HW-1
Make the “High–Low” and “Nerves of Steel” party games

At the beginning of this book, I said that programming is as easy (or difficult) as organizing a party. With that in mind, here are a couple ideas for Python programs that you can use for entertainment at your next party.
You can use the random number generator and the sleep function to make a high–low party game. The game works like this:

1. The program displays a number between 1 and 10, inclusively.

   2. The program then sleeps for 20 seconds. While the program is asleep, the players are invited to decide whether the next number will be higher or lower than the number just printed. Players who choose “high” stand on the right. Players who choose “low” stand on the left.

   3. The program then displays a second number between 1 and 10, and anyone who was wrong is eliminated from the game. The program is then re-run with the players that are left until you have a winner.
   This game can get very tactical, with players taking a chance on an unlikely number just so that they will be one of the people to go forward to the next round.

Another use for the random number generator and the sleep function is to make a “Nerves of Steel” game. This game works like this:

1. The program displays “Players stand.”

2. The program sleeps for a random time between 5 and 20 seconds. While the program is sleeping, players can sit down. Keep track of the last person to sit down.

3. The program displays “Last to sit down wins.” Players still standing are eliminated, and the winner is the last person to sit down.

HW-2

1. Sales Prediction
A company has determined that its annual profit is typically 23 percent of total sales. Write a program that asks the user to enter the projected amount of total sales, and then displays the profit that will be made from that amount.
Hint: Use the value 0.23 to represent 23 percent.

2. Land Calculation
One acre of land is equivalent to 43,560 square feet. Write a program that asks the user to enter the total square feet in a tract of land and calculates the number of acres in the tract.
Hint: Divide the amount entered by 43,560 to get the number of acres.

3. Total Purchase
A customer in a store is purchasing five items. Write a program that asks for the price of each item, and then displays the subtotal of the sale, the amount of sales tax, and the total. Assume the sales tax is 7 percent.

4. Distance Traveled
Assuming there are no accidents or delays, the distance that a car travels down the interstate can be calculated with the following formula:
Distance = Speed x Time

A car is traveling at 70 miles per hour. Write a program that displays the following:
- The distance the car will travel in 6 hours
- The distance the car will travel in 10 hours
- The distance the car will travel in 15 hours

5. Sales Tax
Write a program that will ask the user to enter the amount of a purchase. The program should then compute the state and county sales tax. Assume the state sales tax is 5 percent and the county sales tax is 2.5 percent. The program should display the amount of the purchase, the state sales tax, the county sales tax, the total sales tax, and the total of the sale (which is the sum of the amount of purchase plus the total sales tax).
Hint: Use the value 0.025 to represent 2.5 percent, and 0.05 to represent 5 percent.

6. Miles-per-Gallon
A car's miles-per-gallon (MPG) can be calculated with the following formula:
MPG = Miles driven / Gallons of gas used
Write a program that asks the user for the number of miles driven and the gallons of gas used. It should calculate the car's MPG and display the result.

7. Tip, Tax, and Total
Write a program that calculates the total amount of a meal purchased at a restaurant. The program should ask the user to enter the charge for the food, and then calculate the amount of a 18 percent tip and 7 percent sales tax. Display each of these amounts and the total.

8. Celsius to Fahrenheit Temperature Converter
Write a program that converts Celsius temperatures to Fahrenheit temperatures. The formula is as follows:

$$F = \frac{9}{5}C + 32$$

The program should ask the user to enter a temperature in Celsius, and then display the temperature converted to Fahrenheit.

HW-3

1. Day of the Week

Write a program that asks the user for a number in the range of 1 through 7. The pro- gram should display the corresponding day of the week, where 1 = Monday, 2 = Tuesday, 3 = Wednesday, 4 = Thursday, 5 = Friday, 6 = Saturday, and 7 = Sunday. The program should display an error message if the user enters a number that is outside the range of 1 through 7.

2. Areas of Rectangles

The area of a rectangle is the rectangle's length times its width. Write a program that asks for the length and width of two rectangles. The program should tell the user which rectangle has the greater area, or if the areas are the same.

3. Age Classifier

Write a program that asks the user to enter a person's age. The program should display a message indicating whether the person is an infant, a child, a teenager, or an adult. Following are the guidelines:

- If the person is 1 year old or less, he or she is an infant.
- If the person is older than 1 year, but younger than 13 years, he or she is a child.
- If the person is at least 13 years old, but less than 20 years old, he or she is a teenager.
- If the person is at least 20 years old, he or she is an adult.

4. Roman numerals

Write a program that prompts the user to enter a number within the range of 1 through 10. The program should display the Roman numeral version of that number. If the number is outside the range of 1 through 10, the program should display an error message. The following table shows the Roman numerals for the numbers 1 through 10:

Number	Roman Numeral
	I
	II
	III
	IV
	V
	VI
	VII
	VIII
	IX
	X

5. Mass and Weight

Scientists measure an object's mass in kilograms and its weight in newtons. If you know the amount of mass of an object in kilograms, you can calculate its weight in newtons with the following formula:

$$weight = mass \times 9.8$$

Write a program that asks the user to enter an object's mass, and then calculates its weight. If the object weighs more than 500 newtons, display a message indicating that it is too heavy. If the object weighs less than 100 newtons, display a message indicating that it is too light.

6. Magic Dates

The date June 10, 1960, is special because when it is written in the following format, the month times the day equals the year:

6/10/60

Design a program that asks the user to enter a month (in numeric form), a day, and a two- digit year. The program should then determine whether the month times the day equals the year. If so, it should display a message saying the date is magic. Otherwise, it should display a message saying the date is not magic.

7. Color Mixer

The colors red, blue, and yellow are known as the primary colors because they cannot be made by mixing other colors. When you mix two primary colors, you get a secondary color, as shown here:

When you mix red and blue, you get purple.

When you mix red and yellow, you get orange.

When you mix blue and yellow, you get green.

Design a program that prompts the user to enter the names of two primary colors to mix. If the user enters anything other than "red," "blue," or "yellow," the program should display an error message. Otherwise, the program should display the name of the secondary color that results.

8. Hot Dog Cookout Calculator

Assume that hot dogs come in packages of 10, and hot dog buns come in packages of 8. Write a program that calculates the number of packages of hot dogs and the number of packages of hot dog buns needed for a cookout, with the minimum number of leftovers. The program should ask the user for the number of people attending the cookout and the number of hot dogs each person will be given. The program should display the following details:

- The minimum number of packages of hot dogs required
- The minimum number of packages of hot dog buns required
- The number of hot dogs that will be left over
- The number of hot dog buns that will be left over

9. Roulette Wheel Colors

On a roulette wheel, the pockets are numbered from 0 to 36. The colors of the pockets are as follows:

- Pocket 0 is green.
- For pockets 1 through 10, the odd-numbered pockets are red and the even-numbered pockets are black.
- For pockets 11 through 18, the odd-numbered pockets are black and the even-numbered pockets are red.
- For pockets 19 through 28, the odd-numbered pockets are red and the even-numbered pockets are black.
- For pockets 29 through 36, the odd-numbered pockets are black and the even-numbered pockets are red.

Write a program that asks the user to enter a pocket number and displays whether the pocket is green, red, or black. The program should display an error message if the user enters a number that is outside the range of 0 through 36.

HW-4

1. Bug Collector

A bug collector collects bugs every day for five days. Write a program that keeps a running total of the number of bugs collected during the five days. The loop should ask for the number of bugs collected for each day, and when the loop is finished, the program should display the total number of bugs collected.

2. Calories Burned

Running on a particular treadmill you burn 4.2 calories per minute. Write a program that uses a loop to display the number of calories burned after 10, 15, 20, 25, and 30 minutes.

3. Budget Analysis

Write a program that asks the user to enter the amount that he or she has budgeted for a month. A loop should then prompt the user to enter each of his or her expenses for the month and keep a running total. When the loop finishes, the program should display the amount that the user is over or under budget.

4. Calculating the Factorial of a number

In mathematics, the notation $n!$ represents the factorial of the nonnegative integer n. The factorial of n is the product of all the nonnegative integers from 1 to n. For example,

$7! = 1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7 = 5{,}040$

And

$4! = 1 \times 2 \times 3 \times 4 = 24$

Write a program that lets the user enter a nonnegative integer and then uses a loop to calculate the factorial of that number. Display the factorial.

5. Population

Write a program that predicts the approximate size of a population of organisms. The application should use text boxes to allow the user to enter the starting number of organ- isms, the average daily population increase (as a percentage), and the number of days the organisms will be left to multiply. For example, assume the user enters the following values:

Starting number of organisms: 2 Average daily increase: 30% Number of days to multiply: 10

The program should display the following table of data:

Day	Approximate Population
1	2
2	2.6
3	3.38
4	4.394
5	5.7122
6	7.42586
7	9.653619
8	12.5497

9	16.31462
10	21.209

6. Write a program that uses nested loops to draw this pattern:

```
*******
******
*****
****
***
**
*
```

7. Write a program that uses nested loops to draw this pattern:

```
##
# #
#  #
#    #
#      #
#        #
```

HW-5

1. odd/even Counter

In this chapter, you saw an example of how to write an algorithm that determines whether a number is even or odd. Write a program that generates 100 random numbers and keeps a count of how many of those random numbers are even and how many of them are odd.

2. prime numbers

A prime number is a number that is only evenly divisible by itself and 1. For example, the number 5 is prime because it can only be evenly divided by 1 and 5. The number 6, however, is not prime because it can be divided evenly by 1, 2, 3, and 6.

Write a Boolean function named `is_prime` which takes an integer as an argument and returns true if the argument is a prime number, or false otherwise. Use the function in a program that prompts the user to enter a number and then displays a message indicating whether the number is prime.

tIp: *Recall that the `%` operator divides one number by another and returns the remainder of the division. In an expression such as `num1 % num2`, the `%` operator will return 0 if `num1` is evenly divisible by `num2`.*

3. prime number List

This exercise assumes that you have already written the `is_prime` function in Programming Exercise 17. Write another program that displays all of the prime numbers from 1 to 100. The program should have a loop that calls the `is_prime` function.

4. Future Value

Suppose you have a certain amount of money in a savings account that earns compound monthly interest, and you want to calculate the amount that you will have after a specific number of months. The formula is as follows:

$$F = P\,x\ (1 + i)^t$$

The terms in the formula are:

- F is the future value of the account after the specified time period.
- P is the present value of the account.
- i is the monthly interest rate.
- t is the number of months.

Write a program that prompts the user to enter the account's present value, monthly interest rate, and the number of months that the money will be left in the account. The program should pass these values to a function that returns the future value of the account, after the specified number of months. The program should display the account's future value.

5. Random number Guessing Game

Write a program that generates a random number in the range of 1 through 100 and asks the user to guess what the number is. If the user's guess is higher than the random number, the program should display "Too high, try again." If the user's guess is lower than the random number, the program should display "Too low, try again." If the user guesses the number, the application should congratulate the user and then generate a new random number so the game can start over.

Optional Enhancement: Enhance the game so it keeps count of the number of guesses that the user makes. When the user correctly guesses the random number, the program should display the number of guesses.

HW-6

1. Line numbers

Write a program that asks the user for the name of a file. The program should display the contents of the file with each line preceded with a line number followed by a colon. The line numbering should start at 1.

2. Item Counter

Assume that a file containing a series of names (as strings) is named `names.txt` and exists on the computer's disk. Write a program that displays the number of names that are stored in the file. *(Hint: Open the file and read every string stored in it. Use a variable to keep a count of the number of items that are read from the file.)*

3. sum of numbers

Assume that a file containing a series of integers is named `numbers.txt` and exists on the computer's disk. Write a program that reads all of the numbers stored in the file and calculates their total.

4. Average of numbers

Assume that a file containing a series of integers is named `numbers.txt` and exists on the computer's disk. Write a program that calculates the average of all the numbers stored in the file.

5. random number File Writer

Write a program that writes a series of random numbers to a file. Each random number should be in the range of 1 through 500. The application should let the user specify how many random numbers the file will hold.

6. exception Handing

Modify the program that you wrote for Exercise 5 so it handles the following exceptions:

- It should handle any `IOError` exceptions that are raised when the file is opened, and data is read from it.
- It should handle any `ValueError` exceptions that are raised when the items that are read from the file are converted to a number.

7. golf scores

The Springfork Amateur Golf Club has a tournament every weekend. The club president has asked you to write two programs:

1. A program that will read each player's name and golf score as keyboard input, and then save these as records in a file named `golf.txt`. (Each record will have a field for the player's name and a field for the player's score.)
2. A program that reads the records from the `golf.txt` file and displays them.

HW-7

1. Lottery number Generator

Design a program that generates a seven-digit lottery number. The program should generate seven random numbers, each in the range of 0 through 9, and assign each number to a list element. Then write another loop that displays the contents of the list.

2. Rainfall statistics

Design a program that lets the user enter the total rainfall for each of 12 months into a list. The program should calculate and display the total rainfall for the year, the average monthly rainfall, and the months with the highest and lowest amounts.

3. number Analysis program

Design a program that asks the user to enter a series of 20 numbers. The program should store the numbers in a list and then display the following data:

The lowest number in the list

The highest number in the list

The total of the numbers in the list

The average of the numbers in the list

4. Larger Than *n*

In a program, write a function that accepts two arguments: a list, and a number *n*. Assume that the list contains numbers. The function should display all the numbers in the list that are greater than the number *n*.

5. Lo shu Magic square

The Lo Shu Magic Square is a grid with 3 rows and 3 columns, shown below. The Lo Shu Magic Square has the following properties:

The grid contains the numbers 1 through 9 exactly.

The sum of each row, each column, and each diagonal all add up to the same number.

In a program you can simulate a magic square using a two-dimensional list. Write a function that accepts a two-dimensional list as an argument and determines whether the list is a Lo Shu Magic Square. Test the function in a program.

Diagram next page:

4	9	2
3	5	7
8	1	6

Horizontal / Vertical / Diagonal = 15

1. Course information

Write a program that creates a dictionary containing course numbers and the room numbers of the rooms where the courses meet. The dictionary should have the following key-value pairs:

Course Number (key)	Room Number (value)
CS101	3004
CS102	4501
CS103	6755
NT110	1244
CM241	1411

The program should also create a dictionary containing course numbers and the names of the instructors that teach each course. The dictionary should have the following key-value pairs:

Course Number (key)	Instructor (value)
CS101	Haynes
CS102	Alvarado
CS103	Rich
NT110	Burke
CM241	Lee

The program should also create a dictionary containing course numbers and the meeting times of each course. The dictionary should have the following key-value pairs:

Course Number (key)	Meeting Time (value)
CS101	8:00 a.m.
CS102	9:00 a.m.
CS103	10:00 a.m.
NT110	11:00 a.m.
CM241	1:00 p.m.

The program should let the user enter a course number, and then it should display the course's room number, instructor, and meeting time.

2. Capital Quiz

Write a program that creates a dictionary containing the U.S. states as keys and their capitals as values. (Use the Internet to get a list of the states and their capitals.) The program should then randomly quiz the user by displaying the name of a state and asking the user to enter that state's capital. The program should keep a count of the number of correct and incorrect responses. (As an alternative to the U.S. states, the program can use the names of countries and their capitals.)

3. Unique Words

Write a program that opens a specified text file and then displays a list of all the unique words found in the file.

Hint: Store each word as an element of a set.

4. Word Frequency

Write a program that reads the contents of a text file. The program should create a dictionary in which the keys are the individual words found in the file and the values are the number of times each word appears. For example, if the word "the" appears 128 times, the dictionary would contain an element with `'the'` as the key and 128 as the value. The program should either display the frequency of each word or create a second file containing a list of each word and its frequency.

5. name and email Addresses

Write a program that keeps names and email addresses in a dictionary as key-value pairs. The program should display a menu that lets the user look up a person's email address, add a new name and email address, change an existing email address, and delete an existing name and email address. The program should pickle the dictionary and save it to a file when the user exits the program. Each time the program starts, it should retrieve the dictionary from the file and unpickle it.

HW-9

1. Recursive printing

Design a recursive function that accepts an integer argument, n, and prints the numbers 1 up through n.

2. Recursive Multiplication

Design a recursive function that accepts two arguments into the parameters x and y. The function should return the value of x times y. Remember; multiplication can be performed as repeated addition as follows:

7 3 4 5 4 1 4 1 4 1 4 1 4 1 4

(To keep the function simple, assume that x and y will always hold positive nonzero integers.)

3. Recursive Lines

Write a recursive function that accepts an integer argument, n. The function should display n lines of asterisks on the screen, with the first line showing 1 asterisk, the second line showing 2 asterisks, up to the nth line which shows n asterisks.

4. Largest List item

Design a function that accepts a list as an argument and returns the largest value in the list. The function should use recursion to find the largest item.

5. Recursive List sum

Design a function that accepts a list of numbers as an argument. The function should recursively calculate the sum of all the numbers in the list and return that value.

6. sum of numbers

Design a function that accepts an integer argument and returns the sum of all the integers from 1 up to the number passed as an argument. For example, if 50 is passed as an argument, the function will return the sum of 1, 2, 3, 4... 50. Use recursion to calculate the sum.

7. Recursive power Method

Design a function that uses recursion to raise a number to a power. The function should accept two arguments: the number to be raised and the exponent. Assume that the exponent is a nonnegative integer.

HW-10

1. Pet Class

Write a class named Pet, which should have the following data attributes:

- _ _name(for the name of a pet)
- _ _animal_type (for the type of animal that a pet is. Example values are 'Dog', 'Cat', and 'Bird')
- _ _age(for the pet's age)

The Petclass should have an __init__method that creates these attributes. It should also have the following methods:

- set_name
  This method assigns a value to the _ _name field.
- set_animal_type
  This method assigns a value to the _ _animal_type field.
- set_age
  This method assigns a value to the _ _age field.
- get_name
  This method returns the value of the _ _ namefield.
- get_animal_type
  This method returns the value of the _ _animal_type field.
- get_age
  This method returns the value of the _ _age field.

Once you have written the class, write a program that creates an object of the class and prompts the user to enter the name, type, and age of his or her pet. This data should be stored as the object's attributes. Use the object's accessor methods to retrieve the pet's name, type, and age and display this data on the screen.

2. Car Class

Write a class named Car that has the following data attributes:

- _ _year_model(for the car's year model)
- _ _make(for the make of the car)
- _ _speed(for the car's current speed)

The Carclass should have an __init__method that accepts the car's year model and make as arguments. These values should be assigned to the object's _ _year_model and __make data attributes. It should also assign 0 to the _ _speed data attribute.

The class should also have the following methods:

- accelerate
  The acceleratemethod should add 5 to the speed data attribute each time it is called.
- brake
  The brake method should subtract 5 from the speed data attribute each time it is called.
- get_speed
  The get_speed method should return the current speed.

Next, design a program that creates a Car object and then calls the accelerate method five times. After each call to the accelerate method, get the current speed of the car and display it. Then call the brake method five times. After each call to the brake method, get the current speed of the car and display it.

3. trivia Game

In this programming exercise you will create a simple trivia game for two players. The program will work like this:

- Starting with player 1, each player gets a turn at answering 5 trivia questions. (There should be a total of 10 questions.) When a question is displayed, 4 possible answers are also displayed. Only one of the answers is correct, and if the player selects the correct answer, he or she earns a point.
- After answers have been selected for all the questions, the program displays the number of points earned by each player and declares the player with the highest number of points the winner.

To create this program, write a Question class to hold the data for a trivia question. The Question class should have attributes for the following data:

- A trivia question.
- Possible answer 1
- Possible answer 2
- Possible answer 3
- Possible answer 4
- The number of the correct answer (1, 2, 3, or 4)

The Question class also should have an appropriate _ _init_ _ method, accessors, and mutators.

The program should have a list or a dictionary containing 10 Question objects, one for each trivia question. Make up your own trivia questions on the subject or subjects of your choice for the objects.

HW-11

1. **Employee and ProductionWorker classes**

   Write an Employeeclass that keeps data attributes for the following pieces of information:

- Employee name
- Employee number

   Next, write a class named ProductionWorker that is a subclass of the Employee class. The ProductionWorker class should keep data attributes for the following information:

- Shift number (an integer, such as 1, 2, or 3)
- Hourly pay rate

   The workday is divided into two shifts: day and night. The shift attribute will hold an integer value representing the shift that the employee works. The day shift is shift 1 and the night shift is shift 2. Write the appropriate accessor and mutator methods for each class.

   Once you have written the classes, write a program that creates an object of the ProductionWorker class and prompts the user to enter data for each of the object's data attributes. Store the data in the object and then use the object's accessor methods to retrieve it and display it on the screen.

2. **ShiftSupervisor class**

   In a particular factory, a shift supervisor is a salaried employee who supervises a shift. In addition to a salary, the shift supervisor earns a yearly bonus when his or her shift meets production goals. Write a ShiftSupervisor class that is a subclass of the Employee class you created in Programming Exercise 1. The ShiftSupervisor class should keep a data attribute for the annual salary and a data attribute for the annual production bonus that a shift supervisor has earned. Demonstrate the class by writing a program that uses a ShiftSupervisorobject.

3. **Person and Customer classes**

   Write a class named Person with data attributes for a person's name, address, and tele- phone number. Next, write a class named Customer that is a subclass of the Person class. The Customer class should have a data attribute for a customer number and a Boolean data attribute indicating whether the customer wishes to be on a mailing list. Demonstrate an instance of the Customerclass in a simple program.