Chapter 1 Solutions: True/False, Multiple Choice, Programming Exercises

True/False

1. Computer science is the study of computers. *False.*
2. The CPU is the "brain" of the computer. *True.*
3. Secondary memory is also called RAM. *False.*
4. All information that a computer is currently working on is stored in main memory. *True.*
5. The syntax of a language is its meaning, and semantics is its form. *False.*
6. A function definition is a sequence of statements that defines a new command. *True.*
7. A programming environment refers to a place where programmers work. *False.*
8. A variable is used to give a name to a value to it can be referred to in other places. *True.*
9. A loop is used to skip over a section of a program. *False.*
10. A chaotic function can't be computed by a computer. *False.*

Multiple Choice

1. What is the fundamental question of computer science? *(b) What can be computed*
2. An algorithm is like a *(d) recipe*
3. A problem is intractable when *(d) it is not practical to solve*
4. Which of the following is *not* an example of secondary memory? *(a) RAM*
5. Computer languages designed to be used and understood by humans are *(a) high-level languages*
6. A statement is *(b) a complete computer command*
7. One difference between a compiler and an interpreter is *(c) a compiler is no longer needed after a program is translated*
8. By convention, the statements of a program are often placed in a function called *(b) main*
9. Which of the following is *not* true of comments? *(a) They make a program more efficient*
10. The items listed in the parentheses of a function definition are called *(d) both parameters and arguments*

Programming Exercises

1. Start up an interactive Python session and try typing in each of the following commands.Write down the results you see.

(a) print("Hello, world!")

Hello, world!

(b) print("Hello", "world!")

Hello world!

(c) print(3)

3

(d) print(3.0)

3.0

(e) print(2 + 3)

5

(f) print(2.0 + 3.0)

5.0

(g) print("2" + "3")

23

(h) print("2 + 3 =", 2 + 3)

2 + 3 = 5

(i) print(2 * 3)

6

(j) print(2 ** 3)

8

(k) print(2 / 3)

0.6666666666666

2. Enter and run the Chaos program from Section 1.6. Try it out with various values of input to see that it functions as described in the chapter.3. Modify the Chaos program using 2.0 in place of 3.9 as the multiplier in the logistic function. Your modified line of code should look like this:
x = 2.0 * x * (1 - x)

```
# File: chaos.py
# A simple program illustrating chaotic behavior.

print("This program illustrates a chaotic function")
x = eval(input("Enter a number between 0 and 1: "))
for i in range(10):
     x = 2.0 * x * (1 - x)   # This is the modified line of code.
     print(x)
```

4. Modify the Chaos program so that it prints out 20 values instead of 10.

```
# File: chaos.py
# A simple program illustrating chaotic behavior.

print("This program illustrates a chaotic function")
x = eval(input("Enter a number between 0 and 1: "))
for i in range(20):   # This is the modified line of code.
     x = 3.9 * x * (1 - x)
     print(x)
```

5. Modify the Chaos program so that the number of values to print is determined by the user. You will have to add a line near the top of the program to get another value from the user:

```
n = eval(input("How many numbers should I print? "))
```

Then you will need to change the loop to use n instead of a specific number.

```
# File: chaos.py
# A simple program illustrating chaotic behavior.

print(“This program illustrates a chaotic function”)
n = eval(input("How many numbers should I print? "))   # Added line of code
x = eval(input(“Enter a number between 0 and 1: ”))
for i in range(n):    # Modified line of code
     x = 3.9 * x * (1 - x)
     print(x)
```

6. The calculation performed in the chaos program can be written in a number of ways that are algebraically equivalent. Write a version of the chaos program for each of the following ways of doing the computation. Have your modified programs print out 100 iterations of the function and compare the results when run on the same input. Explain the results of this experiment. Hint: see discussion question number 4, above.

(a) 3.9 * x * (1 - x)

```
# File: chaos.py
# A simple program illustrating chaotic behavior.

print("This program illustrates a chaotic function")
x = eval(input("Enter a number between 0 and 1: "))
for i in range(100):
     x = 3.9 * x * (1 - x)   # Modified line of code
     print(x)
```

(b) 3.9 * (x - x * x)

```
# File: chaos.py
# A simple program illustrating chaotic behavior.

print("This program illustrates a chaotic function")
x = eval(input("Enter a number between 0 and 1: "))
for i in range(100):
     x = 3.9 * (x - x * x)   # Modified line of code
     print(x)
```

(c) 3.9 * x - 3.9 * x * x

```
# File: chaos.py
# A simple program illustrating chaotic behavior.

print("This program illustrates a chaotic function")
x = eval(input("Enter a number between 0 and 1: "))
for i in range(100):
     x = 3.9 * x - 3.9 * x * x   # Modified line of code
     print(x)
```

7. (Advanced) Modify the Chaos program so that it accepts two inputs and then prints a table with two columns similar to the one shown in Section 1.8. (Note: You will probably not be able to get the columns

to line up as nicely as those in the example. Chapter 5 discusses how to print numbers with a fixed number of decimal places.)

```
# File: chaos.py
# A simple program illustrating chaotic behavior.

print("This program illustrates a chaotic function")
num1 = eval(input("Enter a number between 0 and 1: "))
num2 = eval(input("Enter a number between 0 and 1: "))
print("\ninput\t ", num1, "\t\t ", num2)
print("-------------------------------")
for i in range(10):
    num1 = 3.9 * num1 * (1 - num1)
    num2 = 3.9 * num2 * (1 - num2)

    print("\t%.6f\t%.6f" % (num1, num2))
```