use the mnemonic FOIL to remember this pattern, but there is a general rule that works for all product of polynomials, not just binomials: Every term in the first will be multiplied by every term in the second, and then just add them together.

So, for example, if you have a polynomial s with three terms and you multiply it by a polynomial t with five terms, you will get a sum of 15 terms — each term is a product of two monomials, one from s and one from t . (Of course, several of those terms might have the same degree, so they will be combined together when you simplify. Thus you

typically end up with a polynomial with less than 15 terms.)

Using this rule, here is how you would multiply $2x^2 - 3x + 1$ and $5x^2 + 2x - 7$:

$$\begin{aligned}
 (2x^2 - 3x + 1)(5x^2 + 2x - 7) &= \begin{array}{ccccccc} (2x^2)(5x^2) & + & (2x^2)(2x) & + & (2x^2)(-7) & + & \\ (-3x)(5x^2) & + & (-3x)(2x) & + & (-3x)(-7) & + & \\ (1)(5x^2) & + & (1)(2x) & + & (1)(-7) & + & \end{array} \\
 &= 10x^4 + 4x^3 + (-14)x^2 + (-15)x^3 + (-6)x^2 + 21x + 5x^2 + 2x + (-7) \\
 &= 10x^4 + (4 - 15)x^3 + (-14 - 6 + 5)x^2 + (21 + 2)x + (-7) \\
 &= 10x^4 - 11x^3 - 15x^2 + 23x - 7
 \end{aligned}$$

Note that the product (before combining terms with the same degree) has $3 \times 3 = 9$ terms — every possible combination of a term from the first polynomial and a term from the second polynomial.

One common source of error is losing track of the negative signs. You will need to be really careful. We have found that it helps to use $+$ between all terms, and use negative coefficients to express subtraction. For example, if the problem says $4x^2 - 5x - 3$, you should work with that as $4x^2 + (-5)x + (-3)$.

Exercise 12 Multiplying polynomials

Multiply the following pairs of polynomials:

1. $2x + 1$ and $3x - 2$

2. $-3x^2 + 5$ and $4x - 2$

3. $-2x - 1$ and $-3x - \pi$

4. $-2x^5 + 5x$ and $3x^5 + 2x$

Working Space

Answer on Page 83

Exercise 13 Observations

Let us say you have two polynomials, p_1 and p_2 . p_1 has degree 23. p_2 has degree 12. What is the degree of their product?

Working Space

Answer on Page 84

2.5 Multiplying Polynomials in Python

At this point, you have created a nice toolbox of functions for dealing with lists of coefficients as polynomials. Create a file called `poly.py` and copy the following functions into it:

- `evaluate_polynomial`
- `polynomial_to_string`
- `add_polynomials`
- `scalar_polynomial_multiply`
- `subtract_polynomial`

Now, create another file in the same directory called `test.py`. Type this into that file:

```
1 import poly
2
3 polynomial_a = [9.0, -4.0, 3.0, -5.0]
4 print('Polynomial A =', poly.polynomial_to_string(polynomial_a))
5
6 polynomial_b = [-9.0, 0.0, 4.0, 2.0, 1.0]
7 print('Polynomial B =', poly.polynomial_to_string(polynomial_b))
8
9 # Evaluation
10 value_of_b = poly.evaluate_polynomial(polynomial_b, 3)
11 print('Polynomial B at 3 =', value_of_b)
12
13 # Adding
14 a_plus_b = poly.add_polynomials(polynomial_a, polynomial_b)
15 print('A + B =', poly.polynomial_to_string(a_plus_b))
16
17 # Scalar multiplication
```

```

18 b_scalar = poly.scalar_polynomial_multiply(-3.2, polynomial_b)
19 print('-3.2 * Polynomial B =', poly.polynomial_to_string(b_scalar))
20
21 # Subtraction
22 a_minus_b = poly.subtract_polynomial(polynomial_a, polynomial_b)
23 print('A - B =', poly.polynomial_to_string(a_minus_b))

```

When you run it, you should get the following:

```

1 Polynomial A = -5.0x^3 + 3.0x^2 + -4.0x + 9.0
2 Polynomial B = 1.0x^4 + 2.0x^3 + 4.0x^2 + -9.0
3 Polynomial B at 3 = 162.0
4 A + B = 1.0x^4 + -3.0x^3 + 7.0x^2 + -4.0x
5 -3.2 * Polynomial B = -3.2x^4 + -6.4x^3 + -12.8x^2 + 28.8
6 A - B = -1.0x^4 + -7.0x^3 + -1.0x^2 + -4.0x + 18.0

```

You are now ready to implement the multiplication of polynomials. The function will look like this:

```

1 def multiply_polynomials(a, b):
2     ...Your code here...

```

It will return a list of coefficients.

Earlier in this chapter, you found that multiplying a degree 23 polynomial by a degree 12 polynomial gives a polynomial of degree $23 + 12 = 35$.

In our implementation, a polynomial of degree 23 is held in a list of length 24.

In Python, we will be trying to multiply a polynomial *a* and a polynomial *b* represented as lists. What is the degree of that product?

```

1 result_degree = (len(a) - 1) + (len(b) - 1)

```

Now, we need to create an array of zeros that is one longer than that. Here is a cute Python trick: If you have a list, you can replicate it using the `*` operator.

```

1 a = [5,7]
2 b = a * 4
3 print(b)
4 # [5, 7, 5, 7, 5, 7, 5, 7]

```

Here is how you will get a list of zeros:

```
1 result = [0.0] * (result_degree + 1)
```

We will step through `a`, getting the index and value of each entry. You can do this in one line using `enumerate`:

```
1 for a_degree, a_coefficient in enumerate(a):
```

For each of those, we will step through the entire `b` polynomial. As you multiply together each term, you will add it to the appropriate coefficient of the result.

Here is the whole function:

```
1 def multiply_polynomials(a, b):
2     # What is the degree of the resulting polynomial?
3     result_degree = (len(a) - 1) + (len(b) - 1)
4
5     # Make a list of zeros to hold the coefficients
6     result = [0.0] * (result_degree + 1)
7
8     # Iterate over the indices and values of a
9     for a_degree, a_coefficient in enumerate(a):
10
11         # Iterate over the indices and values of b
12         for b_degree, b_coefficient in enumerate(b):
13
14             # Calculate the resulting monomial
15             coefficient = a_coefficient * b_coefficient
16             degree = a_degree + b_degree
17
18             # Add it to the right bucket
19             result[degree] = result[degree] + coefficient
20
21     return result
```

Take a long look at that function. When you understand it, type it into `poly.py`.

In `test.py`, try out the new function:

```
1 # Multiplication
2 a_times_b = poly.multiply_polynomials(polynomial_a, polynomial_b)
3 print('A x B =', poly.polynomial_to_string(a_times_b))
```

You should see:

```
1 A x B = -5.0x^7 + -7.0x^6 + -18.0x^5 + 13.0x^4 + 47.0x^3 + 9.0x^2 + 36.0x + -81.0
```

This is an example of a *nested loop*. The outer loop steps through the polynomial a. For each step it takes, the inner loop steps through the entire polynomial b.

2.6 Something surprising about lists

You can imagine that you might want to create two very similar polynomials. Let us say polynomial c is $x^2 + 2x + 1$ and polynomial d is $x^2 - 2x + 1$. You might think you are very clever to just alter that degree 1 coefficient like this:

```
1 c = [1.0, 2.0, 1.0]
2 d = c
3 d[1] = -2.0
```

If you printed out c, you would get $[1.0, -2.0, 1.0]$. Why? You assigned two variables (c and d) to the *same list*. So, when you use one reference (d) to change the list, you see the change if you look at the list from either reference. Both names point to the same list in memory, so a change made through one name is visible through the other.

To create two separate lists, you would need to explicitly make a copy:

```
1 c = [1.0, 2.0, 1.0]
2 d = c.copy()
3 d[1] = -2.0
```

2.7 Common Polynomial Products

In math and physics, you will run into certain kinds of polynomials over and over again. In this chapter, we will cover some patterns that you will want to be able to recognize as you move through the sequence!

2.7.1 Difference of squares

To start off, watch *Polynomial special products: difference of squares* from Khan Academy at <https://youtu.be/uNweU6I4Icw>.

If you are asked what $(3x-7)(3x+7)$ is, you would use the distributive property to expand

that to $(3x)(3x) + (3x)(7) + (-7)(3x) + (-7)(7)$. Two of the terms cancel each other, so this is $(3x)^2 - (7)^2$. This would simplify to $9x^2 - 49$

You will see this pattern often. Anytime you see $(a + b)(a - b)$, you should immediately recognize it equals $a^2 - b^2$. (Note that the order does not matter: $(a - b)(a + b)$ also equals $a^2 - b^2$.)

Working the other way is important too. Any time you see $a^2 - b^2$, that you should recognize that you can change that into the product $(a + b)(a - b)$. Making something into a product like this is known as *factoring*. You probably have done prime factorization of numbers like $42 = 2 \times 3 \times 7$. In the next couple of chapters, you will learn to factorize polynomials.

Exercise 14 Difference of Squares

Simplify the following products:

Working Space

1. $(2x - 3)(2x + 3)$
2. $(7 + 5x^3)(7 - 5x^3)$
3. $(x - a)(x + a)$
4. $(3 - \pi)(3 + \pi)$
5. $(-4x^3 + 10)(-4x^3 - 10)$
6. $(x + \sqrt{7})(x - \sqrt{7})$ Factor the following polynomials:
7. $x^2 - 9$
8. $49 - 16x^6$
9. $\pi^2 - 25x^8$
10. $x^2 - 5$

Answer on Page 84

We are often interested in the roots of a polynomial. That is, we want to know “For what values of x does the polynomial evaluate to zero?” For example, when you deal with falling bodies, the first question you might ask would be “How many seconds before the hammer hits the ground?” Once you have factored a polynomial into binomials, you can easily find the roots.

For example, what are the roots of $x^2 - 5$? You just factored it into $(x + \sqrt{5})(x - \sqrt{5})$. This

product is zero if and only if one of the factors is zero. The first factor is only zero when x is $-\sqrt{5}$. The second factor is zero only when x is $\sqrt{5}$. Those are the only two roots of this polynomial.

Let us check that result. $\sqrt{5}$ is a little more than 2.2. Using your Python code, you can graph the polynomial:

```

1  import poly
2  import matplotlib.pyplot as plt
3
4  # x**2 - 5
5  pn = [-5.0, 0.0, 1.0]
6
7  # These lists will hold our x and y values
8  x_list = []
9  y_list = []
10
11 # Start at x=-3
12 current_x = -3.0
13
14 # End at x=3.0
15 while current_x < 3.0:
16     current_y = poly.evaluate_polynomial(pn, current_x)
17
18     # Add x and y to respective lists
19     x_list.append(current_x)
20     y_list.append(current_y)
21
22     # Move x forward
23     current_x += 0.1
24
25 # Plot the curve
26 plt.plot(x_list, y_list)
27 plt.grid(True)
28
29 #Additional graph customization (make axes visible, axes legend)
30 plt.axhline(0, color='black', linewidth=1) # x-axis
31 plt.axvline(0, color='black', linewidth=1) # y-axis
32 plt.title("Graph of y = x2 - 5")
33 plt.xlabel("x")
34 plt.ylabel("y")
35
36 plt.show()

```

You should get a plot like Figure 2.1.

It does, indeed, seem to cross the x-axis near -2.2 and 2.2.

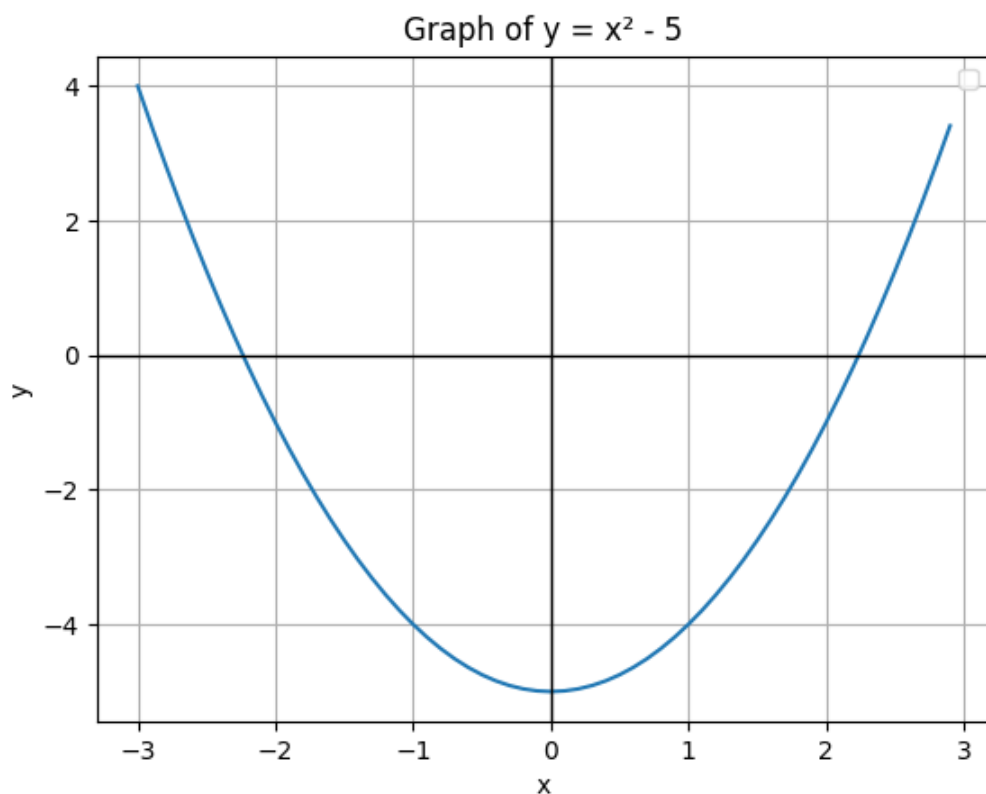


Figure 2.1: A graph of $x^2 - 5$ showing the roots on the x-axis.

2.7.2 Powers of binomials

You can raise whole polynomials to exponents. Raising a polynomial to an exponent means taking the entire expression and multiplying it by itself repeatedly according to the value of the exponent. For example,

$$\begin{aligned}(3x^3 + 5)^2 &= (3x^3 + 5)(3x^3 + 5) \\ &= 9x^6 + 15x^3 + 15x^3 + 25 = 9x^6 + 30x^3 + 25\end{aligned}$$

$(3x^3 + 5)^2$ is a binomial that has been raised to an exponent of 2. A polynomial with two terms is called a *binomial*. $5x^9 - 2x^4$, for example, is a binomial. In this section, we are going to develop some handy techniques for raising a binomial to some power.

Looking at the previous example, you can see that for any monomials a and b , $(a + b)^2 = a^2 + 2ab + b^2$. This is referred to as a perfect square binomial. So, for example, $(7x^3 + \pi)^2 = 49x^6 + 14\pi x^3 + \pi^2$

Exercise 15 **Squaring binomials**

Simplify the following

Working Space

1. $(x + 1)^2$
2. $(3x^5 + 5)^2$
3. $(x^3 - 1)^2$
4. $(x - \sqrt{7})^2$

Answer on Page 84

What about $(x + 2)^3$? You can do it as two separate multiplications:

$$\begin{aligned}
 (x + 2)^3 &= (x + 2)(x + 2)(x + 2) \\
 &= (x + 2)(x^2 + 4x + 4) = x^3 + 4x^2 + 4x + 2x^2 + 8x + 8 \\
 &= x^3 + 6x^2 + 12x + 8
 \end{aligned}$$

In general, we can say that for any monomials a and b , $(a + b)^3 = a^3 + 3a^2b + 3ab^2 + b^3$.

What about higher powers? $(a + b)^4$, for example? You could use the distributive property four times, but it starts to get pretty tedious.

Here is a trick. This is known as *Pascal's triangle*

$$\begin{array}{ccccccc}
 & & & & 1 & & & \\
 & & & 1 & & 1 & & \\
 & & 1 & & 2 & & 1 & \\
 & 1 & & 3 & & 3 & & 1 \\
 & 1 & & 4 & & 6 & & 4 & & 1 \\
 & 1 & & 5 & & 10 & & 10 & & 5 & & 1 \\
 & 1 & & 6 & & 15 & & 20 & & 15 & & 6 & & 1 \\
 & 1 & & 7 & & 21 & & 35 & & 35 & & 21 & & 7 & & 1 \\
 & & & & & & & \dots & & & & & & & &
 \end{array}$$

Each entry is the sum of the two above it.

The coefficients of each term are given by the entries in Pascal's triangle:

$$(a + b)^4 = 1a^4 + 4a^3b + 6a^2b^2 + 4ab^3 + 1b^4$$

Each coefficient is called a *binomial coefficient*, written $\binom{n}{k}$, where n is the exponent and k is the term number (starting with 0). It is read “ n choose k .” The rule that $(a + b)^n$ expands with these coefficients is the *binomial theorem*. For example, in $(a + b)^4$, the coefficient of the third term ($6a^2b^2$) is $\binom{n}{k} = \binom{4}{2} = 6$. We will heavily discuss this in a future chapter on combinatorics.

Exercise 16 Using Pascal’s Triangle

1. What is $(x + \pi)^5$?

Working Space

Answer on Page 85

Exercise 17 Coding Pascal’s Triangle

Write a Python script that finds the row of Pascal’s Triangle at a given index n .

Hint: it does not need to be recursive.

Working Space

Answer on Page 85

CHAPTER 3

Python Lists

Watch CS Dojo's **Introduction to Lists in Python** video at <https://www.youtube.com/watch?v=tw7ror9x32s>

Your playlist, your contacts, the scores in a video game's leaderboard: programs are full of lists. In this chapter, you will review Python lists and then use them to store, evaluate, print, and plot polynomials.

To review, a Python list is an indexed collection. The indices start at zero. You can create a list using square brackets.

You are now going to write a program that makes a list of strings. Type this code into a file called `faves.py`:

```
1 favorites = ["Raindrops", "Whiskers", "Kettles", "Mittens"]
2 favorites.append("Packages")
3 print("Here are all my favorites:", favorites)
4 print("My most favorite thing is", favorites[0])
5 print("My second most favorite is", favorites[1])
6 number_of_faves = len(favorites)
7 print("Number of things I like:", number_of_faves)
8
9 for i in range(number_of_faves):
10     print(i, ": I like", favorites[i])
```

Run it:

```
1 $ python3 faves.py
2 Here are all my favorites: ['Raindrops', 'Whiskers', 'Kettles', 'Mittens',
3 ↵ 'Packages']
4 My most favorite thing is Raindrops
5 My second most favorite is Whiskers
6 Number of things I like: 5
7 0 : I like Raindrops
8 1 : I like Whiskers
9 2 : I like Kettles
10 3 : I like Mittens
    4 : I like Packages
```

After you have run the code, study it until the output makes sense.

Exercise 18 Assign into list

Before you list the items, replace “Mit-tens” with “Gloves”.

Working Space

Answer on Page 85

3.1 Evaluating Polynomials in Python

Worked example: evaluating a polynomial. Let $p(x) = 2x^2 + 3x + 1$. Then $p(2) = 2(4) + 3(2) + 1 = 15$.

First, before you go any further, you need to know that raising a number to a power is done with `**` in Python. For example, to get 5^2 , you would write `5**2`.

Back to polynomials: if you had a polynomial like $2x^3 - 9x + 12$, you could write it like this: $12x^0 + (-9)x^1 + 0x^2 + 2x^3$. We could use this representation to keep a polynomial in a Python list. We would simply store all the coefficients in order:

```
1 pn1 = [12, -9, 0, 2]
```

In the list, the index of each coefficient would correspond to the degree of that monomial. For example, in the list, 2 is at index 3, so that entry represents $2x^3$.

In the last chapter, you evaluated the polynomial $x^3 - 3x^2 + 10x - 12$ at $x = 4$. Now you will write code that does that evaluation. Create a file called `polynomials.py` and type in the following:

```
1 def evaluate_polynomial(pn, x):
2     sum = 0.0
3     for degree in range(len(pn)):
4         coefficient = pn[degree]
5         term_value = coefficient * x ** degree
6         sum = sum + term_value
7     return sum
8
9 pn1 = [-12.0, 10.0, -3.0, 1.0]
10 y = evaluate_polynomial(pn1, 4.0)
11 print("Polynomial 1: When x is 4.0, y is", y)
```

Run it. It should evaluate to 44.0.

3.2 Walking the list backwards

Now you are going to make a function that makes a pretty string to represent your polynomial. Here is how it will be used:

```
1 def polynomial_to_string(pn):
2     ...Your Code Here...
3
4 pn_test = [-12.0, 10.0, 0.0, 1.0]
5 print(polynomial_to_string(pn_test))
```

This would output:

```
1 1.0x^3 + 10.0x + -12.0
```

This is not as simple as you might hope. In particular:

- You should skip the terms with a coefficient of zero
- The term of degree 1 has an x , but no exponent
- The term of degree 0 has neither an x nor an exponent
- Standard form demands that you list the terms in the reverse order from that of your coefficients list. You will need to walk the list from last to first.

Add this function to your `polynomials.py` file after your `evaluate_polynomial` function:

```
1 def polynomial_to_string(pn):
2
3     # Make a list of the monomial strings
4     monomial_strings = []
5
6     # Start at the term with the largest degree
7     degree = len(pn) - 1
8
9     # Go through the list backwards stop after constant term
10    while degree >= 0:
11        coefficient = pn[degree]
12
13        # Skip any term with a zero coefficient
14        if coefficient != 0.0:
15
```

```

16         # Describe the monomial
17         if degree == 0:
18             monomial_string = "{}".format(coefficient)
19         elif degree == 1:
20             monomial_string = "{}x".format(coefficient)
21         else:
22             monomial_string = "{}x^{}".format(coefficient, degree)
23
24         # Add it to the list
25         monomial_strings.append(monomial_string)
26
27         # Move to the previous term
28         degree = degree - 1
29
30     # Deal with the zero polynomial
31     if len(monomial_strings) == 0:
32         monomial_strings.append("0.0")
33
34     # Make a string that joins the terms with a plus sign
35     return " + ".join(monomial_strings)

```

Note that in a list n items, the indices go from 0 to $n - 1$. When we are walking the list backwards, we start at `len(pn) - 1` and stop at zero.

Look over the code and google the functions you are not familiar with. For example, if you want to know about the join function, search for “python join”. (In short, `" + ".join(["a", "b", "c"])` makes the string `"a + b + c"`.)

Now, change your code to use the new function:

```

1 pn1 = [-12.0, 10.0, -3.0, 1.0]
2 y = evaluate_polynomial(pn1, 4.0)
3 print("y =", polynomial_to_string(pn1))
4 print("    When x is 4.0, y is", y)

```

Run the program. Does the function work?

Exercise 19 Evaluate Polynomials

Using the function that you just wrote, add a few lines of code to `polynomials.py` to evaluate the following polynomials:

- Find $4x^4 - 7x^3 - 2x^2 + 5x + 2.5$ at $x = 8.5$. It should be 16481.875

Working Space

- Find $5x^5 - 9$ at $x = 2.0$. It should be 151.0

Answer on Page 85

3.3 Plot the polynomial

We can evaluate a polynomial at many points and plot them on a graph. You are going to write the code to do this. Create a new file called `plot_polynomial.py`. Copy your `evaluate_polynomial` function into the new file.

Add a line at the beginning of the program that imports the plotting library `matplotlib`:

```
1 import matplotlib.pyplot as plt
```

After the `evaluate_polynomial` function:

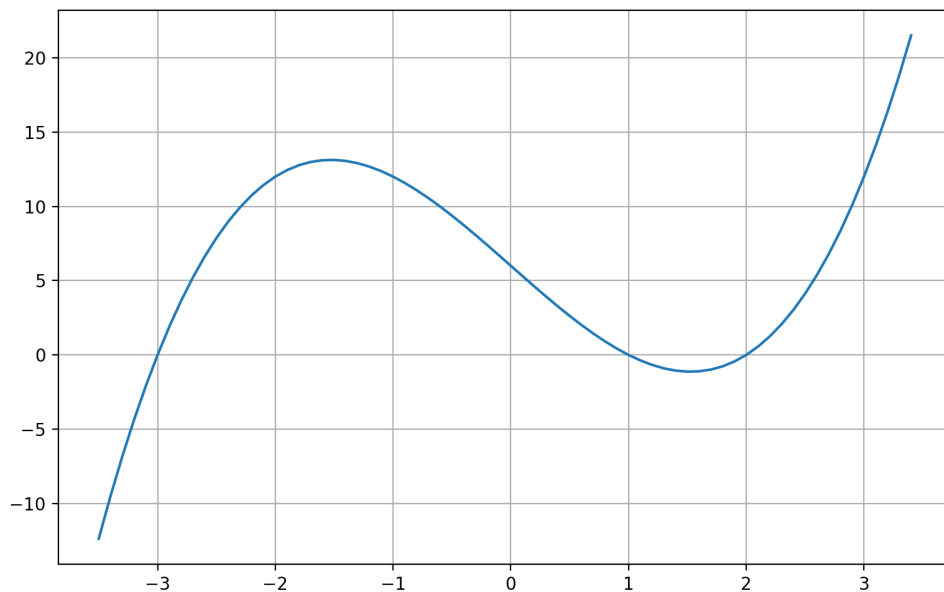
- Create a list with polynomial coefficients.
- Create two empty arrays, one for x values and one for y values.
- Fill the x array with values from -3.5 to 3.5. Evaluate the polynomial at each of these points; put those values in the y array.
- Plot them

Like this:

```
1 # x**3 - 7x + 6
2 pn = [6.0, -7.0, 0.0, 1.0]
3
4 # These lists will hold our x and y values
5 x_list = []
6 y_list = []
7
8 # Start at x=-3.5
9 current_x = -3.5
10
11 # End at x=3.5
12 while current_x <= 3.5:
13     current_y = evaluate_polynomial(pn, current_x)
14
```

```
15     # Add x and y to respective lists
16     x_list.append(current_x)
17     y_list.append(current_y)
18
19     # Move x forward
20     current_x += 0.1
21
22 # Plot the curve
23 plt.plot(x_list, y_list)
24 plt.grid(True)
25 plt.show()
```

You should get a beautiful plot like this:



If you received an error that the matplotlib was not found, use pip to install it:

```
1 $ pip3 install matplotlib
```

Exercise 20 Observations

Where does your polynomial cross the y-axis? Looking at the polynomial $x^3 - 7x + 6$, could you have guessed that value?

Working Space

Where does your polynomial cross the x-axis? The places where a polynomial crosses the x-axis are called *its roots*. Later in the course, you will learn techniques for finding the roots of a polynomial.

Answer on Page 85

Exercise 21 Cubic Polynomial

Let $p(x) = x^3 - x$. Find $p(3)$.

Working Space

Answer on Page 86

Python Classes

4.1 Object-Oriented Programming Introduction

In a video game, every character has a name, health, and position, and can move, jump, and attack. Programmers do not write separate variables for each of a thousand characters. They define a *class*, a blueprint for a kind of thing, and then make as many objects from it as they need. This style is called *object-oriented programming*, and it is how most large programs are organized.

Imagine you want to keep track of several dogs in Python. Without classes, it gets messy fast:

```
1  dog1name = "Teddy"
2  dog1age = 6
3  dog1sound = "woof"
4
5  dog2name = "Fluffers"
6  dog2age = 2
7  dog2sound = "bark"
8
9  dog3name = "Bella"
10 dog3age = 3
11 dog3sound = "grr"
12
13 print(f"{dog1name} is {dog1age} and says {dog1sound}!")
14 print(f"{dog2name} is {dog2age} and says {dog2sound}!")
15 print(f"{dog3name} is {dog3age} and says {dog3sound}!")
16
17
```

Instead, we can use classes, which are a way to create your own datatype. Classes can contain custom methods that either return, print, or calculate different values for you, and they can contain custom variables referred to as attributes.

```
1  class Dog:
2      """A simple model of a dog."""
3      # constructor - which creates a new model of a dog.
4      def __init__(self, name: str, age: int, sound: str):
5          self.name = name
6          self.age = age
7          self.sound = sound
```

```

8     # method speak
9     def speak(self) -> None:
10         """Print a sentence describing the dog."""
11         print(f"{self.name} is {self.age} and says {self.sound}!")
12
13 # create dog objects called "instances" with the given variables
14 dog1 = Dog("Teddy", 6, "woof")
15 dog2 = Dog("Fluffers", 2, "bark")
16 dog3 = Dog("Bella", 3, "grr")
17
18 # call the behavior on each object
19 dog1.speak()
20 dog2.speak()
21 dog3.speak()
22

```

Worked example. Write a class for rectangles that can compute its own area.

Solution.

```

1 class Rectangle:
2     def __init__(self, width, height):
3         self.width = width
4         self.height = height
5
6     def area(self):
7         return self.width * self.height
8
9 r = Rectangle(3, 4)
10 print(r.area())

```

The output is 12. The `__init__` method sets the attributes, and `area` uses them.

Now let us talk about classes in the context of polynomials!

The built-in types, such as strings, have functions associated with them. So, for example, if you needed a string converted to uppercase, you would call its `upper()` function: -

```

1 my_string = "houston, we have a problem!"
2 louder_string = my_string.upper()

```

This would set `louder_string` to "HOUSTON, WE HAVE A PROBLEM!" When a function is associated with a datatype like this, it is called a *method*. A datatype with methods is known as a *class*. Creating a new version of a class in a variable is called an *instance*. For example, in the example, we would say "my_string is an instance of the class str. str has a method called upper"

The function `type` will tell you the type of any data:

```
1 print(type(my_string))
```

This will output

```
1 <class 'str'>
```

A class can also define operators. `+`, for example, is redefined by `str` to concatenate strings together:

```
1 long_string = "I saw " + "15 people"
```

4.2 Parent classes

Classes can have classes they inherit from (called “parent classes”) or classes that inherit from them (called “child classes”). Parent classes (ie ‘Animal’) give a subclass (or child) different attributes or methods. Let us check out an example:

```
1 class Animal:
2     """Generic animal base-class."""
3
4     def __init__(self, name: str, age: int) -> None:
5         self.name = name
6         self.age = age
7
8     def describe(self) -> None:
9         """Print a basic description common to all animals."""
10        print(f"{self.name} is {self.age} years old.")
11
12 class Dog(Animal):
13     """Dog inherits name and age from Animal, adds its own sound."""
14
15     def __init__(self, name: str, age: int, sound: str) -> None:
16         super().__init__(name, age) # initialize the Animal part
17         self.sound = sound
18
19     def speak(self) -> None:
20         """Dog-specific implementation of speak()."""
21         print(f"{self.name} is {self.age} and says {self.sound}!")
22
23
24 # demonstration
25 dogs = [
```

```

26     Dog("Teddy", 6, "woof"),
27     Dog("Fluffers", 2, "bark"),
28     Dog("Bella", 3, "grr")
29 ]
30
31 for dog in dogs:
32     dog.describe()    # common behavior from Animal
33     dog.speak()       # overridden behavior in Dog
34

```

Here, `Animal` is the parent class of `Dog`. A `Dog` automatically gets `describe` from `Animal`, and `super().__init__` lets it reuse the parent's setup code. A `Cat` class could inherit from `Animal` the same way.

4.3 Making a Polynomial class

You have written a bunch of useful Python functions for polynomials, and every one has “polynomial” in its name, like `add_polynomials`. Would it not be more elegant to have a `Polynomial` class, so you could just write `a + b` and `a * b`? Then you could use your polynomials like this:

```

1  a = Polynomial([9.0, 0.0, 2.3])
2  b = Polynomial([-2.0, 4.5, 0.0, 2.1])
3
4  print(a, "plus", b, "is", a+b)
5  print(a, "times", b, "is", a*b)
6  print(a, "times", 3, "is", a*3)
7  print(a, "minus", b, "is", a-b)
8
9  c = b.derivative()
10
11 print("Derivative of", b, "is", c)

```

And it would output:

```

1  2.30x^2 + 9.00 plus 2.10x^3 + 4.50x + -2.00 is 2.10x^3 + 2.30x^2 + 4.50x + 7.00
2  2.30x^2 + 9.00 times 2.10x^3 + 4.50x + -2.00 is 4.83x^5 + 29.25x^3 + -4.60x^2 +
   ↪ 40.50x + -18.00
3  2.30x^2 + 9.00 times 3 is 6.90x^2 + 27.00
4  2.30x^2 + 9.00 minus 2.10x^3 + 4.50x + -2.00 is -2.10x^3 + 2.30x^2 + -4.50x + 11.00
5  Derivative of 2.10x^3 + 4.50x + -2.00 is 6.30x^2 + 4.50

```

The derivative method is a sneak preview of calculus. The derivative of a polynomial is another polynomial that gives the slope of its graph at every point. The rule is simple:

each term cx^n becomes ncx^{n-1} . So the derivative of $2.1x^3 + 4.5x - 2$ is $6.3x^2 + 4.5$. You will learn why this works in the calculus volumes.

How does Python know what $+$ means for a Polynomial? Methods with special names surrounded by double underscores, called *magic methods*, tell it. If a class defines `__add__`, then `a + b` calls `a.__add__(b)`. Likewise, `__mul__` handles `*`, `__sub__` handles `-`, `__repr__` decides how print shows the object, and `__call__` lets you use an object like a function, as in `c(3)`.

Create a file for your class definition called `Polynomial.py`. Enter the following:

```

1 class Polynomial:
2     def __init__(self, coeffs):
3         self.coefficients = coeffs.copy()
4
5     def __repr__(self):
6         # Make a list of the monomial strings
7         monomial_strings = []
8
9         # For standard form we start at the largest degree
10        degree = len(self.coefficients) - 1
11
12        # Go through the list backwards
13        while degree >= 0:
14            coefficient = self.coefficients[degree]
15
16            if coefficient != 0.0:
17                # Describe the monomial
18                if degree == 0:
19                    monomial_string = "{:.2f}".format(coefficient)
20                elif degree == 1:
21                    monomial_string = "{:.2f}x".format(coefficient)
22                else:
23                    monomial_string = "{:.2f}x^{degree}".format(coefficient, degree)
24
25                # Add it to the list
26                monomial_strings.append(monomial_string)
27
28                # Move to the previous term
29                degree = degree - 1
30
31        # Deal with the zero polynomial
32        if len(monomial_strings) == 0:
33            monomial_strings.append("0.0")
34
35        # Separate the terms with a plus sign
36        return " + ".join(monomial_strings)
37
38    def __call__(self, x):
39        sum = 0.0
40        for degree, coefficient in enumerate(self.coefficients):

```

```

41         sum = sum + coefficient * x ** degree
42     return sum
43
44     def __add__(self, b):
45         result_length = max(len(self.coefficients), len(b.coefficients))
46         result = []
47         for i in range(result_length):
48             if i < len(self.coefficients):
49                 coefficient_a = self.coefficients[i]
50             else:
51                 coefficient_a = 0.0
52
53             if i < len(b.coefficients):
54                 coefficient_b = b.coefficients[i]
55             else:
56                 coefficient_b = 0.0
57             result.append(coefficient_a + coefficient_b)
58
59         return Polynomial(result)
60
61     def __mul__(self, other):
62
63         # Not a polynomial?
64         if not isinstance(other, Polynomial):
65             # Try to make it a constant polynomial
66             other = Polynomial([other])
67
68         # What is the degree of the resulting polynomial?
69         result_degree = (len(self.coefficients) - 1) + (len(other.coefficients) -
70             ↪ 1)
71
72         # Make a list of zeros to hold the coefficients
73         result = [0.0] * (result_degree + 1)
74
75         # Iterate over the indices and values of a
76         for a_degree, a_coefficient in enumerate(self.coefficients):
77
78             # Iterate over the indices and values of b
79             for b_degree, b_coefficient in enumerate(other.coefficients):
80
81                 # Calculate the resulting monomial
82                 coefficient = a_coefficient * b_coefficient
83                 degree = a_degree + b_degree
84
85                 # Add it to the right bucket
86                 result[degree] = result[degree] + coefficient
87
88         return Polynomial(result)
89
90     __rmul__ = __mul__
91
92     def __sub__(self, other):
93         return self + other * -1.0

```

```
93
94     def derivative(self):
95
96         # We can ignore the constant term (skip the first coefficient)
97         current_degree = 1
98         result = []
99
100        # Differentiate each monomial
101        while current_degree < len(self.coefficients):
102            coefficient = self.coefficients[current_degree]
103            result.append(coefficient * current_degree)
104            current_degree = current_degree + 1
105
106        # No terms? Make it the zero polynomial
107        if len(result) == 0:
108            result.append(0.0)
109
110        return Polynomial(result)
```

Create a second file called `test_polynomial.py` to test it:

```
1  from Polynomial import Polynomial
2
3  a = Polynomial([9.0, 0.0, 2.3])
4  b = Polynomial([-2.0, 4.5, 0.0, 2.1])
5
6  print(a, "plus", b, "is", a+b)
7  print(a, "times", b, "is", a*b)
8  print(a, "times", 3, "is", a*3)
9  print(a, "minus", b, "is", a-b)
10
11  c = b.derivative()
12
13  print("Derivative of", b, "is", c)
14
15  slope = c(3)
16  print("Value of the derivative at 3 is", slope)
17
```

Run the test code:

```
1  python3 test_polynomial.py
```

You should see the output shown at the start of this section, followed by one more line:

```
1  Value of the derivative at 3 is 61.2
```

That is $6.3(3^2) + 4.5 = 61.2$: the slope of b 's graph at $x = 3$.

Exercise 22 **Evaluate it**

Using your `Polynomial` class, create $p(x) = x^3 - 7x + 6$ and print $p(-3)$, $p(1)$, and $p(2)$. What do the results tell you?

Working Space

Answer on Page 86

Exercise 23 **Second Rectangle**

Create a `Rectangle` with width 5 and height 2, and print its area.

Working Space

Answer on Page 86

Polynomials II: Factoring

5.1 Factoring Polynomials

Multiplying polynomials is mechanical: expand and collect. Going backward, from a polynomial to its factors, takes cleverness, and it pays off. Once a polynomial is factored, you can read its roots right off the page: where the ball lands, where the profit breaks even, where the curve crosses the axis.

We factor a polynomial into two or more polynomials of lower degree. For example, let us say that you wanted to factor $5x^3 - 45x$. You would note that you can factor out $5x$ from every term. Thus,

$$5x^3 - 45x = (5x)(x^2 - 9)$$

You might notice that the second factor looks like the difference of squares, so

$$5x^3 - 45x = (5x)(x + 3)(x - 3)$$

That is as far as we can factorize this polynomial.

Why do we care? The factors make it easy to find the roots of the polynomial. This polynomial evaluates to zero if and only if at least one of the factors is zero. Here, we see that

- The factor $(5x)$ is zero when x is zero.
- The factor $(x + 3)$ is zero when x is -3 .
- The factor $(x - 3)$ is zero when x is 3 .

So, looking at the factorization, you can see that $5x^3 - 45x$ is zero when x is 0 , -3 , or 3 .

This is a graph of that polynomial with its roots circled:

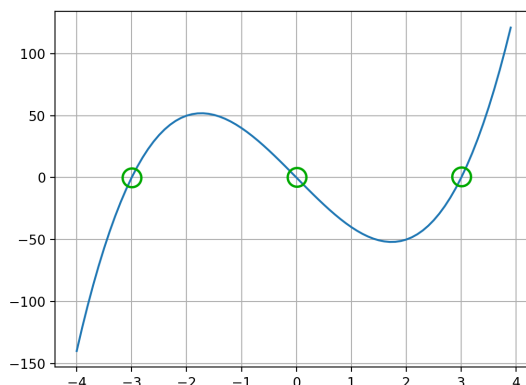


Figure 5.1: Factoring for roots makes it easier to find the roots rather than in expanded form.

5.1.1 How to factor polynomials

The first step when you are trying to factor a polynomial is to find the greatest common divisor for all the terms, then pull that out. In this case, the greatest common divisor will also be a monomial. Its degree is the least of the degrees of the terms, its coefficient will be the greatest common divisor of the coefficients of the terms.

For example, what can you pull out of this polynomial?

$$12x^{100} + 30x^{31} + 42x^{17}$$

The greatest common divisor of the coefficients (12, 30, and 42) is 6. The least of the degrees of terms (100, 31, and 17) is 17. So, you can pull out $6x^{17}$:

$$12x^{100} + 30x^{31} + 42x^{17} = (6x^{17})(2x^{83} + 5x^{14} + 7)$$

Exercise 24 Factoring out the GCD monomial

Factor out the greatest common divisor monomial from each polynomial.

Working Space

1. $18x^3 - 15x$
2. $4x^3 - 12x^2 + 16x$

3. $18x^{50} - 24x^{20} + 30x^8$

Answer on Page 86

So, now you have the product of a monomial and a polynomial. If you are lucky, the polynomial part looks familiar, like the difference of squares or a row from Pascal's triangle.

Often, you are trying to factor a quadratic like $x^2 + 5x + 6$ in a pair of binomials. In this case, the result would be $(x + 3)(x + 2)$. Let us check that:

$$(x + 3)(x + 2) = (x)(x) + (3)(x) + (2)(x) + (3)(2) = x^2 + 5x + 6$$

Notice that 3 and 2 multiply to 6 and add to 5. If you were trying to factor $x^2 + 5x + 6$, you would ask yourself, "What are two numbers that when multiplied equal 6 and when added equal 5?" And you might guess wrong a couple of times. For example, you might say to yourself, "Well, 6 times 1 is 6. Maybe those work. But 6 and 1 add 7. So those do not work."

Solving these sorts of problems are like solving a Sudoku puzzle. You try things and realize they are wrong, so you backtrack and try something else.

The numbers are sometimes negative. For example, $x^2 + 3x - 10$ factors into $(x + 5)(x - 2)$.

Exercise 25 Factoring quadratics

Factor each quadratic into a pair of binomials.

Working Space

1. $x^2 + 14x + 33$

2. $x^2 - 16x + 60$

3. $x^2 + 3x - 54$

4. $x^2 - 3x - 10$

Answer on Page 86

Computing in Python

Python can check a factorization for you. Start by writing the original polynomial and its factored form as two Python functions:

```

1 def p(x):
2     return 5*x**3 - 45*x
3
4 def f(x):
5     return (5*x)*(x + 3)*(x - 3)

```

In Python, `**` means “raised to the power of,” and every multiplication needs a `*`. You have to write `5*x`, since `5x` is an error.

First, confirm the roots we read off the factors:

```

1 for x in [0, -3, 3, 1]:
2     print(x, p(x))

```

```

1 0 0
2 -3 0
3 3 0
4 1 -40

```

The values 0, -3, and 3 all give zero, so they are roots. The value 1 gives -40, so it is not a root.

Next, check that `p` and `f` are the same polynomial. If the factorization is correct, the two functions give the same output for every `x`:

```

1 for x in [0, 1, 2, 3]:
2     print(x, p(x), f(x), p(x) == f(x))

```

```

1 0 0 0 True
2 1 -40 -40 True
3 2 -50 -50 True
4 3 0 0 True

```

Each line shows `x`, the value of the original, the value of the factored form, and whether they match. Every line says **True**.

How many values of `x` do you need to check? Here is a useful fact: if two polynomials of degree at most `n` give the same value at `n + 1` different values of `x`, they are the same polynomial. Our polynomial has degree 3, so four matching values are enough.

Matching at the roots alone can hide a mistake. Suppose you forgot the 5 when you pulled out 5x:

```

1 def g(x):
2     return x*(x + 3)*(x - 3)
3
4 for x in [0, -3, 3, 1]:
5     print(x, p(x), g(x), p(x) == g(x))

```

```

1 0 0 0 True
2 -3 0 0 True
3 3 0 0 True
4 1 -40 -8 False

```

The wrong factorization has the same roots, so it passes at 0, -3, and 3. The value $x = 1$ catches the mistake.

Exercise 26 Predict the output

Without running the code, predict what this prints. Which of the two values of x is a root?

Working Space

```

1 def q(x):
2     return x**2 + 3*x - 4
3
4 print(q(2))
5 print(q(-4))

```

Answer on Page 87

Exercise 27 Spot the sign error

A student claims that $x^2 + 3x - 10 = (x - 5)(x + 2)$ and runs this check:

Working Space

```

1 def a(x):
2     return x**2 + 3*x - 10
3
4 def b(x):

```

```

5     return (x - 5)*(x + 2)
6
7     for x in [0, 1, 2]:
8         print(x, a(x), b(x), a(x) == b(x))

```

The output is:

```

1     0 -10 -10 True
2     1 -6 -12 False
3     2 0 -12 False

```

The first line says **True**. Why is the factorization still wrong? What is the correct factorization?

Answer on Page 87

Exercise 28 Spot the bug

A student wants to check that $x^2 + 5x + 6 = (x + 3)(x + 2)$, which we already know is correct, and writes:

Working Space

```

1     def c(x):
2         return x^2 + 5*x + 6
3
4     def d(x):
5         return (x + 3)*(x + 2)
6
7     for x in [0, 1, 2]:
8         print(x, c(x), d(x), c(x) == d(x))

```

The output is:

```

1     0 8 6 False
2     1 12 12 True
3     2 16 20 False

```

The factorization is correct, so what

went wrong?

[Answer on Page 87](#)

Exercise 29 Factor, check, and find the roots

Factor $2x^3 - 6x^2 - 8x$ completely. Write the original polynomial and your factored form as two Python functions, and check that they match at enough values of x . Then use your factors to list the roots, and confirm that each one gives zero.

[Working Space](#)

[Answer on Page 87](#)

This last part is optional. It uses SymPy, a Python library for algebra. SymPy is a separate install, so you may need to run `python3 -m pip install sympy` first.

```
1 from sympy import symbols, factor, expand
2
3 x = symbols('x')
4 print(factor(5*x**3 - 45*x))
5 print(expand(5*x*(x + 3)*(x - 3)))
```

```
1 5*x*(x - 3)*(x + 3)
2 5*x**3 - 45*x
```

The line `x = symbols('x')` makes x a *symbolic variable*: a letter that stands for any number, which is how x works in algebra. Earlier in this section, x always held one number at a time. Here it holds no number at all.

The function `factor` factors a polynomial, and `expand` multiplies the factors back out. SymPy lists $(x - 3)$ before $(x + 3)$; the order of the factors does not change the product.

SymPy also tells you when a polynomial cannot be factored using integers:

```
1 print(factor(x**2 + 1))
```

```
1 x**2 + 1
```

SymPy returns the polynomial unchanged, so x^2+1 has no factors with integer coefficients.

SymPy can factor every polynomial in this chapter in one line, which means it can also skip the thinking this chapter is teaching. Use it after you factor by hand, to check your answer.

If you get stuck on a problem, open `factor_hint.py`, type your polynomial on the line that starts with `polynomial =`, and run the file. It first shows a hint: the polynomial with its greatest common factor pulled out, found with SymPy's `factor_terms`. Try to finish the problem from there. When you press Enter, the file shows the full factorization.

Exercise 30 Predict what SymPy prints

Without running the code, predict the output.

Working Space

```
1 from sympy import symbols, factor
2
3 x = symbols('x')
4 print(factor(x**2 - 9))
```

Answer on Page 88

Exercise 31 Factor by hand, then check with SymPy

Factor $3x^2+12x-15$ completely by hand. Then use `expand` to multiply your factors back out, and use `factor` to compare your answer with SymPy's.

Working Space

Answer on Page 88

Factoring is also the key to dividing polynomials, which is the subject of the next chapter, Rational Functions.

Rational Functions

If a pizza costs \$12 to share and n friends split it, each pays $\frac{12}{n}$ dollars. As more friends join, the cost per person drops toward zero but never reaches it. Functions like this, one polynomial divided by another, are called *rational functions*. In the Polynomials II: Factoring chapter, you learned to factor polynomials. Here you will use factoring to simplify rational functions, find their asymptotes, and break them into simpler pieces.

6.1 Rational Functions

We have discussed addition, subtraction, and multiplication of polynomials. What about division?

A quotient of polynomials is called a rational expression. When the polynomials are factored and the stars align, we can simplify the rational expression to a single polynomial, just like we might reduce a fraction to lowest terms.

Example

$$\begin{aligned}\frac{(x+1)(x+5)}{x+5} &= (x+1) \cdot \frac{x+5}{x+5} \\ &= x+1 \quad (x \neq -5)\end{aligned}\tag{6.1}$$

1

What if the polynomials are not factored? Factor them first.

Example

$$\frac{x^2 + 6x + 5}{x + 5} = \frac{(x+1)(x+5)}{x+5}$$

and simplify as in the previous example.

Now, let us consider a rational expression which can be simplified to a single polynomial — but in the denominator.

¹We must restrict $x \neq -5$, since substituting $x = -5$ would produce division by zero, making the original expression undefined. Canceling the $(x+5)$ factor is only valid when $x \neq -5$, so the simplified form $x+1$ and the original expression are equal everywhere *except* at $x = -5$.

Example

$$\begin{aligned}
 \frac{x+5}{x^2+6x+5} &= \frac{x+5}{(x+1)(x+5)} \\
 &= \frac{1}{x+1} \cdot \frac{x+5}{x+5} \\
 &= \frac{1}{x+1}
 \end{aligned} \tag{6.2}$$

Consider this expression as a function: $f(x) = \frac{1}{x+1}$. As you might have guessed, this is called a rational function. We did not bother looking at the result of the previous example as a function, because we already know that function type: it is a line with slope 1 and y-intercept 1. However, this rational function is another animal entirely. Let us examine our first rational function with a familiar concept: the y-intercept.

y-intercept: $f(0) = \frac{1}{0+1} = \frac{1}{1} = 1$. The graph contains the point $(0, 1)$.

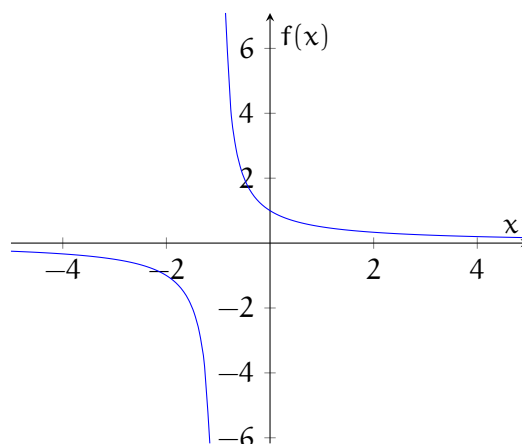
Does f have an x-intercept? That would be an x -value where $f(x) = 0$. But a fraction equals 0 only when its numerator equals 0; since the numerator of this expression is always 1, f has no x-intercept.

Knowing the y-intercept, and that there is no x-intercept, is a comforting start. But things get weird when we consider a concept that has previously seemed quite simple: domain. Recall that the domain of a function is the set of all values which can be used as inputs. In this case, the domain includes all real numbers, with one exception. The number -1 is not a valid input because $f(-1) = \frac{1}{-1+1} = \frac{1}{0}$, which is undefined. So, we say that the domain is all real numbers except -1 . This means the graph contains a point corresponding to every x -value except -1 .

There is no point at $x = -1$, but there is a point at every other x -value, such as, for example, -1.1 , or -0.99999 . So, what is happening near $x = -1$?

x	-1.1	-1.01	-1.001	-0.999	-0.99	-0.9
$f(x)$	-10	-100	-1000	1000	100	10

The function is going haywire. As we choose x -values closer and closer to -1 , the resulting function values are larger and larger in magnitude. Also, they are negative on one side, but positive on the other. So, how does a graph go from y -values of -10 , to -100 , to -1000 , all in a space of less than 0.1 on the x -axis? Then, from there, suddenly to big positive numbers on the other side of $x = -1$? All without ever crossing the x -axis (since there is no x-intercept)? Let us look at the graph.


 Figure 6.1: Graph of $f(x) = \frac{1}{x+1}$

We can see the y-intercept we found above. We can also see that the graph has no x-intercept, as expected. The phenomenon occurring at $x = -1$ is called a vertical asymptote. One other interesting feature of this graph is how it hugs the x-axis toward the left and right edges of the window. This makes the line $y = 0$ (the x-axis) a horizontal asymptote for this function. We can see why this is happening numerically by considering what happens for x-values far from 0. In this function, the result is a fraction with a numerator of 1 and a denominator that is large in size: a fraction that is close to 0.

x	-1000	-100	-10	10	100	1000
f(x)	-0.001	-0.01	-0.1	0.1	0.01	0.001

Let us examine another rational function. Begin by factoring to see if the function can be simplified.

$$g(x) = \frac{x^2 - 3x + 2}{x^2 - 4x + 3} = \frac{(x-1)(x-2)}{(x-1)(x-3)}$$

Consider the domain of g before continuing. Which values of x are valid inputs? Since substituting $x = 1$ or $x = 3$ would result in division by 0, these are not valid inputs. The domain of g is all real numbers except 1 and 3.

Now, for any x -value except 1, $\frac{x-1}{x-1} = 1$. This means that, for all x -values but 1, we can cancel those factors, leaving $g(x) = \frac{x-2}{x-3}$. (We will talk more about what is happening at $x = 1$ in a moment.)

This function has both x- and y-intercepts: y-intercept: $g(0) = \frac{0-2}{0-3} = \frac{2}{3}$. The graph contains the point $(0, \frac{2}{3})$. x-intercept: $g(x) = 0$ where the numerator equals 0 and the denominator does not equal 0. Since $x - 2 = 0$ when $x = 2$, the x-intercept is 2 and the graph contains the point $(2, 0)$.

The graph of g has a vertical asymptote at any x -value where substitution would result in dividing a nonzero number by zero. Thus, g has a vertical asymptote at $x = 3$.

Does g have a horizontal asymptote? Let us see what happens when we substitute x -values far from 0.

x	-1000	-100	-10	10	100	1000
$g(x)$	0.999	0.990	0.923	1.143	1.010	1.001

As we move further away from the y -axis, the y -values become closer to 1. The horizontal asymptote describes the end behavior of the function, or what the graph looks like far from the y -axis. In this case, if we ignore the portion close to the y -axis, the graph begins to look like the line $y = 1$, making this the horizontal asymptote of g .

So, what is happening at $x = 1$? The value is not in the domain of the function, but there is no vertical asymptote there. That is because substituting any other value for x , even values very close to 1, into $\frac{(x-1)(x-2)}{(x-1)(x-3)}$ gives the exact same number as substituting into $\frac{x-2}{x-3}$. So, there is a hole in the graph at $x = 1$, but nothing strange is happening on either side of 1. (Depending on the graphing software, the hole may not be visible.)

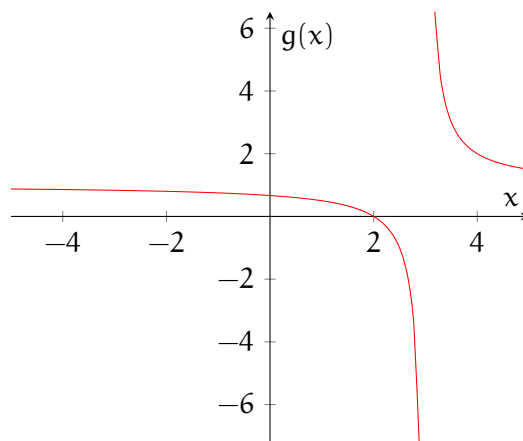


Figure 6.2: Graph of $g(x) = \frac{x^2-3x+2}{x^2-4x+3}$

Exercise 32 Rational Functions Practice 1

Determine the x - and y -intercepts and horizontal and vertical asymptotes of the rational function:

Working Space

1. $\frac{2x+5}{x+4}$

Answer on Page 88

Exercise 33

[This question was originally presented in the no-calculator section of the 2012 AP Calculus BC exam.] The line $y = 5$ is a horizontal asymptote of which of the following functions? Explain.

A. $y = \frac{\sin 5x}{x}$

B. $y = 5x$

C. $y = \frac{1}{x-5}$

D. $y = \frac{5x}{1-x}$

E. $y = \frac{20x^2-x}{1+4x^2}$

Working Space

Answer on Page 89

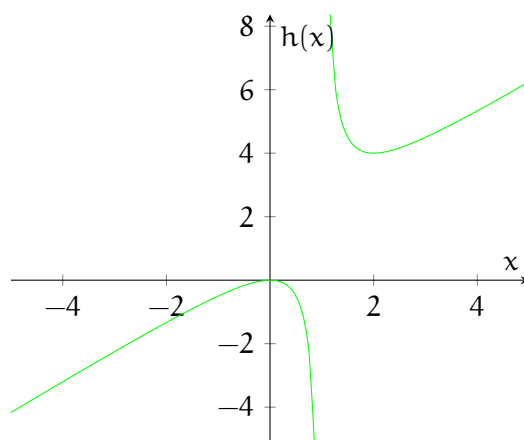
In those examples, common factors cancel, leaving one polynomial. Of course, there is no guarantee that any two polynomials will have common factors, or even be factorable at all. Now, we consider an example that cannot be simplified. We will focus on just the asymptotes here.

$$h(x) = \frac{x^2}{x-1}$$

We see that the x -value 1 gives division of a non-zero number by zero, giving a vertical asymptote at $x = 1$. How about a horizontal asymptote? We examine values of h for values of x far from 0.

x	-1000	-100	-10	10	100	1000
$h(x)$	-999	-99	-9	11	101	1001

Rather than seeing function values leveling off as in the previous examples, we see function values that grow in size along with x . The function h has no horizontal asymptote. Let us examine the graph:

Figure 6.3: Graph of $h(x) = \frac{x^2}{x-1}$

This function exhibits a different type of end behavior: that of a line with slope 1. To see that, cover up the portion of the graph near the y -axis and focus on the left and right. The rather dull and time-consuming technique of polynomial long division can be used to rewrite the function as a quotient and a remainder. We encourage you to watch the Khan Academy video on the topic, but for now, let us instead use our knowledge of factoring techniques and a clever little trick.

$$\begin{aligned}
 h(x) &= \frac{x^2}{x-1} \\
 &= \frac{x^2 - 1 + 1}{x-1} \\
 &= \frac{x^2 - 1}{x-1} + \frac{1}{x-1} \\
 &= \frac{(x-1)(x+1)}{x-1} + \frac{1}{x-1} \\
 &= x + 1 + \frac{1}{x-1}
 \end{aligned} \tag{6.3}$$

We obtain a quotient of $x + 1$ and a remainder of 1. It is the quotient that determines the end behavior of the graph. Why? Substituting x -values far from zero makes the remainder term very small, since it becomes a fraction with a large denominator but a numerator of only 1. So for x -values far from zero, the y -value is x plus 1 plus a very small number (so small that we can justifiably ignore it). This means that far from the y -axis, the function acts like the quotient: the line $y = x + 1$. We call this line an oblique asymptote. See below how the graph of $h(x)$ hugs that line.

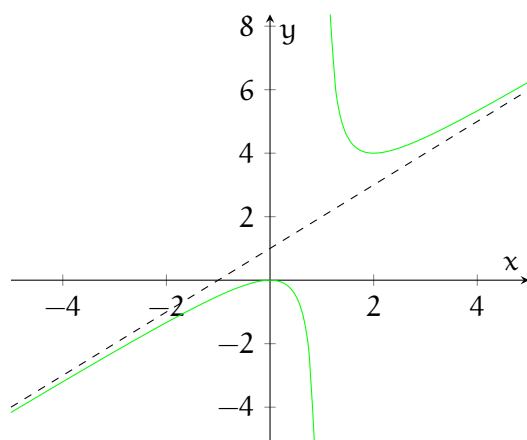


Figure 6.4: Graph of $h(x) = \frac{x^2}{x-1}$ and its oblique asymptote $y = x + 1$

Exercise 34 Rational Functions Practice 2

Factor and simplify the rational function, then determine any holes and vertical and oblique asymptotes of the rational function.

1. $\frac{x^3 + 2x^2}{x^2 + x}$

Working Space

Answer on Page 89

We have seen lines act as end behaviors. Are there other possibilities? Sure! Here is an example with parabolic end behavior.

$$k(x) = \frac{x^3}{x-2}$$

We use our add-subtract trick to reveal the quotient, which describes the end behavior. This time we subtract and add 8, because $x^3 - 8$ is a difference of cubes: $x^3 - 8 = (x - 2)(x^2 + 2x + 4)$. (Check it by multiplying.)

$$\begin{aligned}
 k(x) &= \frac{x^3}{x-2} \\
 &= \frac{x^3 - 8 + 8}{x-2} \\
 &= \frac{x^3 - 8}{x-2} + \frac{8}{x-2} \\
 &= \frac{(x-2)(x^2 + 2x + 4)}{x-2} + \frac{8}{x-2} \\
 &= x^2 + 2x + 4 + \frac{8}{x-2}
 \end{aligned} \tag{6.4}$$

The quotient, $x^2 + 2x + 4$, should describe the end behavior. We confirm by graphing both k and the quotient - the parabolic asymptote.

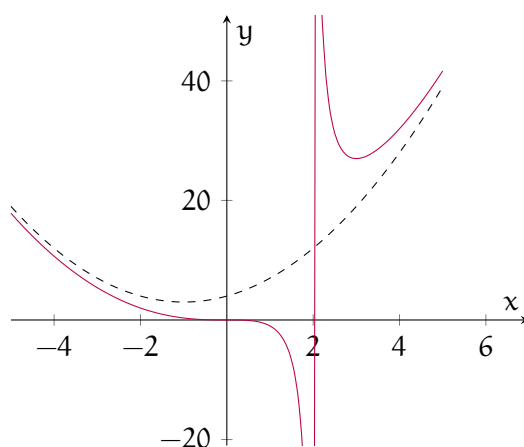


Figure 6.5: Graph of $k(x) = \frac{x^3}{x-2}$ and its parabolic asymptote $y = x^2 + 2x + 4$

6.2 Partial Fractions

How can you add fractions with different denominators, like $\frac{1}{x} + \frac{2}{x+3}$? You would need to make the denominators the same; after that, you could just add the numerators. You achieve this by multiplying the numerator and denominator of each fraction by the denominator of the other fraction:

$$\frac{1}{x} + \frac{2}{x+3} = \frac{1}{x} \left(\frac{x+3}{x+3} \right) + \frac{2}{x+3} \left(\frac{x}{x} \right)$$

Recall that when the numerator and denominator of a fraction are the same, the fraction is equal to one. So, we are not changing the *value* of each fraction, since we are just

multiplying by one. Continuing, we can perform the multiplication and see that:

$$\begin{aligned}\frac{1}{x} + \frac{2}{x+3} &= \frac{x+3}{x(x+3)} + \frac{2x}{x(x+3)} \\ &= \frac{(x+3) + 2x}{x(x+3)} = \frac{3x+3}{x^2+3x} = \frac{3(x+1)}{x^2+3x}\end{aligned}$$

The inverse of this process is called **partial fraction decomposition** (or partial fraction expansion). This method has applications in many fields, but we will find it most useful as a tool to evaluate integrals in a later chapter.

Let $g(x)$ be a rational function such that

$$g(x) = \frac{P(x)}{Q(x)}$$

Where $P(x)$ and $Q(x)$ are polynomials. If $g(x)$ is proper (that is, the degree of P is less than the degree of $Q(x)$) then we can express $g(x)$ as the sum of simpler rational fractions. If $g(x)$ is improper (that is, the degree of P is greater than or equal to the degree of Q), then we must first perform long division to obtain a remainder, $R(x)$, where the degree of R is less than the degree of Q :

$$g(x) = \frac{P(x)}{Q(x)} = S(x) + \frac{R(x)}{Q(x)}$$

Improper fractions

How can we rewrite $\frac{x^3+x}{x-1}$ as a polynomial plus a proper fraction? Using long division (see figure 6.6), just like dividing whole numbers:

$$\frac{x^3+x}{x-1} = x^2 + x + 2 + \frac{2}{x-1}$$

$$\begin{array}{r}
 x^2 + x + 2 \\
 x - 1 \overline{) x^3 + 0x^2 + x} \\
 \underline{-(x^3 - x^2)} \quad \downarrow \\
 x^2 + x \\
 \underline{-(x^2 - x)} \\
 2x \\
 \underline{-(2x - 2)} \\
 2
 \end{array}$$

Figure 6.6: Evaluating $(x^3 + x) \div (x - 1)$ with the long division method

When you start with an improper fraction, use long division to reduce it to a term plus a proper fraction, then use the methods outlined below to further manipulate the proper fraction.

Exercise 35

Use long division to reduce the following improper rational functions to a term plus a proper rational fraction.

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

Working Space

Answer on Page 89

Proper fractions

When the degree of the numerator is less than the degree of the denominator, there are several cases, depending on how the denominator factors.

No repeated linear factors

In the first case, the denominator, $Q(x)$ is composed of distinct linear factors. In this case, we can say that $Q(x) = (a_1x + b_1)(a_2x + b_2) \cdots (a_nx + b_n)$, where no factor is repeated (including constant multiples). Then, there exists $A, B, C, \dots$, such that:

$$\frac{P(x)}{Q(x)} = \frac{A}{a_1x + b_1} + \frac{B}{a_2x + b_2} + \cdots$$

Let us see an example of this by decomposing $\frac{4x^2 - 7x - 12}{x(x+2)(x-3)}$. We start by defining A, B , and C , such that:

$$\frac{4x^2 - 7x - 12}{x(x+2)(x-3)} = \frac{A}{x} + \frac{B}{x+2} + \frac{C}{x-3}$$

Multiplying both sides by $x(x+2)(x-3)$ we get:

$$4x^2 - 7x - 12 = A(x+2)(x-3) + B(x)(x-3) + C(x)(x+2)$$

We have 3 unknowns and only one equation! Do not worry — remember this equation is true for all x , so we can choose a convenient value of x to isolate each unknown in turn. Starting, let $x = 0$. Then:

$$\begin{aligned} 4(0)^2 - 7(0) - 12 &= A(0+2)(0-3) + B(0)(0-3) + C(0)(0+2) \\ -12 &= A(2)(-3) + 0 + 0 \end{aligned}$$

Notice that the B and C disappear, and we can solve for A :

$$A = \frac{-12}{-6} = 2$$

We can solve for B by setting $x = -2$ and for C by setting $x = 3$ (notice, we have used all three zeroes of the denominator polynomial):

$$\begin{aligned} 4(-2)^2 - 7(-2) - 12 &= A(-2+2)(-2-3) + B(-2)(-2-3) + C(-2)(-2+2) \\ 4(4) + 14 - 12 &= 0 + B(-2)(-5) + 0 \\ 18 &= 10B \\ B &= \frac{9}{5} \end{aligned}$$

and

$$\begin{aligned} 4(3)^2 - 7(3) - 12 &= A(3+2)(3-3) + B(3)(3-3) + C(3)(3+2) \\ 4(9) - 21 - 12 &= 0 + 0 + C(3)(5) \\ 36 - 33 &= 15C \\ C &= \frac{1}{5} \end{aligned}$$

We can then decompose our original fraction:

$$\frac{4x^2 - 7x - 12}{x(x+2)(x-3)} = \frac{2}{x} + \frac{9}{5(x+2)} + \frac{1}{5(x-3)}$$

You can check your answer by cross-multiplying and adding. You should get the same rational function back.

Repeated linear factors

The second case is if $Q(x)$ has repeated factors (such as $x^2 + 8x + 16 = (x+4)^2$). Suppose the first linear factor, $(a_1x + b_1)$ is repeated r times (that is, $Q(x)$ contains the factor $(a_1x + b_1)^r$). Then, instead of $\frac{A}{a_1x+b_1}$, we should write:

$$\frac{A_1}{a_1x + b_1} + \frac{A_2}{(a_1x + b_1)^2} + \cdots + \frac{A_r}{(a_1x + b_1)^r}$$

Let us look at a concrete example to see how this works:

Example: Decompose $\frac{x^2+x+1}{(x+1)^2(x+2)}$

Solution: We start by defining:

$$\frac{x^2 + x + 1}{(x+1)^2(x+2)} = \frac{A}{x+1} + \frac{B}{(x+1)^2} + \frac{C}{x+2}$$

Multiplying both sides by $(x+1)^2(x+2)$:

$$x^2 + x + 1 = A(x+1)(x+2) + B(x+2) + C(x+1)^2$$

Since there are only 2 roots to $(x+1)^2(x+2)$, we will use another method called “equating the coefficients” to find A , B , and C . We start by expanding the right side of the equation:

$$x^2 + x + 1 = A(x^2 + 3x + 2) + B(x+2) + C(x^2 + 2x + 1)$$

Distributing and combining, we find that:

$$x^2 + x + 1 = Ax^2 + 3Ax + 2A + Bx + 2B + Cx^2 + 2Cx + C$$

$$x^2 + x + 1 = (A + C)x^2 + (3A + B + 2C)x + (2A + 2B + C)$$

For this equation to be true, we know that:

$$A + C = 1$$

$$3A + B + 2C = 1$$

$$2A + 2B + C = 1$$

(That is, the coefficient for x^2 on the left, 1, must be equal to the coefficient for x^2 on the right, $(A + C)$, and so on.) We now have a system of 3 equations and 3 unknowns. When you solve for each, you should find that:

$$A = -2$$

$$B = 1$$

$$C = 3$$

Therefore,

$$\frac{x^2 + x + 1}{(x + 1)^2(x + 2)} = \frac{-2}{x + 1} + \frac{1}{(x + 1)^2} + \frac{3}{x + 2}$$

Irreducible quadratic factors

Sometimes, we cannot express a polynomial as the product of two linear statements (that is, terms in the form $ax + b$). Take $x^2 + 1$, which cannot be expressed as the product of real, linear terms. What do you do if something like $x^2 + 1$ is in the denominator? In this case, when we write an expression for $\frac{P(x)}{Q(x)}$, we include a term in the form:

$$\frac{Ax + B}{ax^2 + bx + c}$$

For example, we can write:

$$\frac{x}{(x - 2)(x^2 + 1)(x^2 + 4)} = \frac{A}{x - 2} + \frac{Bx + C}{x^2 + 1} + \frac{Dx + E}{x^2 + 4}$$

Example: Decompose $\frac{2x^2 - x + 4}{x^3 + 4x}$

Solution: We begin by factoring the denominator:

$$x^3 + 4x = x(x^2 + 4)$$

Which cannot be factored further. Therefore, we define:

$$\frac{2x^2 - x + 4}{x(x^2 + 4)} = \frac{A}{x} + \frac{Bx + C}{x^2 + 4}$$

$$2x^2 - x + 4 = A(x^2 + 4) + (Bx + C)x$$

$$2x^2 - x + 4 = Ax^2 + 4A + Bx^2 + Cx$$

Which implies that:

$$2 = A + B$$

$$C = -1$$

$$4A = 4$$

Therefore, $A = 1$, $B = 1$, and $C = -1$ and we can say that:

$$\frac{2x^2 - x + 4}{x^3 + 4x} = \frac{1}{x} + \frac{x - 1}{x^2 + 4}$$

Repeated irreducible quadratic factors

Lastly, the denominator might contain repeated irreducible quadratic factors. Similar to repeated linear factors, when setting up your partial fractions, instead of only writing

$$\frac{A}{ax^2 + bx + c}$$

For a quadratic factor that is repeated r times, your equation should include:

$$\frac{A_1}{ax^2 + bx + c} + \frac{A_2}{(ax^2 + bx + c)^2} + \cdots + \frac{A_r}{(ax^2 + bx + c)^r}$$

Exercise 36

Decompose the following proper fractions

Working Space

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

Answer on Page 89

Polynomials III: Graphing and Interpolation

7.1 Graphing Polynomials

A roller coaster designer sketches the track as a smooth curve through a few key heights, and a computer fills in the rest with polynomials. In this chapter, you will learn to sketch a polynomial's graph from a few clues, and then to do the designer's trick: find a polynomial that passes through any points you choose.

You already have several tools for sketching.

For instance, you can determine:

- where the graph crosses the y-axis, which you can find by evaluating the polynomial at $x = 0$.

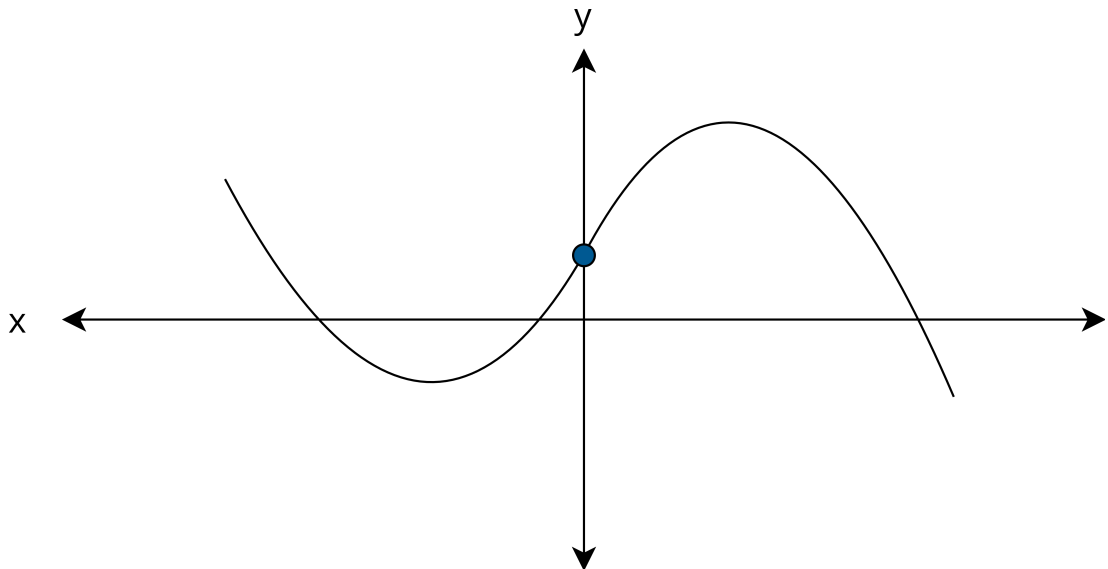


Figure 7.1: A graph showing the y-intercept of a polynomial

- where the graph crosses the x-axis, which you can find by solving for the roots of the polynomial.

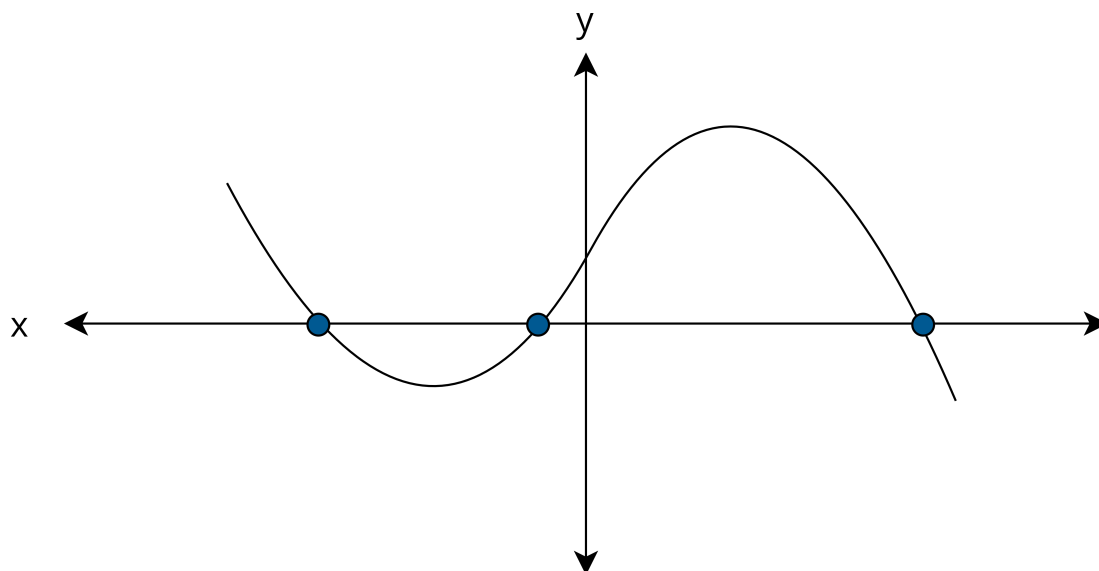


Figure 7.2: A graph showing the roots of a polynomial

- the level spots on the graph (often the top of a hump or the bottom of a dip). Recall the derivative from the classes chapter: a polynomial whose value is the slope of the graph. The level spots are where the slope is zero, so you find the roots of the derivative.

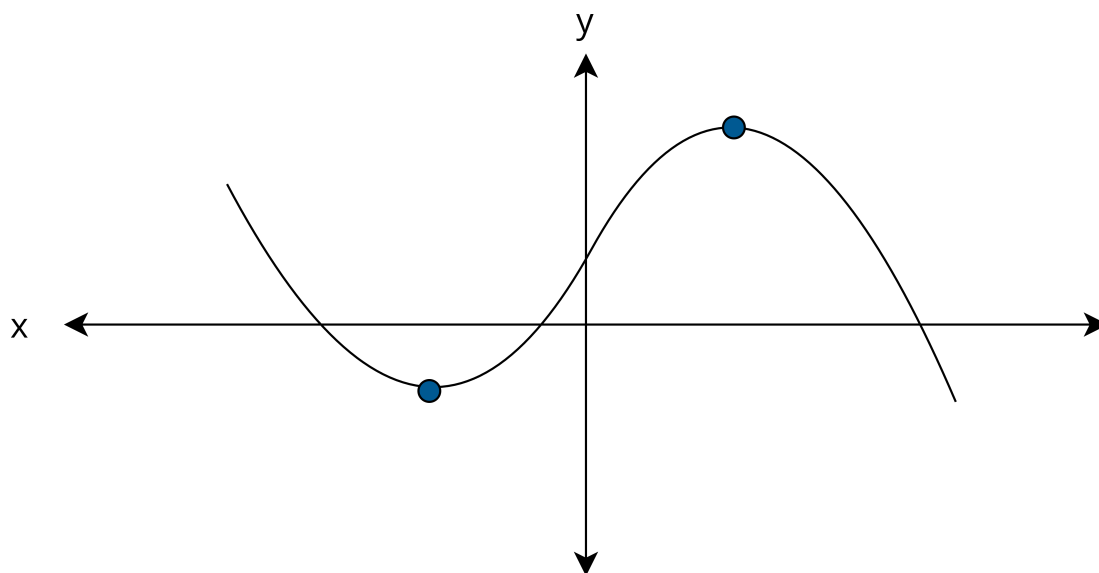


Figure 7.3: A graph showing “level spots” of a polynomial.

For example, if you wanted to graph the polynomial $f(x) = -x^3 - x^2 + 6x$, you might plug in a few values that are easy to compute:

- $f(-2) = -8$
- $f(-1) = -6$
- $f(0) = 0$
- $f(1) = 4$
- $f(2) = 0$

So, right away, we know two roots: $x = 0$ and $x = 2$. Are there others? We will not know until we factor the polynomial:

$$\begin{aligned} -x^3 - x^2 + 6x &= (-1x)(x^2 + x - 6) \\ &= (-1x)(x + 3)(x - 2) \end{aligned}$$

So, yes, there is a third root: $x = -3$

What about the level spots? Using the rule from the classes chapter (each cx^n becomes ncx^{n-1}), the derivative is $f'(x) = -3x^2 - 2x + 6$. Where is that zero? Divide by -3 :

$$\begin{aligned} -3x^2 - 2x + 6 = 0 \\ x^2 + \frac{2}{3}x - 2 = 0 \end{aligned}$$

We have a formula for quadratics like this:

$$\begin{aligned} x &= -\frac{b}{2} \pm \frac{\sqrt{b^2 - 4c}}{2} \\ &= -\frac{\frac{2}{3}}{2} \pm \frac{\sqrt{\left(\frac{2}{3}\right)^2 - 4(-2)}}{2} \\ &= -\frac{1}{3} \pm \frac{\sqrt{\frac{4}{9} + 8}}{2} \\ &= -\frac{1}{3} \pm \frac{\sqrt{\frac{85}{9}}}{2} \\ &= -\frac{1}{3} \pm \frac{\sqrt{85}}{6} \\ &\approx 1.20 \text{ and } -1.87 \end{aligned}$$

Now, you might plug those numbers in:

- $f(1.2) \approx 4.0$

- $f(-1.87) \approx -8.2$

7.1.1 Leading term in graphing

There is one more trick you need before you can draw a good graph of a polynomial. As you go farther and farther to the left and right, where does the function go? In other words, does the graph go up on both ends (like a smile)? Or does it go down on both ends (like a frown)? Or does the negative end go down (frowny) while the positive end go up (smiley)? Or does the negative end go up (smiley) and the positive end go down (frowny)?

Assuming the polynomial is not constant, there are only those four possibilities. It is determined entirely by the leading term of the polynomial. If the degree of the leading term is even, both ends go in the same direction (both are smiley or both are frowny). If the coefficient of the leading term is positive, the right end goes up. (Far from zero, the leading term is so much bigger than all the others that it decides everything.)

Worked example. Describe the graph of $p(x) = x^3 - x$.

Solution. It factors as $x(x - 1)(x + 1)$, so it crosses the x -axis at -1 , 0 , and 1 , and the y -intercept is 0 . The leading term x^3 has odd degree and positive coefficient, so the graph comes up from the bottom left and rises to the top right.

The graph we are working on has a leading term of $-1x^3$. The degree is odd, thus the ends go in different directions. The coefficient is negative, so the positive end points down. Now, you can draw the graph, which should look something like this:

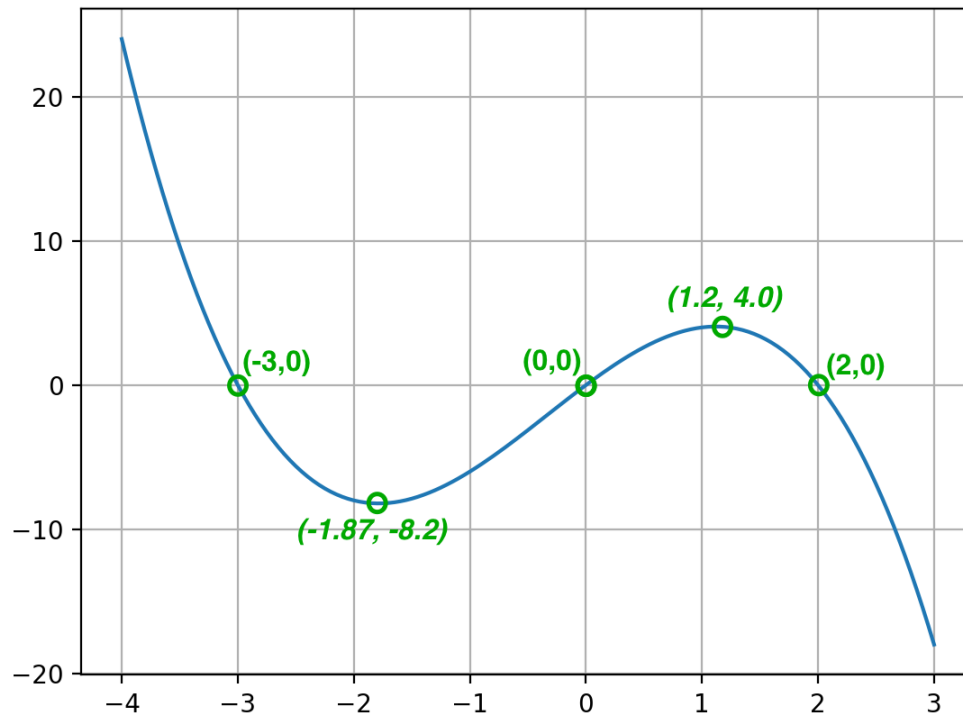


Figure 7.4: A graph with roots and significant points.

7.2 Interpolating with Polynomials

Let us say someone on a distant planet records video of a hammer being thrown up into the air. They send you three random frames of the hammer in flight. Each frame has a timestamp, and you can clearly see how high the hammer is in each one. Can you create a 2nd degree polynomial that explains the entire flight of the hammer?

In other words, you have three points (t_0, h_0) , (t_1, h_1) , (t_2, h_2) . Can you find a, b, c such that the graph of $h = at^2 + bt + c$ passes through all three points?

The answer is yes. In fact, given any n points with different t values, there is exactly one polynomial of degree at most $n-1$ that passes through all of them. (Two points determine a line, which has degree 1. Three points determine a parabola, degree 2.) Finding it is called *interpolation*.

There are a lot of variables floating around. Let us make it concrete: The photos are taken at $t = 2$ seconds, $t = 3$ seconds, and $t = 4$ seconds. In those photos, the height of the hammer is 5m, 7m, and 6m. So, we want our polynomial to pass through these points:

$(2, 5), (3, 7), (4, 6)$.

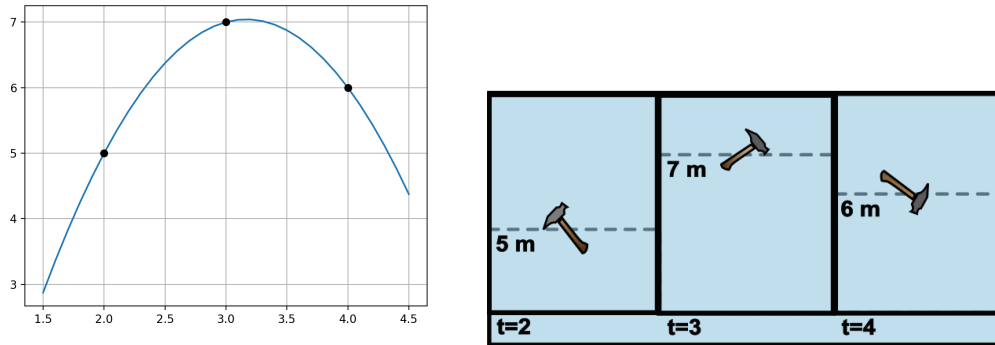


Figure 7.5: Left: the graph of the hammer going through the given points. Right: the three hammer frames.

How can you find that polynomial? Let us do it in small steps. Can you create a 2nd degree polynomial that is not zero at $t = 2$, but is zero at $t = 3$ and $t = 4$? Yes, you can: $(x-3)(x-4)$ has exactly two roots at $t = 3$ and $t = 4$. The value of this polynomial at $t = 2$ is $(2-3)(2-4) = 2$. We really want it to be 5m, so we can divide the whole polynomial by 2 and multiply it by 5.

Now we have the polynomial:

$$f_0(x) = \frac{5}{(2-3)(2-4)}(x-3)(x-4) = \frac{5}{2}x^2 - \frac{35}{2}x + 30$$

This is a second degree polynomial that is 5 at $t = 2$ and 0 at $t = 3$ and $t = 4$.

Now, we create a polynomial that is 7 at $t = 3$ and 0 at $t = 2$ and $t = 4$:

$$f_1(x) = \frac{7}{(3-2)(3-4)}(x-2)(x-4) = -7x^2 + 42x - 56$$

Finally, we create a polynomial that is 6 at $t = 4$ and zero at $t = 2$ and $t = 3$:

$$f_2(x) = \frac{6}{(4-2)(4-3)}(x-2)(x-3) = 3x^2 - 15x + 18$$

Adding these three polynomials together gives you a new polynomial that touches all three points:

$$f(x) = \frac{5}{2}x^2 - \frac{35}{2}x + 30 - 7x^2 + 42x - 56 + 3x^2 - 15x + 18 = -\frac{3}{2}x^2 + \frac{19}{2}x - 8$$

You can test this in Python with your Polynomial class. Create a file called test_interpolation.py. Add this code:

```

1  from Polynomial import Polynomial
2  import matplotlib.pyplot as plt
3
4  in_x = [2,3,4]
5  in_y = [5,7,6]
6
7  pn = Polynomial([-8, 19/2, -3/2])
8  print(pn)
9
10 # These lists will hold our x and y values
11 x_list = []
12 y_list = []
13
14 # Starting x
15 current_x = 1.5
16
17 while current_x <= 4.5:
18     # Evaluate pn at current_x
19     current_y = pn(current_x)
20
21     # Add x and y to respective lists
22     x_list.append(current_x)
23     y_list.append(current_y)
24
25     # Move x forward
26     current_x += 0.05
27
28 # Plot the curve
29 plt.plot(x_list, y_list)
30
31 # Plot black circles on the given points
32 plt.plot(in_x, in_y, "ko")
33 plt.grid(True)
34 plt.show()

```

You should get a nice plot that shows the graph of the polynomial passing through those three points.

In general, then, if you provide any three points $(t_0, h_0), (t_1, h_1), (t_2, h_2)$, there is a second degree polynomial that pass through all three:

$$\frac{h_0}{(t_0 - t_1)(t_0 - t_2)}(x - t_1)(x - t_2) + \frac{h_1}{(t_1 - t_0)(t_1 - t_2)}(x - t_0)(x - t_2) + \frac{h_2}{(t_2 - t_0)(t_2 - t_1)}(x - t_0)(x - t_1)$$

What if you are given 9 points $((t_0, h_0), (t_1, h_1), \dots, (t_8, h_8))$ and want to find a 8th degree

polynomial that passes through all of them? Just what you would expect:

$$\frac{h_0}{(t_0 - t_1)(t_0 - t_2) \dots (t_0 - t_8)}(x - t_1)(x - t_2) \dots (x - t_8) + \dots + \frac{h_8}{(t_8 - t_0) \dots (t_8 - t_7)}(x - t_0) \dots (x - t_7)$$

Mathematicians write this compactly with two symbols. You met Σ (capital sigma) in the data chapters: it means “add up.” Its cousin Π (capital pi) means “multiply together.” Given n points $(t_0, h_0), \dots, (t_{n-1}, h_{n-1})$, the polynomial of degree at most $n - 1$ that goes through them is

$$y = \sum_{i=0}^{n-1} \left(\prod_{\substack{0 \leq j \leq n-1 \\ j \neq i}} \frac{x - t_j}{t_i - t_j} \right) h_i$$

Read it from the inside out: for each point i , multiply together the factors $\frac{x - t_j}{t_i - t_j}$ for every other point j (this is 1 at t_i and 0 at every other t_j), scale by h_i , and add up the results.

That would be tedious for a person to compute, but computers are perfect for this stuff. Let us create a method that creates instances of `Polynomial` using interpolation.¹

7.2.1 Interpolating polynomials in python

Your method will take two lists of numbers: one contains x -values and the other contains y -values. So comment out the line that creates the polynomial in `test_interpolation.py` and create it from two lists:

```
1 in_x = [2,3,4]
2 in_y = [5,7,6]
3 # pn = Polynomial([-8, 19/2, -3/2])
4 pn = Polynomial.from_points(in_x, in_y)
5 print(pn)
```

Add the following method to your `Polynomial` class in `Polynomial.py`

```
1 @classmethod
2 def from_points(cls, x_values, y_values):
3     coef_count = len(x_values)
4
5     # Sums start with a zero polynomial
6     sum_pn = Polynomial([0.0] * coef_count)
7     for i in range(coef_count):
8
9         # Products start with the constant 1 polynomial
```

¹Note that other textbooks or resources may refer to this as *Lagrange Polynomial Interpolation*.

```

10     product_pn = Polynomial([1.0])
11     for j in range(coef_count):
12
13         # Must skip j=i
14         if j != i:
15             # (1x - x_values[j]) has a root at x_values[j]
16             factor_pn = Polynomial([-1 * x_values[j], 1])
17             product_pn = product_pn * factor_pn
18
19         # Scale so product_pn(x_values[i]) = y_values[i]
20         scale_factor = y_values[i] / product_pn(x_values[i])
21         scaled_pn = scale_factor * product_pn
22
23         # Add it to the sum
24         sum_pn = sum_pn + scaled_pn
25
26     return sum_pn

```

It should work exactly the same as before. You should get the same polynomial printed out, $-1.50x^2 + 9.50x + -8.00$, and the same plot of the curve passing through the three points.

The line `@classmethod` makes `from_points` a method of the class itself rather than of one polynomial, so you call it as `Polynomial.from_points(...)` to build a brand-new polynomial.

How about six points? Change `in_x` and `in_y` at the start of `test_interpolation.py`:

```

1 in_x = [1.7, 2, 2.7, 3.5, 4, 4.4]
2 in_y = [8, 12, 1, 4, -1, 6]

```

You should get a degree 5 polynomial that passes through all six points:

$$11.21x^5 - 171.05x^4 + 1019.44x^3 - 2957.53x^2 + 4161.78x - 2258.75$$

It should look like this:

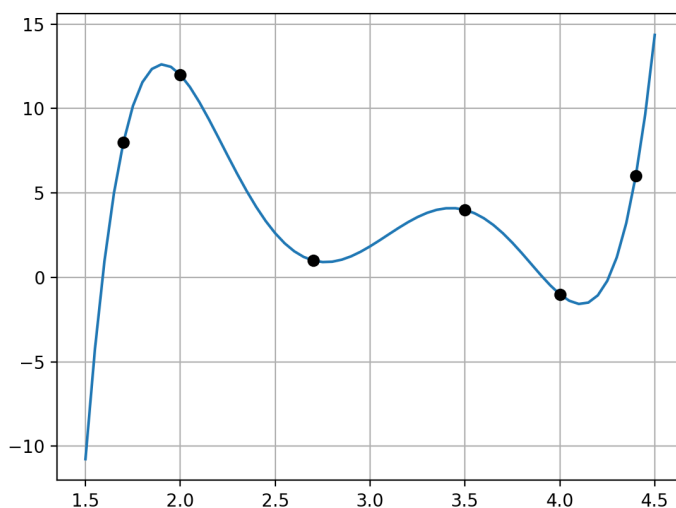


Figure 7.6: Interpolating six points with a degree 5 polynomial.

Exercise 37 Interpolate by hand

Find the parabola through $(0, 1)$, $(1, 0)$, and $(2, 3)$ using the three-polynomial method from this chapter. Check your answer at all three points.

Working Space

Answer on Page 89

Exercise 38 Interpolating a Line

Find the polynomial of degree at most 1 through the points $(0, 1)$ and $(2, 5)$.

Working Space

Answer on Page 90

Answers to Exercises

Answer to Exercise 1 (on page 8)

```

1 import numpy as np
2 import imageio.v3 as iio
3 import sys
4
5 image = iio.imread(sys.argv[1]).astype(float)
6 gray = 0.299 * image[:, :, 0] + 0.587 * image[:, :, 1] + 0.114 * image[:, :, 2]
7 iio.imwrite(sys.argv[2], gray.astype(np.uint8))

```

Green gets the most weight because our eyes are most sensitive to green light, so green contributes the most to how bright a color looks. (The weights add up to 1, so white stays 255.)

Answer to Exercise 2 (on page 9)

(255, 255, 255), with full red, full green, and full blue.

Answer to Exercise 3 (on page 12)

$$-2x^3 + \frac{1}{2}x + 3.9$$

$$(4.5)x^2 + \pi x$$

$$7$$

$$2x^{-10} + 4x - 1$$

$$x^{\frac{2}{3}}$$

$$3x^{20} + 2x^{19} - 5x^{18}$$

Answer to Exercise 4 (on page 12)

Standard form would be $-x^3 + 21x^2 - 1000x + \pi$. The degree is 3. The leading coefficient

is -1

Answer to Exercise 5 (on page 13)

$$4^3 - (3)(4^2) + (10)(4) - 12 = 64 - 48 + 40 - 12. \text{ So } y = 44$$

Answer to Exercise 6 (on page 14)

$$3x^3 - 7x^2 + x - 18 \text{ and } 3x^5 + 2x^4 - 7x^3 + x^2 - 12$$

Answer to Exercise 7 (on page 15)

$$x^3 - 3x^2 + 5x \text{ and } 3x^5 - 2x^4 - 3x^3 + 5x^2 - 2x + 6$$

Answer to Exercise 8 (on page 16)

```
1 def add_polynomials(a, b):
2     degree_of_result = max(len(a), len(b))
3     result = []
4     for i in range(degree_of_result):
5         if i < len(a):
6             coefficient_a = a[i]
7         else:
8             coefficient_a = 0.0
9
10        if i < len(b):
11            coefficient_b = b[i]
12        else:
13            coefficient_b = 0.0
14
15        result.append(coefficient_a + coefficient_b)
16    return result
```

Answer to Exercise 9 (on page 17)

```
1 def subtract_polynomial(a, b):
2     neg_b = scalar_polynomial_multiply(-1.0, b)
```

```
3 | return add_polynomials(a, neg_b)
```

Answer to Exercise 10 (on page 18)

$$(3x^2)(5x^3) = 15x^5$$

$$(2x)(4x^9) = 8x^{10}$$

$$(-5.5x^2)(2x^3) = -11x^5$$

$$(\pi)(-2x^5) = -2\pi x^5$$

$$(2x)(3x^2)(5x^7) = 30x^{10}$$

Answer to Exercise 11 (on page 19)

$$(3x^2)(5x^3 - 2x + 3) = 15x^5 - 6x^3 + 9x^2$$

$$(2x)(4x^9 - 1) = 8x^{10} - 2x$$

$$(-5.5x^2)(2x^3 + 4x^2 + 6) = -11x^5 - 22x^4 - 33x^2$$

$$(\pi)(-2x^5 + 3x^4 + x) = -2\pi x^5 + 3\pi x^4 + \pi x$$

$$(2x)(3x^2)(5x^7 + 2x) = 30x^{10} + 12x^4$$

Answer to Exercise 12 (on page 21)

$$(2x + 1)(3x - 2) = 6x^2 - x - 2$$

$$(-3x^2 + 5)(4x - 2) = -12x^3 + 6x^2 + 20x - 10$$

$$(-2x - 1)(-3x - \pi) = 6x^2 + (3 + 2\pi)x + \pi$$

$$(-2x^5 + 5x)(3x^5 + 2x) = -6x^{10} - 4x^6 + 15x^6 + 10x^2 = -6x^{10} + 11x^6 + 10x^2$$

Answer to Exercise 13 (on page 22)

The degree of the product is determined by the term that is the product of the highest degree term in p_1 and the highest degree term in p_2 . Thus, the product of a degree 23 polynomial and a degree 12 polynomial has degree 35.

Answer to Exercise 14 (on page 26)

$$(2x - 3)(2x + 3) = 4x^2 - 9$$

$$(7 + 5x^3)(7 - 5x^3) = 49 - 25x^6$$

$$(x - a)(x + a) = x^2 - a^2$$

$$(3 - \pi)(3 + \pi) = 9 - \pi^2$$

$$(-4x^3 + 10)(-4x^3 - 10) = 16x^6 - 100$$

$$(x + \sqrt{7})(x - \sqrt{7}) = x^2 - 7$$

$$x^2 - 9 = (x + 3)(x - 3)$$

$$49 - 16x^6 = (7 + 4x^3)(7 - 4x^3)$$

$$\pi^2 - 25x^8 = (\pi + 5x^4)(\pi - 5x^4)$$

$$x^2 - 5 = (x + \sqrt{5})(x - \sqrt{5})$$

Answer to Exercise 15 (on page 29)

$$(x + 1)^2 = x^2 + 2x + 1$$

$$(3x^5 + 5)^2 = 9x^{10} + 30x^5 + 25$$

$$(x^3 - 1)^2 = x^6 - 2x^3 + 1$$

$$(x - \sqrt{7})^2 = x^2 - 2x\sqrt{7} + 7$$

Answer to Exercise 16 (on page 30)

$$(x + \pi)^5 = x^5 + 5\pi x^4 + 10\pi^2 x^3 + 10\pi^3 x^2 + 5\pi^4 x + \pi^5$$

Answer to Exercise 17 (on page 30)

```

1  """Return the nth row (0-indexed) of Pascal's Triangle."""
2  n = int(input("Enter row index n: "))
3  row = [1]
4  for k in range(1, n + 1):
5      row.append(row[-1] * (n - k + 1) // k)
6  return row

```

Answer to Exercise 18 (on page 32)

```

1  favorites[3] = "Gloves"

```

Answer to Exercise 19 (on page 34)

```

1  pn2 = [2.5, 5.0, -2.0, -7.0, 4.0]
2  y = evaluate_polynomial(pn2, 8.5)
3  print("Polynomial 2: When x is 8.5, y is", y)
4
5  pn3 = [-9.0, 0.0, 0.0, 0.0, 0.0, 5.0]
6  y = evaluate_polynomial(pn3, 2.0)
7  print("Polynomial 3: When x is 2.0, y is", y)

```

Answer to Exercise 20 (on page 36)

The polynomial crosses the y-axis at 6. When x is zero, all the terms are zero except the last one. Thus, you can easily tell that $x^3 - 7x + 6$ will cross the y-axis at $y = 6$.

Looking at the graph, you can tell that the curve crosses the x-axis near -3 , 1 , and 2 . If you plug those numbers into the polynomial, you would find that it evaluates to zero at each one. Thus, $x = -3$, $x = 1$, and $x = 2$ are roots. (In fact, $x^3 - 7x + 6 = (x + 3)(x - 1)(x - 2)$,

which you can check by multiplying.)

Answer to Exercise 21 (on page 37)

$$p(3) = 27 - 3 = 24.$$

Answer to Exercise 22 (on page 46)

```
1 p = Polynomial([6.0, -7.0, 0.0, 1.0])
2 print(p(-3), p(1), p(2))
```

All three print 0.0, so -3 , 1, and 2 are the roots, just as you found on the graph in the Python lists chapter.

Answer to Exercise 23 (on page 46)

```
1 r = Rectangle(5, 2)
2 print(r.area())
```

The output is 10.

Answer to Exercise 24 (on page 48)

1. The GCD of 18 and 15 is 3, and the least degree is 1, so $18x^3 - 15x = (3x)(6x^2 - 5)$.
2. The GCD of 4, 12, and 16 is 4, and the least degree is 1, so $4x^3 - 12x^2 + 16x = (4x)(x^2 - 3x + 4)$.
3. The GCD of 18, 24, and 30 is 6, and the least degree is 8, so $18x^{50} - 24x^{20} + 30x^8 = (6x^8)(3x^{42} - 4x^{12} + 5)$.

Answer to Exercise 25 (on page 49)

1. 3 and 11 multiply to 33 and add to 14, so $x^2 + 14x + 33 = (x + 3)(x + 11)$.

2. -6 and -10 multiply to 60 and add to -16 , so $x^2 - 16x + 60 = (x - 6)(x - 10)$.
3. 9 and -6 multiply to -54 and add to 3 , so $x^2 + 3x - 54 = (x + 9)(x - 6)$.
4. -5 and 2 multiply to -10 and add to -3 , so $x^2 - 3x - 10 = (x - 5)(x + 2)$.

Answer to Exercise 26 (on page 51)

```
1 6
2 0
```

At $x = 2$: $4 + 6 - 4 = 6$. At $x = -4$: $16 - 12 - 4 = 0$. So -4 is a root and 2 is not. This matches the factorization $x^2 + 3x - 4 = (x + 4)(x - 1)$.

Answer to Exercise 27 (on page 51)

At $x = 0$ both sides equal the constant term, -10 , because -5 and 2 do multiply to -10 . A single match proves little here. The polynomial has degree 2 , so all three values must match, and two of them fail. The numbers -5 and 2 add to -3 instead of 3 , so the signs are swapped. The correct factorization is $(x + 5)(x - 2)$.

Answer to Exercise 28 (on page 52)

In Python, $^$ does not mean “to the power of.” It is a different operator, so c computes something other than $x^2 + 5x + 6$. The fix is to write $x**2$. Notice that the bad version still matches at $x = 1$ by chance, which is another reason to check several values.

Answer to Exercise 29 (on page 53)

Pull out $2x$ to get $(2x)(x^2 - 3x - 4)$. The numbers -4 and 1 multiply to -4 and add to -3 , so $2x^3 - 6x^2 - 8x = (2x)(x + 1)(x - 4)$.

```
1 def e(x):
2     return 2*x**3 - 6*x**2 - 8*x
3
4 def h(x):
5     return (2*x)*(x + 1)*(x - 4)
6
```

```

7 for x in [0, 1, 2, 3]:
8     print(x, e(x), h(x), e(x) == h(x))
9
10 for r in [0, -1, 4]:
11     print(r, e(r))

```

The polynomial has degree 3, so four values are enough, and all four lines say **True**. The roots are 0, -1 , and 4, and each one prints 0.

Answer to Exercise 30 (on page 54)

```

1 (x - 3)*(x + 3)

```

The polynomial $x^2 - 9$ is a difference of squares, so it factors into $(x - 3)(x + 3)$.

Answer to Exercise 31 (on page 54)

Pull out 3 to get $3(x^2 + 4x - 5)$. The numbers 5 and -1 multiply to -5 and add to 4, so $3x^2 + 12x - 15 = 3(x + 5)(x - 1)$.

```

1 from sympy import symbols, factor, expand
2
3 x = symbols('x')
4 print(expand(3*(x + 5)*(x - 1)))
5 print(factor(3*x**2 + 12*x - 15))

```

```

1 3*x**2 + 12*x - 15
2 3*(x - 1)*(x + 5)

```

Expanding gives back the original polynomial, and SymPy's factors match yours with the binomials in a different order.

Answer to Exercise 32 (on page 60)

x-intercept: $(-5/2, 0)$; y-intercept: $(0, 5/4)$; horizontal asymptote: $y = 2$; vertical asymptote: $x = -4$

Answer to Exercise 33 (on page 61)

The correct answer is E. To explain, we examine the behavior of each function when x is very large.

- A. For large x , $\sin 5x$ stays between -1 and 1 while x grows, so $\frac{\sin 5x}{x}$ shrinks toward 0 , not 5 .
- B. $5x$ grows without bound.
- C. $\frac{1}{x-5}$ shrinks toward 0 (this function has a *vertical* asymptote at $x = 5$).
- D. For large x , $\frac{5x}{1-x}$ behaves like $\frac{5x}{-x} = -5$, so its horizontal asymptote is $y = -5$.
- E. For large x , the x^2 terms dominate, so $\frac{20x^2-x}{1+4x^2}$ behaves like $\frac{20x^2}{4x^2} = 5$.

Answer to Exercise 34 (on page 63)

Factored form: $\frac{x^2(x+2)}{x(x+1)}$; hole: $(0,0)$; vertical asymptote: $x = -1$; oblique asymptote: $y = x + 1$

Answer to Exercise 35 (on page 66)

1. $x^2 + 4x + 12 + \frac{30x-27}{x^2-3x+2}$
2. $2 + \frac{6x^2-4x+13}{x^3-3x^2+2x-4}$
3. $3x - 2 + \frac{8x^2-6x+1}{x^3-3x}$

Answer to Exercise 36 (on page 70)

1. $\frac{10}{7(x+6)} + \frac{-3}{7(x-1)}$
2. $\frac{1}{x^2+1} + \frac{x}{(x^2+1)^2}$
3. $\frac{-2}{x+1} + \frac{1}{(x+1)^2} + \frac{3}{x+2}$

Answer to Exercise 37 (on page 80)

$$f_0 = \frac{1}{(0-1)(0-2)}(x-1)(x-2) = \frac{1}{2}(x^2 - 3x + 2). \quad f_1 = \frac{0}{\dots}(\dots) = 0. \quad f_2 = \frac{3}{(2-0)(2-1)}x(x-1) =$$

$\frac{3}{2}(x^2 - x)$. Adding: $p(x) = 2x^2 - 3x + 1$. Check: $p(0) = 1$, $p(1) = 0$, $p(2) = 8 - 6 + 1 = 3$.

Answer to Exercise 38 (on page 80)

The slope is $\frac{5-1}{2-0} = 2$ and the intercept is 1, so $p(x) = 2x + 1$.



INDEX

- aliasing, [25](#)
- asymptote, [59](#)
- attributes, [39](#)
- binomial
 - definition of, [28](#)
 - multiplying, [20](#)
 - powers of, [28](#)
- binomial coefficient, [30](#)
- binomial theorem, [30](#)
- class in python, [42](#)
- classes, [39](#)
 - child, [41](#)
 - parent, [41](#)
- coefficient
 - polynomial, [11](#)
- color (RGB), [4](#)
- combinatorics, [30](#)
- command-line argument, [4](#)
- degree
 - of a product, [22](#)
 - polynomial, [11](#)
- difference of squares, [25](#)
- distributive property, [19](#)
- domain, [58](#)
- end
 - behavior, [60](#)
- end behavior, [74](#)
- enumerate, [24](#)
- exponential function, [11](#)
- exponentiation, [32](#)
- factoring, [26](#)
 - checking in Python, [49](#)
 - quadratics, [49](#)
- factoring polynomials, [47](#)
- FOIL, [20](#)
- from_points method, [78](#)
- greatest common divisor, [48](#)
- hole, [60](#)
- horizontal
 - asymptote, [59](#)
- image file formats, [7](#)
- imageio, [3](#)
- improper fraction, [65](#)
- indexing
 - zero-based, [31](#)
- infinite series, [13](#)
- inheritance, [41](#)
- instance, [40](#)
- integration, [65](#)
- interpolation, [75](#)
 - general formula, [78](#)
 - in Python, [78](#)
- Lagrange interpolation, [78](#)
- leading coefficient, [12](#)
- leading term, [12](#)

- level spots, 72
- like terms, 13, 20
- list
 - copy, 25
 - replicating, 23
- long division, 65
- magic methods, 43
- matplotlib, 35
- method, 39
- monomial, 11
 - coefficient, 11
 - degree, 11
- monomials
 - adding, 13
 - multiplication, 18
 - multiplying polynomials by, 19
- multiply_polynomials, 23
- negative coefficients, 21
- nested loop, 25
- NumPy array, 3
- object-oriented programming, 39
- oblique
 - asymptote, 62
- operator overloading, 41
- parabolic asymptote, 64
- partial fraction decomposition, 65
 - distinct linear factors, 67
 - irreducible quadratic factors, 69
 - repeated linear factors, 68
- Pascal's triangle, 29
- perfect squares, 28
- pixel, 3
- plotting
 - polynomials, 35
- polynomial, 11
 - adding in Python, 15
 - definition of, 11
 - different degrees, 16
 - evaluating, 32
 - graphing, 72
 - identifying, 12
 - list representation, 15
 - multiplying, 21
 - sketching graphs, 71
 - subtracting in Python, 17
 - through given points, 75
- Polynomial class, 42
- polynomial long division, 62
- polynomials
 - adding, 14
 - multiplication, 18
 - subtracting, 14
- proper fraction, 66
- Python
 - list, 31
- quadratic, 11
- quadratic formula, 73
- rational
 - expression, 57
 - function, 58
 - simplifying, 57
- rational function, 12
- RGB, 3
- roots of a polynomial, 26, 37, 47
- scalar, 17
- scalar multiplication, 17
- standard form
 - polynomial, 12
- subclass, 41
- symbolic variable, 53
- SymPy, 53
- term, 12
- type(), 41
- vertical
 - asymptote, 59
- x-intercept, 58, 71
- y-intercept, 58, 71