Elementary Math for Computer Science

with Python

Eric Bennett

Eric Bennett has taught mathematics in public and private schools in the Austin, Texas, area since 2005. He is the founder and director of Future Set Tech Camp, a summer program for computer science education.

All photographs published herein have been made available by their creators in the public domain as of the publication of this text. Student artwork is published with parental consent. The creators of these photographs and illustrations have not endorsed any part of this text.

Copyright ©2020 Eric Bennett. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written permission of the author.

Contents

Preface

In our increasingly digital world, technology fluency is essential. Computer programming ranks among the most in-demand skill sets for employers today. Computer science instruction is attractive to parents looking to prepare their elementary-aged children for the future and appeals to many children who are naturally curious about technology. In some ways, teaching coding to elementary students is a perfect fit.

Writing a computer program, however, is a complex task which relies on many prerequisite skills. In particular, math concepts such as variable expressions, the order of operations, negative numbers and coordinate systems are essential to developing even the most basic computer programs or video games. Since these concepts are not included in state or national mathematics standards for elementary students, they are not taught in many elementary schools.

This curriculum is an attempt to bridge that gap: the math concepts included here have been selected for their importance to computer programming and video game development. While these concepts are typically presented to older students, they are made accessible to a younger audience through visual models and project-based learning, an approach aimed toward facilitating the meaningful learning necessary for students to craft original work.

Each chapter of this book culminates in a coding project, written in the widely-used Python programming language, which utilizes preceding math concepts. Challenge exercises follow these projects, offering students an opportunity to extend and apply their learning. It is through grappling with these challenges that students will earn the greatest rewards.

What is the purpose of this book?

The purpose of this curriculum is to facilitate meaningful learning in computer science for elementary school students through targeted mathematics and project-based instruction.
Rather than for independent study, this text is designed as a guidebook for a teacher or parent.

What should students know before getting started?

It is recommended that students are reading at a third grade level and familiar with whole-number addition, subtraction, multiplication and division.

Teacher's Guide

This text is designed as a guidebook for a teacher or homeschooling parent, rather than as a self-paced course for students. As such, teachers should employ best practices in presenting this content to students. This section includes suggested strategies for teaching the content in this book, though teachers and parents may also find other strategies effective.

General strategies:

- Read through the introduction to each chapter with students or prepare a slideshow depicting key ideas in advance.
- Break up chapters into smaller lessons. For example, teach adding negatives during one lesson and subtracting negatives in the next lesson. Applications in Python, including challenge exercises, may span several lessons.
- Work example problems with students.
- Allow students to work word problems, answer comprehension questions and attempt challenge exercises in small groups.
- Present key ideas from the written introductions to each coding project.
- Run a working example of each coding project to demonstrate the end product. Afterward, ask students to describe the functions of each part of the code.
- Challenge students to correct their own coding errors. Teach them to read the error messages in the Python shell. When students get stuck, point out the line on which their error occurs, but allow them to find it themselves.

For struggling students:

- Use physical math tiles and reusable number lines to mirror visual models in the text and to create visual models for practice problems that lack them.
- Assign only selected word problems. Work through these together. Use visual models where appropriate. Ask leading questions to guide students toward a process.
- Assign only selected comprehension questions and challenge exercises. Work through these together. Ask leading questions to guide students toward a solution.

For accelerated students:

- Assign fewer practice problems (i.e. every other problem).
- Assign extensions in the appendix.
- Challenge students to write original programs in Python. Generate a list of options appropriate for each chapter and allow students to choose from the list or to pursue their own ideas, where feasible.

Introduction to Coding in Python

Each chapter of this book concludes with a coding project in Python, a programming language popular for its simplicity and wide-ranging applications.

How to install Python

1. Visit the Python Software Foundation website
    - https://www.python.org
2. Download Python
    - Select ”Downloads” and the appropriate operating system to find the necessary installer
    - The programs featured in this book were written for Python 3
3. Install Python
    - During installation, check the boxes to install Python for all users AND add Python to PATH

How to code in Python

1. Open the Python IDLE development environment

   In Microsoft Windows:

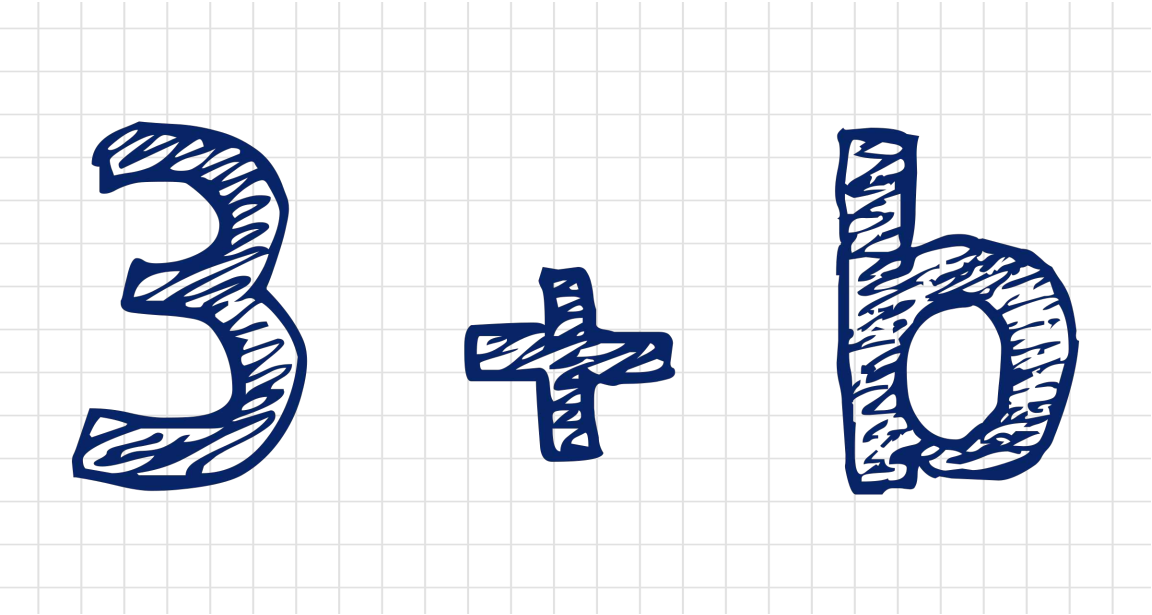
   - From the ”Start” menu, find the Python folder
   - Select ”IDLE...”

   In Mac OS:

   - Open the Terminal
   - Type ”idle”

2. Create a new file
   - Click ”File” and ”New File”
3. Type your code into the editor
   - Try, ”`print("Hello, World!")`”
   - Tips and examples are included at the end of each chapter in this book
4. Save the program
   - Click ”File” and ”Save”
5. Run the program
   - Click, ”Run” and, ”Run Module” or select the file itself and choose, ”Open with Python”

1
Variables



A variable represents a quantity that can change.

Introduction

Variables

Some things never change: the number of inches in a foot; the number of petals on a tulip; the speed of light. Quantities like these are known as constants.

Other things change frequently: the day of the week; the height of a tree; a person's age. Quantities like these are best represented by variables.

Suppose Emily is three years older than Jin. If we know Emily is

twelve years old, we could subtract three years to find Jin's age. Subtracting three from twelve would work this year, but what about next year? If we don't know Emily's age, is it still possible to find Jin's age?

If we represent Emily's age with a variable, x, we can write an expression for Jin's age: $x - 3$. This expression will always find Jin's age, no matter how Emily's age changes.

The height of a tree could be represented by a variable because it changes over time.

Image credit: Carol M. Highsmith

Image credit: Ellie, age 10

What are variables?

- A **variable** represents a quantity that can change.
- A **constant** is a quantity that cannot change.
- In mathematics, it is customary to represent a variable quantity with a letter of the alphabet, like *x* or *y*.
- In computer science, variable quantities are often represented by words or abbreviations to help programmers understand the code.

How are variables used in computer science?

In a computer program, the same variable can store different values at different times, making it possible for a user to achieve varied results. For example, a variable named “brightness” might be adjusted by the user of a photo editor.

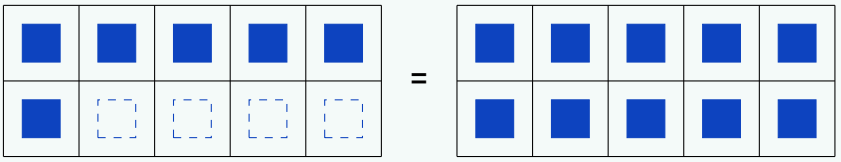
In a video game, the value of a variable named ”score” might change many times during each game.

Examples

Addition with variables

Q: Find the value of the variable:

$$6 + x = 10$$



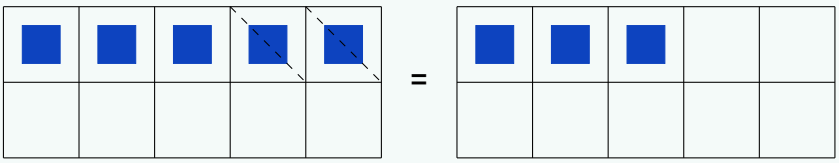
A: Using the ten-frames above, we can see that 4 blue tiles must be added to the left side in order to match the total of 10 tiles on the right side.

In other words, since $6 + 4 = 10$, the value of x must be 4.

Subtraction with variables

Q: Find the value of the variable:

$$5 - c = 3$$

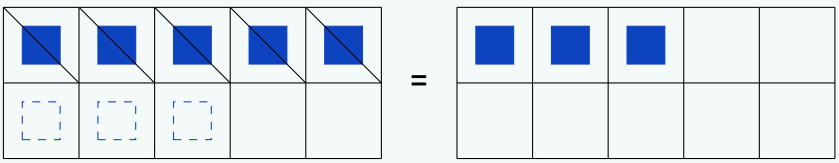


A: Using the ten-frames above, we can see that we must subtract 2 blue tiles from the left side to match the 3 tiles on the right side.

In other words, since $5 - 2 = 3$, the value of c must be 2.

Q: Find the value of the variable:

$$a - 5 = 3$$



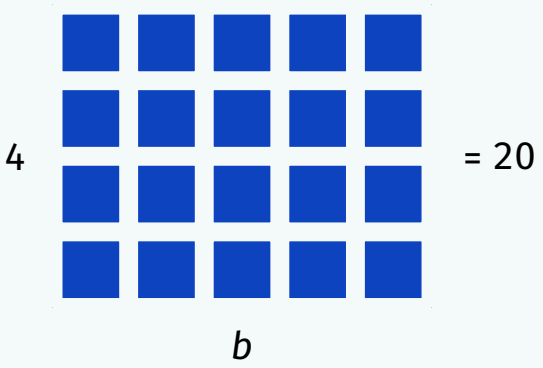
A: Using the ten-frames above, we can see that we must begin with 8 blue tiles on the left side in order to leave 3 after subtracting 5.

In other words, since $8 - 5 = 3$, the value of a must be 8.

Multiplication with variables

Q: Find the value of the variable:

$4 \times b = 20$



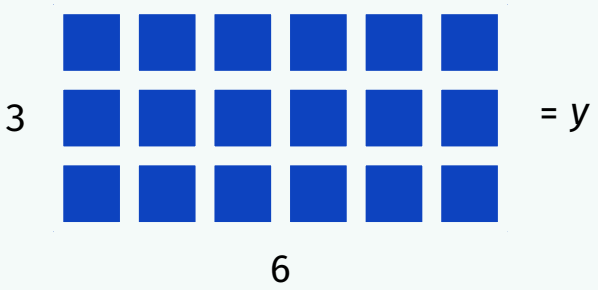
A: Using the grid of tiles above, we see that a 20-tile rectangle with a height of 4 must have a width of 5.

In other words, since $4 \times 5 = 20$, the value of b must be 5.

Division with variables

Q: Find the value of the variable:

$$y \div 3 = 6$$



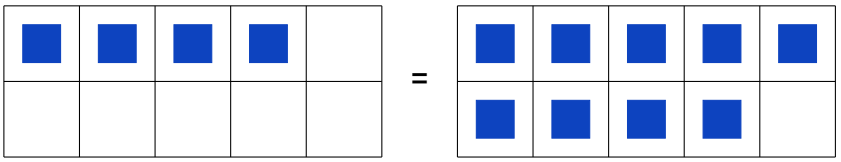
A: Using the grid of tiles above, we see that a rectangle with a height of 3 and width of 6 must be composed of 18 tiles.

In other words, since $18 \div 3 = 6$, the value of y must be 18.

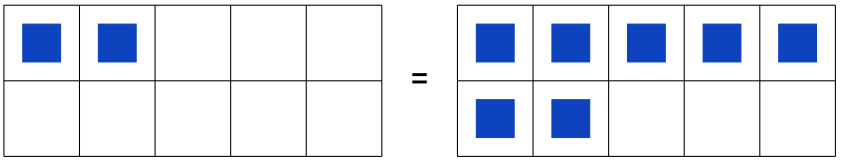
Practice

1. Use the tile models to find the value of the variable.

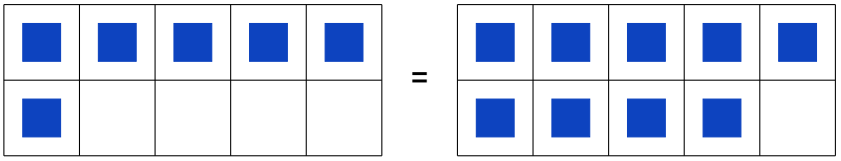
a. $4 + a = 9$



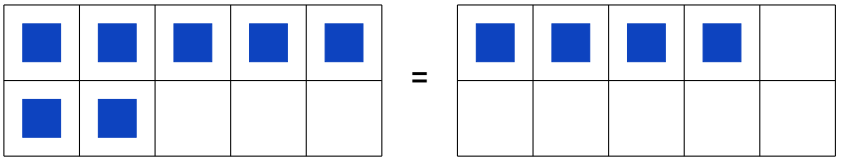
b. $b + 2 = 7$



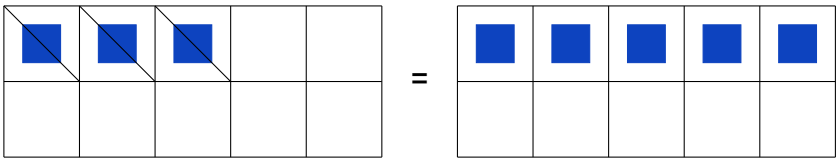
c. $6 + c = 9$



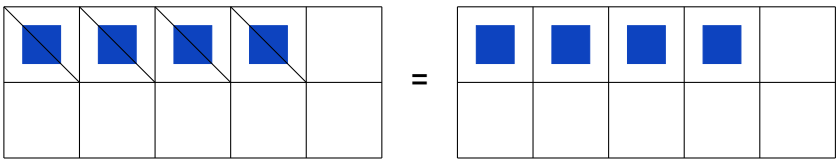
d. $7 - d = 4$



e. $e - 3 = 5$



f. $f - 4 = 4$



2. Find the value of the variable.

a. $a + 2 = 10$

b. $b - 2 = 8$

c. $8 + c = 10$

d. $4 + d = 20$

e. $20 - e = 16$

f. $4 + 16 = f$

g. $13 - g = 4$

h. $h - 7 = 21$

i. $37 + x = 45$

j. $62 - j = 24$

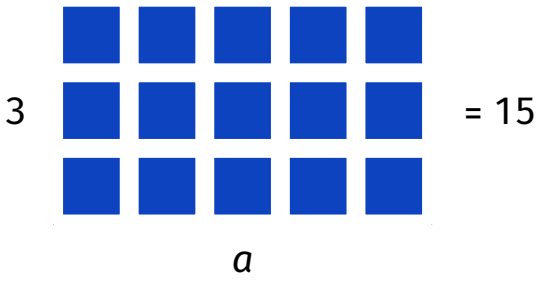
k. $k + 17 = 50$

l. $y - 17 = 50$

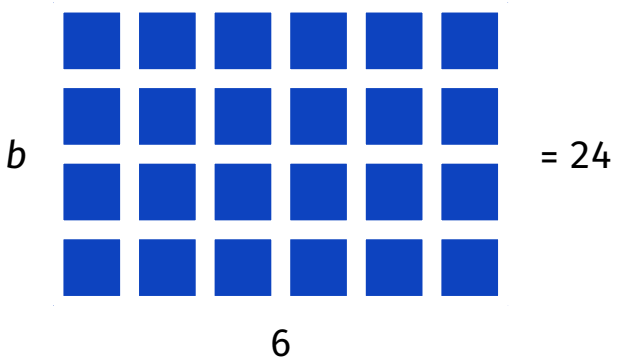
3. Jesper is seven years younger than Chen. Jesper is 8 years old. How old is Chen?

4. Use the tile models to find the value of the variable.

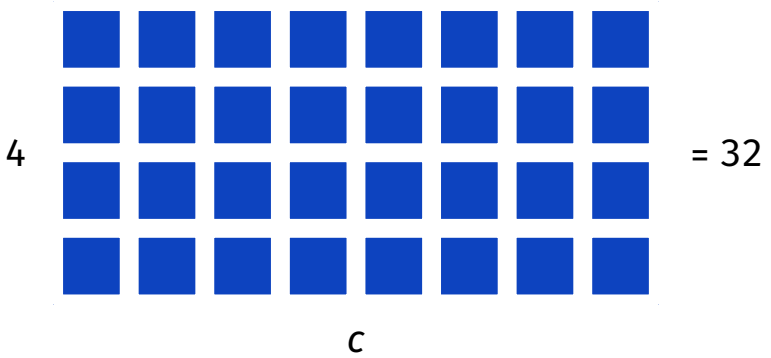
a. $3 \times a = 15$



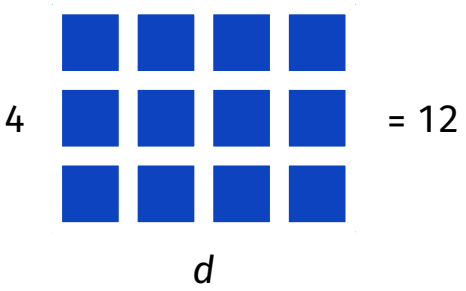
b. $b \times 6 = 24$



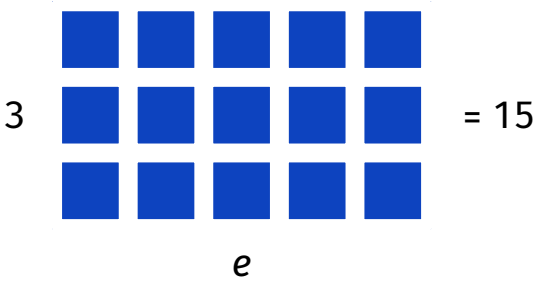
c. $4 \times c = 32$



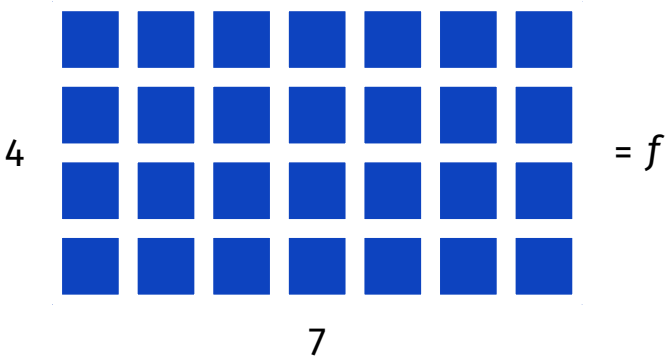
d. $12 \div d = 4$



e. $15 \div e = 3$



f. $f \div 7 = 4$



5. Find the value of the variable.

a. $a \times 2 = 10$

b. $b \div 2 = 8$

c. $8 \times c = 40$

d. $4 \times d = 28$

e. $20 \div e = 2$

f. $4 \times f = 16$

g. $36 \div g = 4$

h. $h \div 7 = 5$

i. $12 \times x = 72$

j. $64 \div j = 4$

k. $k \times 11 = 88$

l. $y \div 13 = 4$

A single bluebonnet flower is composed of many florets.

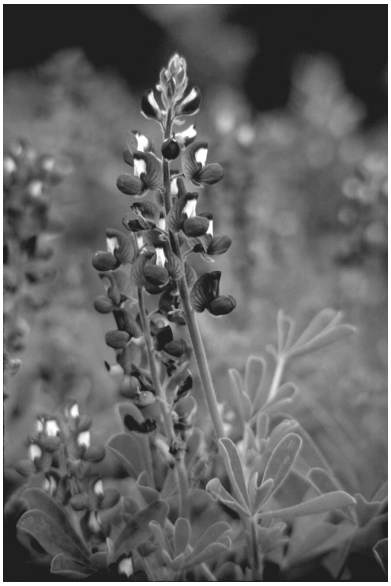


Image credit: Dr. Thomas G. Barnes, U.S. Fish and Wildlife Service

6. A bluebonnet flower is made up of many florets, each with five petals. If there are a total of 70 petals on a particular bluebonnet, how many florets does it have?

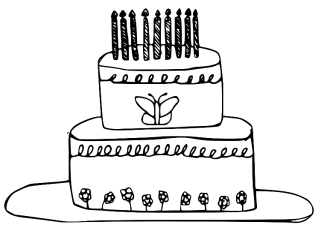


Image credit: Emmaline, age 11

7. **Mark is three times as old as** Carly. Mark is twelve years old. How old is Carly?

8. **Kate is half as old as** Jung was three years ago. Kate is 7 years old. How old is Jung?

9. **A computer programmer uses** the following code to add 100 to a variable named "`score`":

```
score = score + 100
```

   If, after running this code, the value of `score` is 750, what was the previous value of `score`?

10. **In video game development**, all of the parts that interact to make the game are called **game objects**. Player-controlled characters, obstacles and collectible items are all examples of game objects.

    a. A game object moves at a speed of 5 pixels per frame. How many frames would it take for the game object to move 50 pixels? 100 pixels?

    b. If the game runs at 60 frames per second, how many seconds would it take for the game object to move 300 pixels?

Application: *Word Game* in Python

Variables are used to store values in computer programs. In Python, variables are created as they are assigned. For example:

```
score = 100
```

With this code, we create a variable called "`score`" and set its value to 100. Variables are **case-sensitive**, so "`score`" and "`Score`" are different variables.

* * *

We can **print** the value of a variable. For example, the following code will set the variable `score` equal to 100 and print "`100`" on the screen:

```
score = 100
print(score)
```

We could make the output more clear by printing some text along with the value of `score`:

```
score = 100
print("Score:", score)
```

This code sets the value of `score` to 100 and prints:

```
Score: 100
```

* * *

We can also allow a user to **input** values for variables. For example:

```
name = input("Name: ")
age = input("Age: ")
print(name, "is", age, "years old.")
```

With this code, we create two variables and allow the user to input their values. Then, the user's inputs are printed as part of a sentence. If the user enters "`Eric`" and "`many`", then the sentence will read:

```
Eric is many years old.
```

* * *

In the following example program, *Word Game*, the user is asked to input the values for three variables: `adjective`, `verb` and `noun`. These values are then incorporated into a short story.

Word Game in Python

```
print("Word Game!")

adjective = input("Adjective: ")
verb = input("Verb: ")
noun = input("Noun: ")

print("A", adjective, "Day at School")
print("Today at school, we learned how to")
print(verb, "in Python.")
print("I typed my code on a", noun, ".")
print("It was a", adjective, "day at school.")

input("Press ENTER to exit.")
```

Word Game comprehension questions

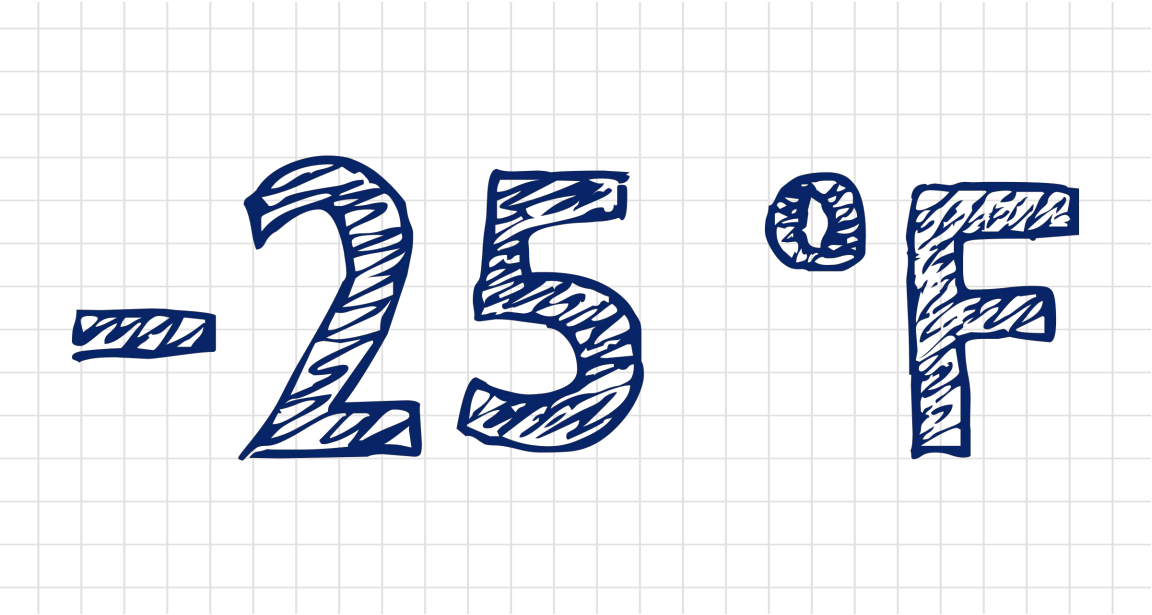
1. How would you store a value to a variable in Python?
2. How would you display the value of a variable in Python?
3. How would you allow another user of your program to enter the value of a variable?

Challenge!

After completing *Word Game* in Python, try these additional challenges:

1. Change the story. Be sure to include all three variables.
2. Add another variable to the game. Include it in the story.
3. Write your own Python program that asks the user to input their name and greets them by that name (i.e. "`Hello, Eric`!").

2
Negative Numbers



The temperature on a cold night might dip below zero.

Introduction

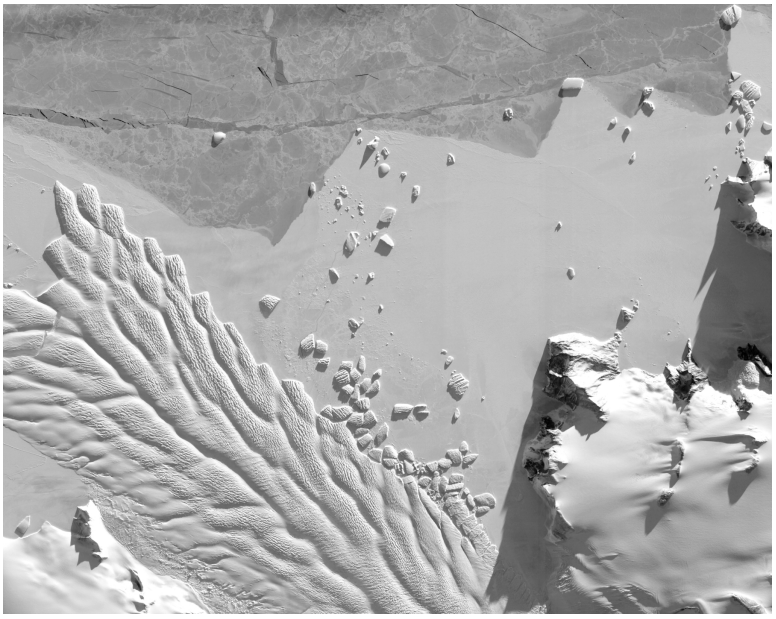
Negative Numbers

Anything that can be counted can be represented by a positive number: the number of students in a classroom; the number of books those students have read this year; the number of minutes in a lunch break. More abstract ideas can also be described by positive numbers: the amount of money in a bank account; the height of a mountain peak; the temperature outside on a summer afternoon.

Where there is something, there can also be nothing: there are times when every classroom is empty; when the year begins, no

books have been read; some lunch breaks are spoiled by lunch detention. The concept of zero is common enough, but what about numbers less than zero? If we can count with positive numbers, can we also count with negative numbers?

It turns out that some ideas are most easily described by numbers less than zero: when someone spends more money than they have, they acquire debt, a negative amount of money; when a scuba diver descends into the ocean, they acquire depth, a negative height; on a cold night in a cold place, the temperature drops below zero degrees, a negative temperature.



A glacier flows in Antarctica, where the temperature is often well below zero degrees Celsius.

Image credit: NASA

What are negative numbers?

- Any number less than zero is a **negative number**.
- All of the negative whole numbers, along with zero and all of the positive whole numbers, are known as **integers**.
- On a **number line**, numbers always increase (become "more positive") to the right and decrease (become "more negative") to the left.
- The **absolute value** of a number, positive or negative, is its distance from 0 on the number line.

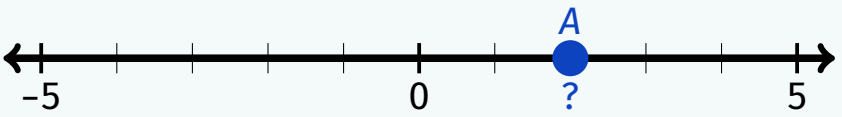
How are negative numbers used in computer science?

Negative numbers can be used to describe the position of an object in a video game or to model concepts like money, temperature or elevation in a computer program.

Examples

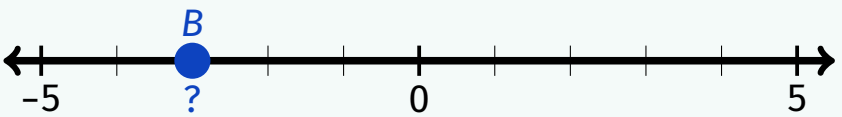
Points on a number line

Q: Where is point *A* located on the number line?



A: Since numbers become ”more positive” to the right on the number line, we can count forward from zero to find that point *A* is located at 2.

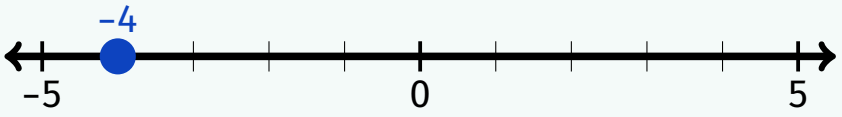
Q: Where is point *B* located on the number line?



A: Since numbers become ”more negative” to the left on the number line, we can count back from zero to find that point *B* is located at −3.

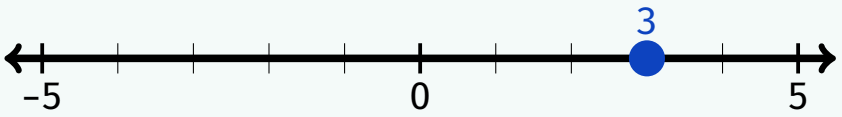
Absolute value

Q: What is the absolute value of -4?



A: Since −4 is located 4 units from 0, the absolute value of −4 is 4.
In other words, the distance from −4 to 0 is 4.

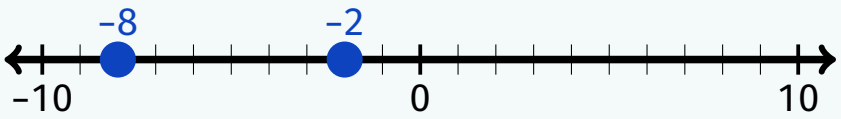
Q: What is the absolute value of 3?



A: Since 3 is located 3 units from 0, the absolute value of 3 is 3.
In other words, the distance from 3 to 0 is 3.

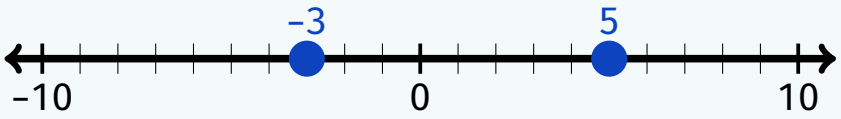
Distance between points

Q: What is the distance between –8 and –2?



A: The distance between –8 and –2 is 6 units.

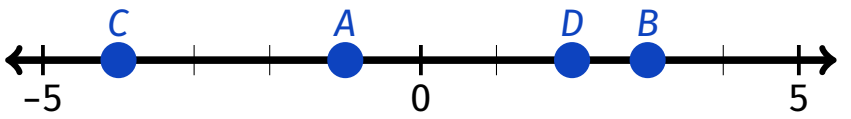
Q: What is the distance between –3 and 5?



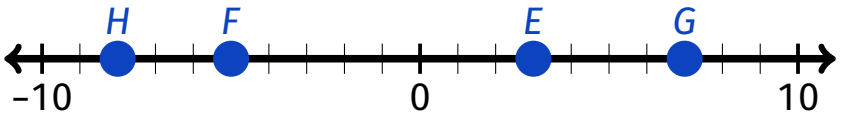
A: Since –3 is 3 units from zero and 5 is 5 units from zero, the distance between –3 and 5 is a total of 8 units.

Practice

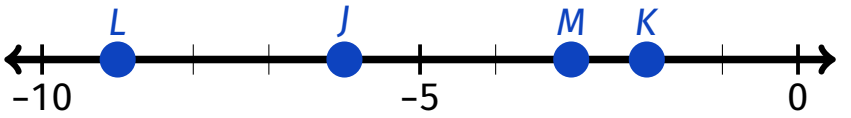
1. Find the location of each labeled point on the number line.



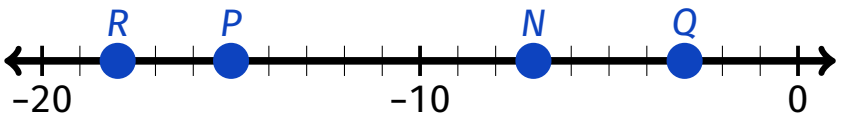
A = _____ *B* = _____ *C* = _____ *D* = _____



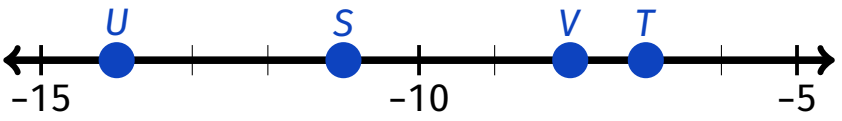
E = _____ *F* = _____ *G* = _____ *H* = _____



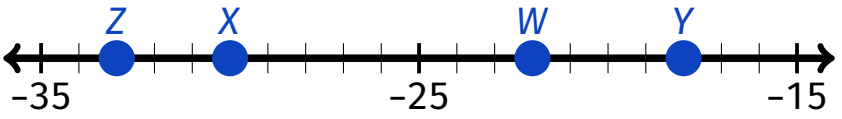
J = _____ *K* = _____ *L* = _____ *M* = _____



N = _____ *P* = _____ *Q* = _____ *R* = _____



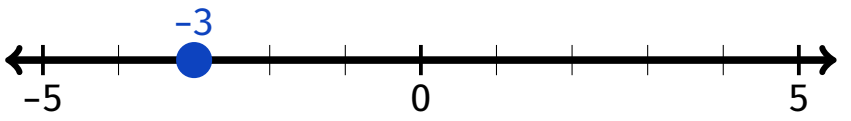
S = _____ *T* = _____ *U* = _____ *V* = _____



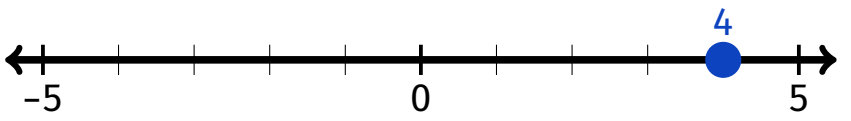
W = _____ *X* = _____ *Y* = _____ *Z* = _____

2. Find the absolute value of each integer.

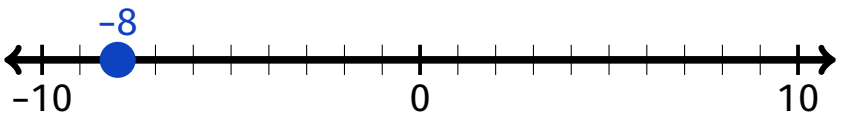
a. −3



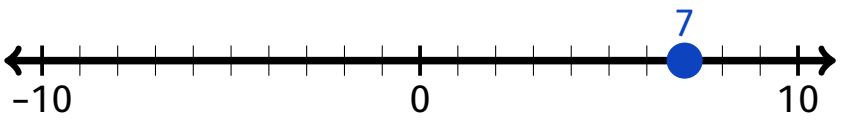
b. 4



c. −8



d. 7



e. 3

f. −4

g. 6

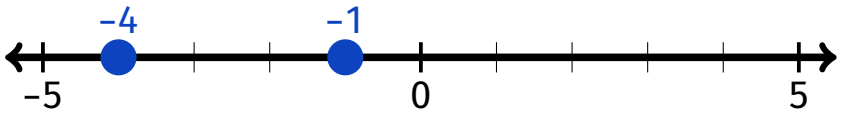
h. 0

i. 17

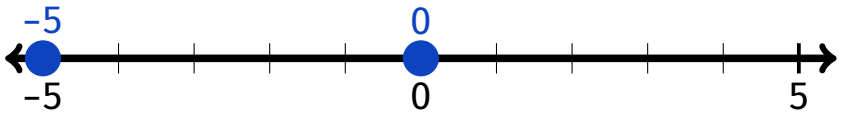
j. −23

3. Find the distance between each pair of points on the number line.

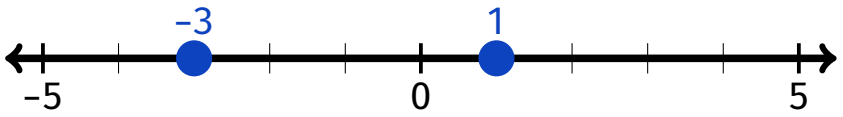
a. –4 and –1



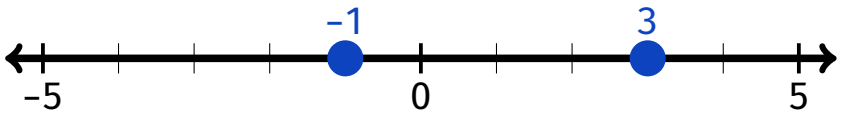
b. 0 and –5



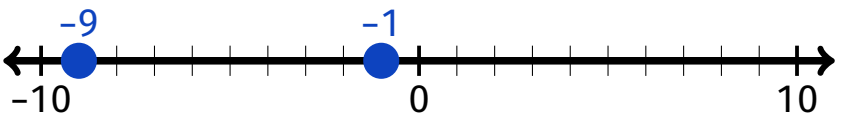
c. –3 and 1



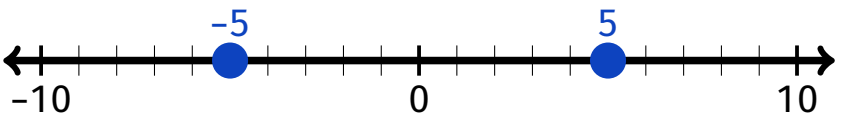
d. 3 and –1



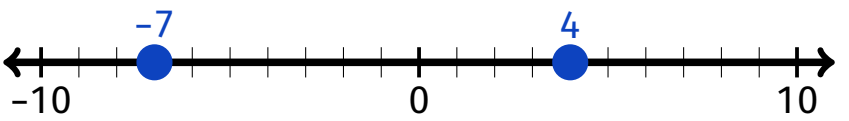
e. –9 and –1



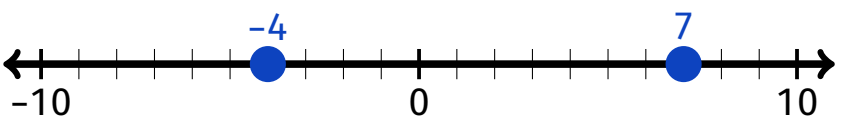
f. –5 and 5



g. 4 and –7



h. –4 and 7



4. **Plot each pair of integers on a number line and find the distance between them.**

   a. –4 and –2 **c.** –3 and 2 **e.** 8 and –3

   b. –2 and –1 **d.** –6 and 0 **f.** 6 and –7

5. **Find the distance between each pair of integers.**

   a. –3 and –1 **e.** –8 and 0 **i.** –1 and 15

   b. –3 and –2 **f.** –6 and 6 **j.** –20 and –7

   c. –5 and 1 **g.** 0 and –12 **k.** –18 and 8

   d. –1 and 5 **h.** –15 and 1 **l.** –27 and –13

6. **Two different integers are a distance** of exactly 12 units from –5. List both integers.

The climate in Austin, Texas is much warmer than in Boise, Idaho.

Image credit: Carol M. Highsmith

7. **The coldest recorded temperature** in Boise, Idaho occurred on December 23rd, 1990. On that day, the temperature reached -25 °F. The coldest recorded temperature in Austin, Texas occurred on February 2nd, 2011. On that day, the temperature reached 17 °F. How many degrees warmer is the record-low in Austin than the record-low in Boise?

8. **To create branches in a computer program** that lead to different outcomes, a programmer will often use "**if-then**" constructs. For example: *If* the player completes a challenge before the time expires, *then* the player earns a perfect score. Complete the following "if-then" constructs about absolute values:

   a. If a number is positive, then its absolute value is...

   b. If a number is negative, then its absolute value is...

   c. If two numbers have the same absolute value, then the numbers are...

Application: *Distance Game* in Python

We can import **modules** to add functionality to the Python programming language. For example, we can use the following code to import the **random module** and add functions pertaining to random numbers:

```
import random
```

Using the random module, we can generate a **random integer** between -10 and 10:

```
X = random.randint(-10, 10)
```

The distance between two integers is the absolute value of their difference. Here, we store the distance between `X` and `Y` as a variable called `distance`:

```
D = abs(X - Y)
```

We'll learn more about subtracting integers in Chapter 5.

* * *

When we store a response to the `input()` function as a variable, that response is stored as a **string**, or text. Since math uses numbers, not text, a string must first be converted to an **integer** if it will be used for math:

```
N = input("Enter an integer: ")
N = int(N)
```

Where we use the "=" operator to store a value to a variable, we must use the "==" operator to compare the values of two variables:

```
if N == D:
    print("The variables are equal!")
```

In the above example, we use an **if statement** to control the flow of the program. We print "`The variables are equal!`" only if `N` is equal to `D`.

In Python, **indentation**, or spacing at the beginning of a line, is important to the structure of a program. We indent any lines of code following an `if` statement to create a **code block** that only runs when the if statement is true.

* * *

In the following example program, *Distance Game*, two random integers are generated between -10 and 10. The user is then tasked with calculating the distance between the two integers.

Distance Game in Python

```
import random

print("Distance Game")

A = random.randint(-10, 10)
B = random.randint(-10, 10)
distance = abs(A - B)

print("What is the distance between")
print(A, "and", B, "?")
guess = input("? ")
guess = int(guess)

if guess == distance:
    print("Correct!")
else:
    print("Incorrect.")

print("The distance between")
print(A, "and", B, "is", distance)

input("Press ENTER to exit.")
```

Distance Game comprehension questions

1. In Python, what module is necessary for functions involving random numbers?
2. How would you generate a random integer greater than or equal to –5 and less than or equal to 0?
3. What is indentation? How is it used in Python?

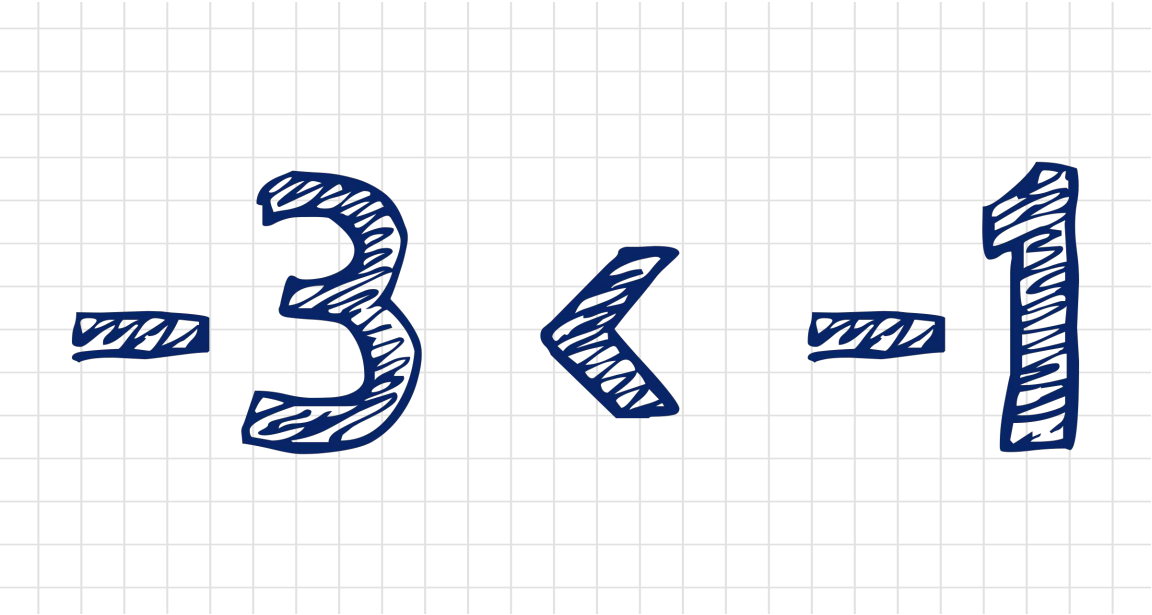
Challenge!

After completing *Distance Game* in Python, try these additional challenges:

1. Change the message the user receives with correct and incorrect guesses.
2. Make the game more difficult by increasing the range of possible random integers.
3. Write a new program that asks the user to guess a random number, determines whether the guess is correct or incorrect, and displays the random number on the screen.
4. It is possible to break *Distance Game* by entering a value that is not an integer. For a solution to this problem, see *Extension: Distance Game in Python* in the appendix.

3

Comparing Negative Numbers



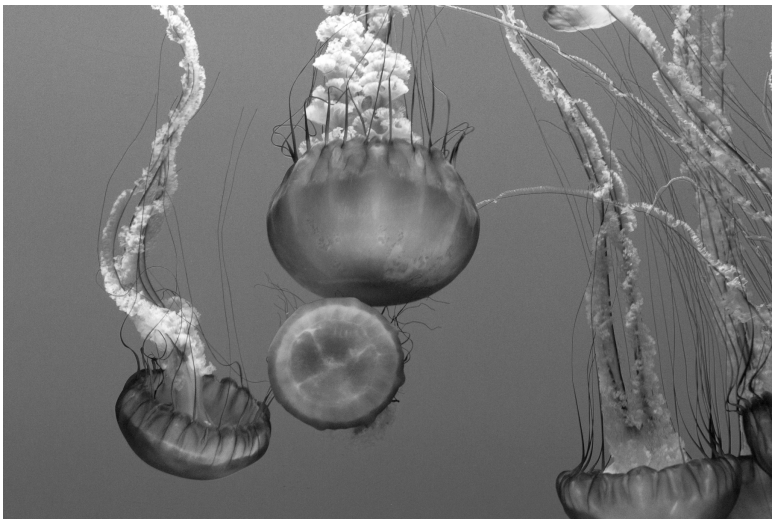
"Negative three is less than negative one."

Introduction

Comparing Negative Numbers

Some problems have many possible solutions: as long as we get to school before the bell rings, we will be on time for class (the number of hours that have passed in the day must be less than a value); as long as we have enough gas in our gas tank, our car can get to our destination (the number of gallons of gas we have must be greater than a value). When comparing numbers, then, it can be useful to describe one number as less than, greater than, or equal to another.

We can compare negative numbers in the same way that we compare positive numbers: just as one tree might be taller than another, one scuba diver might dive deeper than another. The closer a scuba diver gets to the surface, the greater we would consider his height. Thus, negative numbers closer to zero are greater than negative numbers further from zero.



Ocean depths are often described by negative numbers.

Image credit: Carol M. Highsmith

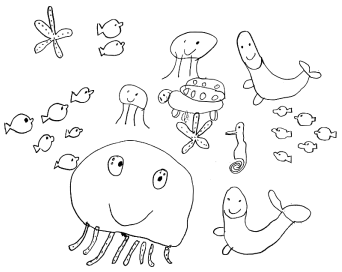


Image credit: Lexi, age 6

How are negative numbers compared?

- On the **number line**, numbers always increase (become "more positive") to the right and decrease (become "more negative") to the left.
- Thus, numbers to the right are **greater than** numbers to the left and numbers to the left are **less than** numbers to the right.
- To describe one number as less than another, we use the symbol "<".
- To describe one number as greater than another, we use the symbol ">".
- To describe one number as **equal to** another, we use the symbol "=".

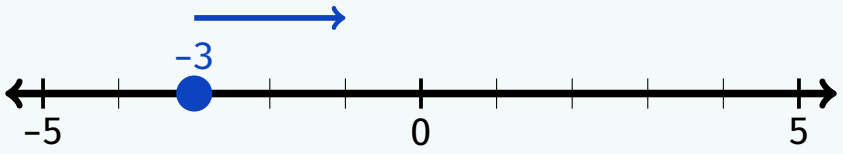
How are negative numbers compared in computer science?

In a video game, a player might restart a level if their avatar falls from a platform. This could be achieved by comparing the vertical position of the player object with a position below the level of the platform, often represented by a negative number.

Examples

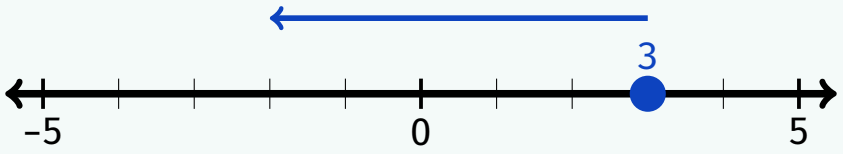
Less and more

Q: What number is 2 more than –3?



A: To find the number 2 more than –3, we move 2 units to the right on the number line. Thus, the solution is –1.

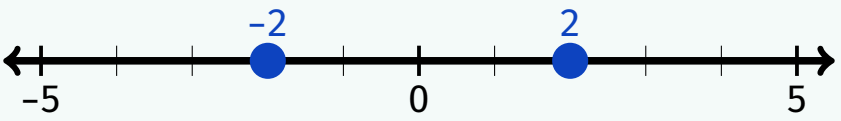
Q: What number is 5 less than 3?



A: To find the number 5 less than 3, we move 5 units to the left on the number line. Thus, the solution is –2.

Less than and greater than

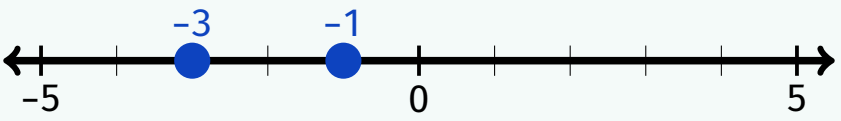
Q: Compare 2 and –2 using <, > or =.



A: Since 2 is greater than –2, the solution is:

$$2 > -2$$

Q: Compare –3 and –1 using <, > or =.



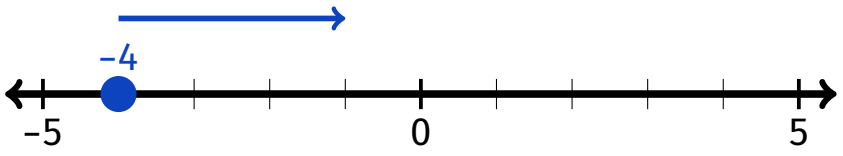
A: Since –3 is less than –1, the solution is:

$$-3 < -1$$

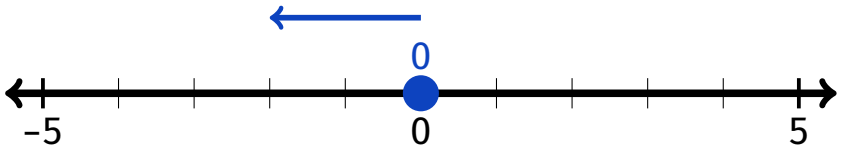
Practice

1. Use the number lines to find the indicated value.

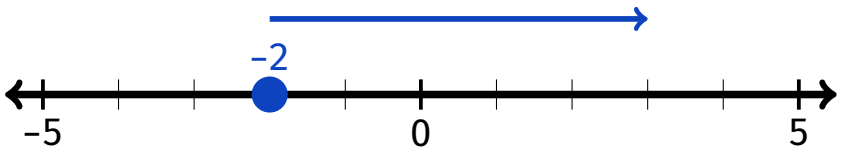
a. 3 more than −4



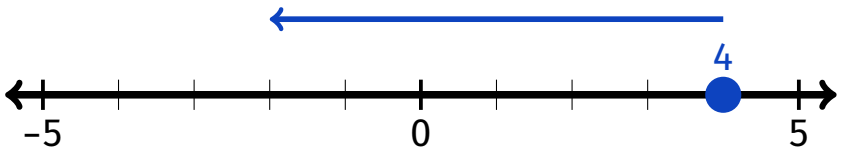
b. 2 less than 0



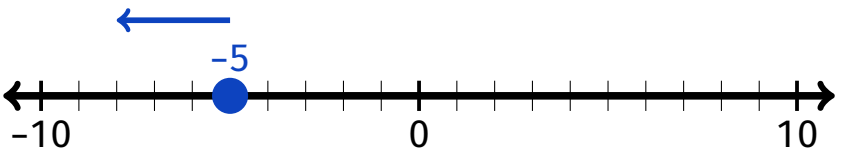
c. 5 more than −2



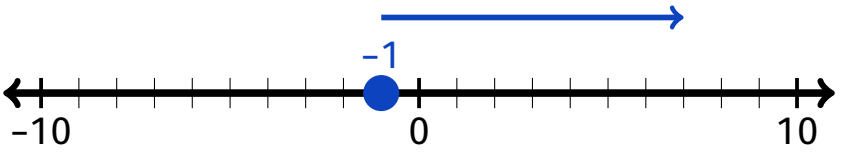
d. 6 less than 4



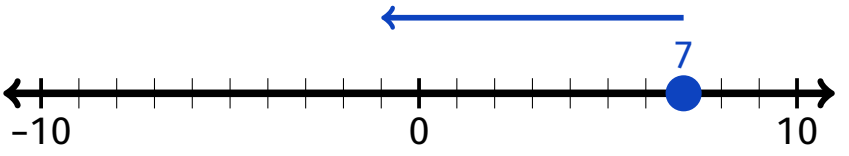
e. 3 less than –5



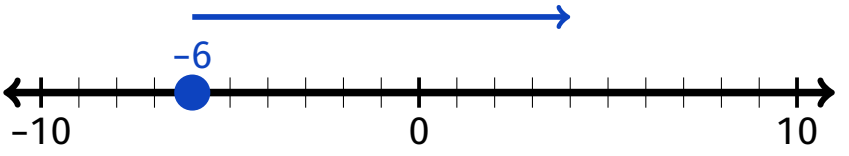
f. 8 more than –1



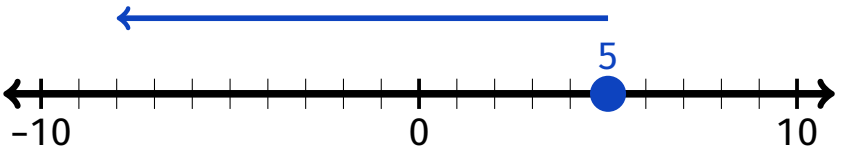
g. 8 less than 7



h. 10 more than –6



i. 13 less than 5



2. Find the indicated value. If you get stuck, try plotting points on a number line.

a. 1 more than –2

b. 3 less than –1

c. 4 more than –2

d. 6 more than –9

e. 9 less than 0

f. 12 more than –7

g. 10 less than –5

h. 11 less than 2

i. 13 more than –3

j. 14 less than –5

k. 17 less than 4

l. 23 more than –6

3. Plot each pair of points on a number line and compare them using <, > or =.

a. 3 and 4

b. 0 and 1

c. –1 and 0

d. 1 and –1

e. –3 and 3

f. –2 and –2

g. –4 and –5

h. –8 and 2

i. –9 and –1

j. 5 and –6

k. 6 and –5

l. –3 and –7

4. Compare each pair of integers using <, > or =.

a. 3 and 2

b. 4 and 0

c. –4 and 0

d. –2 and 2

e. 6 and –3

f. –7 and –8

g. –12 and –12

h. –10 and –11

i. –21 and 21

j. –16 and 8

k. –23 and –17

l. –26 and –27

Crater Lake, in Oregon, is the deepest lake in the United States.

Image credit: Jeff Hopper

5. The deepest lake in the world is Lake Baikal in Russia, which plunges to a depth of 5,387 ft (-5,387 ft). The deepest lake in the United States is Crater Lake in Oregon with a depth of 1,949 ft (-1,949 ft). Compare the depths of Lake Baikal and Crater Lake using <, > or =.

6. **Based on their relative positions** on the number line, we can make a comparison between two integers. Complete the following "if-then" constructs with a comparison using <, > or =:

   a. If n is to the right of m on the number line, then...

   b. If n is to the left of m on the number line, then...

   c. If r and s are both positive and r is closer to zero, then...

   d. If r and s are both negative and r is closer to zero, then...

   e. If x and y are both positive and x is farther from zero, then...

   f. If x and y are both negative and x is farther from zero, then...

Application: *Guess the Number* in Python

Great games give players feedback with which they can modify their gameplay. In a *Guess the Number* game, this means giving the user an opportunity to guess again. In Python, we can repeat a set of commands **while** a given condition is true:

```
while (guess != number):
    guess = input("What's the number? ")
```

With the code above, we allow the user to input a guess while that guess is *not* equal to ("`!=`") a previously determined number. If this code is to work, however, we must first define both the guess and the number, such that it is not possible for the guess to equal the number until the user makes it so:

```
number = random.randint(-10, 10)
guess = 100
while (guess != number):
    guess = input("What's the number? ")
    guess = int(guess)
```

Since our while loop will only run while the guess is *not* equal to the number, the program will move forward once the guess *is* equal to the number.

* * *

In the following example program, *Guess the Number*, two random integers are generated between -10 and 10. The user is then tasked with calculating the distance between the two integers.

Guess the Number in Python

```
import random

print("Guess the Number")

number = random.randint(-10, 10)
guess = 100

print("I am thinking of a number")
print("between -10 and 10.")

while (guess != number):
    guess = input("What's the number? ")
    guess = int(guess)
    if (guess > number):
        print("Too high.")
    if (guess < number):
        print("Too low.")

print("Correct!  The number was", number)

input("Press ENTER to exit.")
```

Guess the Number comprehension questions

1. Why must we set the variable `guess` equal to 100? List three examples of other numbers that would also work.
2. How is indentation used in a `while` loop?
3. In our previous program, *Distance Game*, how would you use a `while` loop to allow the user to guess a distance until that guess is correct?
4. In our *Guess the Number* program, we print, "Correct!" without first testing whether the guess is equal to the number. Why is it not necessary to test whether the guess is equal to the number?

Challenge!

After completing *Guess the Number* in Python, try these additional challenges:

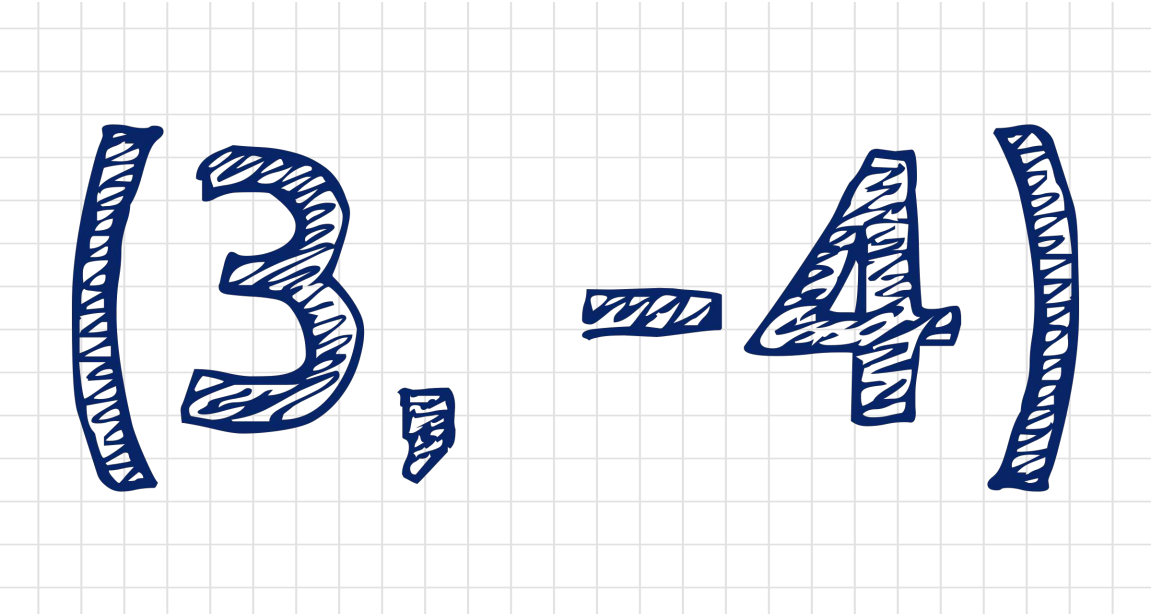
1. Change the message the user receives with a correct guess.
2. Make the game more difficult by increasing the range of possible random integers.
3. Use a variable to track the number of guesses made by the user and display the value at the end of the game.

4. Write a new program that rolls a 6-sided die (generates a random integer between 1 and 6) until the result is a 6. Display the result of each roll and the number of total rolls.

5. It is possible to break *Guess the Number* by entering a value that is not an integer. For a solution to this problem, see *Extension: Guess the Number in Python* in the appendix.

4

Coordinate Systems



A location on a coordinate plane might be described by negative numbers.

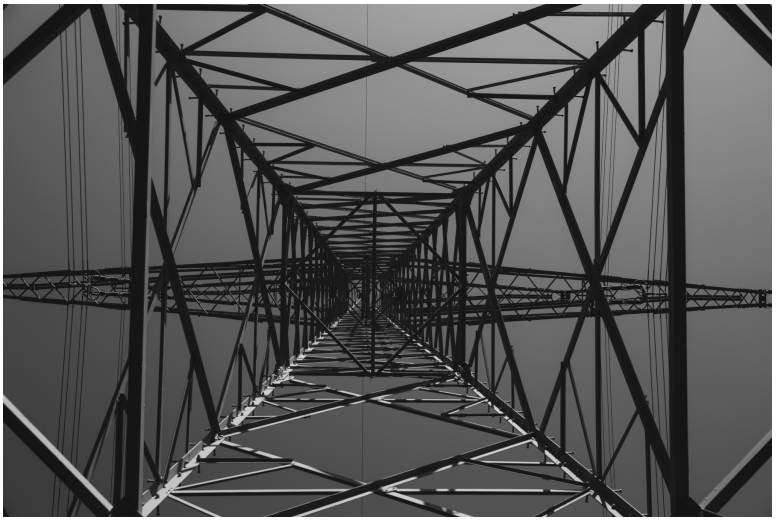
Introduction

Coordinate Systems

Geometry can be seen in the world around us: our buildings are made strong by the rigidity of triangles; beehives are built with hexagons; our roads appear to converge at a point as they stretch away from us toward the horizon. While it is possible to describe these geometric figures with words, it is useful to describe them with numbers. For this, we use coordinates.

Every point in space can be described by a set of coordinates. Points on a plane, or flat surface, can be described using two coordinates.

In the classic example of a coordinate system, the first coordinate gives a horizontal position and the second coordinate gives a vertical position.



Triangles are used in construction for their rigidity and strength.

Image credit: Markus Spiske

What is a coordinate system?

- A **coordinate system** uses numbers, or **coordinates**, to describe locations.
- The **coordinate plane** consists of a horizontal number line, the ***x*-axis**, and a vertical number line, the ***y*-axis**.
- The *x*- and *y*-axes extend to positive and negative infinity.
- Points on the coordinate plane are given in **ordered pairs** of the form (x, y).
- The point where the *x*-axis and *y*-axis intersect is called the **origin**.
- The coordinates at the origin are $(0, 0)$.

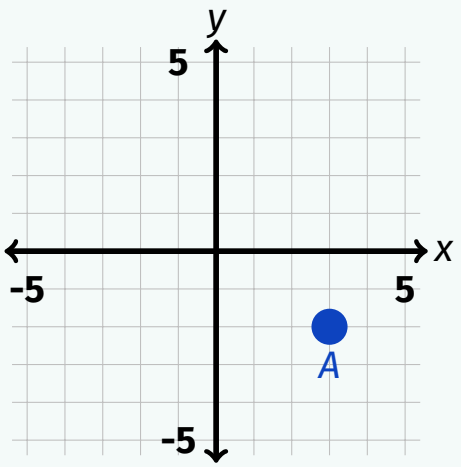
How are coordinate systems used in computer science?

The positions of graphical elements in a computer program or objects in a video game can be described by coordinates. These coordinates are used to calculate movements or determine interactions with other elements or objects.

Examples

x- and *y*-coordinates

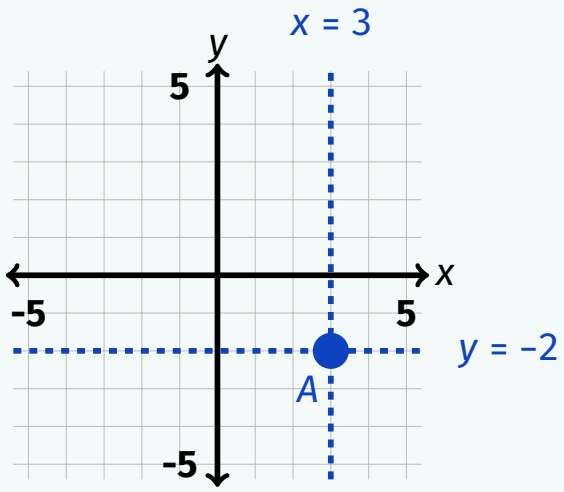
Q: Classify the *x*- and *y*-coordinates of point *A* as positive, negative or zero.



A: The *x*-coordinate of point *A* is positive. The *y*-coordinate of point *A* is negative.

Ordered pairs

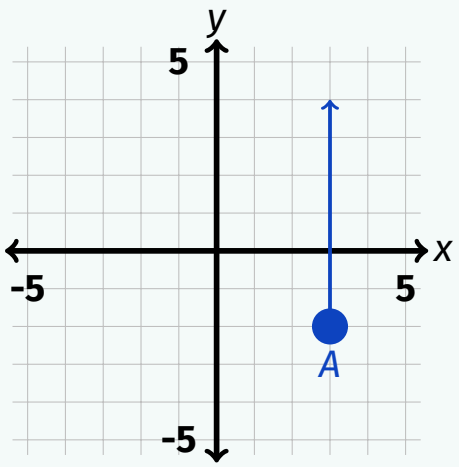
Q: What are the coordinates of point *A*?



A: The coordinates of point *A* are (3, –2).

Ordered pair translations

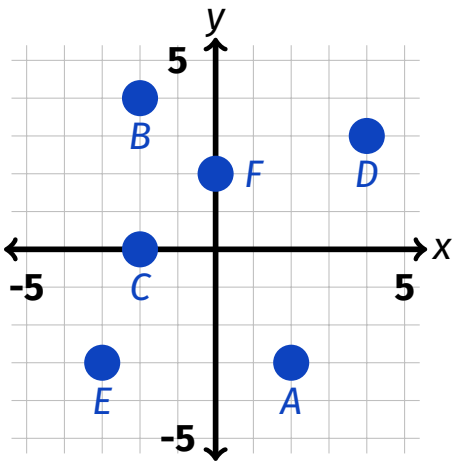
Q: Translate point *A* up 6 units.



A: If point *A* were translated up 6 units, its new location would be $(3, 4)$.

Practice

1. **Classify the *x*- and *y*-coordinates of each point as positive, negative or zero.**



   a. The *x*-coordinate of *A* is [*positive*, *negative*, *zero*].
   The *y*-coordinate of *A* is [*positive*, *negative*, *zero*].

   b. The *x*-coordinate of *B* is [*positive*, *negative*, *zero*].
   The *y*-coordinate of *B* is [*positive*, *negative*, *zero*].

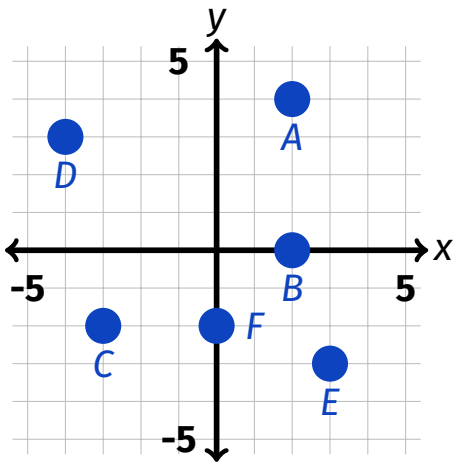
   c. The *x*-coordinate of *C* is [*positive*, *negative*, *zero*].
   The *y*-coordinate of *C* is [*positive*, *negative*, *zero*].

   d. The *x*-coordinate of *D* is [*positive*, *negative*, *zero*].
   The *y*-coordinate of *D* is [*positive*, *negative*, *zero*].

   e. The *x*-coordinate of *E* is [*positive*, *negative*, *zero*].
   The *y*-coordinate of *E* is [*positive*, *negative*, *zero*].

   f. The *x*-coordinate of *F* is [*positive*, *negative*, *zero*].
   The *y*-coordinate of *F* is [*positive*, *negative*, *zero*].

2. Write the coordinates of each point as an ordered pair.



a. *A* is located at _____ .

b. *B* is located at _____ .

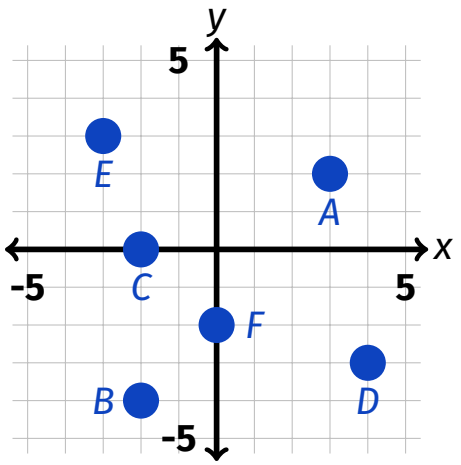
c. *C* is located at _____ .

d. *D* is located at _____ .

e. *E* is located at _____ .

f. *F* is located at _____ .

3. Perform the indicated translations. Write the new location as an ordered pair.



a. Translate *A* up 2 units.

b. Translate *B* right 2 units.

c. Translate *C* down 4 units.

d. Translate *D* left 4 units.

e. Translate *E* right 6 units.

f. Translate *F* up 3 units.

g. Translate *A* left 3 units and down 2 units.

h. Translate *B* left 1 unit and up 5 units.

i. Translate *C* right 7 units and up 5 units.

j. Translate *D* left 4 units and up 2 units.

k. Translate *E* left 1 unit and down 3 units.

l. Translate *F* right 4 units and up 6 units.

4. **Describe the process** of translating a point right or left. Which coordinate (*x* or *y*) changes? How does it change?

5. **Describe the process** of translating a point up or down. Which coordinate (*x* or *y*) changes? How does it change?

6. **Perform the indicated translations. Write the new location as an ordered pair.**

   a. Translate the point (2, 1) up 3 units.

   b. Translate the point (–3, –2) right 3 units.

   c. Translate the point (0, –1) down 4 units.

   d. Translate the point (3, –4) left 5 units.

   e. Translate the point (5, 0) right 6 units.

   f. Translate the point (2, –6) up 7 units.

   g. Translate the point (–10, 3) left 2 units and down 3 units.

   h. Translate the point (1, 8) left 1 unit and up 3 units.

   i. Translate the point (0, 21) right 10 units and up 10 units.

   j. Translate the point (–5, –5) left 5 units and up 5 units.

   k. Translate the point (0, 0) left 16 units and down 12 units.

   l. Translate the point (7, –8) right 13 units and up 17 units.

7. **A game object moves** 36 pixels toward the left side of the screen. If the game object was initially located at the origin, what are the coordinates of its new location?

8. **A game object moves** from the location (–240, 120) to the location (–60, –180). Describe the movement of the game object: did it move up or down? Left or right? How far?

Application: *Draw a Cube* in Python

We can use the **turtle module** and our knowledge of the coordinate plane to draw graphics in Python:

```
import turtle
```

We can instruct our ”turtle” to **go to** different locations on the coordinate plane:

```
turtle.goto(-50, 50)
```

The above code will move our turtle to a position left of and above the origin.

* * *

As our turtle moves, it will draw along its path. To move the turtle without drawing, we can lift our **pen up** before the turtle moves, then replace the **pen down** after that move:

```
turtle.penup()
turtle.goto(-50, 50)
turtle.pendown()
```

* * *

To improve the readability of our code, we can use **comments** to describe each section:

```
#move the pen to the top left corner
turtle.penup()
turtle.goto(-50, 50)
turtle.pendown()
```

Comments are visible when reading the source code but have no effect when the program runs.

* * *

In the following example program, *Draw a Cube*, we use turtle graphics to draw a three-dimensional cube.

Draw a Cube in Python

```
import turtle

#move the pen to the top left corner
turtle.penup()
turtle.goto(-50, 50)
turtle.pendown()

#draw the front face of the cube
turtle.goto(50, 50)
turtle.goto(50, -50)
turtle.goto(-50, -50)
turtle.goto(-50, 50)

#make it three-dimensional
turtle.goto(0, 100)
turtle.goto(100, 100)
turtle.goto(100, 0)
turtle.goto(50, -50)

#one more line
turtle.penup()
turtle.goto(50, 50)
turtle.pendown()
turtle.goto(100, 100)

turtle.hideturtle()
turtle.exitonclick()
```

Draw a Cube comprehension questions

1. Using the turtle module, how would you move the turtle to the origin?
2. How would you move the turtle without drawing its path?
3. How would you include a note in your code that will have no effect on the program when it runs?

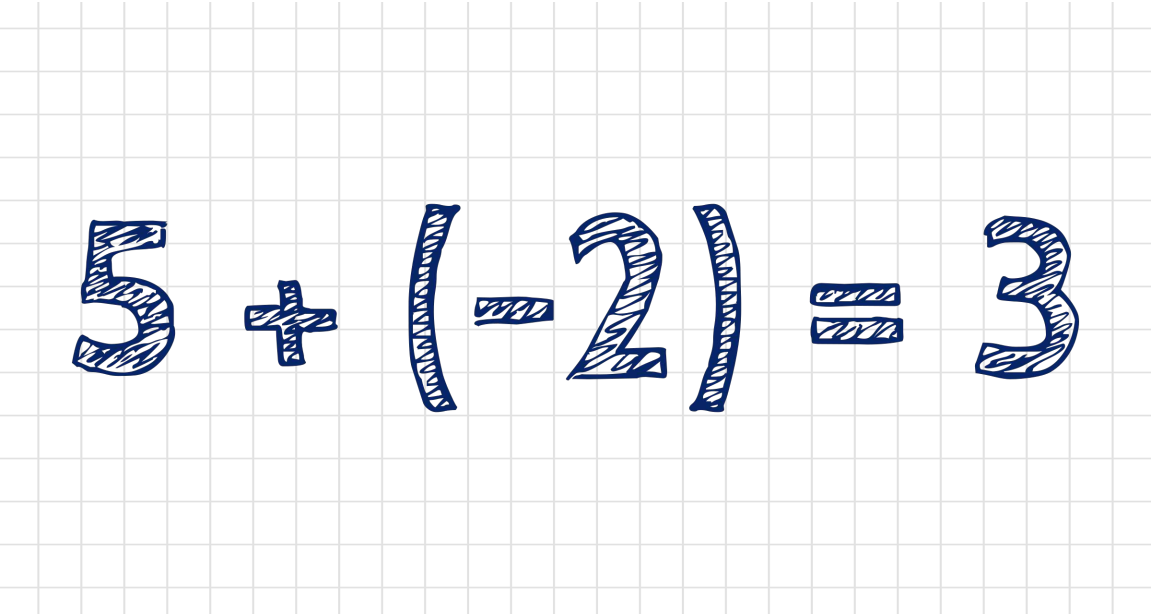
Challenge!

After completing *Draw a Cube* in Python, try these additional challenges:

1. Make the cube larger by adjusting the coordinates in the code.
2. Use `turtle.fillcolor("pink")`, `turtle.begin_fill()` and `turtle.end_fill()` to make the cube pink.
3. Write a new program that uses the turtle module to draw your initials.

5

Adding and Subtracting Negative Numbers



Adding negative numbers is like subtracting positive numbers.

Introduction

Adding and Subtracting Negative Numbers

Money, temperature and elevation can all be counted with negative numbers. When working with larger amounts, it makes sense to use counting shortcuts like addition and subtraction. How can we think about adding and subtracting negative numbers?

If we count the money we have with positive numbers, we can count the money we owe, debt, with negative numbers. Adding debt to the money we have would result in a decrease in our wealth, so adding negative numbers is like subtracting positive numbers. Subtract-

ing debt from the money we have would result in an increase in our wealth, so subtracting negative numbers is like adding positive numbers.

A five dollar bill from 1899. When we spend more money than we have, we acquire debt, a negative amount of money.

Image credit: National Numismatic Collection at the Smithsonian Institution

How are negative numbers added and subtracted?

Adding and subtracting negative numbers can be summarized with the following rules:

- Adding a positive results in a move to the right on the number line
- Adding a negative results in a move to the left on the number line
- Subtracting a positive results in a move to the left on the number line
- Subtracting a negative results in a move to the right on the number line

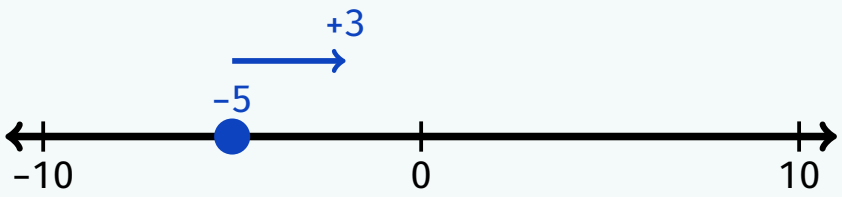
How are negative numbers added and subtracted in computer science?

In a video game, the position of an object in a particular dimension (x, y or, in a three-dimensional game, z) might be negative, in which case it would be necessary to add or subtract negative numbers to calculate changes to that object's position.

Examples

Adding negative numbers

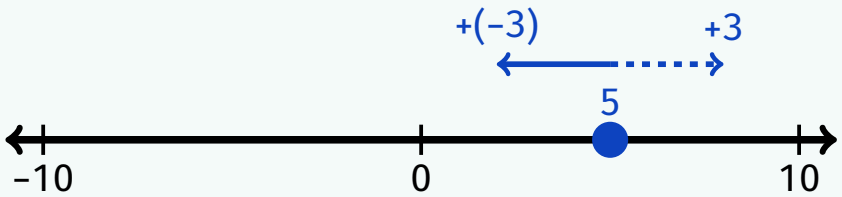
Q: Evaluate: −5 + 3



A: Adding a positive results in a move to the right on the number line. Thus:

$$-5 + 3 = -2$$

Q: Evaluate: 5 + (−3)

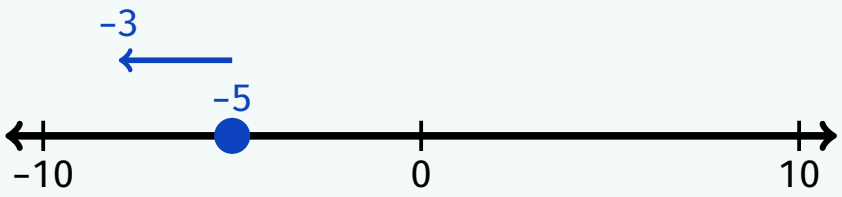


A: Adding a negative results in a move to the left on the number line. Thus:

$$5 + (-3) =$$
$$5 - 3 = 2$$

Subtracting negative numbers

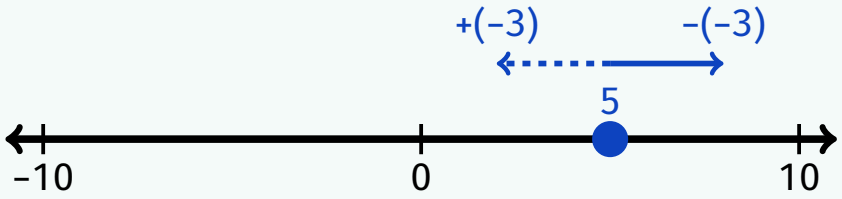
Q: Evaluate: –5 – 3



A: Subtracting a positive results in a move to the left on the number line. Thus:

$$-5 - 3 = -8$$

Q: Evaluate: 5 – (–3)



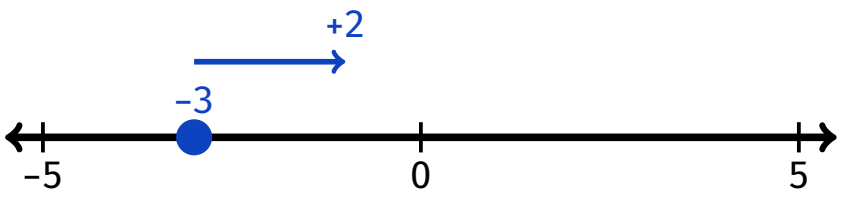
A: Subtracting a negative results in a move to the right on the number line. Thus:

$$5 - (-3) =$$
$$5 + 3 = 8$$

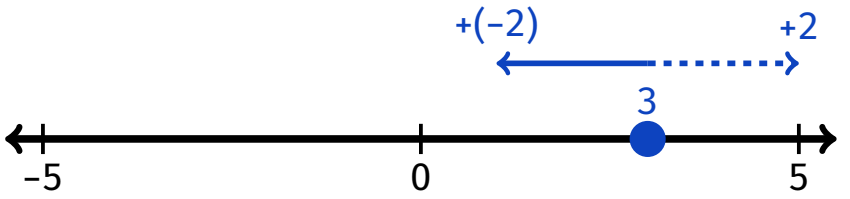
Practice

1. Use the number lines to find each sum.

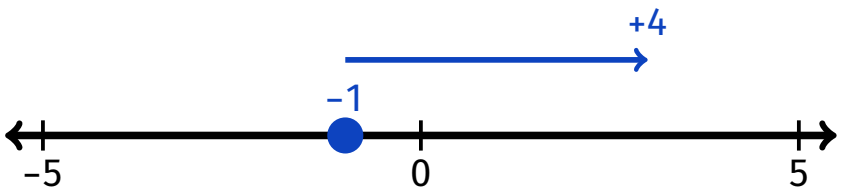
a. −3 + 2



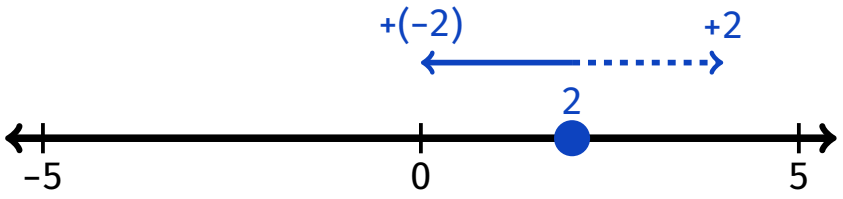
b. 3 + (−2)



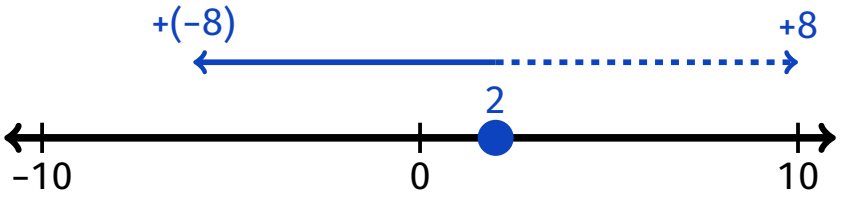
c. −1 + 4



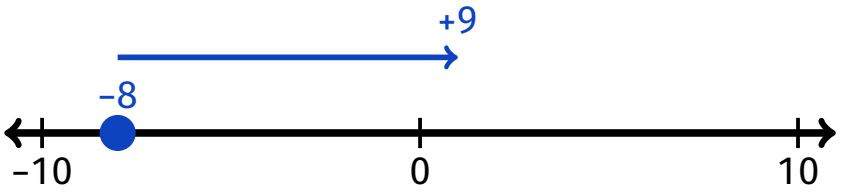
d. 2 + (−2)



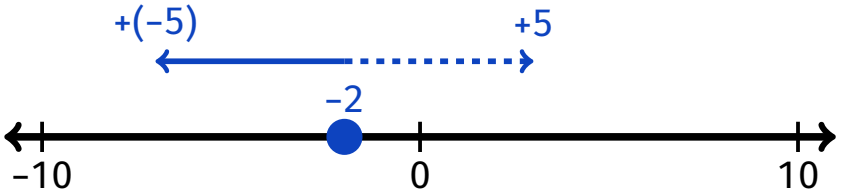
e. 2 + (−8)



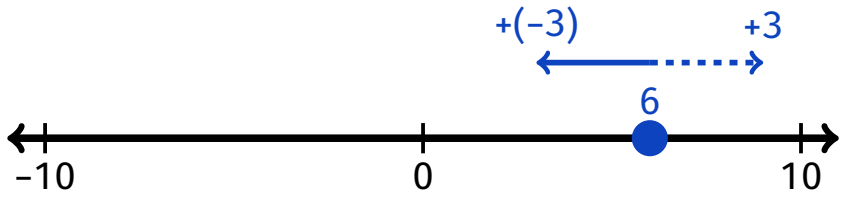
f. −8 + 9



g. −2 + (−5)



h. 6 + (−3)



2. Find the sum. If you get stuck, try plotting points on a number line.

a. −1 + 1

b. −2 + 4

c. 3 + (−1)

d. 4 + (−3)

e. −3 + (−5)

f. −7 + 6

g. 8 + (−9)

h. −4 + 12

i. 6 + (−10)

j. −11 + 5

k. 17 + (−8)

l. 13 + (−20)

m. −22 + 27

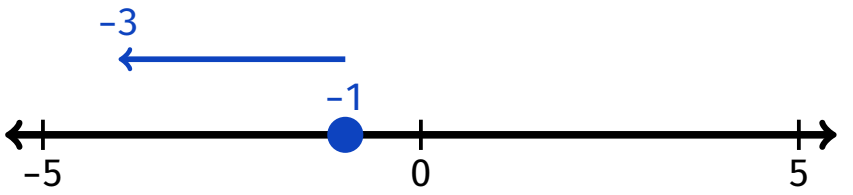
n. 15 + (−30)

o. 21 + (−32)

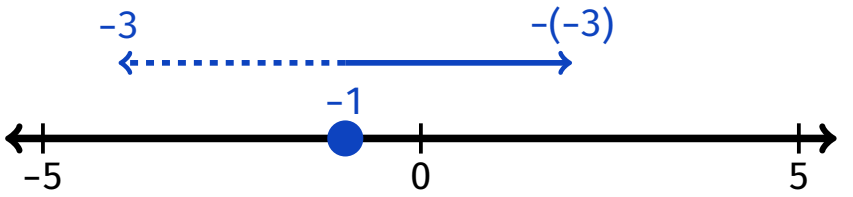
3. Sperm whales are known to dive incredibly deep in search of squid. Imagine that a sperm whale dives to a depth of 2,100 ft. Still searching for squid, it dives a further 1,080 ft to its maximum observed diving depth. Use integer addition to find the maximum observed diving depth of a sperm whale.

4. Use the number lines to find each difference.

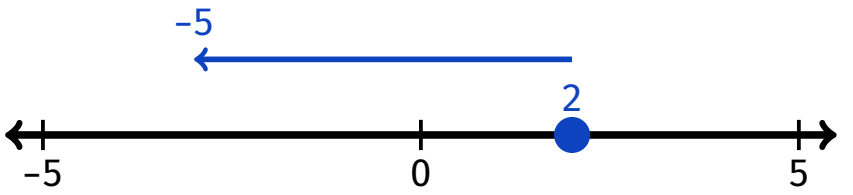
a. −1 − 3



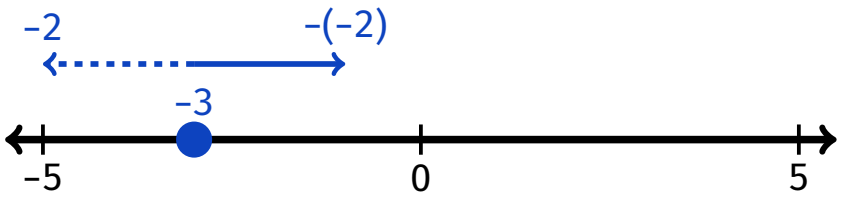
b. −1 − (−3)



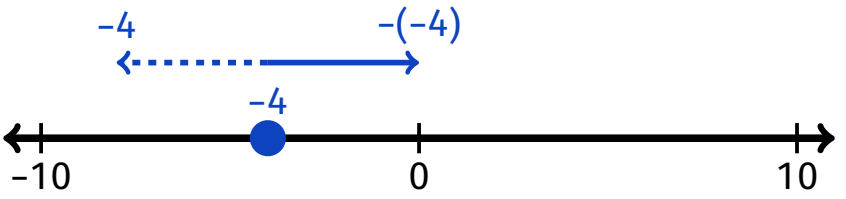
c. 2 − 5



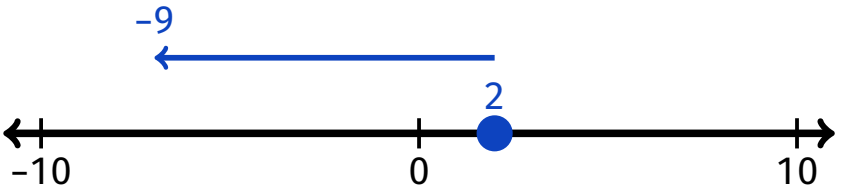
d. −3 − (−2)



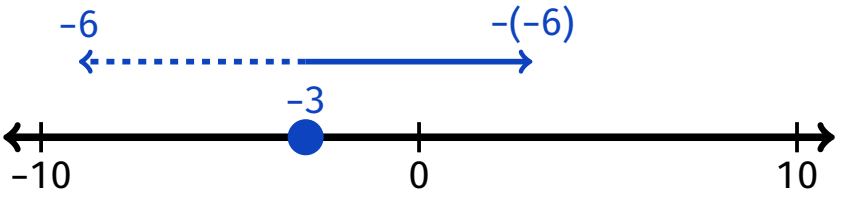
e. −4 − (−4)



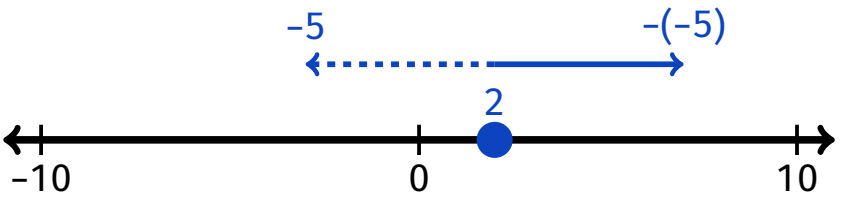
f. 2 − 9



g. −3 − (−6)



h. 2 − (−5)



5. Find the difference. If you get stuck, try plotting points on a number line.

a. –1 – 1	**f.** 8 – (–2)	**k.** 19 – (–9)
b. –1 – (–1)	**g.** –9 – (–8)	**l.** 12 – 22
c. 2 – (–3)	**h.** –4 – 7	**m.** –20 – 14
d. 5 – (–4)	**i.** 6 – (–10)	**n.** 15 – (–30)
e. –5 – 4	**j.** –5 – 11	**o.** –24 – (–32)

6. Javier owes his friend, Sarah, $13 for a movie ticket she bought for him last week. Today, Javier paid for Sarah's lunch, which cost $9. Use integer subtraction to find Javier's debt to Sarah after lunch.

7. **Mixed practice. If you get stuck, try plotting points on a number line.**

   a. 4 + (–5)
   b. 4 – (–5)
   c. 2 + (–7)
   d. 7 + (–2)
   e. –3 – 8
   f. –9 – (–9)
   g. 8 – 13
   h. 8 – (–13)
   i. 16 + (–11)
   j. 5 + (–18)
   k. –15 + 45
   l. 15 – 45
   m. 17 – (–17)
   n. 24 + (–33)
   o. –40 – (–27)

8. **A computer programmer uses** the following code to display the value of the variable `answer`:

```
answer = 10 - (-12)
print(answer)
```

   After running this code, what value will be displayed?

9. **The average annual temperature** at the South Pole in Antarctica is -57 °F. The average annual temperature in Ushuaia, Argentina, the southernmost city in the world, is 98 degrees warmer than the South Pole. Use integer addition to find the average annual temperature in Ushuaia.

Application: *Integer Game* in Python

Rather than repeating a bit of code *while* a condition is true, sometimes it is necessary to loop through code a set number of times. In that case, we can use a **for** loop:

```
for x in range(5):
    print(x)
```

With this code, we use the variable `x` to count up from 0 until we reach 5, resulting in the following output:

```
0
1
2
3
4
```

This for loop stops when it reaches 5, so it does not print ”5” on the screen. A for loop like this one could be used in a turn-based game where the number of turns is limited to 5.

* * *

At times, we may want to use text input from a user to determine the outcome of a program. For example:

```
operation = input("Add or subtract? ")
if operation == "add":
    print(2 + 2)
else:
    print(2 - 2)
```

In this example, we ask the user for input and store that input as `operation`. If `operation` is equal to "`add`", then we print the sum of 2 and 2. Otherwise, we print the difference of 2 and 2.

The problem with this example is that it does not account for capitalization: if the user enters "`Add`" or "`ADD`", then `operation` will not be equal to "`add`", since Python is case-sensitive. To address this, we can convert any input to lowercase:

```
operation = "ADD"
operation = operation.lower()
print(operation)
```

Here, we use the built-in string method **lower()** to convert the string variable `operation` to lowercase and print the result, "`add`", on the screen.

* * *

In the following example program, *Integer Game*, we give the user five turns to get a high score. On each turn, we generate two random integers. The user decides whether to add or subtract the random integers and the result is added to their score.

Integer Game in Python

```
import random

print("Integer Game")
score = 0

for x in range(5):
    A = random.randint(-10, 10)
    B = random.randint(-10, 10)

    print(A, "and", B)

    operation = input("Add or subtract? ")
    operation = operation.lower()

    if operation == "add":
        print(A, "+", B, "=", A+B)
        score = score + A + B
    else:
        print(A, "-", B, "=", A-B)
        score = score + A - B

    print("Your score is now", score)
    print("You have", 4-x, "turns remaining")

input("Press ENTER to exit.")
```

Integer Game comprehension questions

1. How would you use a `for` loop to print "`Hello, World!`" exactly 100 times?
2. How would you use a `for` loop to print every integer from 0 to 100? From 1 to 100?
3. In the *Integer Game* program, why was it necessary to convert `operation` to lowercase?

Challenge!

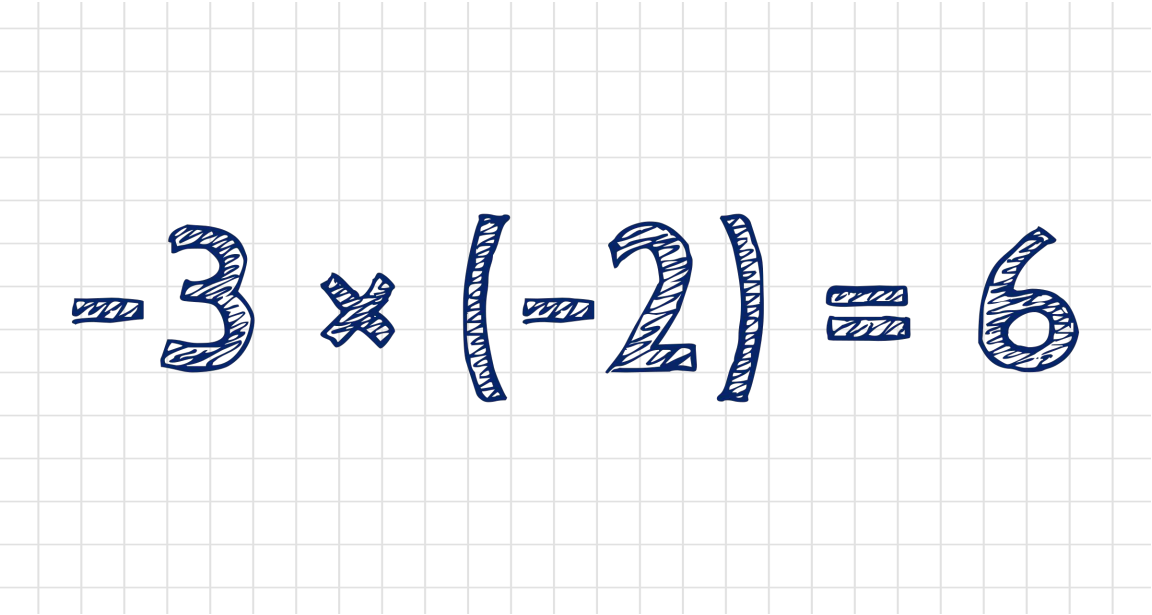
After completing *Integer Game* in Python, try these additional challenges:

1. When the game ends, display the text, "`Game Over`".
2. Use `upper()` to convert `operation` to uppercase instead of lowercase (remember to change the `if` statement).
3. Make the game last 10 turns. Be sure to display the number of turns remaining correctly.
4. Write a program in which the user can roll a 6-sided die 10 times. Display the result of each roll and the total of all 10 rolls.
5. Though we instruct the user, in *Integer Game*, to enter "`add`" or "`subtract`", there is no mechanism for preventing other en-

tries. For a solution to this problem, see *Extension: Integer Game in Python* in the appendix.

6

Multiplying and Dividing Negative Numbers



Subtracting three groups of negative two makes positive six.

Introduction

Multiplying and Dividing Negative Numbers

Multiplication is a shortcut for repeated addition: when we need to add multiples of the same number, we can use multiplication. And, since multiplication is simply a shortcut for repeated addition, it follows that multiplying negative numbers is similar to adding negative numbers.

Every month, many people receive bills for things like electricity, water or internet service. Each of these bills implies a debt, or money owed. If we receive 5 bills for $10 each, then we would owe

a total of $50. Thus, multiplying a positive number (5 bills) and a negative number (–$10) results in a negative number (–$50). On the other hand, if we send out 5 bills for $10 each, we would receive $50. Thus, multiplying two negative numbers (–5 bills, –$10) results in a positive number ($50).

Dividing negative numbers works much the same way. If we receive 5 bills for a total of $50, then we would owe an average of $10 on each bill. Thus, dividing a negative number (–$50) and a positive number (5 bills) results in a negative number (–$10). Finally, if we send out 5 bills for a total of $50, we would receive an average of $10 for each bill. Thus, dividing two negative numbers (–$50, –5 bills) results in a positive number.

Bills for utilities such as electricty indicate a debt, represented by a negative number.

Image credit: Carol M. Highsmith

How are negative numbers multiplied and divided?

Multiplying and dividing negative numbers can be summarized with the following rules:

- Multiplying a negative and a positive results in a negative
- Multiplying two negatives results in a positive
- Dividing a negative and a positive results in a negative
- Dividing two negatives results in a positive

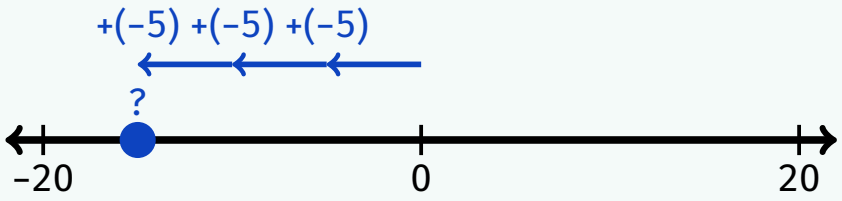
How are negative numbers multiplied and divided in computer science?

Since computer programs are often used to model concepts that can be described by negative numbers like financial transactions or climate, it is common to multiply or divide negative numbers as well.

Examples

Multiplying negative numbers

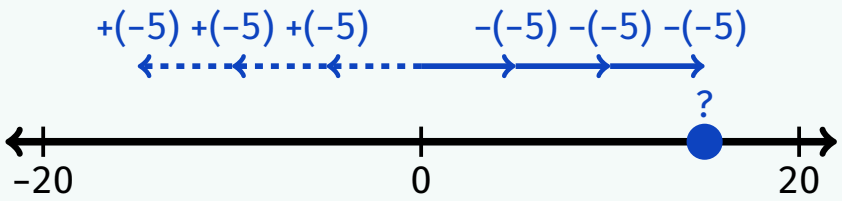
Q: Evaluate: 3 × (–5)



A: Multiplying 3 and –5 means adding 3 groups of –5. Adding negatives results in a move to the left on the number line. Thus:

$$3 \times (-5) = -15$$

Q: Evaluate: –3 × (–5)

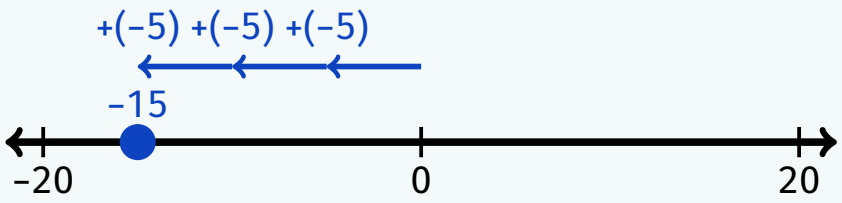


A: Multiplying –3 and –5 means subtracting 3 groups of –5. Subtracting negatives results in a move to the right on the number line. Thus:

$$-3 \times (-5) = 15$$

Dividing negative numbers

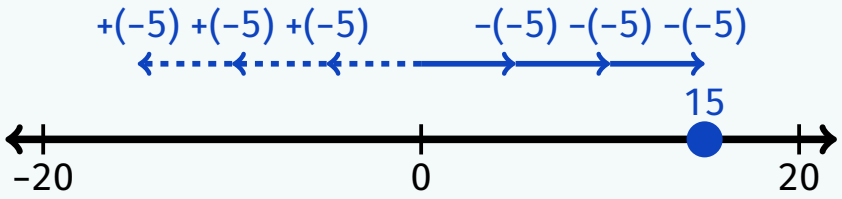
Q: Evaluate: −15 ÷ (−5)



A: Dividing −15 and −5 means counting groups of −5 until we get to −15. Since *adding* negatives results in a move to the left on the number line (toward −15), our quotient will be *positive*. Thus:

$$-15 \div (-5) = 3$$

Q: Evaluate: 15 ÷ (−5)



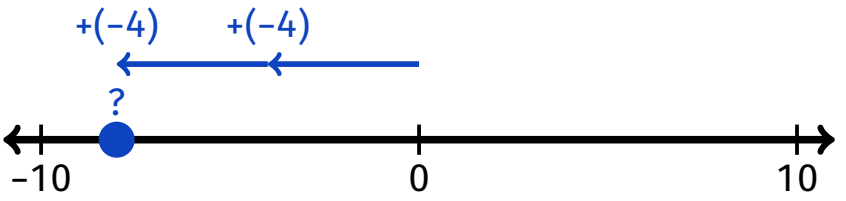
A: Dividing 15 and −5 means counting groups of −5 until we get to 15. Since *subtracting* negatives results in a move to the right on the number line (toward 15), our quotient will be *negative*. Thus:

$$15 \div (-5) = -3$$

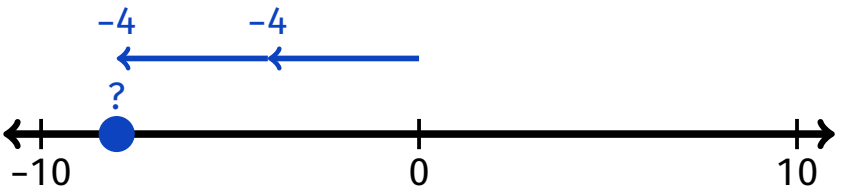
Practice

1. Use the number lines to find each product.

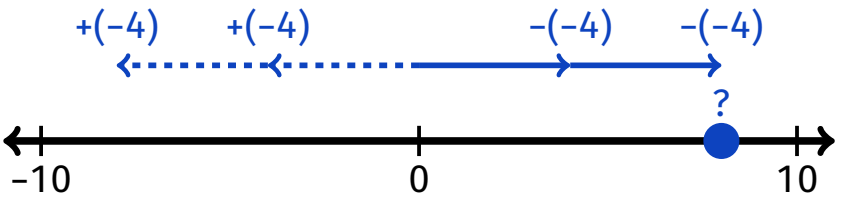
a. $2 \times (-4)$



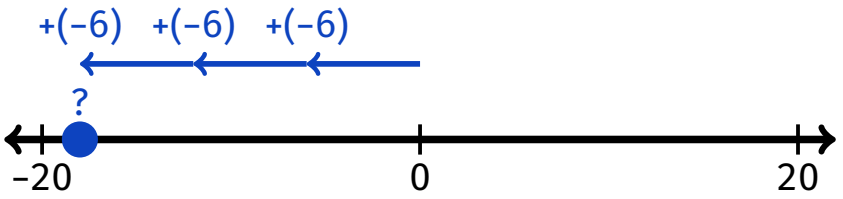
b. -2×4



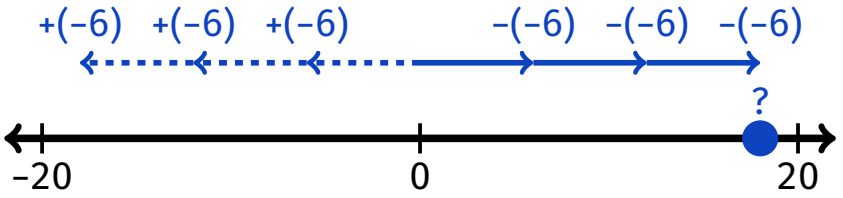
c. $-2 \times (-4)$



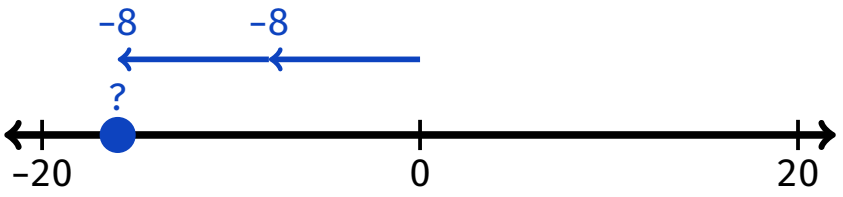
d. 3 × (−6)



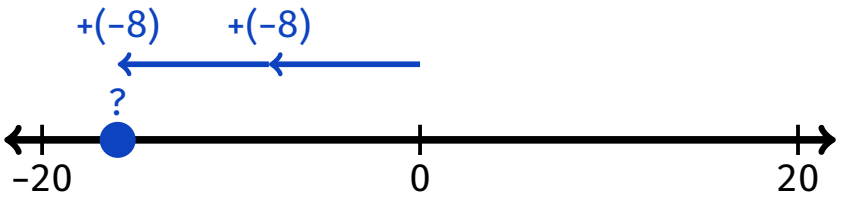
e. −3 × (−6)



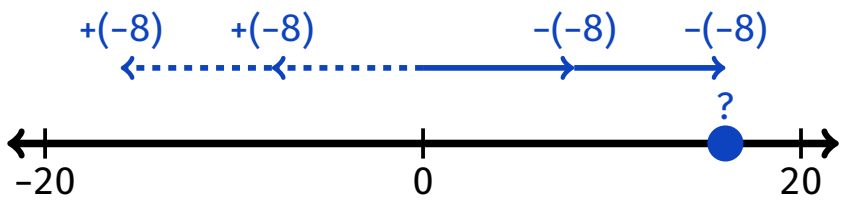
f. −2 × 8



g. 2 × (−8)



h. $-2 \times (-8)$



2. Find the product. If you get stuck, try plotting points on a number line.

a. -1×1

b. -3×4

c. $3 \times (-4)$

d. $-3 \times (-4)$

e. $-4 \times (-3)$

f. -2×6

g. $6 \times (-2)$

h. $-6 \times (-4)$

i. $4 \times (-10)$

j. -8×7

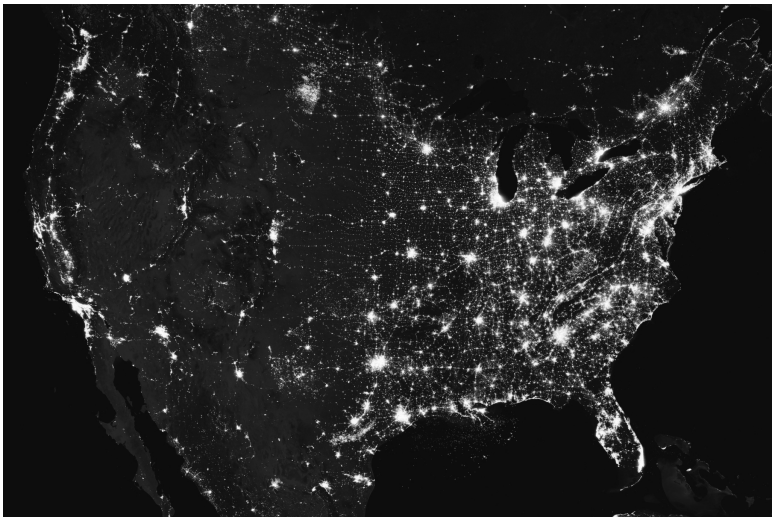
k. $-3 \times (-8)$

l. $5 \times (-9)$

m. $-20 \times (-3)$

n. $15 \times (-4)$

o. $-12 \times (-10)$



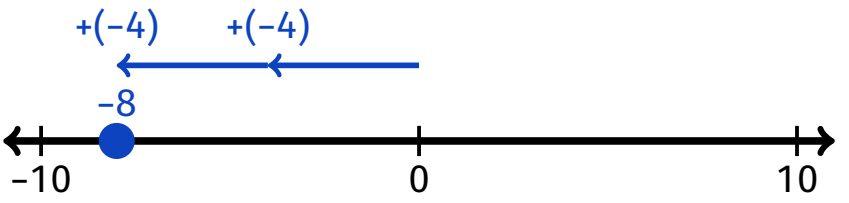
The United States at night, taken in 2012.

Image credit: NASA

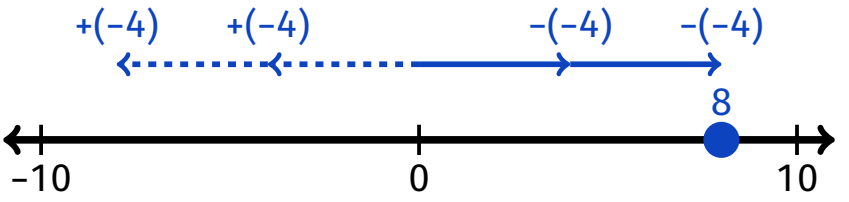
3. **Overnight, the temperature drops** three degrees Fahrenheit every hour for eight hours. Use integer multiplication to find the total change in temperature overnight.

4. Use the number lines to find each quotient.

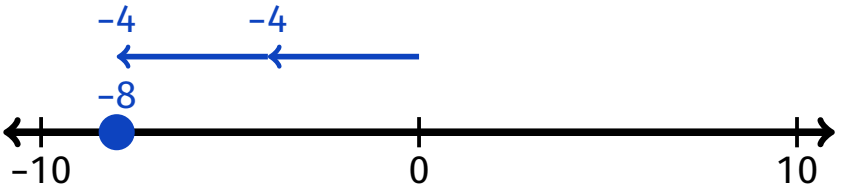
a. $-8 \div (-4)$



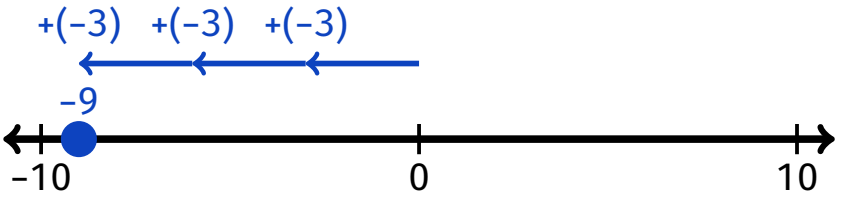
b. $8 \div (-4)$



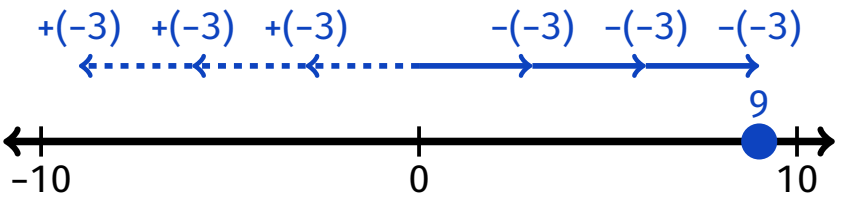
c. $-8 \div 4$



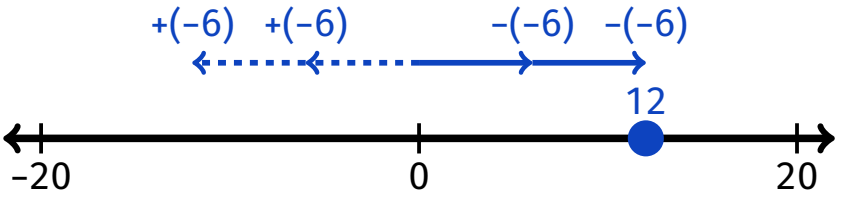
d. $-9 \div (-3)$



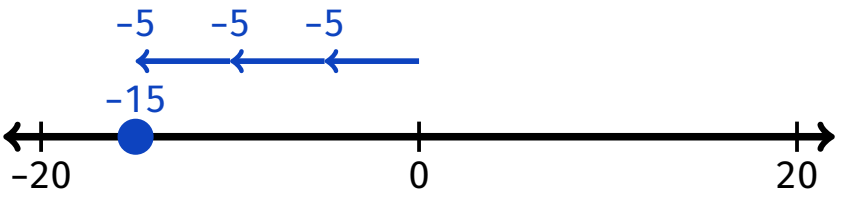
e. 9 ÷ (−3)



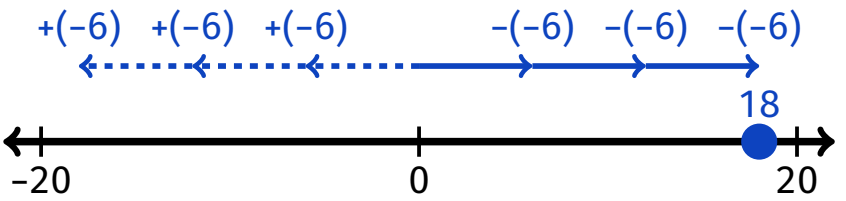
f. 12 ÷ (−6)



g. −15 ÷ 5



h. 18 ÷ (−6)



5. Find the quotient. If you get stuck, try plotting points on a number line.

a. $-1 \div (-1)$	**f.** $-8 \div 2$	**k.** $-20 \div (-4)$
b. $-1 \div 1$	**g.** $12 \div (-3)$	**l.** $21 \div (-7)$
c. $1 \div (-1)$	**h.** $-12 \div 3$	**m.** $-24 \div (-8)$
d. $-8 \div (-2)$	**i.** $-15 \div (-3)$	**n.** $30 \div (-15)$
e. $10 \div (-5)$	**j.** $-20 \div 5$	**o.** $-42 \div (-14)$

6. On Thursday, it is twenty-eight degrees (°F) colder than it was last Thursday. Use integer division to find the average temperature change per day over the past 7 days.

7. Jaime's mother loans him $65 to buy a new skateboard. Since Jaime earns $5 every week by doing chores, he is able to pay off his debt by $5 per week. Use integer division to find the number of weeks it will take Jaime to pay off his debt.

8. Mixed practice. If you get stuck, try plotting points on a number line.

a.	$-1 \times (-1)$	**f.**	$16 \div (-2)$	**k.**	$-15 \times (-3)$
b.	-1×10	**g.**	$-12 \div 4$	**l.**	$45 \div (-3)$
c.	$-10 \div (-1)$	**h.**	-3×4	**m.**	$45 \div (-15)$
d.	$-8 \times (-2)$	**i.**	$-10 \times (-3)$	**n.**	$32 \div (-8)$
e.	$16 \div (-8)$	**j.**	$-30 \div (-10)$	**o.**	$-8 \times (-4)$

9. Complete the following "if-then" constructs regarding products and quotients of integers:

a. If two integers are negative, then their product will be...

b. If one integer is positive and another is negative, then their quotient will be...

c. If the product of two integers is positive, then...

d. If the quotient of two integers is negative, then...

Application: *Funny Face* in Python

Python's turtle module contains a function for collecting **numerical input** from the user. Here, we generate a dialog window with the title, "Integer" and the prompt, "Enter an Integer" with a default value of 1:

```
A = turtle.numinput("Integer", "Enter an integer", 1)
```

* * *

We can use the turtle module to draw a **circle**:

```
turtle.circle(100)
```

The size of a circle can be determined by its **radius**, or the distance from the center of a circle to a point on the circle.

The code above will draw a circle with a radius of 100 at the current location of the turtle.

With variables, we can give a user control over the size of the circle by multiplying the radius. The following code will draw a circle with a radius of 100 × `A`:

```
turtle.circle(100*A)
```

* * *

In the following example program, *Funny Face*, we use turtle graphics to draw a face that depends upon user input.

Funny Face in Python

```
import turtle
turtle.speed(10)

print("Funny Face")

A = turtle.numinput("Face", "Face multiplier?", 1)
B = turtle.numinput("Mouth", "Mouth multiplier?", 1)
C = turtle.numinput("Eye", "Eye multiplier?", 1)

#draw the outline of the face
turtle.penup()
turtle.goto(0, -200*A)
turtle.pendown()
turtle.circle(200*A)

#draw the mouth
turtle.penup()
turtle.goto(-100*B, -40*B)
turtle.pendown()
turtle.setheading(270)
turtle.circle(100*B, 180)
turtle.goto(-100*B, -40*B)
turtle.setheading(0)
```

```
#draw the right eye
turtle.penup()
turtle.goto(60*C, 40*C)
turtle.pendown()
turtle.circle(50*C)
turtle.circle(20*C)

#draw the left eye
turtle.penup()
turtle.goto(-60*C, 40*C)
turtle.pendown()
turtle.circle(50*C)
turtle.circle(20*C)

turtle.hideturtle()
turtle.exitonclick()
```

Funny Face comprehension questions

1. How would you use the turtle module to collect numerical input from a user?
2. How would you use the turtle module to draw a circle with a radius of 20 units?
3. In the *Funny Face* program, how did we allow the user to control the location of the eyes and mouth?

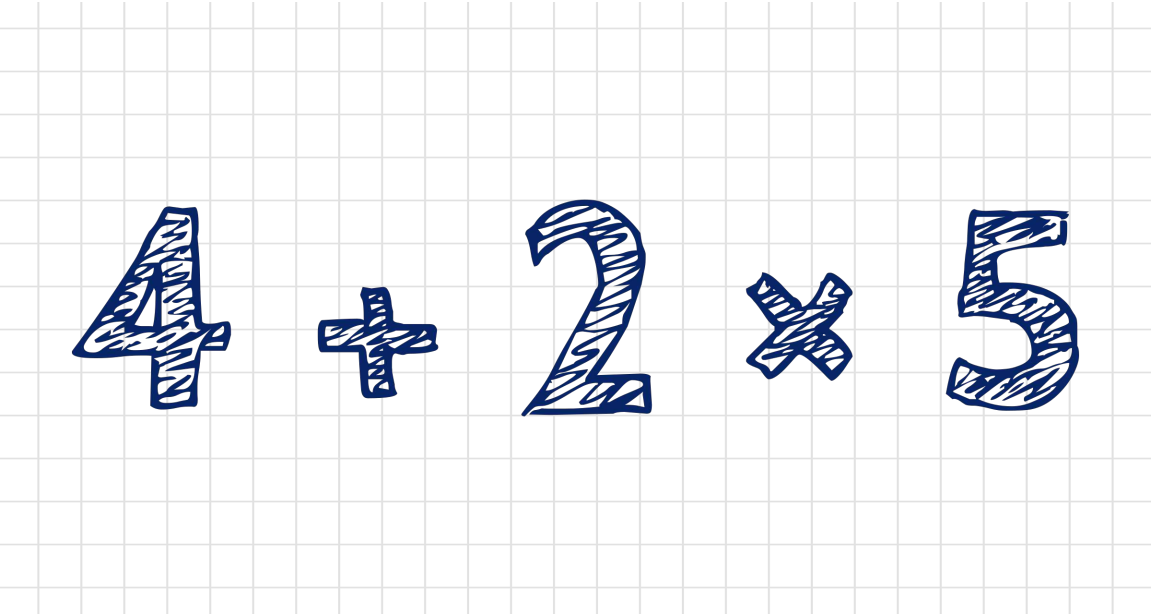
Challenge!

After completing *Funny Face* in Python, try these additional challenges:

1. Add a nose to the *Funny Face* program. Allow the user to change its size and position.
2. Use `turtle.fillcolor()`, `turtle.begin_fill()` and `turtle.end_fill()` to add color to the features drawn with the *Funny Face* program.
3. Write your own program that allows a user to control the height of a building drawn with the turtle module.

7

The Order of Operations



The order of operations dictates that we multiply before we add.

Introduction

The Order of Operations

In many situations, we must take care to perform steps in a particular order: if we want to make it to first base in a game of kickball, we must first kick the ball, then run to first base; if we want to bake a cake, we must first mix the ingredients, then place the cake in the oven; when we go shopping, we must first choose the items we want to buy, then pay for them.

In each of these cases, the order is important: it wouldn't work to run to first base before we kick the ball; we can't place our cake in

the oven before we mix the ingredients; we can't purchase items we haven't added to our cart.

When multiple steps are involved in mathematics, the order is important. For example, multiplying before we add will give us a different answer than adding before we multiply. For this reason, we use a specific order, the order of operations, to solve multi-step math problems.

The steps one takes to bake a cake must be performed in a particular order.

Image credit: Markus Spiske

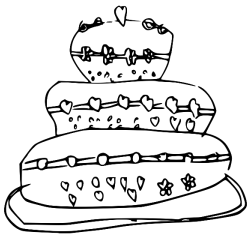


Image credit: Lexi, age 6

What is the order of operations?

Considering addition, subtraction, multiplication and division, operations are evaluated in the following order:

1. Evaluate operations within parentheses
2. Evaluate multiplication and division from left to right
3. Evaluate addition and subtraction from left to right

How is the order of operations used in computer science?

Many computer programming languages, including Python, evaluate mathematical expressions according to the order of operations.

Examples

The order of operations (no parentheses)

Q: Evaluate: $3 + 4 \times 5$

A: The order of operations dictates that multiplication must be evaluated before addition:

$$3 + 4 \times 5 =$$
$$3 + 20 = 23$$

Q: Evaluate: $6 \div 3 \times 2$

A: The order of operations dictates that multiplication and division are evaluated from left to right:

$$6 \div 3 \times 2 =$$
$$2 \times 2 = 4$$

The order of operations (with parentheses)

Q: Evaluate: $(3 + 4) \times 5$

A: The inclusion of parentheses changes the order in which the operations must be evaluated:

$$(3 + 4) \times 5 =$$
$$7 \times 5 = 35$$

Q: Evaluate: $6 \div (3 \times 2)$

A: The inclusion of parentheses changes the order in which the operations must be evaluated:

$$6 \div (3 \times 2) =$$
$$6 \div 6 = 1$$

Practice

1. **Evaluate each expression.**

a. $2 + 3 \times 4$	**f.** $4 \div 2 \times 6$	**k.** $12 + 6 - 6$
b. $2 \times 3 + 4$	**g.** $10 - 10 \div 5$	**l.** $12 - 6 \div 6$
c. $3 \times 4 + 2$	**h.** $10 - 10 + 5$	**m.** $8 \times 8 - 8$
d. $6 \div 2 \times 4$	**i.** $10 + 10 \times 5$	**n.** $8 - 8 \times 8$
e. $6 \times 2 \div 4$	**j.** $12 + 6 \div 6$	**o.** $8 \div 8 \times 8$

2. **Mila is seven years old.** Dustin is older than Mila: his age is four less than three times Mila's age. Write a numerical expression for Dustin's age.

3. **Mitchell is four feet and** seven inches tall. Write a numerical expression for Mitchell's height in inches (one foot is equal to twelve inches).

4. Evaluate each expression.

a. $3 + (-4) \times 5$

b. $3 \times (-4) + 5$

c. $3 + (-4) - 5$

d. $-8 \div 2 \times (-4)$

e. $-8 \times 2 \div (-4)$

f. $-8 \div (-4) \times 2$

g. $12 - (-9) \div 3$

h. $12 - (-9) + 3$

i. $12 - (-9) \times 3$

j. $-15 \div (-5) + 5$

k. $-15 + (-5) \div 5$

l. $-15 - (-5) + 5$

m. $3 \times 14 \div (-7)$

n. $3 + 14 \div (-7)$

o. $3 \times 14 - (-7)$

5. The high temperature in Minneapolis on Tuesday was 21 °F. The temperature dropped 28 degrees overnight before warming 23 degrees the next day. Write a numerical expression for the high temperature in Minneapolis on Wednesday.

6. Evaluate each expression.

a. $(2 + 3) \times 4$	**f.** $(7 - 5) \times 2$	**k.** $3 \times (6 \div 2)$
b. $4 \times (2 + 3)$	**g.** $9 - (5 + 3)$	**l.** $2 \times (6 \div 3)$
c. $2 \times (4 + 3)$	**h.** $9 + (5 - 3)$	**m.** $(16 + 5) \div 7$
d. $7 \times (5 - 2)$	**i.** $9 - (5 - 3)$	**n.** $28 - (11 - 5)$
e. $(5 - 2) \times 7$	**j.** $6 \div (3 \times 2)$	**o.** $9 \times (35 \div 5)$

7. Yosef made cookies for his friends. He counted 19 cookies. Including himself, Yosef intended to divide the cookies evenly among four friends. Thinking quickly, he ate three cookies before his friends arrived and divided the rest into four equal groups. Write a numerical expression for the number of cookies each person receives after Yosef's friends arrive.

8. Evaluate each expression.

a. $(-3 + 4) \times (-5)$

b. $-5 \times (-3 + 4)$

c. $-3 \times (-5 + 4)$

d. $-9 \times (-6 - 2)$

e. $(-6 - 2) \times (-9)$

f. $(-9 - 2) \times (-6)$

g. $8 + (3 - 7)$

h. $8 - (3 - 7)$

i. $-8 - (3 + 7)$

j. $-12 \div (4 \times 3)$

k. $3 \times (-12 \div 4)$

l. $4 \times (-12 \div 3)$

m. $(7 - 34) \div 9$

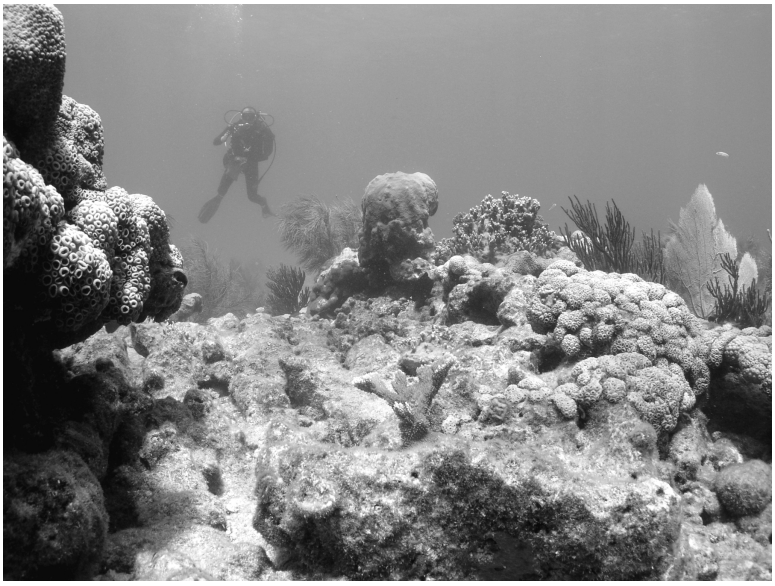
n. $-24 - (5 - 12)$

o. $11 \times (-18 \div 6)$

9. Python evaluates numerical expressions according to the order of operations. Consider the following code:

```
answer = 10 + 3 * (-5) - 4
print(answer)
```

Note that the symbol for division, in Python, is "/".
After running this code, what value is displayed on the screen?



A scuba diver performs research for the U.S. Geological Survey (USGS).

Image credit: Ilsa B. Kuffner, USGS

10. **A scuba diver observes a coral reef** at a depth of 75 feet below the surface of the ocean. To get a better view, she dives an additional 45 feet. Since it is only safe to ascend at a rate of 30 feet per minute, write a numerical expression for the number of minutes required for the scuba diver to surface.

Application: *24 Game* in Python

Sometimes, it is useful to store many values to the same variable. In Python, we can accomplish this with a **list**:

```
students = ["Grant", "Esperanza", "Makena", "Delun"]
print(students[1])
```

This code stores four names in a list called `students`. Then, because list items are numbered beginning at 0, it prints the name, "`Esperanza`".

We can use a for loop to print every item in a list:

```
for x in students:
    print(x)
```

With this code, we generate the following output:

```
Grant
Esperanza
Makena
Delun
```

* * *

In Python, arithmetic expressions are evaluated according to the order of operations. For example:

```
answer = 2 + 3 * 4
print(answer)
```

Since multiplication ("`*`") is evaluated before addition, this code will print:

```
14
```

We can use the **eval() function** to evaluate user input as it would be evaluated in Python:

```
answer = input("Make 24: ")
if eval(answer) == 24:
    print("You win!")
```

With this code, we ask the user to "`Make 24`". If, when evaluated, the user's input is equal to 24, we print, "`You win!`"

For example, if the user entered "`6 * 4`", then the program would print:

```
You win!
```

* * *

In the following example program, *24 Game*, we store four random numbers in a list and task the user with arranging those numbers in an arithmetic expression equal to 24.

24 Game in Python

```
import random

numbers = [0, 0, 0, 0]

print("24 Game")
print("Your numbers are:")

#Store 4 random integers in a list, print each
for x in range(4):
    numbers[x] = random.randint(0, 10)
    print(numbers[x])

guess = input("Make 24: ")
guess = eval(guess)

if guess == 24:
    print("You win!")
else:
    print("That equals", guess)
```

24 Game comprehension questions

1. How would you print the first item from a list named "`movies`"?
2. How would you print every item from a list named "`movies`"?
3. Describe the output from the following code:

```
answer = "3 + 4 * 5"
print(answer)
print(eval(answer))
```

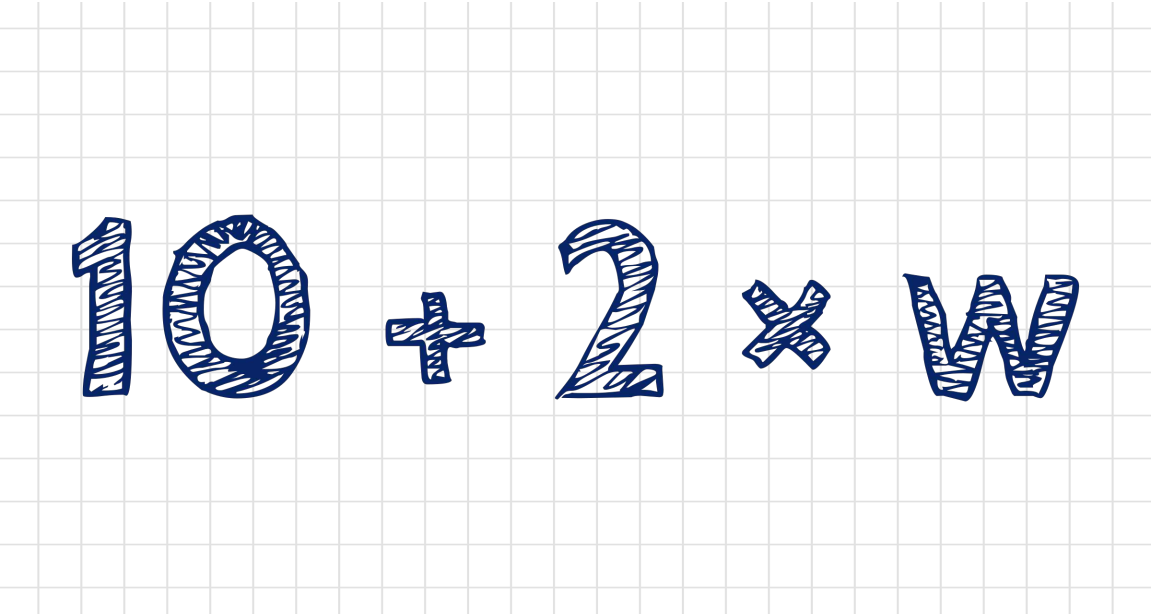
Challenge!

After completing *24 Game* in Python, try these additional challenges:

1. Change the game to the *36 Game*.
2. Allow the user to keep guessing until they win the game.
3. Write a program that selects a random name from a list of five names.
4. In our *24 Game*, it is possible for the user to trick the computer by entering numbers not given or by omitting some of the given numbers. For a solution to this problem, see *Extension: 24 Game in Python* in the appendix.

8

Variable Expressions



"Olivia has ten dollars and earns two more every week."

Introduction

Variable Expressions

In mathematics, a variable represents a quantity, or number. When we use a variable in the place of a number, we are able to use different numbers to fit different situations. We saw in an earlier example that, for Emily who is 3 years older than Jin, we can use a variable expression, $x - 3$, to find Jin's age, no matter how Emily's age changes. In this way, variable expressions are quite useful.

Just as mathematics includes multi-step problems, it also includes multi-step variable expressions. For example, suppose that Olivia

receives an allowance of $2 per week. If Olivia has already saved $10, we can determine how much money she will have in the future by first multiplying $2 by the number of weeks that have passed, then adding the $10 she has already saved. Thus, we could predict Olivia's savings with the expression $10 + 2 \times w$, where w represents the number of weeks that have passed.

We can model a fixed income, like a weekly allowance, with a variable expression.

Image credit: Eric Bennett

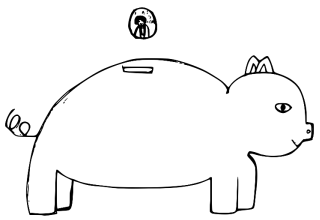


Image credit: Emmaline, age 11

How are variable expressions evaluated?

- To evaluate a **variable expression**, we first **substitute** a value for the variable, then apply the order of operations.
- Since we can substitute any value for a variable, variable expressions can be used to predict an outcome for a given condition.

How are variable expressions used in computer science?

In computer programming, variables are often included in expressions that change their value based on user input.

In video game development, variables that represent the player's position can be included in expressions that incorporate user input to change their values and move the player.

Examples

Evaluating variable expressions (one-step)

Q: Evaluate for $z = 5$: $z + 9$

A: First, we substitute for z. Then, we add:

$$z + 9 =$$
$$5 + 9 = 14$$

Q: Evaluate for $a = 5$: $8 \times a$

A: First, we substitute for a. Then, we multiply:

$$8 \times a =$$
$$8 \times 5 = 40$$

Evaluating variable expressions (two-step)

Q: Evaluate for $x = 8$: $x - 6 \div 2$

A: First, we substitute for x. Then, we apply the order of operations:

$$x - 6 \div 2 =$$
$$8 - 6 \div 2 =$$
$$8 - 3 = 5$$

Q: Evaluate for $x = 8$: $(x - 6) \div 2$

A: First, we substitute for x. Remember that parentheses change the order in which operations are applied:

$$(x - 6) \div 2 =$$
$$(8 - 6) \div 2 =$$
$$2 \div 2 = 1$$

Practice

1. Evaluate each variable expression.

a. $a + 5$ for $a = 3$

b. $a + 5$ for $a = -3$

c. $7 + b$ for $b = 7$

d. $7 + b$ for $b = -7$

e. $c + 8$ for $c = 11$

f. $c + 8$ for $c = -11$

g. $10 - d$ for $d = 4$

h. $10 - d$ for $d = -4$

i. $e - 6$ for $e = 6$

j. $e - 6$ for $e = -6$

k. $11 - f$ for $f = 16$

l. $11 - f$ for $f = -16$

2. Elsa is five years older than Miriam.

a. Write an expression for Elsa's age in terms of Miriam's age.

b. If Miriam is ten years old, how old is Elsa?

3. Henry is four years older than Bradley.

a. Write an expression for Bradley's age in terms of Henry's age.

b. If Henry is twelve years old, how old is Bradley?

4. Evaluate each variable expression.

a. $g \times 4$ for $g = 3$

b. $g \times 4$ for $g = -3$

c. $7 \times h$ for $h = 1$

d. $7 \times h$ for $h = -1$

e. $j \times 6$ for $j = 8$

f. $j \times 6$ for $j = -8$

g. $48 \div k$ for $k = 6$

h. $48 \div k$ for $k = -6$

i. $m \div 5$ for $m = 35$

j. $m \div 5$ for $m = -35$

k. $72 \div n$ for $n = 12$

l. $72 \div n$ for $n = -12$

5. Connor receives an allowance of three dollars per week.

a. Write an expression for the money Connor has received in terms of the number of weeks that have passed.

b. How much money will Connor receive after 6 weeks?

6. Lexi has twenty-four pieces of candy that she plans to divide evenly among the guests at her birthday party.

a. Write an expression for the number of pieces of candy each guest will receive in terms of the number of guests at Lexi's party.

b. If there are 8 guests at Lexi's party, how many pieces of candy will each guest receive?

7. Evaluate each variable expression.

a. $p + 3 \times 5$ for $p = 7$

b. $(p + 3) \times 5$ for $p = 7$

c. $9 - q \times 4$ for $q = 2$

d. $(9 - q) \times 4$ for $q = 2$

e. $r + 12 \div 3$ for $r = 6$

f. $(r + 12) \div 3$ for $r = 6$

g. $18 - s + 10$ for $s = 3$

h. $18 - (s + 10)$ for $s = 3$

i. $t + 16 - 11$ for $t = 14$

j. $t + (16 - 11)$ for $t = 14$

k. $24 \div u \times 3$ for $u = 8$

l. $24 \div (u \times 3)$ for $u = 8$

8. Noah was exactly four feet tall at the beginning of the school year.

a. Write a variable expression for Noah's current height in inches, with n representing the number of inches Noah has grown during the school year.

b. If Noah has grown three inches during the school year, how many inches tall is he now?

9. Python evaluates variable expressions according to the order of operations. Consider the following code:

```
A = 12
answer = (A - 8) * A
print(answer)
```

After running this code, what value is displayed on the screen?

10. Evaluate each variable expression.

a. $v \times 4 - 7$ for $v = -3$

b. $v \times (4 - 7)$ for $v = -3$

c. $-1 + w \times (-5)$ for $w = 7$

d. $(-1 + w) \times (-5)$ for $w = 7$

e. $8 + x \div 4$ for $x = -16$

f. $(8 + x) \div 4$ for $x = -16$

g. $y - 13 - 17$ for $y = -1$

h. $y - (13 - 17)$ for $y = -1$

i. $15 + z - 21$ for $z = 5$

j. $15 + (z - 21)$ for $z = 5$

k. $a \times (-9) \div 3$ for $a = -4$

l. $a \times (-9 \div 3)$ for $a = -4$

11. Ji-yeon borrows twenty dollars from her parents to start a lemonade stand. She plans to sell lemonade for two dollars per cup.

a. Considering her initial debt of $20, write a variable expression for Ji-yeon's profit in terms of c, the number of cups of lemonade she has sold.

b. How much profit will Ji-yeon generate after selling 12 cups of lemonade?

12. A video game developer might use an expression for an x- or y-coordinate to check for a collision. For example, $(x+5, y)$ could be used to check five pixels to the right of a particular game object.

a. What coordinates would the developer use to check five pixels to the left of the game object?

b. Ten pixels above? Thirty-two pixels below?

Application: *Lemonade Stand* in Python

In computer programming, it is sometimes useful to generate a random number that depends on the value of a variable:

```
maximum = input("How high? ")
maximum = int(maximum)
print(random.randint(1, maximum))
```

Here, we ask the user to enter a value for `maximum`. Then, we print a random number between 1 and `maximum`.

* * *

We can use a variable expression to set the value of a variable:

```
price = input("What is the selling price? ")
price = int(price)
sales = 10 - price
```

With this code, the number of sales depends on the price: as `price` increases, `sales` would decrease.

If we are selling cups of lemonade, the number of sales might depend on both the selling price and the weather forecast: customers might purchase more lemonade when the temperature is high:

```
price = input("What is the selling price? ")
price = int(price)
temp = random.randint(50, 100)
sales = temp - 10*price
```

In Python, variable expressions like these are evaluated according to the order of operations: first, `price` is multiplied by 10, then the result is subtracted from `temp`. If `price` is too high, it could result in a negative number of sales, which wouldn't make sense. We can account for that possibility:

```
price = input("What is the selling price? ")
price = int(price)
temp = random.randint(50, 100)
sales = temp - 10*price
if sales < 0:
    sales = 0
```

Now, if the number of sales is negative, it will be set to 0.

* * *

In the following example program, *Lemonade Stand*, the user runs a lemonade stand, deciding how many cups of lemonade to make and the selling price based on the forecast for the day.

Lemonade Stand in Python

```
import random

weather = ["cloudy", "sunny", "hot and dry"]
money = 0

print("The cost to make lemonade is $1 per cup")

#Run the game loop 12 times
for x in range(12):
    print(12-x, "days remaining")
    print("You have", money, "dollars")

    print("The forecast for today is")
    forecast = random.randint(0, 2)
    print(weather[forecast])

    print("How many lemonades do you make today?")
    lemonade = input("? ")
    lemonade = int(lemonade)

    money = money - lemonade
    print("You have", money, "dollars")

    print("What is your selling price today?")
    price = input("? ")
    price = int(price)

    #Generate random integers based on forecast
```

```
    #and price
    A = random.randint(0, forecast)
    B = random.randint(0, price)

    #Use forecast and price to determine sales
    sales = 5 + 20*A - B

    #Make sure number of sales is reasonable
    if sales > lemonade:
        sales = lemonade
    if sales < 0:
        sales = 0

    print("You sold", sales, "lemonades")
    money = money + sales*price

print("Game over")
input("Press ENTER to exit")
```

Lemonade Stand comprehension questions

1. How can we use a variable to influence the value of a random integer?
2. How can we use one variable to determine the value of another variable?
3. How are variable expressions evaluated in Python?

Challenge!

After completing *Lemonade Stand* in Python, try these additional challenges:

1. Make the game longer or shorter by changing the number of days allowed. Make sure the number of days remaining is displayed correctly.
2. Add a victory condition: if the user has over $100 when the game ends, display a congratulatory message.
3. Add the possibility of rain: if it rains, no sales are made and all cups of lemonade are lost.
4. Write a program that asks a user for their height in feet and inches, then converts that height to inches and displays the result.

Appendix

Extension: *Distance Game* in Python

In our original *Distance Game*, it was possible for the user to break the program by entering a value that was not a distance. Here, we address that issue.

In Python, we can determine whether a string is composed entirely of digits (numerical characters) with the **isdigit() method**:

```
guess = input("Enter a positive number: ")
if guess.isdigit():
    print("Well done!")
else:
    print("That is NOT a positive number!")
```

With this code, the user is asked to type a positive number (since "-" is not a digit, isdigit() is only true for positive numbers).

If the user types a positive number, we print:

```
Well done!
```

If the user does not type a positive number, we print:

```
That is NOT a positive number!
```

* * *

In this extension to *Distance Game*, we set the user's input to -1 if that input is not a positive number, ensuring that non-integer, and negative integer, responses are evaluated as incorrect but do not break the program.

Distance Game in Python

```
import random

print("Distance Game")

A = random.randint(-10, 10)
B = random.randint(-10, 10)
distance = abs(A - B)

print("What is the distance between")
print(A, "and", B, "?")
guess = input("? ")

if guess.isdigit() == False:
    guess = -1
```

```
guess = int(guess)

if guess == distance:
    print("Correct!")
else:
    print("Incorrect.")

print("The distance between")
print(A, "and", B, "is", distance)

input("Press ENTER to exit.")
```

Extension: *Guess the Number* in Python

In our original *Guess the Number*, it was possible for the user to break the program by entering a value that was not an integer. Here, we address that issue.

For this, it is necessary to determine whether the user's input is an integer:

```
guess = input("Enter an integer: ")
try:
    guess = int(guess)
except:
    print("That's not an integer.")
else:
    print("Well done!")
```

The **try block** tests a block of code ("`guess = int(guess)`") for errors.

The **except block** determines the code that is executed when the try block contains an error.

This **else block** determines the code that is executed when the try block does not contain an error.

So, with the above code, we first try to convert `guess`, a string variable, into an integer. If that is not possible, we print:

```
That's not an integer!
```

On the other hand, if it is possible to convert `guess` into an integer, we print:

```
Well done!
```

* * *

In this extension to *Guess the Number*, we try to convert the user's input into an integer and only compare that input with the random number if our conversion does not generate an error.

Guess the Number in Python

```
import random

print("Guess the Number")

number = random.randint(-10, 10)
guess = 100

print("I am thinking of a number")
print("between -10 and 10.")

while (guess != number):
    guess = input("What's the number? ")
    try:
        guess = int(guess)
    except:
```

```
        print("That's not an integer.")
    else:
        if (guess > number):
            print("Too high.")
        if (guess < number):
            print("Too low.")

print("Correct!  The number was", number)

input("Press ENTER to exit.")
```

Extension: *Integer Game* in Python

In our original *Integer Game*, it was possible for the user to enter values other than "`add`" or "`subtract`", leading to some unexpected results. Here, we address that issue.

Since we are asking the user to choose from a list of options ("`add`" or "`subtract`), it would make sense to define this list in Python:

```
operations = ["add", "subtract"]
```

Now, we can test whether the user's input is included in our list with the **not in operator**:

```
operation = input("Add or subtract? ")
operation = operation.lower()
if operation not in operations:
    print("That's not an option.")
```

We can also prompt the user to alter their input if it is not in our list:

```
operation = "none"
while operation not in operations:
    operation = input("Add or subtract? ")
    operation = operation.lower()
```

* * *

In this extension to *Integer Game*, we ask the user for input until that input, when converted to lowercase, is either "`add`" or "'`subtract`.

Integer Game in Python

```
import random

print("Integer Game")
operations = ["add", "subtract"]
score = 0

for x in range(5):
    A = random.randint(-10, 10)
    B = random.randint(-10, 10)

    print(A, "and", B)

    operation = "none"
    while operation not in operations:
        operation = input("Add or subtract? ")
        operation = operation.lower()

    if operation == "add":
        print(A, "+", B, "=", A+B)
        score = score + A + B
    else:
        print(A, "-", B, "=", A-B)
        score = score + A - B
```

```
    print("Your score is now", score)
    print("You have", 4-x, "turns remaining")

input("Press ENTER to exit.")
```

Extension: *24 Game* in Python

In our original *24 Game*, it was possible for the user to trick the computer by entering numbers not given or by omitting some of the given numbers. Here, we address that issue.

We can use a for loop to print every character in a string:

```
for x in "Delun":
    print(x)
```

With this code, we print:

```
D
e
l
u
n
```

* * *

Recall our list of students:

```
students = ["Grant", "Esperanza", "Makena", "Delun"]
```

We can use the **in operator** to check if a specific name is in our list:

```
if "Makena" in students:
    print("Makena is here!")
```

```
if "Talia" not in students:
    print("Talia is not here.")
```

With this code, we print:

```
Makena is here!
Talia is not here.
```

* * *

In this extension to *24 Game*, we require the user to enter all given numbers and prevent the user from entering any numbers not given.

24 Game in Python

```
import random
numbers = [0, 0, 0, 0]
countGood = 0
countBad = 0

print("24 Game")
print("Your numbers are:")

#Store 4 random integers in a list, print each
for x in range(4):
    numbers[x] = random.randint(1, 10)
    print(numbers[x])
```

```
guess = input("Make 24: ")

#Check how many given numbers are in the guess
for x in numbers:
    if str(x) in guess:
        countGood = countGood + 1

#Check if any numbers used were not given
for x in guess:
    if x.isdigit():
        if int(x) not in numbers:
            countBad = countBad + 1

#If guess uses all given numbers and no others,
#check if guess equals 24
if countGood == 4 and countBad == 0:
    guess = eval(guess)
    if guess == 24:
        print("You win!")
    else:
        print("That equals", guess)
else:
    if countGood < 4:
        print("You didn't use all the numbers.")
    if countBad > 0:
        print("You used some wrong numbers.")
```

Application: *Spiral Maker* in Python

The cover art for this textbook was generated using this program.

With the turtle module, we can draw a line by instructing the ”turtle” to move **forward**:

```
turtle.forward(100)
```

With this code, the turtle moves forward a distance of 100 units in its current direction. By default, this direction is to the right, but we can change the direction with the **right function**:

```
turtle.right(90)
turtle.forward(100)
```

Here, we first instruct the turtle to turn to its right 90 degrees before moving forward. Thus, we draw a line down the screen.

With variables, we can give a user control over the length of a line. The following code will draw a line with length 100 × `A`:

```
turtle.forward(100*A)
```

We can use a for loop, in combination with the `forward()` and `right()` functions, to draw a square:

```
for x in range(4):
    turtle.forward(100)
    turtle.right(90)
```

This for loop will move the turtle forward, then turn right, a total of four times. We can also **nest** for loops to draw many squares:

```
for x in range(180):
    for y in range(4):
        turtle.forward(100)
        turtle.right(90)
    turtle.right(2)
```

Here, we draw 180 squares, each rotated 2 degrees from the last. Since 2 × 180 = 360, the result is a perfect ”circle of squares”.

We can use a variable to give a user control over the number of squares drawn by our program:

```
B = turtle.numinput("Squares", "How many squares?", 36)
for x in range(int(B)):
    for y in range(4):
        turtle.forward(100)
        turtle.right(90)
    turtle.right(360/B)
```

We convert *B* to an integer because our range cannot be a decimal (our for loop cannot run a decimal number of times). We can also use the absolute value function to ensure that result is a positive whole number:

```
for x in range(int(abs(B))):
```

* * *

In the following example program, *Spriral Maker*, we use turtle graphics to draw a spiral pattern based on user input.

Spiral Maker in Python

```
import turtle
turtle.speed(0)

A = turtle.numinput("Length", "Length?", 3)
B = turtle.numinput("Offset", "Offset?", 3)
C = turtle.numinput("Step", "Step?", 3)
D = round(360/abs(C))

turtle.penup()
turtle.goto(0,50)
turtle.pendown()

for x in range(D):
    for y in range(4):
        turtle.forward(50*A)
        turtle.right(90)
    turtle.right(C)
    turtle.forward(2*B)

turtle.hideturtle()
```

Solutions

Practice problems, comprehension questions and selected *Challenge!* exercises.

1 Variables

1. a) $a = 5$
   b) $b = 5$
   c) $c = 3$
   d) $d = 3$
   e) $e = 8$
   f) $f = 8$
2. a) $a = 8$
   b) $b = 10$
   c) $c = 2$
   d) $d = 16$
   e) $e = 4$
   f) $f = 20$
   g) $g = 9$
   h) $h = 28$
   i) $x = 8$
   j) $j = 38$
   k) $k = 33$
   l) $y = 67$
3. Chen is 15 years old.
4. a) $a = 5$
   b) $b = 4$
   c) $c = 8$
   d) $d = 3$
   e) $e = 5$
   f) $f = 28$
5. a) $a = 5$
   b) $b = 16$
   c) $c = 5$
   d) $d = 7$
   e) $e = 10$
   f) $f = 4$
   g) $g = 9$
   h) $h = 35$
   i) $x = 6$
   j) $j = 16$
   k) $k = 8$
   l) $y = 52$
6. The bluebonnet has 14 florets.
7. Carly is 4 years old.
8. Jung is 17 years old.
9. The previous value of `score` was 650.
10. a) It would take 10 frames for the game

object to move 50 pixels and 20 frames for the game object

to move 100 pixels.

b) It would take 1 second for the game object to move 300 pixels.

Word Game comprehension questions

1. For example:
   ```
   name = "Eric"
   ```
2. For example:
   ```
   print(name)
   ```
3. For example:
   ```
   name = input("Name? ")
   ```

Challenge! exercises

3.
```
name = input("What is your name? ")
print("Hello,", name, "!")

input("Press ENTER to exit.")
```

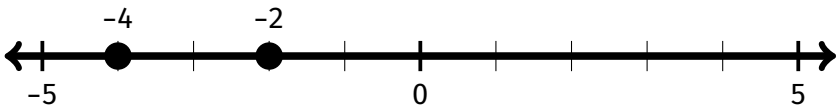
2 Negative Numbers

1. $A = -1$
   $B = 3$
   $C = -4$
   $D = 2$
   $E = 3$
   $F = -5$
   $G = 7$
   $H = -8$
   $J = -6$
   $K = -2$
   $L = -9$
   $M = -3$
   $N = -7$
   $P = -15$
   $Q = -3$
   $R = -18$
   $S = -11$
   $T = -7$
   $U = -14$
   $V = -8$
   $W = -22$
   $X = -30$
   $Y = -18$
   $Z = -33$

2. a) 3
   b) 4
   c) 8
   d) 7
   e) 3
   f) 4
   g) 6
   h) 0
   i) 17
   j) 23

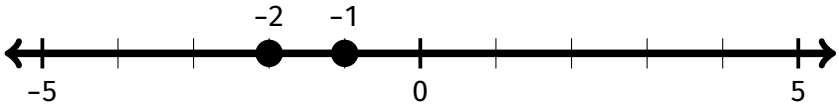
3. a) 3
   b) 5
   c) 4
   d) 4
   e) 8
   f) 10
   g) 11
   h) 11

4. a)



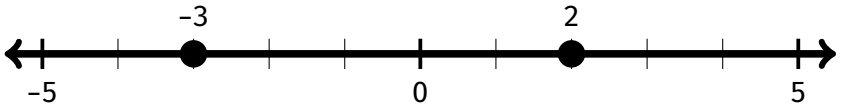
The distance between −4 and −2 is 2.

b)



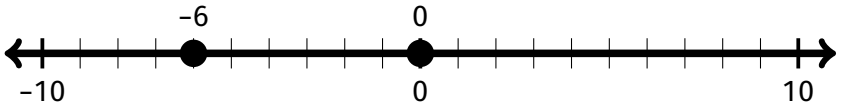
The distance between −2 and −1 is 1.

c)



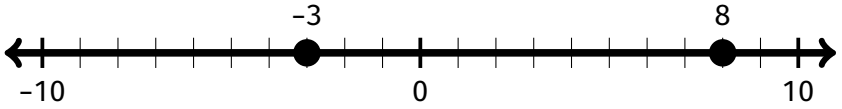
The distance between −3 and 2 is 5.

d)



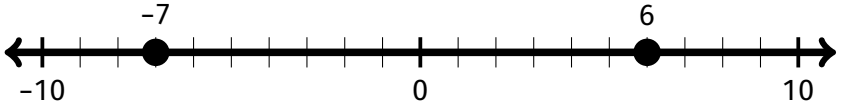
The distance between −6 and 0 is 6.

e)



The distance between 8 and −3 is 11.

f)



The distance between 6 and −7 is 13.

5.
a) 2
b) 1
c) 6
d) 6
e) 8
f) 12
g) 12
h) 16
i) 16
j) 13
k) 26
l) 14

6. −17 and 7 are each 12 units from −5.

7. The record-low temperature in Austin is 42 degrees warmer than the record-low temperature in Boise.

8. a) ...positive.

b) ...positive.

c) ...equal or opposite (i.e. 3 and -3).

Distance Game comprehension questions

1. The `random` module
2. `random.randint(-5, 0)`
3. Indentation is the spacing at the beginning of a line of code. It is used to create code blocks, as in the case of an `if` statement.

Challenge! exercises

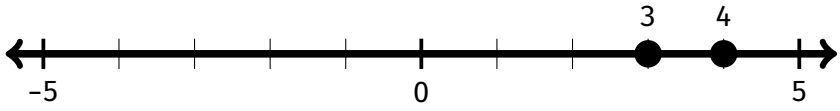
3.
```
import random

print("Guess the Number")

number = random.randint(-10, 10)

print("I am thinking of a number")
print("between -10 and 10.")

guess = input("What is the number? ")
guess = int(guess)

if guess == number:
    print("Correct!")
else:
    print("Incorrect.")

print("The number was", number)

input("Press ENTER to exit.")
```

3 Comparing Negative Numbers

1. a) −1
   b) −2
   c) 3
   d) −2
   e) −8
   f) 7
   g) −1
   h) 4
   i) −8

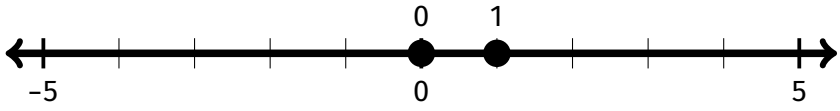
2. a) −1
   b) −4
   c) 2
   d) −3
   e) −9
   f) 5
   g) −15
   h) −9
   i) 10
   j) −19
   k) −13
   l) 17

3. a)



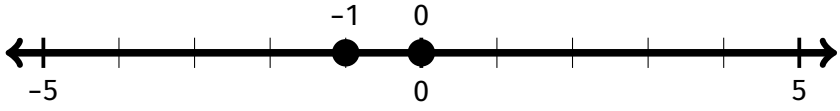
3 < 4

b)



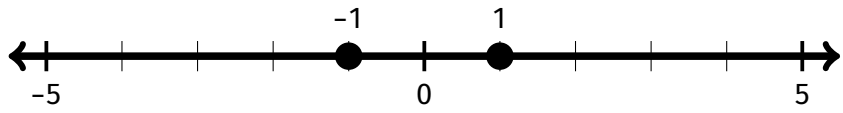
0 < 1

c)



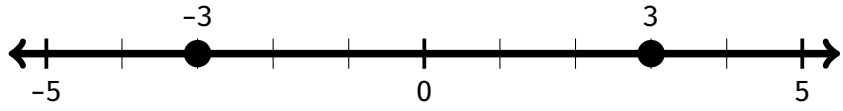
−1 < 0

d)



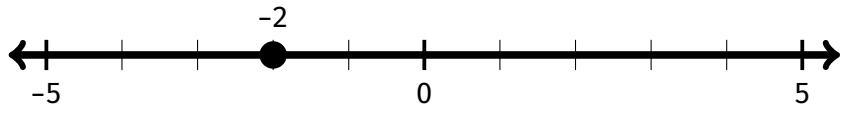
1 > −1

e)



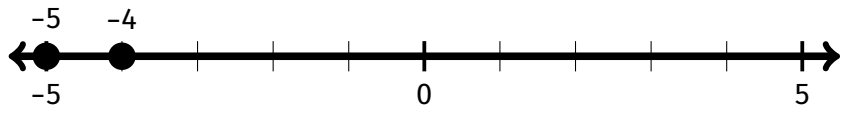
−3 < 3

f)



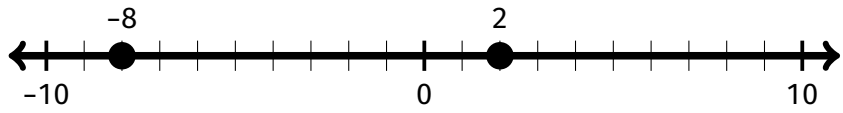
−2 = −2

g)



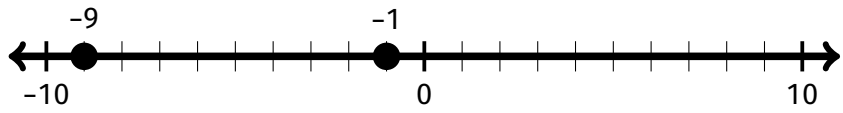
−4 > −5

h)



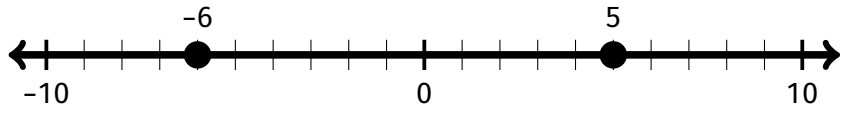
−8 < 2

i)



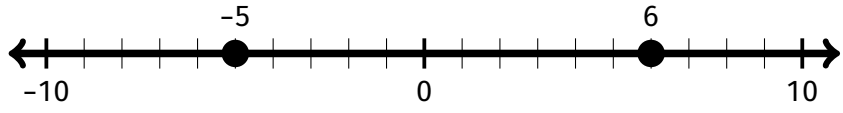
$-9 < -1$

j)



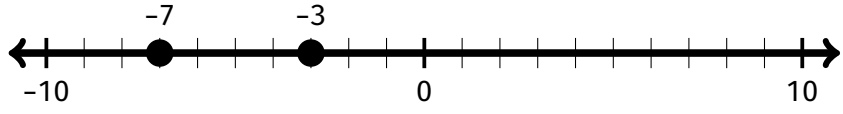
$5 > -6$

k)



$6 > -5$

l)



$-3 > -7$

4. a) $3 > 2$
   b) $4 > 0$
   c) $-4 < 0$
   d) $-2 < 2$
   e) $6 > -3$
   f) $-7 > -8$
   g) $-12 = -12$
   h) $-10 > -11$
   i) $-21 < 21$
   j) $-16 < 8$
   k) $-23 < -17$
   l) $-26 > -27$

5. $-5,387 < -1,949$

6. a) $n > m$ or $m < n$
   b) $n < m$ or $m > n$
   c) $r < s$ or $s > r$

d) $r > s$ or $s < r$

e) $x > y$ or $y < x$

f) $x < y$ or $y > x$

Guess the Number comprehension questions

1. We set `guess` to 100 because our `while` loop will only run if `guess` is not equal to the random number. Since the random number could never be 100, we are certain that our `while` loop will run every time. We could set `guess` to any number greater than 10 or less than -10 to achieve the same result.
2. Indentation is used in a `while` loop to create a block of code that will run repeatedly as long as the condition defined in the `while` loop remains true.
3. ```
   guess = 100
   while guess != distance:
       print("What is the distance between")
       print(A, "and", B, "?")
       guess = input("? ")
       guess = int(guess)
   ```
4. Because our `while` loop only ends when `guess` is equal to `number`, we don't need to test this condition again.
   ```

Challenge! exercises

3.
```
import random

print("Guess the Number")

number = random.randint(-10, 10)
guess = 100
guesses = 0

print("I am thinking of a number")
print("between -10 and 10.")

while (guess != number):
    guess = input("What's the number? ")
    guess = int(guess)
    guesses = guesses + 1
    if (guess > number):
        print("Too high.")
    if (guess < number):
        print("Too low.")

print("Correct!  The number was", number)
print("You used", guesses, "guesses")

input("Press ENTER to exit.")
```

4.
```
import random

print("Roll a 6")

number = 0
rolls = 0

while (number != 6):
    number = random.randint(1, 6)
    rolls = rolls + 1
    print(number)

print("It took", rolls, "rolls to get a 6.")

input("Press ENTER to exit.")
```

4 Coordinate Systems

1. a) *positive negative*
   b) *negative positive*
   c) *negative zero*
   d) *positive positive*
   e) *negative negative*
   f) *zero positive*
2. a) (2, 4)
   b) (2, 0)
   c) (–3, –2)
   d) (–4, 3)
   e) (3, –3)
   f) (0, –2)
3. a) (3, 4)
   b) (0, –4)
   c) (–2, –4)
   d) (0, –3)
   e) (3, 3)
   f) (0, 1)
   g) (0, 0)
   h) (–3, 1)
   i) (5, 5)
   j) (0, –1)
   k) (–4, 0)
   l) (4, 4)
4. When translating a point to the right, the *x*-coordinate increases. When translating a point to the left, the *x*-coordinate decreases.
5. When translating a point upward, the *y*-coordinate increases. When translating a point downward, the *y*-coordinate decreases.
6. a) (2, 4)
   b) (0, –2)
   c) (0, –5)
   d) (–2, –4)
   e) (11, 0)
   f) (2, 1)
   g) (–12, 0)
   h) (0, 11)
   i) (10, 31)
   j) (–10, 0)
   k) (–16, –12)
   l) (20, 9)
7. The coordinates of the game object's new location are (–36, 0).
8. The game object moved 180 pixels to the right and 300 pixels down.

Draw a Cube comprehension questions

1. `turtle.goto(0, 0)`
2. Use the "`penup()`" function; for example:
   ```
   turtle.penup()
   turtle.goto(10, 10)
   ```

3. Use a hashtag; for example:
   `#move the turtle without drawing`

Challenge! exercises

2.
```
import turtle

#move the pen to the top left corner
turtle.penup()
turtle.goto(-50, 50)
turtle.pendown()

turtle.fillcolor("pink")
turtle.begin_fill()

#draw the front face of the cube
turtle.goto(50, 50)
turtle.goto(50, -50)
turtle.goto(-50, -50)
turtle.goto(-50, 50)

#make it three-dimensional
turtle.goto(0, 100)
turtle.goto(100, 100)
turtle.goto(100, 0)
turtle.goto(50, -50)

turtle.end_fill()

#one more line
turtle.penup()
turtle.goto(50, 50)
turtle.pendown()
turtle.goto(100, 100)

turtle.hideturtle()
turtle.exitonclick()
```

5 Adding and Subtracting Negative Numbers

1. a) −1
   b) 1
   c) 3
   d) 0
   e) −6
   f) 1
   g) −7
   h) 3

2. a) 0
   b) 2
   c) 2
   d) 1
   e) −8
   f) −1
   g) −1
   h) 8
   i) −4
   j) −6
   k) 9
   l) −7
   m) 5
   n) −15
   o) −11

3. −2, 100 + (−1, 080) = −3, 180
   The maximum observed diving depth of a sperm whale is 3,180 ft.

4. a) −4
   b) 2
   c) −3
   d) −1
   e) 0
   f) −7
   g) 3
   h) 7

5. a) −2
   b) 0
   c) 5
   d) 9
   e) −9
   f) 10
   g) −1
   h) −11
   i) 16
   j) −16
   k) 28
   l) −10
   m) −34
   n) 45
   o) 8

6. −13 − (−9) = −4
   Javier still owes Sarah $4.
   a) −1
   b) 9
   c) −5
   d) 5
   e) −11
   f) 0
   g) −5
   h) 21
   i) 5
   j) −13
   k) 30
   l) −30
   m) 34
   n) −9
   o) −13

7. 22

8. −57 + 98 = 41
   The average annual temperature in Ushuaia is 41 °F.

Integer Game comprehension questions

1.
```
for x in range(100):
    print("Hello, World!")
```

2.
```
#print every integer from 0 to 100
for x in range(101):
    print(x)
#print every integer from 1 to 100
for x in range(100):
    print(x + 1)
```

3. We convert `operation` to lowercase in order to compare it to the string "`add`"; if a user were to enter "`Add`" or "`ADD`", it would not initially equal "`add`" because Python is case-sensitive.

Challenge! exercises

4.
```
import random

print("10 rolls")
score = 0

for x in range(10):
    roll = random.randint(1, 6)
    print(roll)
    score = score + roll

print("Your total score is", score)

input("Press ENTER to exit.")
```

6 Multiplying and Dividing Negative Numbers

1. a) −8
   b) −8
   c) 8
   d) −18
   e) 18
   f) −16
   g) −16
   h) 16

2. a) −1
   b) −12
   c) −12
   d) 12
   e) 12
   f) −12
   g) −12
   h) 24
   i) −40
   j) −56
   k) 24
   l) −45
   m) 60
   n) −60
   o) 120

3. −3 × 8 = −24
   The total change in temperature overnight was 24 °F.

4. a) 2
   b) −2
   c) −2
   d) 3
   e) −3
   f) −2
   g) −3
   h) −3

5. a) 1
   b) −1
   c) −1
   d) 4
   e) −2
   f) −4
   g) −4
   h) −4
   i) 5
   j) −4
   k) 5
   l) −3
   m) 3
   n) −2
   o) 3

6. −28 ÷ 7 = −4
   The average temperature change per day was -4 °F.

7. −65 ÷ (−5) = 13
   Jaime will pay off his debt in 13 weeks.

8. a) 1
   b) −10
   c) 10
   d) 16
   e) −2
   f) −8
   g) −3
   h) −12
   i) 30
   j) 3
   k) 45
   l) −15
   m) −3
   n) −4
   o) 32

9. a) ...positive.
   b) ...negative.
   c) ...both integers are positive or both integers are negative.
   d) ...one integer is postive and the other integer is negative.

Funny Face comprehension questions

1. For example:
   ```
   X = turtle.numinput("How old are you? ")
   ```
2. `turtle.circle(20)`
3. The user is able to control the location of the eyes and mouth because the coordinates for those locations are written as variable expressions where the variable is stored input from the user.

Challenge! exercises

3.
```
import turtle
turtle.speed(10)

print("Draw a Building")

H = turtle.numinput("Height", "How tall?", 100)
if H < 100:
    H = 100

#draw the building
turtle.penup()
turtle.goto(-100, -200)
turtle.pendown()
turtle.goto(-100, H-200)
turtle.goto(100, H-200)
turtle.goto(100, -200)
turtle.goto(-100, -200)

#draw the door
turtle.goto(-10, -200)
turtle.goto(-10, -160)
turtle.goto(10, -160)
turtle.goto(10, -200)

#draw the windows
for y in range(int(H/100)):
    for x in range(5):
        turtle.penup()
        turtle.goto(-90+40*x, -130+100*y)
        turtle.pendown()
        turtle.goto(-90+40*x, -110+100*y)
        turtle.goto(-70+40*x, -110+100*y)
        turtle.goto(-70+40*x, -130+100*y)
        turtle.goto(-90+40*x, -130+100*y)

turtle.hideturtle()
turtle.exitonclick()
```

7 The Order of Operations

1. a) 14
   b) 10
   c) 14
   d) 12
   e) 3
   f) 12
   g) 8
   h) 5
   i) 60
   j) 13
   k) 12
   l) 11
   m) 56
   n) −56
   o) 8
2. 3 × 7 − 4
3. 4 × 12 + 7
4. a) −17
   b) −7
   c) −6
   d) 16
   e) 4
   f) 4
   g) 15
   h) 24
   i) 39
   j) 8
   k) −16
   l) −5
   m) −6
   n) 1
   o) 49
5. 21 − 28 + 23
6. a) 20
   b) 20
   c) 14
   d) 21
   e) 21
   f) 4
   g) 1
   h) 11
   i) 7
   j) 1
   k) 9
   l) 4
   m) 3
   n) 22
   o) 63
7. (19 − 3) ÷ 4
8. a) −5
   b) −5
   c) 3
   d) 72
   e) 72
   f) 66
   g) 4
   h) 12
   i) −18
   j) −1
   k) −9
   l) −16
   m) −3
   n) −17
   o) −33
9. −9
10. (−75 + (−45)) ÷ (−30)

24 Game comprehension questions

1. `print(movies[0])`
2. ```
   for x in movies:
       print(x)
   ```
3. The output would look like this:
   ```
   3 + 4 * 5
   23
   ```

Challenge! exercises

2.
```
import random

numbers = [0, 0, 0, 0]

print("24 Game")
print("Your numbers are:")

#Store 4 random integers in a list, print each
for x in range(4):
    numbers[x] = random.randint(-6, 6)
    print(numbers[x])

guess = 0
while guess != 24:
    guess = input("Make 24: ")
    guess = eval(guess)

    if guess != 24:
        print("That equals", guess)

print("You win!")

input("Press ENTER to exit.")
```
```

3.
```
import random

names = ["","","","",""]

for x in range(5):
    print("Name", x+1)
    names[x] = input("? ")

x = random.randint(0, 4)
print("I choose", names[x])

input("Press ENTER to exit.")
```

8 Variable Expressions

1. a) 8
   b) 2
   c) 14
   d) 0
   e) 19
   f) -3
   g) 6
   h) 14
   i) 0
   j) -12
   k) -5
   l) 27

2. a) $m + 5$
   b) Elsa is 15 years old.

3. a) $h - 4$
   b) Bradley is 8 years old.

4. a) 12
   b) -12
   c) 7
   d) -7
   e) 48
   f) -48
   g) 8
   h) -8
   i) 7
   j) -7
   k) 6
   l) -6

4. a) $w \times 3$
   b) $18

5. a) $24 \div g$
   b) Each guest will receive 3 pieces of candy.

6. a) 22
   b) 50
   c) 1
   d) 28
   e) 10
   f) 6
   g) 25
   h) 5
   i) 19
   j) 19
   k) 9
   l) 1

7. a) $4 \times 12 + n$
   b) Noah is now 51 inches tall.

8. 48

9. a) -19
   b) 9
   c) -36
   d) -30
   e) 4
   f) -2
   g) -31
   h) 3
   i) -1
   j) -1
   k) 12
   l) 12

10. a) $-20 + 2 \times c$
    b) Ji-yeon will generate a profit of $4.

11. a) $(x - 5, y)$
    b) $(x, y + 10)$; $(x, y - 32)$

Lemonade Stand comprehension questions

1. We can use a variable as a parameter in a `randint()` function. For example:
   ```
   X = input("Maximum? ")
   print(random.randint(1, X))
   ```
2. We can determine the value of any variable with a variable expression. For example:
   ```
   X = random.randint(1, 10)
   Y = X * 10
   ```
   Here, *Y* is always exactly 10 times the value of *X*.
3. In Python, variable expressions are evaluated according to the order of operations.

Challenge! exercises

3.
```
import random

weather = ["cloudy", "sunny", "hot and dry"]
money = 0

print("The cost to make lemonade is $1 per cup")

#Run the game loop 12 times
for x in range(12):
    print(12-x, "days remaining")
    print("You have", money, "dollars")
```

```
        print("The forecast for today is")
        forecast = random.randint(0, 2)
        print(weather[forecast])

        print("How many lemonades do you make today?")
        lemonade = input("? ")
        lemonade = int(lemonade)

        money = money - lemonade
        print("You have", money, "dollars")

        print("What is your selling price today?")
        price = input("? ")
        price = int(price)

        #Generate random integers based on forecast
        #and price
        A = random.randint(0, forecast)
        B = random.randint(0, price)

        #Use forecast and price to determine sales
        sales = 5 + 20*A - B
        rain = random.randint(1, 10)
        if forecast == 0 and rain > 5:
            sales = 0
            print("It started raining!")
            print("All of your lemonades are lost.")

        #Make sure number of sales is reasonable
        if sales > lemonade:
            sales = lemonade
        if sales < 0:
            sales = 0

        print("You sold", sales, "lemonades")
        money = money + sales*price

print("Game over")
input("Press ENTER to exit")
```

4.
```
print("Enter your height:")
feet = input("How many feet? ")
inches = input("How many inches? ")

feet = int(feet)
inches = int(inches)

inches = feet*12 + inches

print("You are", inches, "inches tall.")
input("Press ENTER to exit.")
```

Index