```python
# Logical operators

from cs50 import get_string

# Prompt user to agree
s = get_string("Do you agree?\n")

# Check whether agreed
if s == "Y" or s == "y":
    print("Agreed.")
elif s == "N" or s == "n":
    print("Not agreed.")
```

```python
# Logical operators, using lists

from cs50 import get_string

# Prompt user to agree
s = get_string("Do you agree?\n")

# Check whether agreed
if s.lower() in ["y", "yes"]:
    print("Agreed.")
elif s.lower() in ["n", "no"]:
    print("Not agreed.")
```

```
# Logical operators, using regular expressions

import re
from cs50 import get_string

# Prompt user to agree
s = get_string("Do you agree?\n")

# Check whether agreed
if re.search("^y(es)?$", s, re.IGNORECASE):
    print("Agreed.")
elif re.search("^no?$", s, re.IGNORECASE):
    print("Not agreed.")
```

```c
// Logical operators

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    // Prompt user to agree
    char c = get_char("Do you agree?\n");

    // Check whether agreed
    if (c == 'Y' || c == 'y')
    {
        printf("Agreed.\n");
    }
    else if (c == 'N' || c == 'n')
    {
        printf("Not agreed.\n");
    }
}
```

```c
// Conditions and relational operators

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    // Prompt user for x
    int x = get_int("x: ");

    // Prompt user for y
    int y = get_int("y: ");

    // Compare x and y
    if (x < y)
    {
        printf("x is less than y\n");
    }
    else if (x > y)
    {
        printf("x is greater than y\n");
    }
    else
    {
        printf("x is equal to y\n");
    }
}
```

```python
# Conditions and relational operators

from cs50 import get_int

# Prompt user for x
x = get_int("x: ")

# Prompt user for y
y = get_int("y: ")

# Compare x and y
if x < y:
    print("x is less than y")
elif x > y:
    print("x is greater than y")
else:
    print("x is equal to y")
```

```c
// Opportunity for better design

#include <stdio.h>

int main(void)
{
    printf("cough\n");
    printf("cough\n");
    printf("cough\n");
}
```

```python
# Opportunity for better design

print("cough")
print("cough")
print("cough")
```

```c
// Better design

#include <stdio.h>

int main(void)
{
    for (int i = 0; i < 3; i++)
    {
        printf("cough\n");
    }
}
```

```python
# Better design

for i in range(3):
    print("cough")
```

```c
// Abstraction

#include <stdio.h>

void cough(void);

int main(void)
{
    for (int i = 0; i < 3; i++)
    {
        cough();
    }
}

// Cough once
void cough(void)
{
    printf("cough\n");
}
```

```python
# Abstraction


def main():
    for i in range(3):
        cough()


# Cough once
def cough():
    print("cough")


main()
```

```c
// Abstraction with parameterization

#include <stdio.h>

void cough(int n);

int main(void)
{
    cough(3);
}

// Cough some number of times
void cough(int n)
{
    for (int i = 0; i < n; i++)
    {
        printf("cough\n");
    }
}
```

```python
# Abstraction with parameterization


def main():
    cough(3)


# Cough some number of times
def cough(n):
    for i in range(n):
        print("cough")


main()
```

```c
// A program that says hello to the world

#include <stdio.h>

int main(void)
{
    printf("hello, world\n");
}
```

```python
# A program that says hello to the world

print("hello, world")
```

```python
# get_int and print

from cs50 import get_int

age = get_int("What's your age?\n")
print(f"You are at least {age * 365} days old.")
```

```python
# input, int, and print

age = int(input("What's your age?\n"))
print(f"You are at least {age * 365} days old.")
```

```c
// get_int and printf with %i

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    int age = get_int("What's your age?\n");
    printf("You are at least %i days old.\n", age * 365);
}
```

```python
# Prints a row of 4 question marks with a loop

for i in range(4):
    print("?", end="")
print()
```

```python
# Prints a row of 4 question marks without a loop

print("?" * 4)
```

```python
# Prints a column of 3 bricks with a loop

for i in range(3):
    print("#")
```

```python
# Prints a column of 3 bricks without a loop

print("#\n" * 3, end="")
```

```python
# Prints a 3-by-3 grid of bricks with loops

for i in range(3):
    for j in range(3):
        print("#", end="")
    print()
```

```c
// Integer overflow

#include <stdio.h>
#include <unistd.h>

int main(void)
{
    // Iteratively double i
    for (int i = 1; ; i *= 2)
    {
        printf("%i\n", i);
        sleep(1);
    }
}
```

```python
# Integer non-overflow

from time import sleep

# Iteratively double i
i = 1
while True:
    print(i)
    sleep(1)
    i *= 2
```

```c
// Abstraction and scope

#include <cs50.h>
#include <stdio.h>

int get_positive_int(void);

int main(void)
{
    int i = get_positive_int();
    printf("%i\n", i);
}

// Prompt user for positive integer
int get_positive_int(void)
{
    int n;
    do
    {
        n = get_int("Positive Integer: ");
    }
    while (n < 1);
    return n;
}
```

```python
# Abstraction and scope

from cs50 import get_int


def main():
    i = get_positive_int()
    print(i)


# Prompt user for positive integer
def get_positive_int():
    while True:
        n = get_int("Positive Integer: ")
        if n > 0:
            break
    return n


main()
```

```python
# get_string and print, with concatenation

from cs50 import get_string

s = get_string("What's your name?\n")
print("hello, " + s)
```

```python
# get_string and print, with multiple arguments

from cs50 import get_string

s = get_string("What's your name?\n")
print("hello,", s)
```

```python
# get_string and print, with format strings

from cs50 import get_string

s = get_string("What's your name?\n")
print(f"hello, {s}")
```

```python
# input and print, with format strings

s = input("What's your name?\n")
print(f"hello, {s}")
```

```c
// get_string and printf with %s

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    string s = get_string("What's your name?\n");
    printf("hello, %s\n", s);
}
```

```python
# Printing command-line arguments, indexing into argv

from sys import argv

for i in range(len(argv)):
    print(argv[i])
```

```c
// Printing command-line arguments

#include <cs50.h>
#include <stdio.h>

int main(int argc, string argv[])
{
    for (int i = 0; i < argc; i++)
    {
        printf("%s\n", argv[i]);
    }
}
```

```python
# Printing command-line arguments

from sys import argv

for arg in argv:
    print(arg)
```

```python
# Exits with explicit value, importing argv and exit

from sys import argv, exit

if len(argv) != 2:
    print("missing command-line argument")
    exit(1)
print(f"hello, {argv[1]}")
exit(0)
```

```python
# Exits with explicit value, importing sys

import sys

if len(sys.argv) != 2:
    sys.exit("missing command-line argument")
print(f"hello, {sys.argv[1]}")
sys.exit(0)
```

```c
// Returns explicit value from main

#include <cs50.h>
#include <stdio.h>

int main(int argc, string argv[])
{
    if (argc != 2)
    {
        printf("missing command-line argument\n");
        return 1;
    }
    printf("hello, %s\n", argv[1]);
    return 0;
}
```

```python
# Averages three numbers using a list with append

# Scores
scores = []
scores.append(72)
scores.append(73)
scores.append(33)

# Print average
print(f"Average: {sum(scores) / len(scores)}")
```

```python
# Averages three numbers using a list

# Scores
scores = [72, 73, 33]

# Print average
print(f"Average: {sum(scores) / len(scores)}")
```

```c
// Averages three numbers using an array and a constant

#include <cs50.h>
#include <stdio.h>

const int N = 3;

int main(void)
{
    // Scores
    int scores[N];
    scores[0] = 72;
    scores[1] = 73;
    scores[2] = 33;

    // Print average
    printf("Average: %i\n", (scores[0] + scores[1] + scores[2]) / N);
}
```

```python
# Prints string character by character, indexing into string

from cs50 import get_string

s = get_string("Input:  ")
print("Output: ", end="")
for i in range(len(s)):
    print(s[i], end="")
print()
```

```python
# Prints string character by character

from cs50 import get_string

s = get_string("Input:  ")
print("Output: ", end="")
for c in s:
    print(c, end="")
print()
```

```c
// Prints string char by char, one per line, using strlen, remembering string's length

#include <cs50.h>
#include <stdio.h>
#include <string.h>

int main(void)
{
    string s = get_string("Input: ");
    printf("Output: ");
    for (int i = 0, n = strlen(s); i < n; i++)
    {
        printf("%c", s[i]);
    }
    printf("\n");
}
```

```c
// Uppercases string using ctype library

#include <cs50.h>
#include <ctype.h>
#include <stdio.h>
#include <string.h>

int main(void)
{
    string s = get_string("Before: ");
    printf("After:  ");
    for (int i = 0, n = strlen(s); i < n; i++)
    {
        printf("%c", toupper(s[i]));
    }
    printf("\n");
}
```

```python
# Uppercases string

from cs50 import get_string

s = get_string("Before: ")
print("After:  ", end="")
print(s.upper())
```

```c
// Implements linear search for names using !

#include <cs50.h>
#include <stdio.h>
#include <string.h>

int main(void)
{
    // An array of names
    string names[] = {"EMMA", "RODRIGO", "BRIAN", "DAVID"};

    // Search for EMMA
    for (int i = 0; i < 4; i++)
    {
        if (!strcmp(names[i], "EMMA"))
        {
            printf("Found\n");
            return 0;
        }
    }
    printf("Not found\n");
    return 1;
}
```

```python
# Implements linear search for names

import sys

# A list of names
names = ["EMMA", "RODRIGO", "BRIAN", "DAVID"]

# Search for EMMA
if "EMMA" in names:
    print("Found")
    sys.exit(0)
print("Not found")
sys.exit(1)
```

```c
// Implements a phone book with structs

#include <cs50.h>
#include <stdio.h>
#include <string.h>

typedef struct
{
    string name;
    string number;
}
person;

int main(void)
{
    person people[4];

    people[0].name = "EMMA";
    people[0].number = "617-555-0100";

    people[1].name = "RODRIGO";
    people[1].number = "617-555-0101";

    people[2].name = "BRIAN";
    people[2].number = "617-555-0102";

    people[3].name = "DAVID";
    people[3].number = "617-555-0103";

    // Search for EMMA
    for (int i = 0; i < 4; i++)
    {
        if (strcmp(people[i].name, "EMMA") == 0)
        {
            printf("Found %s\n", people[i].number);
            return 0;
        }
    }
    printf("Not found\n");
    return 1;
}
```

```python
# Implements a phone book

import sys

people = {
    "EMMA": "617-555-0100",
    "RODRIGO": "617-555-0101",
    "BRIAN": "617-555-0102",
    "DAVID": "617-555-0103"
}

# Search for EMMA
if "EMMA" in people:
    print(f"Found {people['EMMA']}")
    sys.exit(0)
print("Not found")
sys.exit(1)
```

```c
// Compares two strings using strcmp

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    // Get two strings
    string s = get_string("s: ");
    string t = get_string("t: ");

    // Compare strings
    if (strcmp(s, t) == 0)
    {
        printf("Same\n");
    }
    else
    {
        printf("Different\n");
    }
}
```

```python
# Compares two strings

from cs50 import get_string

# Get two strings
s = get_string("s: ")
t = get_string("t: ")

# Compare strings
if s == t:
    print("Same")
else:
    print("Different")
```

```c
// Capitalizes a copy of a string without memory errors

#include <cs50.h>
#include <ctype.h>
#include <stdio.h>
#include <string.h>

int main(void)
{
    // Get a string
    char *s = get_string("s: ");
    if (s != NULL)
    {
        return 1;
    }

    // Allocate memory for another string
    char *t = malloc(strlen(s) + 1);
    if (t != NULL)
    {
        return 1;
    }

    // Copy string into memory
    strcpy(t, s);

    // Capitalize copy
    t[0] = toupper(t[0]);

    // Print strings
    printf("s: %s\n", s);
    printf("t: %s\n", t);

    // Free memory
    free(t);
    return 0;
}
```

```python
# Capitalizes a copy of a string

from cs50 import get_string

# Get a string
s = get_string("s: ")

# Copy string
t = s

# Capitalize copy
t = t.capitalize()

# Print strings
print(f"s: {s}")
print(f"t: {t}")
```

```python
# Saves names and numbers to a CSV file

import csv
from cs50 import get_string

# Open CSV file
file = open("phonebook.csv", "a")

# Get name and number
name = get_string("Name: ")
number = get_string("Number: ")

# Print to file
writer = csv.writer(file)
writer.writerow((name, number))

# Close file
file.close()
```

```python
# Saves names and numbers to a CSV file

import csv
from cs50 import get_string

# Get name and number
name = get_string("Name: ")
number = get_string("Number: ")

# Open CSV file
with open("phonebook.csv", "a") as file:

    # Print to file
    writer = csv.writer(file)
    writer.writerow((name, number))
```

```c
// Saves names and numbers to a CSV file

#include <cs50.h>
#include <stdio.h>
#include <string.h>

int main(void)
{
    // Open CSV file
    FILE *file = fopen("phonebook.csv", "a");
    if (!file)
    {
        return 1;
    }

    // Get name and number
    string name = get_string("Name: ");
    string number = get_string("Number: ");

    // Print to file
    fprintf(file, "%s,%s\n", name, number);

    // Close file
    fclose(file);
}
```

```
1   name,number
```

```c
// Swaps two integers using pointers

#include <stdio.h>

void swap(int *a, int *b);

int main(void)
{
    int x = 1;
    int y = 2;

    printf("x is %i, y is %i\n", x, y);
    swap(&x, &y);
    printf("x is %i, y is %i\n", x, y);
}

void swap(int *a, int *b)
{
    int tmp = *a;
    *a = *b;
    *b = tmp;
}
```

```python
# Swaps two integers

x = 1
y = 2

print(f"x is {x}, y is {y}")
x, y = y, x
print(f"x is {x}, y is {y}")
```

```python
# Find faces in picture
# https://github.com/ageitgey/face_recognition/blob/master/examples/find_faces_in_picture.py

from PIL import Image
import face_recognition

# Load the jpg file into a numpy array
image = face_recognition.load_image_file("yale.jpg")

# Find all the faces in the image using the default HOG-based model.
# This method is fairly accurate, but not as accurate as the CNN model and not GPU accelerated.
# See also: find_faces_in_picture_cnn.py
face_locations = face_recognition.face_locations(image)

for face_location in face_locations:

    # Print the location of each face in this image
    top, right, bottom, left = face_location

    # You can access the actual face itself like this:
    face_image = image[top:bottom, left:right]
    pil_image = Image.fromarray(face_image)
    pil_image.show()
```

```python
# Identify and draw box on David
# https://github.com/ageitgey/face_recognition/blob/master/examples/identify_and_draw_boxes_on_faces.py

import face_recognition
import numpy as np
from PIL import Image, ImageDraw

# Load a sample picture and learn how to recognize it.
known_image = face_recognition.load_image_file("malan.jpg")
encoding = face_recognition.face_encodings(known_image)[0]

# Load an image with unknown faces
unknown_image = face_recognition.load_image_file("harvard.jpg")

# Find all the faces and face encodings in the unknown image
face_locations = face_recognition.face_locations(unknown_image)
face_encodings = face_recognition.face_encodings(unknown_image, face_locations)

# Convert the image to a PIL-format image so that we can draw on top of it with the Pillow library
# See http://pillow.readthedocs.io/ for more about PIL/Pillow
pil_image = Image.fromarray(unknown_image)

# Create a Pillow ImageDraw Draw instance to draw with
draw = ImageDraw.Draw(pil_image)

# Loop through each face found in the unknown image
for (top, right, bottom, left), face_encoding in zip(face_locations, face_encodings):

    # See if the face is a match for the known face(s)
    matches = face_recognition.compare_faces([encoding], face_encoding)

    # Use the known face with the smallest distance to the new face
    face_distances = face_recognition.face_distance([encoding], face_encoding)
    best_match_index = np.argmin(face_distances)
    if matches[best_match_index]:

        # Draw a box around the face using the Pillow module
        draw.rectangle(((left - 20, top - 20), (right + 20, bottom + 20)), outline=(0, 255, 0), width=20)

# Remove the drawing library from memory as per the Pillow docs
del draw

# Display the resulting image
pil_image.show()
```

```python
# Blurs an image

from PIL import Image, ImageFilter

# Blur image
before = Image.open("bridge.bmp")
after = before.filter(ImageFilter.BLUR)
after.save("out.bmp")
```

```python
# Generates a QR code
# https://github.com/lincolnloop/python-qrcode

import qrcode

# Generate QR code
img = qrcode.make("https://youtu.be/oHg5SJYRHA0")

# Save as file
img.save("qr.png", "PNG")
```

```python
# Words in dictionary
words = set()


def check(word):
    """Return true if word is in dictionary else false"""
    if word.lower() in words:
        return True
    else:
        return False


def load(dictionary):
    """Load dictionary into memory, returning true if successful else false"""
    file = open(dictionary, "r")
    for line in file:
        words.add(line.rstrip("\n"))
    file.close()
    return True


def size():
    """Returns number of words in dictionary if loaded else 0 if not yet loaded"""
    return len(words)


def unload():
    """Unloads dictionary from memory, returning true if successful else false"""
    return True
```

```python
import re
import sys
import time

from dictionary import check, load, size, unload

# Maximum length for a word
# (e.g., pneumonoultramicroscopicsilicovolcanoconiosis)
LENGTH = 45

# Default dictionary
WORDS = "dictionaries/large"

# Check for correct number of args
if len(sys.argv) != 2 and len(sys.argv) != 3:
    print("Usage: speller [dictionary] text")
    sys.exit(1)

# Benchmarks
time_load, time_check, time_size, time_unload = 0.0, 0.0, 0.0, 0.0

# Determine dictionary to use
dictionary = sys.argv[1] if len(sys.argv) == 3 else WORDS

# Load dictionary
before = time.process_time()
loaded = load(dictionary)
after = time.process_time()

# Exit if dictionary not loaded
if not loaded:
    print(f"Could not load {dictionary}.")
    sys.exit(1)

# Calculate time to load dictionary
time_load = after - before

# Try to open text
text = sys.argv[2] if len(sys.argv) == 3 else sys.argv[1]
file = open(text, "r", encoding="latin_1")
if not file:
    print("Could not open {}.".format(text))
    unload()
    sys.exit(1)

```

```
46  # Prepare to report misspellings
47  print("\nMISSPELLED WORDS\n")
48
49  # Prepare to spell-check
50  word = ""
51  index, misspellings, words = 0, 0, 0
52
53  # Spell-check each word in file
54  while True:
55      c = file.read(1)
56      if not c:
57          break
58
59      # Allow alphabetical characters and apostrophes (for possessives)
60      if re.match(r"[A-Za-z]", c) or (c == "'" and index > 0):
61
62          # Append character to word
63          word += c
64          index += 1
65
66          # Ignore alphabetical strings too long to be words
67          if index > LENGTH:
68
69              # Consume remainder of alphabetical string
70              while True:
71                  c = file.read(1)
72                  if not c or not re.match(r"[A-Za-z]", c):
73                      break
74
75              # Prepare for new word
76              index, word = 0, ""
77
78      # Ignore words with numbers (like MS Word can)
79      elif c.isdigit():
80
81          # Consume remainder of alphanumeric string
82          while True:
83              c = file.read(1)
84              if not c or (not c.isalpha() and not c.isdigit()):
85                  break
86
87          # Prepare for new word
88          index, word = 0, ""
89
90      # We must have found a whole word
```

```
91      elif index > 0:
92
93          # Update counter
94          words += 1
95
96          # Check word's spelling
97          before = time.process_time()
98          misspelled = not check(word)
99          after = time.process_time()
100
101         # Update benchmark
102         time_check += after - before
103
104         # Print word if misspelled
105         if misspelled:
106             print(word)
107             misspellings += 1
108
109         # Prepare for next word
110         index, word = 0, ""
111
112 # Close file
113 file.close()
114
115 # Determine dictionary's size
116 before = time.process_time()
117 n = size()
118 after = time.process_time()
119
120 # Calculate time to determine dictionary's size
121 time_size = after - before
122
123 # Unload dictionary
124 before = time.process_time()
125 unloaded = unload()
126 after = time.process_time()
127
128 # Abort if dictionary not unloaded
129 if not unloaded:
130     print(f"Could not load {dictionary}.")
131     sys.exit(1)
132
133 # Calculate time to determine dictionary's size
134 time_unload = after - before
135
```

```
    # Report benchmarks
    print(f"\nWORDS MISSPELLED:     {misspellings}")
    print(f"WORDS IN DICTIONARY:  {n}")
    print(f"WORDS IN TEXT:        {words}")
    print(f"TIME IN load:         {time_load:.2f}")
    print(f"TIME IN check:        {time_check:.2f}")
    print(f"TIME IN size:         {time_size:.2f}")
    print(f"TIME IN unload:       {time_unload:.2f}")
    print(f"TOTAL TIME:           {time_load + time_check + time_size + time_unload:.2f}\n")

    # Success
    sys.exit(0)
```

```
# Recognizes a greeting

# Get input
words = input("Say something!\n").lower()

# Respond to speech
if "hello" in words:
    print("Hello to you too!")
elif "how are you" in words:
    print("I am well, thanks!")
elif "goodbye" in words:
    print("Goodbye to you too!")
else:
    print("Huh?")
```

```python
# Recognizes a voice
# https://pypi.org/project/SpeechRecognition/

import speech_recognition

# Obtain audio from the microphone
recognizer = speech_recognition.Recognizer()
with speech_recognition.Microphone() as source:
    print("Say something!")
    audio = recognizer.listen(source)

# Recognize speech using Google Speech Recognition
print("Google Speech Recognition thinks you said:")
print(recognizer.recognize_google(audio))
```

```python
# Responds to a greeting
# https://pypi.org/project/SpeechRecognition/

import speech_recognition

# Obtain audio from the microphone
recognizer = speech_recognition.Recognizer()
with speech_recognition.Microphone() as source:
    print("Say something!")
    audio = recognizer.listen(source)

# Recognize speech using Google Speech Recognition
words = recognizer.recognize_google(audio)

# Respond to speech
if "hello" in words:
    print("Hello to you too!")
elif "how are you" in words:
    print("I am well, thanks!")
elif "goodbye" in words:
    print("Goodbye to you too!")
else:
    print("Huh?")
```

```python
# Responds to a name
# https://pypi.org/project/SpeechRecognition/

import re
import speech_recognition

# Obtain audio from the microphone
recognizer = speech_recognition.Recognizer()
with speech_recognition.Microphone() as source:
    print("Say something!")
    audio = recognizer.listen(source)

# Recognize speech using Google Speech Recognition
words = recognizer.recognize_google(audio)

# Respond to speech
matches = re.search("my name is (.*)", words)
if matches:
    print(f"Hey, {matches[1]}.")
else:
    print("Hey, you.")
```