# Contents

Chapter 1

# Python Lists

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

To review, 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 an array of strings. Type this code into a file called `faves.py`:

```
favorites = ["Raindrops", "Whiskers", "Kettles", "Mittens"]
favorites.append("Packages")
print("Here are all my favorites:", favorites)
print("My most favorite thing is", favorites[0])
print("My second most favorite is", favorites[1])
number_of_faves = len(favorites)
print("Number of things I like:", number_of_faves)

for i in range(number_of_faves):
    print(i, ": I like", favorites[i])
```

Run it:

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

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

### *Exercise 1* Assign into list

Before you list the items, replace ”Mittens” with ”Gloves”.

*Working Space*

*Answer on Page 31*

## 1.1 Evaluating Polynomials in Python

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:

```
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 evalution. Create a file called `polynomials.py` and type in the following:

```
def evaluate_polynomial(pn, x):
    sum = 0.0
    for degree in range(len(pn)):
        coefficient = pn[degree]
        term_value = coefficient * x ** degree
        sum = sum + term_value
    return sum

pn1 = [-12.0, 10.0, -3.0, 1.0]
y = evaluate_polynomial(pn1, 4.0)
print("Polynomial 1: When x is 4.0, y is", y)
```

Run it. It should evaluate to 44.0.

## 1.2 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:

```
def polynomial_to_string(pn):
    ...Your Code Here...

pn_test = [-12.0, 10.0, 0.0, 1.0]
print(polynomial_to_string(pn1))
```

This would output:

```
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:

```
def polynomial_to_string(pn):

  # Make a list of the monomial strings
  monomial_strings = []

  # Start at the term with the largest degree
  degree = len(pn) - 1

  # Go through the list backwards stop after constant term
  while degree >= 0:
      coefficient = pn[degree]

      # Skip any term with a zero coefficient
      if coefficient != 0.0:

          # Describe the monomial
          if degree == 0:
```

```
                monomial_string = "{}".format(coefficient)
            elif degree == 1:
                monomial_string = "{}x".format(coefficient)
            else:
                monomial_string = "{}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")

    # Make a string that joins the terms with a plus sign
    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 aren't familar with. For example, if you want to know about the (join) function, google for "python join".

Now, change your code to use the new function:

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

Run the program. Does the function work?

### *Exercise 2* 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
- Find $5x^5 - 9$ at $x = 2.0$. It should be 151.0

*Working Space*

*Answer on Page 31*

## 1.3 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:

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

After the `evaluate_polynomial` function:

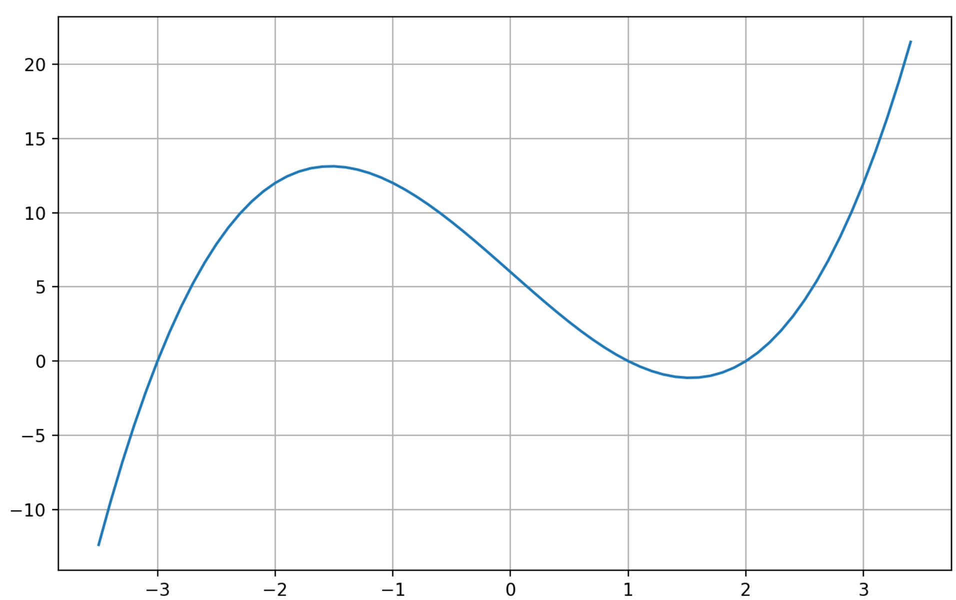
- Create a list with polynomial coefficients.
- Create two empty arrays, one for x values and one for y values.
- Fill the x array with values from -3.5 to 3.5. Evaluate the polynomial at each of these points; put those values in the y array.
- Plot them

Like this:

```
# x**3 - 7x + 6
pn = [6.0, -7.0, 0.0, 1.0]
```

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

# Start at x=-3.5
current_x =-3.5

# End at x=3.5
while current_x <= 3.5:
    current_y = 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 beautiful plot like this:



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

```
$ pip3 install matplotlib
```

## *Exercise 3* Observations

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

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

*Working Space*

*Answer on Page 31*

## Chapter 2

# Adding and Subtracting Polynomials

Watch Khan Academy's **Adding polynomias** 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:

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

## *Exercise 4* Adding Polynomials Practice

*Working Space*

Add the following polynomials:

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 31*

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

One more tricky thing can happen — sometimes the coefficients don't add nicely. For example:

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

That is as far as you can simplify it.

## 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 make 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 5 Subtracting Polynomials Practice

Add the following polynomials:

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

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

*Working Space*

*Answer on Page 32*

## 2.2 Adding Polynomials in Python

As a reminder, in our Python code, we are representing a polynomial with a list of coefficients. The first coefficient is the constant term. The last coefficient is the leading coefficient. So, we can imagine $-5x^3 + 3x^2 - 4x + 9$ and $2x^3 + 4x^2 - 9$ would look like this: *FIXME: Diagram here*

To add the two polynomials then, we sum the coefficients for each degree. *FIXME: Diagram here*

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

```
def add_polynomials(a, b):
    degree_of_result = len(a)
    result = []
    for i in range(degree_of_result):
        coefficient_a = a[i]
        coefficient_b = b[i]
        result.append(coefficient_a + coefficient_b)
    return result
```

```
polynomial1 = [9.0, -4.0, 3.0, -5.0]
polynomial2 = [-9.0, 0.0, 4.0, 2.0]
polynomial3 = add_polynomials(polynomial1, polynomial2)

print('Sum =', polynomial3)
```

Run the program.

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

```
# x**4 - 5x**3 + 3x**2 - 4x + 9
polynomial1 = [9.0, -4.0, 3.0, -5.0, 1.0]

# 2x**3 + 4x**2 - 9
polynomial2 = [-9.0, 0.0, 4.0, 2.0]
polynomial3 = add_polynomials(polynomial1, polynomial2)
print('Sum =', polynomial3)
```

See the problem?

## *Exercise 6* 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.

```
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 32*

## 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:

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

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

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

## *Exercise 7* Subtract polynomials in Python

*Working Space*

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

It should look like this:

```
def subtract_polynomial(a, b):
    ...Your code here...

polynomial5 = [9.0, -4.0, 3.0, -5.0]
polynomial6 = [-9.0, 0.0, 4.0, 2.0, 1.0]
polynomial7 = subtract_polynomial(polynomial5, polynomial6)
print('Difference =', polynomial_to_string(polynomial7))
```

*Answer on Page 32*

## Chapter 3

# Multiplying Polynomials

Watch Khan Academy's **Multiplying monomials** at `https://youtu.be/Vm7H0VTlIco`.

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 8 Multiplying monomials

Multiply these monomials:

1. $(3x^2)(5x^3)$
2. $(2x)(4x^9)$
3. $(-5.5x^2)(2x^3)$
4. $(\pi)(-2x^5)$
5. $(2x)(3x^2)(5x^7)$

*Working Space*

*Answer on Page 32*

## 3.1 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 the distributive property.

After that, it is just multiplying several pairs of monomials:

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

## *Exercise 9* Multiplying a monomial and a polynomial

*Working Space*

Multiply these monomials:

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 33*

## 3.2 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,

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

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 $5x^2+2x-7$:

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

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 10* 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 33*

## *Exercise 11* Observations

Let's 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 33*

## Chapter 4

# 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 folowing 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:

```
import poly

polynomial_a = [9.0, -4.0, 3.0, -5.0]
print('Polynomial A =', poly.polynomial_to_string(polynomial_a))

polynomial_b = [-9.0, 0.0, 4.0, 2.0, 1.0]
print('Polynomial B =', poly.polynomial_to_string(polynomial_b))

# Evaluation
value_of_b = poly.evaluate_polynomial(polynomial_b, 3)
print('Polynomial B at 3 =', value_of_b)

# Adding
a_plus_b = poly.add_polynomials(polynomial_a, polynomial_b)
print('A + B =', poly.polynomial_to_string(a_plus_b))

# Scalar multiplication
b_scalar = poly.scalar_polynomial_multiply(-3.2, polynomial_b)
print('-3.2 * Polynomial B =', poly.polynomial_to_string(b_scalar))

# Subtraction
```

```
a_minus_b = poly.subtract_polynomial(polynomial_a, polynomial_b)
print('A - B =', poly.polynomial_to_string(a_minus_b))
```

When you run it, you should get the following:

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

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

It will return a list of coefficients.

In an exercise in the last chapter, you were asked " Let's say I 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?" The answer was $23 + 12 = 35$.

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

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

```
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.

```
a = [5,7]
b = a * 4
print(b)
# [5, 7, 5, 7, 5, 7, 5, 7]
```

Here iss how you will get a list of zeros:

```
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`:

```
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:

```
def multiply_polynomials(a, b): # What is the degree of the resulting
polynomial?  result_degree = (len(a) - 1) + (len(b) - 1)

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

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

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

            # 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 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:

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

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.

## 4.1 Something surprising about lists

You can imagine that you might want to create two very similar polynomials. Let's 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:

```
c = [1.0, 2.0, 1.0]
d = c
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 *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. *FIXME: Diagram of two references to the same list here.*

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

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

## Chapter 5

# 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 $abt^{b-1}$

So, for example, if we tell you that the height 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 12* Differentiation of polynomials

Differentiate the following polynomials.

1. $f(t) = 2t^3 - 3t^2 - 4t$
2. $g(t) = 2t^{-3/4}$
3. $F(r) = \frac{5}{r^3}$
4. $H(u) = (3u - 1)(u + 2)$

*Working Space*

*Answer on Page 33*

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 13* Differentiation of polynomials in Python

*Working Space*

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

```
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 34*

## 5.1 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, $s''(t)$:

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

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 14* 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.

(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?

*Working Space*

*Answer on Page 34*

Appendix A

# Answers to Exercises

## Answer to Exercise 1 (on page 4)

```
favorites[3] = "Gloves"
```

## Answer to Exercise 2 (on page 7)

```
pn2 = [2.5, 5.0, -2.0, -7.0, 4.0]
y = evaluate_polynomial(pn2, 8.5)
print("Polynomia 2: When x is 8.5, y is", y)

pn3 = [-9.0, 0.0, 0.0, 0.0, 0.0, 5.0]
y = evaluate_polynomial(pn3, 2.0)
print("Polynomial 3: When x is 2.0, y is", y)
```

## Answer to Exercise 3 (on page 9)

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 tell that the curve crosses the y-axes near -3, 1 and 2. If you plug those numbers into the polynomial, you would find that it evalutes to zero at each one. Thus, $x = -3$, $x = 1$, and $x = 2$ are roots.

## Answer to Exercise 4 (on page 12)

$3x^3 - 7x^2 + x - 18$ and $3x^5 - 7x^3 + x^2 - 12$

## Answer to Exercise 5 (on page 13)

$x^3 - 3x^2 + 5x$ and $x^5 - 3x^3 + 5x^2 - 2x + 6$

## Answer to Exercise 6 (on page 14)

```
def add_polynomials(a, b):
    degree_of_result = max(len(a), len(b))
    result = []
    for i in range(degree_of_result):
        if i < len(a):
            coefficient_a = a[i]
        else:
            coefficient_a = 0.0

        if i < len(b):
            coefficient_b = b[i]
        else:
            coefficient_b = 0.0

        result.append(coefficient_a + coefficient_b)
    return result
```

## Answer to Exercise 7 (on page 15)

```
def subtract_polynomial(a, b):
    neg_b = scalar_polynomial_multiply(-1.0, b)
    return add_polynomials(a, neg_b)
```

## Answer to Exercise 8 (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 9 (on page 19)

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

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

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

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

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

## Answer to Exercise 10 (on page 21)

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

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

$(-2x - 1)(-3x - \pi) = 6x^2 + (4 + 2\pi)x + \pi$

$(-2x^5 + 5x)(3x^5 + 2x) = -6x^{10} + 12x^6 + 10x^2$

## Answer to Exercise 11 (on page 21)

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 12 (on page 27)

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 13 (on page 28)

```
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 14 (on page 30)

(a) Velocity is the first derivative of the position function, $s'(t) = 4t^3 - 6t^2 + 2t - 1$. cceleration 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 $\frac{m}{s}$, so our final answer is $s'(1.5s) = 2\frac{m}{s}$.

(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 $s''(1.5s) = 11\frac{m}{s^2}$.

(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 $s'(1.5s)$ and $s''(1.5s)$ are both $> 0$, the speed of the object is increasing.

# Index