



CONTENTS

1	Differentiating Polynomials	3
1.1	Second order and higher derivatives	5
2	Python Classes	7
2.1	Object-Oriented Programming Introduction	7
2.2	Parent classes	9
2.3	Making a Polynomial class	9
3	Common Polynomial Products	15
3.1	Difference of squares	15
3.2	Powers of binomials	17
4	Factoring Polynomials	21
4.1	How to factor polynomials	21
5	Practice with Polynomials	25
A	Answers to Exercises	27
	Index	31

Differentiating Polynomials

If you had a function that gave you the height of an object, it would be handy to be able to figure out a function that gave you the velocity at which it was rising or falling. The process of converting the position function into a velocity function is known as *differentiation* or *finding the derivative*. There are a bunch of rules for finding a derivative, but differentiating polynomials only requires three:

- The derivative of a sum is equal to the sum of the derivatives.
- The derivative of a constant is zero.
- The derivative of a nonconstant monomial at^b (a and b are constant numbers, t is time) is abt^{b-1} . This is referred to as the **power rule**.

Power Rule

For $f(x) = ax^n$, the derivative is $f'(x) = anx^{n-1}$.

So, for example, if we tell you that the height (position) in meters of a quadcopter at second t is given by $2t^3 - 5t^2 + 9t + 200$. You could tell us that its vertical velocity is $6t^2 - 10t + 9$.

We indicate the derivative of a function with an apostrophe (read as "prime") between the name of the function and the variable. For example, the derivative of $h(t)$ is $h'(t)$ (which is read out loud as "h prime of t").

Exercise 1 Differentiation of polynomials

Differentiate the following polynomials.

Working Space

1. $f(t) = 2t^3 - 3t^2 - 4t$

2. $g(t) = 2t^{-3/4}$

3. $F(r) = \frac{5}{r^3}$

4. $H(u) = (3u - 1)(u + 2)$

Answer on Page 27

Notice that the degree of the derivative is one less than the degree of the original polynomial. (Unless, of course, the degree of the original is already zero.)

Now, if you know that a position is given by a polynomial, you can differentiate it to find the object's velocity at any time.

The same trick works for acceleration: Let's say you know a function that gives an object's velocity. To find its acceleration at any time, you take the derivative of the velocity function (the second derivative).

Exercise 2 Differentiation of polynomials in Python

Write a function that returns the derivative of a polynomial in `poly.py`. It should look like this:

Working Space

```
def derivative_of_polynomial(pn):
    ...Your code here...
```

When you test it in `test.py`, it should look like this:

```
# 3x**3 + 2x + 5
p1 = [5.0, 2.0, 0.0, 3.0]
d1 = poly.derivative_of_polynomial(p1)
# d1 should be 9x**2 + 2
print("Derivative of", poly.polynomial_to_string(p1),"is", poly.polynomial_to_string(d1))

# Check constant polynomials
p2 = [-9.0]
d2 = poly.derivative_of_polynomial(p2)
# d2 should be 0.0
print("Derivative of", poly.polynomial_to_string(p2),"is", poly.polynomial_to_string(d2))
```

Answer on Page 27

1.1 Second order and higher derivatives

As seen from the example, with height, velocity, and acceleration, you can take the derivative of a derivative, which is called the second derivative and is indicated with two marks, like so:

$$\frac{d}{dx}f'(x) = f''(x)$$

When you have the height function (or position function, in the case of horizontal motion) of an object, the first derivative describes the velocity of the object, and the second derivative describes the acceleration. Suppose the motion of a particle is given by $s(t) = t^3 - 5t$, where s is in meters and t is in seconds. What is the acceleration when the velocity is 0? First, we find the velocity function, $s'(t)$, and the acceleration function, $s''(t)$:

$$s'(t) = 3t^2 - 5$$

$$s''(t) = 6t$$

To find where the velocity is 0, set $s'(t) = 0$:

$$3t^2 - 5 = 0$$

$$3t^2 = 5$$

$$t^2 = \frac{5}{3}$$

$$t = \sqrt{\frac{5}{3}} \approx 1.29s$$

(we ignore the other solution, $t = -\sqrt{\frac{5}{3}}$ because it is usual for time to start at zero.)

Next, we use $t \approx 1.29s$ in the acceleration function, $s''(t)$:

$$s''\left(\sqrt{\frac{5}{3}}\right) = 6\sqrt{\frac{5}{3}} \approx 7.75 \frac{m}{s^2}$$

For higher order derivatives, you just keep taking the derivative! So a third derivative is found by taking the derivative of the second derivative, and so on.

Exercise 3 Using Derivatives to Describe Motion

The position of a particle is described by the equation $s(t) = t^4 - 2t^3 + t^2 - t$, where s is in meters and t is in seconds.

Working Space

(a) Find the velocity and acceleration as functions of t .

(b) Find the velocity after 1.5 s.

(c) Find the acceleration after 1.5 s.

(d) Is the object speeding up or slowing down at $t = 1.5$? How do you know?

Answer on Page 28

Python Classes

FIXME integrate this better with polynomials, but we need to cover the following

2.1 Object-Oriented Programming Introduction

Imagine you want to implement multiple dogs in python. It gets a bit complicated to do the following:

```
dog1name = "Teddy"
dog1age = 6
dog1sound = "woof"

dog2name = "Fluffers"
dog2age = 2
dog2sound = "bark"

dog3name = "Bella"
dog3age = 3
dog3sound = "grr"

print(f"{dog1name} is {dog1age} and says {dog1sound}!")
print(f"{dog2name} is {dog2age} and says {dog2sound}!")
print(f"{dog3name} is {dog3age} and says {dog3sound}!")
```

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.

```
class Dog:
    """A simple model of a dog."""
    # constructor - which creates a new model of a dog.
    def __init__(self, name: str, age: int, sound: str):
        self.name = name
        self.age = age
        self.sound = sound
    # method speak
    def speak(self) -> None:
        """Print a sentence describing the dog."""
```

```
print(f"{self.name} is {self.age} and says {self.sound}!")

# create dog objects called "instances" with the given variables
dog1 = Dog("Teddy", 6, "woof")
dog2 = Dog("Fluffers", 2, "bark")
dog3 = Dog("Bella", 3, "grr")

# call the behavior on each object
dog1.speak()
dog2.speak()
dog3.speak()
```

Now lets talk about classes in 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: -

```
my_string = "houston, we have a problem!"
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 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:

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

This will output

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

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

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


2.2 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. Lets check out an example:

```
class Animal:
    """Generic animal base-class."""

    def __init__(self, name: str, age: int) -> None:
        self.name = name
        self.age = age

    def describe(self) -> None:
        """Print a basic description common to all animals."""
        print(f"{self.name} is {self.age} years old.")

class Dog(Animal):
    """Dog inherits name and age from Animal, adds its own sound."""

    def __init__(self, name: str, age: int, sound: str) -> None:
        super().__init__(name, age)      # initialize the Animal part
        self.sound = sound

    def speak(self) -> None:
        """Dog-specific implementation of speak()."""
        print(f"{self.name} is {self.age} and says {self.sound}!")

# demonstration
dogs = [
    Dog("Teddy", 6, "woof"),
    Dog("Fluffers", 2, "bark"),
    Dog("Bella", 3, "grr")
]

for dog in dogs:
    dog.describe()    # common behavior from Animal
    dog.speak()       # overridden behavior in Dog
```

Here, we have made a parent class for ‘dog’ called ‘animal’.

2.3 Making a Polynomial class

You have created a bunch of useful python functions for dealing with polynomials. Notice how each one has the word “polynomial” in the function name like `derivative_of_polynomial`.

Wouldn't it be more elegant if you had a Polynomial class with a derivative method? Then you could use your polynomial like this:

```
a = Polynomial([9.0, 0.0, 2.3])
b = Polynomial([-2.0, 4.5, 0.0, 2.1])
```

```
print(a, "plus", b, "is", a+b)
print(a, "times", b, "is", a*b)
print(a, "times", 3, "is", a*3)
print(a, "minus", b, "is", a-b)
```

```
c = b.derivative()
```

```
print("Derivative of", b, "is", c)
```

And it would output:

```
2.30x^2 + 9.00 plus 2.10x^3 + 4.50x + -2.00 is 2.10x^3 + 2.30x^2 + 4.50x + 7.00
2.30x^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
2.30x^2 + 9.00 times 3 is 6.90x^2 + 27.00
2.30x^2 + 9.00 minus 2.10x^3 + 4.50x + -2.00 is -2.10x^3 + 2.30x^2 + -4.50x + 11.00
Derivative of 2.10x^3 + 4.50x + -2.00 is 6.30x^2 + 4.50
```

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

```
class Polynomial:
    def __init__(self, coeffs):
        self.coefficients = coeffs.copy()

    def __repr__(self):
        # Make a list of the monomial strings
        monomial_strings = []

        # For standard form we start at the largest degree
        degree = len(self.coefficients) - 1

        # Go through the list backwards
        while degree >= 0:
            coefficient = self.coefficients[degree]

            if coefficient != 0.0:
                # Describe the monomial
                if degree == 0:
                    monomial_string = "{:.2f}".format(coefficient)
                elif degree == 1:
```

```

        monomial_string = "{:.2f}x".format(coefficient)
    else:
        monomial_string = "{:.2f}x^{ {}".format(coefficient, degree)

    # Add it to the list
    monomial_strings.append(monomial_string)

    # Move to the previous term
    degree = degree - 1

# Deal with the zero polynomial
if len(monomial_strings) == 0:
    monomial_strings.append("0.0")

# Separate the terms with a plus sign
return " + ".join(monomial_strings)

def __call__(self, x):
    sum = 0.0
    for degree, coefficient in enumerate(self.coefficients):
        sum = sum + coefficient * x ** degree
    return sum

def __add__(self, b):
    result_length = max(len(self.coefficients), len(b.coefficients))
    result = []
    for i in range(result_length):
        if i < len(self.coefficients):
            coefficient_a = self.coefficients[i]
        else:
            coefficient_a = 0.0

        if i < len(b.coefficients):
            coefficient_b = b.coefficients[i]
        else:
            coefficient_b = 0.0
        result.append(coefficient_a + coefficient_b)

    return Polynomial(result)

def __mul__(self, other):

    # Not a polynomial?
    if not isinstance(other, Polynomial):
        # Try to make it a constant polynomial
        other = Polynomial([other])

```

```
# What is the degree of the resulting polynomial?
result_degree = (len(self.coefficients) - 1) + (len(other.coefficients) - 1)

# Make a list of zeros to hold the coefficients
result = [0.0] * (result_degree + 1)

# Iterate over the indices and values of a
for a_degree, a_coefficient in enumerate(self.coefficients):

    # Iterate over the indices and values of b
    for b_degree, b_coefficient in enumerate(other.coefficients):

        # Calculate the resulting monomial
        coefficient = a_coefficient * b_coefficient
        degree = a_degree + b_degree

        # Add it to the right bucket
        result[degree] = result[degree] + coefficient

    return Polynomial(result)

__rmul__ = __mul__

def __sub__(self, other):
    return self + other * -1.0

def derivative(self):

    # What is the degree of the resulting polynomial?
    original_degree = len(self.coefficients) - 1
    if original_degree > 0:
        degree_of_derivative = original_degree - 1
    else:
        degree_of_derivative = 0

    # We can ignore the constant term (skip the first coefficient)
    current_degree = 1
    result = []

    # Differentiate each monomial
    while current_degree < len(self.coefficients):
        coefficient = self.coefficients[current_degree]
        result.append(coefficient * current_degree)
        current_degree = current_degree + 1
```

```
# No terms? Make it the zero polynomial
if len(result) == 0:
    result.append(0.0)

return Polynomial(result)
```

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

```
from Polynomial import Polynomial

a = Polynomial([9.0, 0.0, 2.3])
b = Polynomial([-2.0, 4.5, 0.0, 2.1])

print(a, "plus", b, "is", a+b)
print(a, "times", b, "is", a*b)
print(a, "times", 3, "is", a*3)
print(a, "minus", b, "is", a-b)

c = b.derivative()

print("Derivative of", b, "is", c)

slope = c(3)
print("Value of the derivative at 3 is", slope)
```

Run the test code:

```
python3 test_polynomial.py
```


Common Polynomial Products

In math and physics, you will run into certain kinds of polynomials over and over again. In this chapter, We are going to cover some patterns that you will want to be able to recognize.

3.1 Difference of squares

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 doesn't matter: $(a-b)(a+b)$ also $a^2 - b^2$.)

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 4 **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 28

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

```
import poly.py
import matplotlib.pyplot as plt

# x**2 - 5
pn = [-5.0, 0.0, 1.0]
```



```

# These lists will hold our x and y values
x_list = []
y_list = []

# Start at x=-3
current_x = -3.0

# End at x=3.0
while current_x < 3.0:
    current_y = poly.evaluate_polynomial(pn, current_x)

    # Add x and y to respective lists
    x_list.append(current_x)
    y_list.append(current_y)

    # Move x forward
    current_x += 0.1

# Plot the curve
plt.plot(x_list, y_list)
plt.grid(True)
plt.show()

```

You should get a plot like this:

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

3.2 Powers of binomials

You can raise whole polynomials to exponents. For example,

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

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,

$$(7x^3 + \pi)^2 = 49x^6 + 14\pi x^3 + \pi^2$$

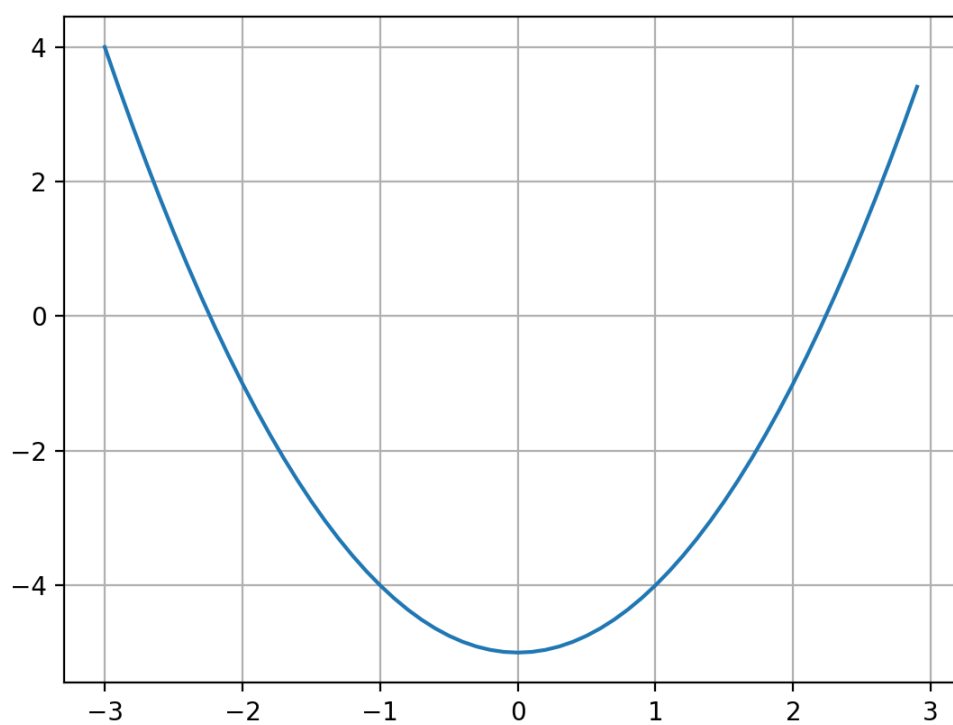


Figure 3.1: A graph of $x^2 - 5$ showing the roots on the x-axis.

Exercise 5 Squaring binomials

Simply the following

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

Working Space

Answer on Page 29

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:

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

Exercise 6 **Using Pascal's Triangle**

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

Working Space

Answer on Page 29

Factoring Polynomials

We factor a polynomial into two or more polynomials of lower degree. For example, let's 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:

4.1 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?

$$12x^{100} + 30x^3 + 42x^7$$

The greatest common divisor of the coefficients (12 , 30 , and 42) is 6 . The least of the

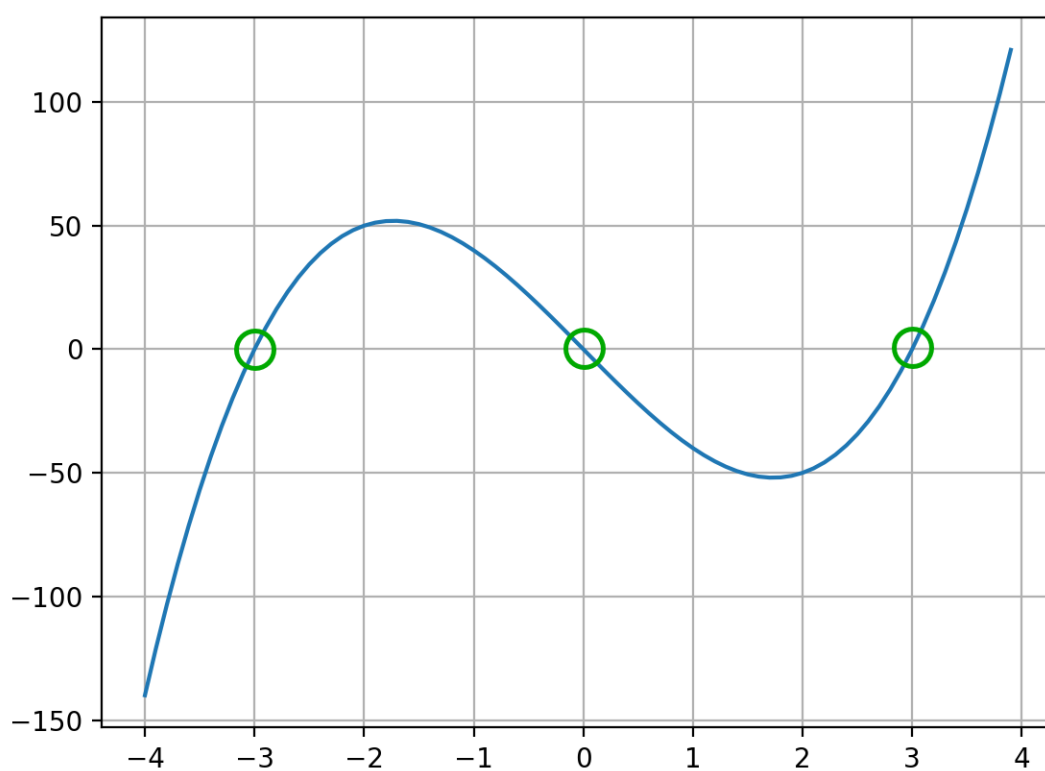


Figure 4.1: Factoring for roots makes it easier to find the roots rather than in expanded form.

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 7 Factoring out the GCD monomial

FIXME Exercise here

Working Space

Answer on Page 29

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 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's 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 don't 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 8 Factoring quadratics

Working Space

Answer on Page 29

CHAPTER 5

Practice with Polynomials

At this point, you know all the pieces necessary to solve problems involving polynomials. In this chapter, you are going to practice using all of these ideas together.

Watch Khan Academy's **Polynomial identities introduction** here: <https://youtu.be/EvNKKyhLSpQ> Also, watch the follow up here: <https://youtu.be/-6qi049Q180>

FIXME: Lots of practice problems here

Answers to Exercises

Answer to Exercise 1 (on page 4)

1. $f'(t) = 3t^2 - 6t - 4$
2. $g'(t) = (\frac{-3}{4})2t^{-3/4-1} = \frac{-3}{2}t^{-7/4}$
3. $F'(r) = \frac{-15}{r^4}$
4. First, we expand the function by multiplying out the two binomials: $(3u-1)(u+2) = 3u^2 + 6u - u - 2$. Therefore, $H(u) = 3u^2 + 5u - 2$, and we can differentiate using what we have learned about differentiating polynomials. $H'(u) = 6u + 5$. In a later chapter, you will learn the Product rule, which will allow you to differentiate this function without multiplying out the binomials.

Answer to Exercise 2 (on page 5)

```
def derivative_of_polynomial(pn):  
  
    # What is the degree of the resulting polynomial?  
    original_degree = len(pn) - 1  
    if original_degree > 0:  
        degree_of_derivative = original_degree - 1  
    else:  
        degree_of_derivative = 0  
  
    # We can ignore the constant term (skip the first coefficient)  
    current_degree = 1  
    result = []  
  
    # Differentiate each monomial  
    while current_degree < len(pn):  
        coefficient = pn[current_degree]  
        result.append(coefficient * current_degree)  
        current_degree = current_degree + 1  
  
    # No terms? Make it the zero polynomial
```

```
if len(result) == 0:
    result.append(0.0)

return result
```

Answer to Exercise 3 (on page 6)

(a) Velocity is the first derivative of the position function, $s'(t) = 4t^3 - 6t^2 + 2t - 1$. Acceleration is the derivative of the velocity function, $s''(t) = 12t^2 - 12t + 2$.

(b) $s'(1.5) = 4(1.5)^3 - 6(1.5)^2 + 2(1.5) - 1 = 2$ We should note that this is a measurement and needs units to make sense. Since s is in meters and t is in seconds, our velocity should have units of $\frac{\text{m}}{\text{s}}$, so our final answer is $s'(1.5\text{s}) = 2\frac{\text{m}}{\text{s}}$.

(c) $s''(1.5) = 12(1.5)^2 - 12(1.5) + 2 = 11$. Similarly to part (b), our answer needs units. The units for acceleration are the units for velocity divided by the unit for time (because acceleration is a rate of change of velocity), and our final answer should be $s''(1.5\text{s}) = 11\frac{\text{m}}{\text{s}^2}$.

(d) When velocity and acceleration are occurring in the same direction (i.e. have the same sign), the speed (the absolute value of velocity) is increasing. Since $s'(1.5\text{s})$ and $s''(1.5\text{s})$ are both > 0 , the speed of the object is increasing.

Answer to Exercise 4 (on page 16)

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

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

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

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

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

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

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

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

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

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

Answer to Exercise 5 (on page 19)

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

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

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

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

Answer to Exercise 6 (on page 20)

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

Answer to Exercise 7 (on page 23)**Answer to Exercise 8 (on page 23)**



INDEX

attributes, [7](#)

class in python, [10](#)

classes, [7](#)

 child, [9](#)

 parent, [9](#)

derivative

 definition of, [3](#)

factoring polynomials, [21](#)

instance, [8](#)

perfect squares, [17](#)

polynomials

 differentiation, [3](#)

power rule, [3](#)

subclass, [9](#)