```
# Prints four question marks

print("????")
```

```python
# Prints four question marks using a loop

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

```python
# Prints any number of question marks, as specified by user

from cs50 import get_int

n = get_int("Number: ")
for i in range(n):
    print("?", end="")
print()
```

```python
# Prints a positive number of question marks, as specified by user

from cs50 import get_int

# Prompt user for a positive number
while True:
    n = get_int("Positive number: ")
    if n > 0:
        break

# Print out that many bricks
for i in range(n):
    print("#")
```

```python
# Prints a square of bricks, sized as specified by user

from cs50 import get_int

# Prompt user for a positive number
while True:
    n = get_int("Positive number: ")
    if n > 0:
        break

# Print out this many rows
for i in range(n):

    # Print out this many columns
    for j in range(n):
        print("#", end="")
    print()
```

```
Pillow
```

```python
import sys
from PIL import Image

if len(sys.argv) != 4:
    sys.exit("Usage: python resize.py n infile outfile")

n = int(sys.argv[1])
infile = sys.argv[2]
outfile = sys.argv[3]

inimage = Image.open(infile)
width, height = inimage.size
outimage = inimage.resize((width * n, height * n))

outimage.save(outfile)
```

```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"""
    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)
```

```
45
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
136   # Report benchmarks
137   print(f"\nWORDS MISSPELLED:     {misspellings}")
138   print(f"WORDS IN DICTIONARY:  {n}")
139   print(f"WORDS IN TEXT:        {words}")
140   print(f"TIME IN load:         {time_load:.2f}")
141   print(f"TIME IN check:        {time_check:.2f}")
142   print(f"TIME IN size:         {time_size:.2f}")
143   print(f"TIME IN unload:       {time_unload:.2f}")
144   print(f"TOTAL TIME:           {time_load + time_check + time_size + time_unload:.2f}\n")
145
146   # Success
147   sys.exit(0)
```

```python
# Logical operators

from cs50 import get_char

# Prompt user for answer
c = get_string("Answer: ")

# Check answer
if c == "Y" or c == "y":
    print("yes")
elif c == "N" or c == "n":
    print("no")
```

```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")
```

```python
# Opportunity for better design

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

```python
# Better design

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

```python
# Abstraction


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


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


if __name__ == "__main__":
    main()
```

```python
# Abstraction with parameterization


def main():
    cough(3)


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


if __name__ == "__main__":
    main()
```

```
# get_int and print

from cs50 import get_int

f = get_float("Float: ")
print("hello,", f)
```

```python
# Floating-point arithmetic

from cs50 import get_float

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

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

# Perform division
print(f"x / y = {(x / y):.50f}")
```

```
1  print("hello, world")
```

```python
# Floating-point imprecision

print(f"{1/10:.55f}")
```

```python
# get_int and print

from cs50 import get_int

i = get_int("Integer: ")
print("hello,", i)
```

```python
# Integer arithmetic

from cs50 import get_int

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

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

# Perform arithmetic
print(f"x + y = {x + y}")
print(f"x - y = {x - y}")
print(f"x * y = {x * y}")
print(f"x / y = {x / y}")
print(f"x // y = {x // y}")
print(f"x mod y = {x % y}")
```

```python
# Integer overflow

from time import sleep

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

```python
# Remainder operation

from cs50 import get_int

# Prompt user for integer
n = get_int("n: ");

# Check parity of integer
if n % 2 == 0:
    print("even")
else:
    print("odd")
```

```python
# Abstraction and scope

from cs50 import get_int


def main():
    i = get_positive_int("Positive integer: ")
    print(i)


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


if __name__ == "__main__":
    main()
```

```python
# Return value

from cs50 import get_int


def main():
    x = get_int("x: ")
    print(square(x))


def square(n):
    """Return square of n"""
    return n**2


if __name__ == "__main__":
    main()
```

```python
# Conditions and relational operators

from cs50 import get_int

# Prompt user for number
i = get_int("number: ")

# Check sign of number
if i < 0:
    print("negative")
elif i > 0:
    print("positive")
else:
    print("zero")
```

```python
# get_string and print

from cs50 import get_string

s = get_string("Name: ")
print("hello,", s)
```

```python
# Demonstrates format

from cs50 import get_string

s = get_string("Name: ")
print(f"hello, {s}")
```

```python
# Printing 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

from sys import argv

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

```python
# Printing characters in an array of strings

from sys import argv

for s in argv:
    for c in s:
        print(c)
    print()
```

```python
# Explicitly casts chars to ints

from cs50 import get_string

s = get_string("String: ")
for c in s:
    print(f"{c} {ord(c)}")
```

```python
# Buggy example for help50

s = get_string("Name: ")
print(f"hello, {s}")
```

```python
# Capitalizes string using str method

from cs50 import get_string

s = get_string()
for c in s:
    print(c.upper(), end="")
print()
```

```
# Exits with explicit value

import sys

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

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

```python
# Generates a bar chart of three scores

from cs50 import get_int

# Get scores from user
score1 = get_int("Score 1: ")
score2 = get_int("Score 2: ")
score3 = get_int("Score 3: ")

# Generate first bar
print("Score 1: ", end="");
for i in range(score1):
    print("#", end="")
print()

# Generate second bar
print("Score 2: ", end="");
for i in range(score2):
    print("#", end="")
print()

# Generate third bar
print("Score 3: ", end="");
for i in range(score3):
    print("#", end="")
print()
```

```python
# Generates a bar chart of three scores

from cs50 import get_int


def main():

    # Get scores from user
    score1 = get_int("Score 1: ")
    score2 = get_int("Score 2: ")
    score3 = get_int("Score 3: ")

    # Chart first score
    print("Score 1: ", end="")
    chart(score1)

    # Chart second score
    print("Score 2: ", end="")
    chart(score2)

    # Chart third score
    print("Score 3: ", end="")
    chart(score3)


def chart(score):

    # Output one hash per point
    for i in range(score):
        print("#", end="")
    print()


if __name__ == "__main__":
    main()
```

```python
# Generates a bar chart of three scores using a list

from cs50 import get_int


def main():

    # Get scores from user
    scores = []
    for i in range(3):
        scores.append(get_int(f"Score {i + 1}: "))

    # Chart scores
    for i in range(len(scores)):
        print(f"Score {i + 1}: ", end="")
        chart(scores[i])

def chart(score):
    """Generate bar"""

    # Output one hash per point
    for i in range(score):
        print("#", end="")
    print()


if __name__ == "__main__":
    main()
```

```python
# Prints string char by char

from cs50 import get_string

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

```python
# Prints string char by char, one per line

from cs50 import get_string

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

```python
# Determines the length of a string

from cs50 import get_string

s = get_string("Name: ")
print(len(s))
```

```python
# Compares two strings for equality

from cs50 import get_string

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

# Compare strings for equality
if s == t:
    print("same")
else:
    print("different")
```

```python
# Compares two strings for equality while checking for errors

import sys
from cs50 import get_string

# Get a string
s = get_string("s: ")
if s is None:
    sys.exit(1)

# Get another string
t = get_string("t: ")
if t is None:
    sys.exit(1)

# Compare strings for equality
if s == t:
    print("same")
else:
    print("different")
```

```python
# Compares two strings for equality while checking (succinctly) for errors

import sys
from cs50 import get_string

# Get a string
s = get_string("s: ")
if not s:
    sys.exit(1)

# Get another string
t = get_string("t: ")
if not t:
    sys.exit(1)

# Compare strings for equality
if s == t:
    print("same")
else:
    print("different")
```

```python
# Capitalizes a copy of a string while checking for errors

import sys
from cs50 import get_string

# Get a string
s = get_string("s: ")
if not s:
    sys.exit(1)

# Capitalize first letter in copy
t = s.capitalize()

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

sys.exit(0)
```

```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
# Implements a list of unique numbers

from cs50 import get_int

# Memory for numbers
numbers = []

# Prompt for numbers (until EOF)
while True:

    # Prompt for number
    number = get_int("number: ")

    # Check for EOF
    if not number:
        break

    # Check whether number is already in list
    if number not in numbers:

        # Add number to list
        numbers.append(number)

# Print numbers
print()
for number in numbers:
    print(number)
```

```python
# Demonstrates objects

from cs50 import get_string

# Space for students
students = []

# Prompt for students' names and dorms
for i in range(3):
    name = get_string("name: ")
    dorm = get_string("dorm: ")
    students.append({"name": name, "dorm": dorm})

# Print students' names and dorms
for student in students:
    print(f"{student['name']} is in {student['dorm']}.")
```

```python
# Demonstrates file I/O

import csv
from cs50 import get_string

# Space for students
students = []

# Prompt for students' names and dorms
for i in range(3):
    name = get_string("name: ")
    dorm = get_string("dorm: ")
    students.append({"name": name, "dorm": dorm})

with open("students.csv", "w") as file:
    writer = csv.writer(file)
    for student in students:
        writer.writerow((student["name"], student["dorm"]))
```