```
"""
File: myinfo.py
Project 1.2
Prints my name, address, and phone number.
"""

print("Ken Lambert")
print("Virginia")
print("345-9110")


```

```
"""
File: rectangle.py
Project 1.4
Computes and prints the area of a rectangle, given an
input width and height.
"""

width = float(input("Enter the width: "))
height = float(input("Enter the height: "))
area = width * height
print("The area is", area, "square units.")


```

```
"""
File: triangle.py
Project 1.5
Computes and prints the area of a triangle, given an
input base and height.
"""

base = float(input("Enter the base: "))
height = float(input("Enter the height: "))
area = .5 * base * height
print("The area is", area, "square units.")


```

```
"""
File: circle.py
Project 1.6
Computes and prints the area of a circle, given an
input radius.
"""

radius = float(input("Enter the radius: "))
area = 3.14 * radius ** 2
print("The area is", area, "square units.")


```

```
"""
File: myinfo.py
Project 1.7
Inputs the user's name and age and outputs a sentence
containing them.
"""

name = input("Enter your name: ")
years = int(input("Enter your age: "))
print(name, "is", years, "years old.")


```

```
"""
Program: taxform.py
Project 2.1

Compute a person's income tax.

1. Significant constants
        tax rate
        standard deduction
        deduction per dependent
2. The inputs are
        gross income
        number of dependents
3. Computations:
        net income = gross income - the standard deduction -
                        a deduction for each dependent
        income tax = is a fixed percentage of the net income
4. The outputs are
        the income tax, rounded to two figures
"""

# Initialize the constants
TAX_RATE = 0.20
STANDARD_DEDUCTION = 10000.0
DEPENDENT_DEDUCTION = 3000.0

# Request the inputs
grossIncome = float(input("Enter the gross income: "))
numDependents = int(input("Enter the number of dependents: "))

# Compute the income tax
taxableIncome = grossIncome - STANDARD_DEDUCTION - \
                DEPENDENT_DEDUCTION * numDependents
incomeTax = taxableIncome * TAX_RATE

# Display the income tax
print("The income tax is $" + str(round(incomeTax, 2)))

```

```
"""
Program: cube.py
Project 2.2

Compute the surface area of a cube.

The input is the cube's edge.
The output is the surface area of the cube.
"""

# Request the input
edge = float(input("Enter the cube's edge: "))

# Compute the surface area
surfaceArea = edge * edge * 6

# Display the surface area
print("The surface area is", surfaceArea, "square units.")

```

```
"""
Program: fivestar.py
Project 2.3

Calculate the total charge for a customer's video rentals,
given the number of each type of video.

New video rental = $3.00
Oldie rental = $2.00

"""

# Initialize constants
NEW_RENTAL = 3.00
OLDIE_RENTAL = 2.00

# Request the inputs
newOnes = int(input("Enter the number of new videos: "))
oldOnes = int(input("Enter the number of oldies: "))

# Compute the results
totalCost = NEW_RENTAL * newOnes + OLDIE_RENTAL * oldOnes

# Display the results
print("The total cost is $" + str(round(totalCost, 2)))

```

```
"""
Program: sphere.py
Project 2.4

Given the radius compute the diameter, circumference, and volume
of a sphere.

Useful facts:
   diameter = 2 * radius
   circumference = diameter * 3.14
   surface area = 4 * PI * radius * radius
   volume = 4/3 * PI * radius * radius * radius

"""

import math

# Request the input
radius = float(input("Enter the sphere's radius: "))

# Compute the results
diameter = 2 * radius
circumference = diameter * math.pi
surfaceArea = 4 * math.pi * radius * radius
volume = 4/3.0 * math.pi * radius * radius * radius

# Display the results
print("Diameter     :", diameter)
print("Circumference:", circumference)
print("Surface area :", surfaceArea)
print("Volume       :", volume)

```

```
"""
Program: momentum.py
Project 2.5

Given an object's mass and velocity, compute its momentum.

"""

# Request the input
mass = float(input("Enter the object's mass: "))
velocity = float(input("Enter the object's velocity: "))

# Compute the results
momentum = mass * velocity

# Display the results
print("The object's momentum is", momentum)

```

```
"""
Program: momentum.py
Project 2.6

Given an object's mass and velocity, compute its momentum
and kinetic energy.

"""

# Request the input
mass = float(input("Enter the object's mass: "))
velocity = float(input("Enter the object's velocity: "))

# Compute the results
momentum = mass * velocity
kineticEnergy = (mass * velocity ** 2) / 2

# Display the results
print("The object's momentum is", momentum)
print("The object's kinetic energy is", kineticEnergy)

```

```
"""
Program: minutes.py
Project 2.7

Compute the number of minutes in a year.

Useful facts:
   1 year = 365 days (we ignore leap years)
   1 day = 24 hours
   1 hour = 60 minutes

"""

# Compute the result
minutes = 365 * 24 * 60

# Display the result
print("The number of minutes in a year is", minutes)

```

```
"""
Program: lightyear.py
Project 2.8

Compute the distance that light travels in a year.

Useful facts:
   rate = 3 * 10 ** 8 meters per second
   seconds in a year = 365 * 24 * 60 ** 2

"""

# Compute the result
rate = 3 * 10 ** 8
seconds = 365 * 24 * 60 ** 2
distance = rate * seconds

# Display the result
print("Light travels", distance, "meters in a year.")

```

```
"""
Program: klickstonauts.py
Project 2.9

Convert kilometers to nautical miles.

Useful facts:
   1 kilometer = 1/10000 of the distance between the North Pole and the Equator
   there are 90 degrees between the North Pole and the Equator
   1 degree = 60 minutes of arc
   1 nautical mile =  1 minute of arc


"""

# Request the input
klicks = float(input("Enter the number of kilometers: "))

# Compute the result
nauts = klicks * 90 * 60 / 10000.0

# Display the result
print("The number of nautical miles is", nauts)

```

```
"""
Program: employeepay.py
Project 2.10

Compute an employee's total weekly pay.

Useful facts:
     An employee's total weekly pay equals the
     hourly wage multiplied by the total number
     of regular hours plus any overtime pay.  Overtime
     pay equals the total overtime hours multiplied by 1.5
     times the hourly wage.

Write a program that takes as inputs the hourly wage,
total regular hours, and total overtime hours and
displays an employee's total weekly pay.

"""

# Request the inputs
wage = float(input("Enter the wage: $"))
regularHours = float(input("Enter the regular hours: "))
overtimeHours = float(input("Enter the overtime hours: "))


# Compute the result
totalPay = regularHours * wage + overtimeHours * wage * 1.5

# Display the result
print("The total weekly pay is $" + str(round(totalPay, 2)))

```

```
"""
Program: equilateral.py
Project 3.1

Determine whether or not three input sides compose an
equilateral triangle.
"""

# Request the inputs
side1 = int(input("Enter the first side: "))
side2 = int(input("Enter the second side: "))
side3 = int(input("Enter the third side: "))

# Determine the result and display it
if side1 == side2 and side2 == side3:
    print("The triangle is equilateral.")
else:
    print("The triangle is not equilateral.")


```

```
"""
Program: right.py
Project 3.2

Determine whether or not three input sides compose a
right triangle.
"""

# Request the inputs
side1 = int(input("Enter the first side: "))
side2 = int(input("Enter the second side: "))
side3 = int(input("Enter the third side: "))

# Compute the squares
square1 = side1 ** 2
square2 = side2 ** 2
square3 = side3 ** 2

# Determine the result and display it
if square1 + square2 == square3 or \
   square2 + square3 == square1 or \
   square1 + square3 == square2:
    print("The triangle is a right triangle.")
else:
    print("The triangle is not a right triangle.")


```

```
"""
Program: guess.py
Project 3.3

The computer guesses the user's number using the minimum
number of attempts and prevents cheating by the user.
"""

import math

smaller = int(input("Enter the smaller number: "))
larger = int(input("Enter the larger number: "))
count = 0
maxGuesses = round(math.log(larger - smaller + 1, 2))
while True:
    count += 1
    print(smaller, larger)
    yourNumber = (smaller + larger) // 2
    print("Your number is", yourNumber)
    answer = input("Enter =, <, or >: ")
    if answer == "=":
        print("Hooray, I've got it in", count, "tries!")
        break
    elif count == maxGuesses:
        print("I'm out of guesses, and you cheated!")
        break
    elif answer == "<":
        larger = yourNumber - 1
    else:
        smaller = yourNumber + 1

```

```
"""
Program: bouncy.py
Project 3.4

This program calculates the total distance a ball travels as it bounces given:
1. the initial height of the ball
2. its bounciness index
3. the number of times the ball is allowed to continue bouncing

"""

height = float(input("Enter the height from which the ball is dropped: "))
bounciness = float(input("Enter the bounciness index of the ball: "))
distance = 0
bounces = int(input("Enter the number of times the ball is allowed to continue bouncing: "))
for eachPass in range(bounces):
    distance += height
    height *= bounciness
    distance += height
print('\nTotal distance traveled is:', distance, 'units.')


```

```
"""
Program: population.py
Project 3.5

Predict popultation growth, assuming that no
organisms die.

Inputs:
   initial number of organisms
   rate of growth (a float > 1)
   the number of hours to achieve the rate
   number of hours of growth
"""

# Accept the inputs

number = int(input("Enter the initial number of organisms
: "))
rate = float(input("Enter the rate of growth [a real
number > 1]: "))
cycleHours = int(input("Enter the number of hours to
achieve the rate of growth: "))
totalHours = int(input("Enter the total hours of growth: "
))

# Calculate the number of cycles

cycles = totalHours // cycleHours

# Calculate the population after an integral number of
cycles

for eachPass in range(cycles):
    number = number * rate

print("The total population is", int(number))

```

```
"""
Program: leibniz.py
Project 3.6

This program approximates the value of pi using an algorithm
designed by the German mathematician Gottfried Leibniz. The
algorithm is as follows:

pi = 4 - 4 / 3 + 4 / 5 - 4 / 7 + . . .

This program allows the user to specify the number of iterations
to use in the approximation.
"""

import math


iterations = int(input("Enter the number of iterations: "))
pioverfour = 0
numerator = 1
denominator = 1
for count in range(iterations):
    pioverfour += numerator / denominator
    numerator = -numerator
    denominator += 2
print("The approximation of pi is", pioverfour * 4)
print("Compare this to the computer's estimation: ", math.pi)

```

```
"""
Program: salary.py
Project 3.7

Compute a school district's salary schedule.

Inputs
   starting salary
   annual percentage increase
   number of years for which to print the schedule

Outputs
    Two columns containing the year and the salary
    after the increase.
"""

# Accept the inputs
startSal = float(input("Enter the starting salary: $"))
increase = int(input("Enter the annual % increase: "))
years = int(input("Enter the number of years: "))

# Compute and display the results
print("Year   Salary\n-------------")
multiplier = 1 + increase / 100
nextSal = startSal
for year in range(1, years + 1):
    print("%2d%12.2f" % (year, nextSal))
    nextSal *= multiplier

```

```
"""
Program: gcd.py
Project 3.8

Compute and print the greatest common divisor of two input
integers.
"""

first = int(input("Enter the smaller number: "))
second = int(input("Enter the larger number: "))

while first > 0:
    remainder = second % first
    second = first
    first = remainder

print("The greatest common divisor is", second)

```

```
"""
Program: sum.py
Project 3.9

Computes the sum and average of a series of input numbers.
"""

theSum = 0
count = 0
while True:
    number = input("Enter a number or press Enter to quit
: ")
    if number == "":
        break
    theSum += float(number)
    count += 1

print("The sum is", theSum)
if count > 0:
    print("The average is", theSum / count)

```

```
"""
Program: tidbit.py
Project 3.10

Print a payment schedule for a loan to purchase a computer.

Input
   purchase price

Constants
   annual interest rate = 12%
   downpayment = 10% of purchase price
   monthly payment = 5% of purchase price

"""

ANNUAL_RATE = .12
MONTHLY_RATE = ANNUAL_RATE / 12


purchasePrice = float(input("Enter the puchase price: "))

monthlyPayment = .05 * purchasePrice
month = 1
balance = purchasePrice
print("Month  Starting Balance  Interest to Pay  Principal to Pay  Payment  Ending Balance")
while balance > 0:
    if monthlyPayment > balance:
        monthlyPayment = balance
        interest = 0
    else:
        interest = balance * MONTHLY_RATE
    principal = monthlyPayment - interest
    remaining = balance - monthlyPayment
    print("%2d%15.2f%15.2f%17.2f%17.2f%17.2f" % \
          (month, balance, interest, principal, monthlyPayment, remaining))
    balance = remaining
    month += 1


```

```
"""
Program: sevens.py
Project 3.11

Simulate the game of lucky sevens until all funds are depleted.

1) Rules:
        roll two dice
        if the sum equals 7, win $4, else lose $1
2) The input is:
        the amount of money the user is prepared to lose
3) Computations:
        use a random number generator to simulate rolling the dice
        loop until the funds are depleted
        count the number of rolls
        keep track of the maximum amount
4) The outputs are:
        the number of rolls it takes to deplete the funds
        the maximum amount

"""

import random

# Request the input
dollars = int(input("How many dollars do you have? "))

# Initialize variables
maxDollars = dollars
countAtMax = 0
count = 0

# Loop until the money is gone
while dollars > 0:
    count += 1
    # Roll the dice
    die1 = random.randint(1, 6)  # 1-6
    die2 = random.randint(1, 6)  # 1-6
    #Calculate the winnings or losses
    if die1 + die2 == 7:
        dollars += 4
    else:
        dollars -= 1
    #If this is a new maximum, remember it
```

```
45      if dollars > maxDollars:
46          maxDollars = dollars
47          countAtMax = count
48
49  # Display the results
50  print("You are broke after " + str(count) + " rolls.\n" +
    \
51        "You should have quit after " + str(countAtMax) + \
52        " rolls when you had $" + str(maxDollars) + ".")
53
54
55
56
57
58
59
```

```
"""
File: encrypt.py
Probject 4.1

Encypts an input string of the ASCII characters and prints
the result.  The other input is the distance value.
"""

# The ASCII values range from 0 through 127

plainText = input("Enter a message: ")
distance = int(input("Enter the distance value: "))
code = ""
for ch in plainText:
    ordValue = ord(ch)
    cipherValue = ordValue + distance
    if cipherValue > 127:
        cipherValue = distance - (127 - ordValue + 1)
    code +=  chr(cipherValue)
print(code)

```

```
"""
File: decrypt.py
Project 4.2

Decypts an input string characters and prints
the result.  The other input is the distance value.
"""

# The ASCII values range from 0 through 127

code = input("Enter the coded text: ")
distance = int(input("Enter the distance value: "))
plainText = ''
for ch in code:
    ordValue = ord(ch)
    cipherValue = ordValue - distance
    if cipherValue < 0:
        cipherValue = 127 - \
                      (distance - (1 - ordValue))
    plainText +=  chr(cipherValue)
print(plainText)

```

```
"""
File: decrypt.py
Project 4.3

Decypts a file of encrypted text. and prints
the result.  The other input is the distance value.
"""

# The ASCII values range from 0 through 127

# Take the inputs
inName = input("Enter the input file name: ")
outName = input("Enter the output file name: ")
distance = int(input("Enter the distance value: "))

# Open the input file and read the encrypted text
inputFile = open(inName, 'r')
code = inputFile.read()


# Open the output file and write the decrypted text
outFile = open(outName, 'w')
plainText = ''
for ch in code:
    ordValue = ord(ch)
    cipherValue = ordValue - distance
    if cipherValue < 0:
        cipherValue = 127 - \
                      (distance - (1 - ordValue))
    plainText +=  chr(cipherValue)
outFile.write(plainText)
outFile.close()

```

```
"""
File: encrypt.py
Probject 4.3

Encypts a text file.  The inputs are the names of
the input file and the output file and the distance value.
The encrypted code is witten to a new file.
"""

# The ASCII values range from 0 through 127

# Take the inputs
inName = input("Enter the input file name: ")
outName = input("Enter the output file name: ")
distance = int(input("Enter the distance value: "))

# Open the input file and read the plain text
inputFile = open(inName, 'r')
plainText = inputFile.read()

# Open the output file and write the encrypted text
outFile = open(outName, 'w')
code = ""
for ch in plainText:
    ordValue = ord(ch)
    cipherValue = ordValue + distance
    if cipherValue > 127:
        cipherValue = distance - (127 - ordValue + 1)
    code +=  chr(cipherValue)
outFile.write(code)
outFile.close()

```

```
"""
File: decimaltooctal.py
Project 4.4

Converts a decimal integer to a string of octal digits.
"""

decimal = int(input("Enter a decimal integer: "))
if decimal == 0:
    print(0)
else:
    print("Quotient Remainder Octal")
    ostring = ""
    while decimal > 0:
        remainder = decimal % 8
        decimal = decimal // 8
        ostring = str(remainder) + ostring
        print("%5d%8d%12s" % (decimal, remainder, ostring)
)
    print("The octal representation is", ostring)

```

```
"""
File: octaltodecimal.py
Project 4.4

Converts a string of octal digits to a decimal integer.
"""

ostring = input("Enter a string of octal digits: ")
decimal = 0
exponent = len(ostring) - 1
for digit in ostring:
    decimal = decimal + int(digit) * 8 ** exponent
    exponent = exponent - 1
print("The integer value is", decimal)

```

```
"""
File: shiftleft.py
Project 4.5

Shifts the bits in an input string one place to the left.
The leftmost bit wraps around to the rightmost position.
"""

bits = input("Enter a string of bits: ")
if len(bits) > 1:
    bits = bits[1:] + bits[0]
print(bits)

```

```
"""
File: shiftright.py
Project 4.5

Shifts the bits in an input string one place to the right.
The rightmost bit wraps around to the leftmost position.
"""

bits = input("Enter a string of bits: ")
if len(bits) > 1:
    bits = bits[-1] + bits[:-1]
print(bits)

```

```
"""
File: encrypt.py
Probject 4.6

Encypts an input string of characters and prints
the result.
"""

plainText = input("Enter a message: ")

code = ""
for ch in plainText:
    # Add 1 to ASCII value
    ordValue = ord(ch) + 1
    # Convert to binary
    bstring = ""
    while ordValue > 0:
        remainder = ordValue % 2
        ordValue = ordValue // 2
        bstring = str(remainder) + bstring
    # Shift one bit to left
    if len(bstring) > 1:
        bstring = bstring[1:] + bstring[0]
    # Add encrypted character to code string
    code += bstring + " "
print(code)

```

```
"""
File: decrypt.py
Project 4.7

Decypts an input string of characters and prints
the result.
"""

code = input("Enter the coded text: ")
wordList = code.split()
plainText = ""
for word in wordList:
    # Shift one bit to right
    word = word[-1] + word[:-1]
    # Convert to decimal
    decimal = 0
    exponent = len(word) - 1
    for digit in word:
        decimal = decimal + int(digit) * 2 ** exponent
        exponent = exponent - 1
    # Subtract 1 from ASCII value
    decimal -= 1
    # Convert to a character and add to code string
    plainText += chr(decimal)
print(plainText)

```

```
"""
File: copyfile.py
Project 4.8

Copies the text from a given input file to a given
output file.
"""

# Take the inputs
inName = input("Enter the input file name: ")
outName = input("Enter the output file name: ")

# Open the input file and read the text
inputFile = open(inName, 'r')
text = inputFile.read()

# Open the output file and write the text
outFile = open(outName, 'w')
outFile.write(text)
outFile.close()

```

```
"""
File: numberlines.py
Project 4.9

Copies the text from a given input file to a given
output file, numbering them as it goes.
"""

# Take the inputs
inName = input("Enter the input file name: ")
outName = input("Enter the output file name: ")

# Open the files
inputFile = open(inName, 'r')
outputFile = open(outName, 'w')

# Output the numbered lines
lineNumber = 0
for line in inputFile:
    lineNumber += 1
    outputFile.write("%4d> %s" % (lineNumber, line))
outputFile.close()

```

```
"""
File: dif.py
Project 4.10

Deterines whether or not the contents of two text
files are the same.  Outputs "Yes" if that is the
case or "No" and the first two lines that differ if
that is not the case.
"""

# Take the inputs
fileName1 = input("Enter the first file name: ")
fileName2 = input("Enter the second file name: ")

# Open the input files
inputFile1 = open(fileName1, 'r')
inputFile2 = open(fileName2, 'r')

# Read each pair of lines and compare them
while True:
    line1 = inputFile1.readline()
    line2 = inputFile2.readline()
    if line1 == "" and line2 == "":    # Ends of both files
        print("Yes")
        break
    elif line1 != line2:
        print("No")
        print(line1)
        print(line2)
        break


```

```
"""
Program: textanalysis.py
Project 4.9

Computes and displays the Flesch Index and the Grade
Level Equivalent for the readability of a text file.
Updated to count syllables containing multiple vowels
as single syllables.
"""

# Take the inputs
fileName = input("Enter the file name: ")
inputFile = open(fileName, 'r')
text = inputFile.read()

# Count the sentences
sentences = text.count('.') + text.count('?') + \
            text.count(':') + text.count(';') + \
            text.count('!')

# Count the words
words = len(text.split())

# Count the syllables
syllables = 0
vowels = "aeiouAEIOU"
for word in text.split():
    vowelSeen = False         # Corrects for multi-vowel
syllables
    for character in word:
        if not vowelSeen and character in vowels:
            syllables += 1
            vowelSeen = True
        elif not character in vowels:
            vowelSeen = False
    for ending in ['es', 'ed', 'e']:
        if word.endswith(ending):
            syllables -= 1
    if word.endswith('le'):
        syllables += 1

# Compute the Flesch Index and Grade Level
index = 206.835 - 1.015 * (words / sentences) - \
        84.6 * (syllables / words)
level = int(round(0.39 * (words / sentences) + 11.8 * \
                  (syllables / words) - 15.59))
```

```
46
47 # Output the results
48 print("The Flesch Index is", index)
49 print("The Grade Level Equivalent is", level)
50 print(sentences, "sentences")
51 print(words, "words")
52 print(syllables, "syllables")
53
54
55
```

```
"""
Program: payroll.py
Project 4.12

Print a payroll report.

Input
   A file in which each line has the form

   <last name> <hourly wage> <hours worked>

Output
   A report in tabular format.  Each line has the form

   <last name> <hours worked> <total wages>

"""

# Take the inputs
fileName = input("Enter the file name: ")

# Open the input file
inputFile = open(fileName, 'r')

# Read the data and print the report
print("%-15s%6s%15s" % ("Name", "Hours", "Total Pay"))
for line in inputFile:
    dataList = line.split()
    name = dataList[0]
    hours = int(dataList[1])
    payRate = float(dataList[2])
    totalPay = hours * payRate
    print("%-15s%6d%15.2f" % (name, hours, totalPay))


```

```
1 Lambert 34 10.50
2 Osborne 22 6.25
3 Giacometti 5 100.70
```

```
"""
File: stats.py
Probject 5.1

Defines functions to compute the mean, median, and mode
of a list of numbers.
"""

def mean(lyst):
    """Returns the mean of a list of numbers."""
    sum = 0
    for number in lyst:
        sum += number
    if len(lyst) == 0:
        return 0
    else:
        return sum / len(lyst)

def mode(lyst):
    """Returns the mode of a list of numbers."""
    # Obtain the set of unique numbers and their
    # frequencies, saving these associations in
    # a dictionary
    theDictionary = {}
    for number in lyst:
        freq = theDictionary.get(number, None)
        if freq == None:
            # number entered for the first time
            theDictionary[number] = 1
        else:
            # number already seen, increment its freq
            theDictionary[number] = freq + 1

    # Find the mode by obtaining the maximum freq
    # in the dictionary and determining its key
    if len(theDictionary) == 0:
        return 0
    else:
        theMaximum = max(theDictionary.values())
        for key in theDictionary:
            if theDictionary[key] == theMaximum:
                return key

def median(lyst):
    """Returns the median of a list of numbers."""
    # Create a copy of lyst before sorting
```

```
    numbers = []
    for number in lyst:
        numbers.append(number)
    # Sort the list and return the number at its midpoint
    numbers.sort()
    if len(numbers) == 0:
        return 0
    else:
        midpoint = len(numbers) // 2
        if len(numbers) % 2 == 1:
            return numbers[midpoint]
        else:
            return (numbers[midpoint] + numbers[midpoint - 1]) / 2

def main():
    """Tests the functions."""
    lyst = [3, 1, 7, 1, 4, 10]
    print("List:", lyst)
    print("Mode:", mode(lyst))
    print("Median:", median(lyst))
    print("Mean:", mean(lyst))

# The entry point for program execution
if __name__ == "__main__":
    main()
```

```
"""
File: navigate.py
Project 5.2

Allows the user to navigate to any line in a text file.
"""

# Take the input file name
inName = input("Enter the input file name: ")

# Open the input file and read the text
inputFile = open(inName, 'r')
lines = []
for line in inputFile:
    lines.append(line)

# Loop for line numbers from the user until she enters 0
# and prints the line's number followed by the line
while True:
    print("The file has", len(lines), "lines.")
    if len(lines) == 0:
        break
    lineNumber = int(input("Enter a line number [0 to quit
]: "))
    if lineNumber == 0:
        break
    elif lineNumber >= len(lines):
        print("ERROR: line number must be less than", len(
lines))
    else:
        print(lineNumber, ": ", lines[lineNumber])


```

```
"""
File: generator.py
Project 5.3
Generates and displays sentences using a simple grammar
and vocabulary.  Words are chosen at random.  Vocabulary
is
input from the files nouns.txt, verbs.txt, articles.txt,
and prepositions.txt.
"""

import random

def getWords(fileName):
    """Builds and returns a tuple of words
    from an input file."""
    inputFile = open(fileName, 'r')
    words = []
    for line in inputFile:
        words.extend(line.split())
    return tuple(words)

articles = getWords("articles.txt")

nouns = getWords("nouns.txt")

verbs = getWords("verbs.txt")

prepositions = getWords("prepositions.txt")

def sentence():
    """Builds and returns a sentence."""
    return nounPhrase() + " " + verbPhrase()

def nounPhrase():
    """Builds and returns a noun phrase."""
    return random.choice(articles) + " " + random.choice(
nouns)

def verbPhrase():
    """Builds and returns a verb phrase."""
    return random.choice(verbs) + " " + nounPhrase() + " "
 + \
           prepositionalPhrase()

def prepositionalPhrase():
    """Builds and returns a prepositional phrase."""
```

```
44      return random.choice(prepositions) + " " + nounPhrase(
    )
45
46  def main():
47      """Allows the user to input the number of sentences
48      to generate."""
49      number = int(input("Enter the number of sentences: "))
50      for count in range(number):
51          print(sentence())
52
53  # The entry point for program execution
54  if __name__ == "__main__":
55      main()
56
57
```

1 A THE

1 BOY GIRL BAT BALL

1 WITH BY

1 HIT SAW LIKED

```
"""
File: generator.py
Project 5.4
Generates and displays sentences using a simple grammar
and vocabulary.  Words are chosen at random.  Adds optional
adjectives to noun phrases, optional prepositional phrases,
and optional independent clauses connected by conjunctions.
"""

import random

articles = ("A", "THE")

nouns = ("BOY", "GIRL", "BAT", "BALL")

verbs = ("HIT", "SAW", "LIKED")

prepositions = ("WITH", "BY")

adjectives = ("RED", "LITTLE")

conjunctions = ("AND", "BUT")

def sentence():
    """Builds and returns a sentence.
    Now allows for optional independent clauses
    connected by conjunctions."""
    first = independentClause()
    if random.randint(1, 5) == 1:
        return first + " " + random.choice(conjunctions) + \
               " " + independentClause()
    else:
        return first

def independentClause():
    """Builds and returns an independent clause."""
    return nounPhrase() + " " + verbPhrase()

def nounPhrase():
    """Builds and returns a noun phrase.
    Now includes an optional adjective."""
    if random.randint(1, 2) == 1:
```

```
43          adj = random.choice(adjectives) + " "
44      else:
45          adj = ""
46      return random.choice(articles) + " " + adj + random.
    choice(nouns)
47
48  def verbPhrase():
49      """Builds and returns a verb phrase.
50      The prepositional phrase is now optional."""
51      if random.randint(1, 3) == 1:
52          prepPhrase = " " + prepositionalPhrase()
53      else:
54          prepPhrase = ""
55      return random.choice(verbs) + " " + nounPhrase() +
    prepPhrase
56
57  def prepositionalPhrase():
58      """Builds and returns a prepositional phrase."""
59      return random.choice(prepositions) + " " + nounPhrase(
    )
60
61  def main():
62      """Allows the user to input the number of sentences
63      to generate."""
64      number = int(input("Enter the number of sentences: "))
65      for count in range(number):
66          print(sentence())
67
68  # The entry point for program execution
69  if __name__ == "__main__":
70      main()
71
72
```

```
"""
File: convert.py
Project 5.5
Defines a function repToDecimal that converts
a number in any base to decimal.
"""

# Table of digits in bases 2-16 with integer values
repTable = {'0': 0, '1': 1, '2': 2, '3': 3, '4': 4,
            '5': 5, '6':6, '7': 7, '8' : 8, '9' : 9,
            'A' : 10, 'B' : 11, 'C' : 12, 'D' : 13,
            'E' : 14, 'F' : 15}

def repToDecimal(rep, base):
    """Converts the string rep of a number in base
    to decimal and returns the decimal as an int."""
    decimal = 0
    exp = len(rep) - 1
    for digit in rep:
        decimal += repTable[digit] * base ** exp
        exp -= 1
    return decimal

def main():
    """Tests the function."""
    print(repToDecimal('10', 10))
    print(repToDecimal('10', 8))
    print(repToDecimal('10', 2))
    print(repToDecimal('10', 16))

# The entry point for program execution
if __name__ == "__main__":
    main()
```

```
"""
File: convert.py
Project 5.6
Defines a function decimalToRep that converts
a number in decimal to a number in a given base.
"""

# Table of integers in bases 2-16 with digits
repTable = {0 : '0', 1 : '1', 2 : '2', 3 : '3', 4 :'4',
            5 : '5', 6 : '6', 7 : '7', 8 : '8', 9 : '9',
            10 : 'A', 11 : 'B', 12 : 'C', 13 : 'D',
            14 : 'E', 15 : 'F'}

def decimalToRep(decimal, base):
    """Converts the integer value in decimal to
    a rep in the given base returns the rep as a string
."""
    if decimal == 0:
        return '0'
    else:
        rep = ""
        while decimal > 0:
            remainder = decimal % base
            decimal = decimal // base
            rep = repTable[remainder] + rep
        return rep

def main():
    """Tests the function."""
    print(decimalToRep(10, 10))
    print(decimalToRep(10, 8))
    print(decimalToRep(10, 2))
    print(decimalToRep(10, 16))

# The entry point for program execution
if __name__ == "__main__":
    main()
```

```
"""
File: unique.py
Project 5.7

Prints the unique words in a text file in alphabetical
order.
"""

# Take the input file name
inName = input("Enter the input file name: ")

# Open the input file and initialize list of unique words
inputFile = open(inName, 'r')
uniqueWords = []

# Add the unique words in the file to the list
for line in inputFile:
    words = line.split()
    for word in words:
        if not word in uniqueWords:
            uniqueWords.append(word)
uniqueWords.sort()

# Prints the unique words
for word in uniqueWords:
    print(word)
```

```
"""
File: concordance.py
Project 5.8

Prints the unique words in a text file and their
frequences.
"""

# Take the input file name
inName = input("Enter the input file name: ")

# Open the input file and initialize list of unique words
inputFile = open(inName, 'r')
uniqueWords = {}

# Add the unique words in the file to the list
for line in inputFile:
    words = line.split()
    for word in words:
        freq = uniqueWords.get(word, None)
        if freq == None:
            uniqueWords[word] = 1
        else:
            uniqueWords[word] = freq + 1

# Prints the unique words and their frequencies,
# in alphabetical order
words = list(uniqueWords.keys())
words.sort()
for word in words:
    print(word, uniqueWords[word])

```

```
"""
File: doctor.py
Project 5.9
Conducts an interactive session of nondirective psychotherapy.
Fixes problem of responding to sentences that address the doctor
using second-person pronouns.
"""

import random

hedges = ("Please tell me more.",
          "Many of my patients tell me the same thing.",
          "Please coninue.")

qualifiers = ("Why do you say that ",
              "You seem to think that ",
              "Can you explain why ")

# The fix is in this dictionary, third line of data
replacements = {"I":"you", "me":"you", "my":"your",
                "we":"you", "us":"you", "mine":"yours",
                "you":"I", "your":"my", "yours":"mine"}

def reply(sentence):
    """Implements two different reply strategies."""
    probability = random.randint(1, 4)
    if probability == 1:
        return random.choice(hedges)
    else:
        return random.choice(qualifiers) + changePerson(sentence)

def changePerson(sentence):
    """Replaces first person pronouns with second person
    pronouns."""
    words = sentence.split()
    replyWords = []
    for word in words:
        replyWords.append(replacements.get(word, word))
    return " ".join(replyWords)

def main():
    """Handles the interaction between patient and doctor."""
```

```
43      print("Good morning, I hope you are well today.")
44      print("What can I do for you?")
45      while True:
46          sentence = input("\n>> ")
47          if sentence.upper() == "QUIT":
48              print("Have a nice day!")
49              break
50          print(reply(sentence))
51
52  # The entry point for program execution
53  if __name__ == "__main__":
54      main()
55
56
```

```
"""
File: doctor.py
Project 5.10
Conducts an interactive session of nondirective psychotherapy.
Adds a history list of earlier patient sentences, which can
be chosen for replies to shift the conversation to an earlier topic.
"""

import random

history = []

hedges = ("Please tell me more.",
          "Many of my patients tell me the same thing.",
          "Please coninue.")

qualifiers = ("Why do you say that ",
              "You seem to think that ",
              "Can you explain why ")

replacements = {"I":"you", "me":"you", "my":"your",
                "we":"you", "us":"you", "mine":"yours",
                "you":"I", "your":"my", "yours":"mine"}

def reply(sentence):
    """Implements three different reply strategies."""
    probability = random.randint(1, 5)
    if probability in (1, 2):
        # Just hedge
        answer = random.choice(hedges)
    elif probability == 3 and len(history) > 3:
        # Go back to an earlier topic
        answer = "Earlier you said that " + \
                 changePerson(random.choice(history))
    else:
        # Transform the current input
        answer = random.choice(qualifiers) + changePerson(sentence)
    # Always add the current sentence to the history list
    history.append(sentence)
    return answer

def changePerson(sentence):
```

```
    """Replaces first person pronouns with second person
    pronouns."""
    words = sentence.split()
    replyWords = []
    for word in words:
        replyWords.append(replacements.get(word, word))
    return " ".join(replyWords)

def main():
    """Handles the interaction between patient and doctor
    ."""
    print("Good morning, I hope you are well today.")
    print("What can I do for you?")
    while True:
        sentence = input("\n>> ")
        if sentence.upper() == "QUIT":
            print("Have a nice day!")
            break
        print(reply(sentence))

# The entry point for program execution
if __name__ == "__main__":
    main()
```

```
"""
File: newton.py
Project 6.1

Compute the square root of a number (uses function with
loop).

1. The input is a number, or enter/return to halt the
   input process.

2. The outputs are the program's estimate of the square
root
   using Newton's method of successive approximations, and
   Python's own estimate using math.sqrt.
"""

import math


# Initialize the tolerance
TOLERANCE = 0.000001

def newton(x):
    """Returns the square root of x."""
    # Perform the successive approximations
    estimate = 1.0
    while True:
        estimate = (estimate + x / estimate) / 2
        difference = abs(x - estimate ** 2)
        if difference <= TOLERANCE:
            break
    return estimate

def main():
    """Allows the user to obtain square roots."""
    while True:
        # Receive the input number from the user
        x = input("Enter a positive number or enter/return
 to quit: ")
        if x == "":
            break
        x = float(x)
        # Output the result
        print("The program's estimate is", newton(x))
        print("Python's estimate is     ", math.sqrt(x))

```

```
44 if __name__ == "__main__":
45     main()
46
```

```
"""
File: newton.py
Project 6.2

Compute the square root of a number (uses recursive function).

1. The input is a number, or enter/return to halt the
   input process.

2. The outputs are the program's estimate of the square root
   using Newton's method of successive approximations, and
   Python's own estimate using math.sqrt.
"""

import math


# Initialize the tolerance
TOLERANCE = 0.000001

def newton(x, estimate):
    """Returns the square root of x."""
    # Compute the difference and test for the base case
    difference = abs(x - estimate ** 2)
    if difference <= TOLERANCE:
        return estimate
    else:
        # Recurse after improving the estimate
        return newton(x, (estimate + x / estimate) / 2)

def main():
    """Allows the user to obtain square roots."""
    while True:
        # Receive the input number from the user
        x = input("Enter a positive number or enter/return to quit: ")
        if x == "":
            break
        x = float(x)
        # Output the result
        print("The program's estimate is", newton(x, 1))
        print("Python's estimate is     ", math.sqrt(x))

if __name__ == "__main__":
```

```
44      main()
45
```

```
"""
File: newton.py
Project 6.3

Compute the square root of a number (uses recursive function with
default second parameter).

1. The input is a number, or enter/return to halt the
   input process.

2. The outputs are the program's estimate of the square root
   using Newton's method of successive approximations, and
   Python's own estimate using math.sqrt.
"""

import math

# Initialize the tolerance
TOLERANCE = 0.000001

def newton(x, estimate = 1):
    """Returns the square root of x."""
    # Compute the difference and test for the base case
    difference = abs(x - estimate ** 2)
    if difference <= TOLERANCE:
        return estimate
    else:
        # Recurse after improving the estimate
        return newton(x, (estimate + x / estimate) / 2)

def main():
    """Allows the user to obtain square roots."""
    while True:
        # Receive the input number from the user
        x = input("Enter a positive number or enter/return to quit: ")
        if x == "":
            break
        x = float(x)
        # Output the result
        print("The program's estimate is", newton(x))
        print("Python's estimate is     ", math.sqrt(x))

if __name__ == "__main__":
```

```
44      main()
45
```

```
"""
File: newton.py
Project 6.4

Compute the square root of a number (uses recursive function with
default second argument).

1. The input is a number, or enter/return to halt the
   input process.

2. The outputs are the program's estimate of the square root
   using Newton's method of successive approximations, and
   Python's own estimate using math.sqrt.
"""

import math

# Initialize the tolerance
TOLERANCE = 0.000001

def newton(x, estimate = 1):
    """Returns the square root of x."""
    # Compute the difference and test for the base case
    if limitReached(x, estimate):
        return estimate
    else:
        # Recurse after improving the estimate
        return newton(x, improveEstimate(x, estimate))

def limitReached(x, estimate):
    """Returns True if the estimate is within
    the tolerance or False otherwise."""
    difference = abs(x - estimate ** 2)
    return difference <= TOLERANCE

def improveEstimate(x, estimate):
    """Returns an improved estimate."""
    return (estimate + x / estimate) / 2

def main():
    """Allows the user to obtain square roots."""
    while True:
        # Receive the input number from the user
        x = input("Enter a positive number or enter/return
```

```
44  to quit: ")
45          if x == "":
46               break
47          x = float(x)
48          # Output the result
49          print("The program's estimate is", newton(x))
50          print("Python's estimate is     ", math.sqrt(x))
51
52 if __name__ == "__main__":
53     main()
54
```

```
"""
File: testsort.py
Project 6.5

Defines a predicate to test lists for being sorted.

"""

def isSorted(lyst):
    """Returns True if lyst is sorted in ascending
    order or False otherwise."""
    if len(lyst) == 0 or len(lyst) == 1:
        return True
    else:
        for index in range(len(lyst) - 1):
            if lyst[index] > lyst[index + 1]:
                return False
    return True

def main():
    lyst = []
    print(isSorted(lyst))
    lyst = [1]
    print(isSorted(lyst))
    lyst = list(range(10))
    print(isSorted(lyst))
    lyst[9] = 3
    print(isSorted(lyst))

if __name__ == "__main__":
    main()

```

```
"""
File: filesys.py
Project 6.6

Provides a menu-driven tool for navigating a file system
and gathering information on files.

Adds a command to view a file's contents.
"""

import os, os.path

QUIT = '8'

COMMANDS = ('1', '2', '3', '4', '5', '6', '7', '8')

MENU = """1   List the current directory
2   Move up
3   Move down
4   Number of files in the directory
5   Size of the directory in bytes
6   Search for a file name
7   View the contents of a file
8   Quit the program"""

def main():
    while True:
        print(os.getcwd())
        print(MENU)
        command = acceptCommand()
        runCommand(command)
        if command == QUIT:
            print("Have a nice day!")
            break

def acceptCommand():
    """Inputs and returns a legitimate command number."""
    while True:
        command = input("Enter a number: ")
        if not command in COMMANDS:
            print("Error: command not recognized")
        else:
            return command

def runCommand(command):
    """Selects and runs a command."""
```

```
47      if command == '1':
48          listCurrentDir(os.getcwd())
49      elif command == '2':
50          moveUp()
51      elif command == '3':
52          moveDown(os.getcwd())
53      elif command == '4':
54          print("The total number of files is", \
55                countFiles(os.getcwd()))
56      elif command == '5':
57          print("The total number of bytes is", \
58                countBytes(os.getcwd()))
59      elif command == '6':
60          target = raw_input("Enter the search string: ")
61          fileList = findFiles(target, os.getcwd())
62          if not fileList:
63              print("String not found")
64          else:
65              for f in fileList:
66                  print(f)
67      elif command == '7':
68          viewFile(os.getcwd())
69
70  def viewFile(dirName):
71      lyst = list(filter(os.path.isfile, os.listdir(dirName)
    ))
72      if len(lyst) == 0:
73          print("There are no files in this directory")
74      else:
75          while True:
76              print("Files in " + dirName + ":")
77              for element in lyst: print(element)
78              fileName = input("Enter a file name from these
     names: ")
79              if not fileName in lyst:
80                  print("Sorry, there is an error in your
    file name.")
81              else:
82                  f = open(fileName, 'r')
83                  print(f.read())
84                  break
85
86  def listCurrentDir(dirName):
87      """Prints a list of the cwd's contents."""
88      lyst = os.listdir(dirName)
89      for element in lyst: print(element)
```

```
 90
 91 def moveUp():
 92     """Moves up to the parent directory."""
 93     os.chdir("..")
 94
 95 def moveDown(currentDir):
 96     """Moves down to the named subdirectory if it exists
    ."""
 97     newDir = input("Enter the directory name: ")
 98     if os.path.exists(currentDir + os.sep + newDir) and \
 99        os.path.isdir(newDir):
100         os.chdir(newDir)
101     else:
102         print("ERROR: no such name")
103
104 def countFiles(path):
105     """Returns the number of files in the cwd and
106     all its subdirectories."""
107     count = 0
108     lyst = os.listdir(path)
109     for element in lyst:
110         if os.path.isfile(element):
111             count += 1
112         else:
113             os.chdir(element)
114             count += countFiles(os.getcwd())
115             os.chdir("..")
116     return count
117
118 def countBytes(path):
119     """Returns the number of bytes in the cwd and
120     all its subdirectories."""
121     count = 0
122     lyst = os.listdir(path)
123     for element in lyst:
124         if os.path.isfile(element):
125             count += os.path.getsize(element)
126         else:
127             os.chdir(element)
128             count += countBytes(os.getcwd())
129             os.chdir("..")
130     return count
131
132 def findFiles(target, path):
133     """Returns a list of the file names that contain
134     the target string in the cwd and all its
```

```
134 subdirectories."""
135     files = []
136     lyst = os.listdir(path)
137     for element in lyst:
138         if os.path.isfile(element):
139             if target in element:
140                 files.append(path + os.sep + element)
141         else:
142             os.chdir(element)
143             files.extend(findFiles(target, os.getcwd()))
144             os.chdir("..")
145     return files
146 
147 if __name__ == "__main__":
148     main()
149 
```

```
"""
File: viewfiles.py
Project 6.7

Allows the user to visit all of the files in the current
path and view them.

"""

import os, os.path

def main():
    displayFiles(os.getcwd())

def displayFiles(path):
    """Visits all of the files and directories in
    path and displays the files' contents."""
    if os.path.isfile(path):
        print("File name: " + path)
        f = open(path, 'r')
        print(f.read())
    else:
        print("Directory name: " + path)
        lyst = os.listdir(path)
        for element in lyst: displayFiles(element)

if __name__ == "__main__":
    main()

```

```
"""
File: testprintlist.py
Project 6.8

Defines the printAll function with a trace.

Before each recursive call, the function creates
a slice of its nonempty list argument.  The hidden cost
is that each slice produces a copy of the list, less
its first item.  This process requires time and memory.

"""


def printAll(seq):
    if seq:
        print(seq, "->", seq[0])
        printAll(seq[1:])

printAll(list(range(10)))


```

```
"""
File: average.py
Project 6.9
Prints the average of the numbers in a text file
"""

from functools import reduce

# Accept the input file name and open the file
fileName = input("Enter the input file name: ")
inputFile = open(fileName, 'r')

# Read the numbers as strings into a list
lyst = []
for line in inputFile:
    lyst.extend(line.split())

# Convert all the strings in the list to numbers
lyst = list(map(float, lyst))

# Compute the sum of the numbers
summation = reduce(lambda x, y: x + y, lyst)

# Print the average
if len(lyst) == 0:
    average = 0
else:
    average= summation / len(lyst)
print("The average is", average)

```

1 45 66 88
2 100 22 98

```
"""
File: testmyrange.py
Project 6.10

Defines a function that behaves like Python's range
function.

"""

def myRange(start, stop = None, step = None):
    lyst = []
    if stop == None and step == None:
        stop = start
        start = 0
        step = 1
    if start < stop:
        if step == None:
            step = 1
        elif step <= 0:
            return lyst
        while start < stop:
            lyst.append(start)
            start += step
    else:
        if step == None or step > -1:
            return lyst
        while start > stop:
            lyst.append(start)
            start += step
    return lyst

def main():
    print(myRange(10))
    print(myRange(1, 10))
    print(myRange(1, 10, 2))
    print(myRange(10, 1, -1))

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