# Prints four question marks

```python
print("????")
```
# Prints four question marks using a loop

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

# 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()
# Prints a positive number of question marks, as specified by user

```python
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("#")
```
# 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()
1 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)
```
# Words in dictionary

```python
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
```
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)
# Prepare to report misspellings
print("\nMISSPELLED WORDS\n")

# Prepare to spell-check
word = ""
index, misspellings, words = 0, 0, 0

# Spell-check each word in file
while True:
    c = file.read(1)
    if not c:
        break

    # Allow alphabetical characters and apostrophes (for possessives)
    if re.match(r"[A-Za-z]", c) or (c == "" and index > 0):

        # Append character to word
        word += c
        index += 1

        # Ignore alphabetical strings too long to be words
        if index > LENGTH:
            # Consume remainder of alphabetical string
            while True:
                c = file.read(1)
                if not c or not re.match(r"[A-Za-z]", c):
                    break

            # Prepare for new word
            index, word = 0, ""

    # Ignore words with numbers (like MS Word can)
    elif c.isdigit():

        # Consume remainder of alphanumeric string
        while True:
            c = file.read(1)
            if not c or (not c.isalpha() and not c.isdigit()):
                break

        # Prepare for new word
        index, word = 0, ""
# We must have found a whole word
elif index > 0:
    # Update counter
    words += 1
    # Check word's spelling
    before = time.process_time()
    misspelled = not check(word)
    after = time.process_time()
    # Update benchmark
    time_check += after - before
    # Print word if misspelled
    if misspelled:
        print(word)
        misspellings += 1
    # Prepare for next word
    index, word = 0, ""

# Close file
file.close()
# Determine dictionary's size
before = time.process_time()
N = size()
after = time.process_time()
# Calculate time to determine dictionary's size
time_size = after - before
# Unload dictionary
before = time.process_time()
unloaded = unload()
after = time.process_time()
# Abort if dictionary not unloaded
if not unloaded:
    print(f"Could not load {dictionary}.")
    sys.exit(1)
# Calculate time to determine dictionary's size

time_unload = after - before

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

# Success
sys.exit(0)
# Logical operators

```python
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")
```
# Conditions and relational operators

```python
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")
```
# Opportunity for better design

```python
print("cough")
print("cough")
print("cough")
```
# Better design

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

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

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

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

```python
from cs50 import get_int

f = get_float("Float: ")
print("hello,", f)
```
# 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\}"
print("hello, world")
# Floating-point imprecision

```python
print(f"{1/10:.55f}"
```
# get_int and print

from cs50 import get_int

i = get_int("Integer: ")
print("hello,", i)
# Integer arithmetic

```python
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}"
```

# Integer overflow

```python
time.sleep
```

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

```python
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")
```
# Abstraction and scope

```python
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()
```

# Return value

```python
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()
```
# Conditions and relational operators

```python
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")
```
# get_string and print

```python
from cs50 import get_string

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

```python
from cs50 import get_string

s = get_string("Name: ")
print(f"hello, {s}"")
```
# Printing a command-line argument

```python
from sys import argv

if len(argv) == 2:
    print(f"hello, {argv[1]}")
else:
    print("hello, world")
```
# Printing command-line arguments

```python
from sys import argv

for s in argv:
    print(s)
```
# Printing characters in an array of strings

```python
from sys import argv

for s in argv:
    for c in s:
        print(c)
    print()
```
Explicitly casts chars to ints

```python
from cs50 import get_string

s = get_string("String: ")
for c in s:
    print(f"{c} \{ord(c)}")
```
# Buggy example for help50

```
s = get_string("Name: ")
print(f"hello, {s}")
```
# Capitalizes string using str method

```python
from cs50 import get_string

s = get_string()
for c in s:
    print(c.upper(), end="")
print()
```
# Exits with explicit value

```python
import sys

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

print(f"hello, {argv[1]}")
sys.exit(0)
```
# Generates a bar chart of three scores

```python
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="\n")
for i in range(score1):
    print("#", end="\n")
print()

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

# Generate third bar
print("Score 3: ", end="\n")
for i in range(score3):
    print("#", end="\n")
print()
```
# 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()
# Generates a bar chart of three scores using a list

```python
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()
```
# Prints string char by char

from cs50 import get_string

s = get_string("Input: ")
print("Output: ")
for c in s:
    print(c)
# Prints string char by char, one per line
rom cs50 import get_string

s = get_string("Input: ");
print("Output:");
for c in s:
    print(c)
# Determines the length of a string

```python
from cs50 import get_string

s = get_string("Name: ")
print(len(s))
```
# Compares two strings for equality

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

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

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

```python
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)
```
# Swaps two integers

```python
x = 1
y = 2
print(f"x is {x}, y is {y}\n"
)x, y = y, x
print(f"x is {x}, y is {y}\n")
```
# Implements a list of unique numbers

```python
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)
```

# Demonstrates objects

```python
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']}.")
```
# Demonstrates file I/O

```python
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"]))
```