```
../README.txt

1/hello0.c
1/hello0.py

6/speller/dictionary.py
6/filter/{blur,edges}.py
6/faces/{detect,recognize}.py

1/hello1.c
1/hello{1,2,3}.py

1/calculator0.c
1/calculator{0,1}.py

1/compare3.c
1/compare3.py

4/compare.py
1/agree.c
1/agree{0,1,2}.py
4/copy.py
2/uppercase{0,1}.py

1/meow0.c
1/meow0.py
1/meow1.c
1/meow1.py
1/meow2.c
1/meow2.py
1/meow3.c
1/meow3.py
1/meow4.c
1/meow4.py
1/meow5.c
1/meow5.py

1/calculator{2,3,4,5}.py

1/mario0.py
1/mario1.c
1/mario{1,2,3,4,5,6,7}.py
```

```
43
44  2/scores{0,1,2}.py
45  3/phonebook{0,1,2,3,4}.py
46
47  2/greet{0,1,2}.py
48  2/exit.py
49
50  4/phonebook*.py
51
52  6/moo.py
53  6/qr/qr.py
```

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


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


def load(dictionary):
    """Load dictionary into memory, returning true if successful else false"""
    with open(dictionary) as file:
        words.update(file.read().splitlines())
    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
# Blurs an image

from PIL import Image, ImageFilter

# Blur image
before = Image.open("bridge.bmp")
after = before.filter(ImageFilter.BoxBlur(1))
after.save("out.bmp")
```

```python
# Blurs an image

from PIL import Image, ImageFilter

# Find edges
before = Image.open("bridge.bmp")
after = before.filter(ImageFilter.FIND_EDGES)
after.save("out.bmp")
```

```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("office.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("toby.jpg")
encoding = face_recognition.face_encodings(known_image)[0]

# Load an image with unknown faces
unknown_image = face_recognition.load_image_file("office.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

```

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

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

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

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

```python
# get_string and print, with concatenation

from cs50 import get_string

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

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

from cs50 import get_string

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

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

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

```c
// Addition with int

#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: ");

    // Perform addition
    printf("%i\n", x + y);
}
```

```python
# Addition with int [using get_int]

from cs50 import get_int

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

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

# Perform addition
print(x + y)
```

```python
# Addition with int [using input]

# Prompt user for x
x = int(input("x: "))

# Prompt user for y
y = int(input("y: "))

# Perform addition
print(x + y)
```

```c
// Conditionals, Boolean expressions, relational operators

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

int main(void)
{
    // Prompt user for integers
    int x = get_int("What's x? ");
    int y = get_int("What's y? ");

    // Compare integers
    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
# Conditionals, Boolean expressions, relational operators

from cs50 import get_int

# Prompt user for integers
x = get_int("What's x? ")
y = get_int("What's y? ")

# Compare integers
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")
```

```python
# Compares two strings

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

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

```c
// Logical operators

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

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

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

```python
# Logical operators

from cs50 import get_string

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

# 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? ")

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

```python
# Logical operators, using lists

from cs50 import get_string

# Prompt user to agree
s = get_string("Do you agree? ").lower()

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

```python
# Capitalizes a copy of a string

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

# Capitalize copy of string
t = s.capitalize()

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

```python
# Uppercases string one character at a time

before = input("Before: ")
print("After:  ", end="")
for c in before:
    print(c.upper(), end="")
print()
```

```python
# Uppercases string all at once

before = input("Before: ")
after = before.upper()
print(f"After:  {after}")
```

```c
// Opportunity for better design

#include <stdio.h>

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

```python
# Opportunity for better design

print("meow")
print("meow")
print("meow")
```

```c
// Demonstrates while loop

#include <stdio.h>

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

```python
# Demonstrates while loop

i = 0
while i < 3:
    print("meow")
    i += 1
```

```c
// Demonstrates for loop

#include <stdio.h>

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

```python
# Opportunity for better design

for i in [0, 1, 2]:
    print("meow")
```

```python
# Better design

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

```c
// Abstraction

#include <stdio.h>

void meow(void);

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

// Meow once
void meow(void)
{
    printf("meow\n");
}
```

```python
# Abstraction

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

# Meow once
def meow():
    print("meow")


main()
```

```c
// Abstraction with parameterization

#include <stdio.h>

void meow(int n);

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

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

```python
# Abstraction with parameterization

def main():
    meow(3)


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


main()
```

```python
# Division with integers, demonstration lack of truncation

# Prompt user for x
x = int(input("x: "))

# Prompt user for y
y = int(input("y: "))

# Divide x by y
z = x / y
print(z)
```

```python
# Floating-point imprecision

# Prompt user for x
x = int(input("x: "))

# Prompt user for y
y = int(input("y: "))

# Divide x by y
z = x / y
print(f"{z:.50f}")
```

```python
# Implements get_int

def get_int(prompt):
    return int(input(prompt))


def main():

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

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

    # Perform addition
    print(x + y)


main()
```

```python
# Implements get_int with a loop

def get_int(prompt):
    while True:
        try:
            return int(input(prompt))
        except ValueError:
            print("Not an integer")


def main():

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

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

    # Perform addition
    print(x + y)


main()
```

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

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

```c
// Prints a column of height n

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

int main(void)
{
    // Get height of column
    int n;
    do
    {
        n = get_int("Height: ");
    }
    while (n < 1);

    // Print column of bricks
    for (int i = 0; i < n; i++)
    {
        printf("#\n");
    }
}
```

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

from cs50 import get_int

while True:
    n = get_int("Height: ")
    if n > 0:
        break

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

```python
# Prints a column of bricks, using a helper function to get input

from cs50 import get_int


def main():
    height = get_height()
    for i in range(height):
        print("#")


def get_height():
    while True:
        n = get_int("Height: ")
        if n > 0:
            return n


main()
```

```python
# Prints a column of bricks, catching exceptions


def main():
    height = get_height()
    for i in range(height):
        print("#")


def get_height():
    while True:
        try:
            n = int(input("Height: "))
            if n > 0:
                return n
        except ValueError:
            print("Not an integer")


main()
```

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

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

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

print("?" * 4)
```

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

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

```python
# Prints a 3-by-3 grid of bricks with loop and * operator

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

```python
# Averages three numbers using a list

# Scores
scores = [72, 73, 33]

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

```python
# Averages three numbers using a list and a loop

from cs50 import get_int

# Get scores
scores = []
for i in range(3):
    score = get_int("Score: ")
    scores.append(score)

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

```python
# Averages three numbers using a list and a loop with + operator

from cs50 import get_int

# Get scores
scores = []
for i in range(3):
    score = get_int("Score: ")
    scores += [score]

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

```python
# Implements linear search for names using loop

# A list of names
names = ["Carter", "David", "John"]

# Ask for name
name = input("Name: ")

# Search for name
for n in names:
    if name == n:
        print("Found")
        break
else:
    print("Not found")
```

```python
# Implements linear search for names using `in`

# A list of names
names = ["Carter", "David", "John"]

# Ask for name
name = input("Name: ")

# Search for name
if name in names:
    print("Found")
else:
    print("Not found")
```

```python
# Implements a phone book as a list of dictionaries

from cs50 import get_string

people = [
    {"name": "Carter", "number": "+1-617-495-1000"},
    {"name": "David", "number": "+1-617-495-1000"},
    {"name": "John", "number": "+1-949-468-2750"},
]

# Search for name
name = get_string("Name: ")
for person in people:
    if person["name"] == name:
        number = person["number"]
        print(f"Found {number}")
        break
else:
    print("Not found")
```

```python
# Implements a phone book as a list of dictionaries, without a variable

from cs50 import get_string

people = [
    {"name": "Carter", "number": "+1-617-495-1000"},
    {"name": "David", "number": "+1-617-495-1000"},
    {"name": "John", "number": "+1-949-468-2750"},
]

# Search for name
name = get_string("Name: ")
for person in people:
    if person["name"] == name:
        print(f"Found {person['number']}")
        break
else:
    print("Not found")
```

```python
# Implements a phone book using a dictionary

from cs50 import get_string

people = {
    "Carter": "+1-617-495-1000",
    "David": "+1-617-495-1000",
    "John": "+1-949-468-2750",
}

# Search for name
name = get_string("Name: ")
if name in people:
    print(f"Number: {people[name]}")
else:
    print("Not found")
```

```python
# Prints a command-line argument

from sys import argv

if len(argv) == 2:
    print(f"hello, {argv[1]}")
else:
    print("hello, world")
```

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

from sys import argv

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

```python
# Printing command-line arguments

from sys import argv

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

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

import sys

if len(sys.argv) != 2:
    print("Missing command-line argument")
    sys.exit(1)

print(f"hello, {sys.argv[1]}")
sys.exit(0)
```

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

import csv

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

# Get name and number
name = input("Name: ")
number = input("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

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

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

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

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

import csv

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

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

    # Print to file
    writer = csv.DictWriter(file, fieldnames=["name", "number"])
    writer.writerow({"name": name, "number": number})
```

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

import os
import qrcode

# Generate QR code
img = qrcode.make("https://youtu.be/xvFZjo5PgG0")

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

# Open file
os.system("open qr.png")
```