If Statement Exercises

2

The programming constructs that you used to solve the exercises in the previous chapter will continue to be useful as you tackle these problems. In addition, the exercises in this chapter will require you to use decision making constructs so that your programs can handle a variety of different situations that might arise. You should expect to use some or all of these Python features when completing these problems:

- Make a decision with an if statement
- Select one of two alternatives with an if-else statement
- Select from one of several alternatives by using an if-elif or if-elif-else statement
- Construct a complex condition for an if statement that includes the Boolean operators `and`, `or` and `not`
- Nest an if statement within the body of another if statement

Exercise 34: Even or Odd?

(*Solved—13 Lines*)

Write a program that reads an integer from the user. Then your program should display a message indicating whether the integer is even or odd.

Exercise 35: Dog Years

(*22 Lines*)

It is commonly said that one human year is equivalent to 7 dog years. However this simple conversion fails to recognize that dogs reach adulthood in approximately two years. As a result, some people believe that it is better to count each of the first two human years as 10.5 dog years, and then count each additional human year as 4 dog years.

Write a program that implements the conversion from human years to dog years described in the previous paragraph. Ensure that your program works correctly for conversions of less than two human years and for conversions of two or more human years. Your program should display an appropriate error message if the user enters a negative number.

Exercise 36: Vowel or Consonant

(*Solved—16 Lines*)

In this exercise you will create a program that reads a letter of the alphabet from the user. If the user enters a, e, i, o or u then your program should display a message indicating that the entered letter is a vowel. If the user enters y then your program should display a message indicating that sometimes y is a vowel, and sometimes y is a consonant. Otherwise your program should display a message indicating that the letter is a consonant.

Exercise 37: Name that Shape

(*Solved—31 Lines*)

Write a program that determines the name of a shape from its number of sides. Read the number of sides from the user and then report the appropriate name as part of a meaningful message. Your program should support shapes with anywhere from 3 up to (and including) 10 sides. If a number of sides outside of this range is entered then your program should display an appropriate error message.

Exercise 38: Month Name to Number of Days

(*Solved—18 Lines*)

The length of a month varies from 28 to 31 days. In this exercise you will create a program that reads the name of a month from the user as a string. Then your program should display the number of days in that month. Display "28 or 29 days" for February so that leap years are addressed.

Exercise 39: Sound Levels

(*30 Lines*)

The following table lists the sound level in decibels for several common noises.

Noise	Decibel level (dB)
Jackhammer	130
Gas lawnmower	106
Alarm clock	70
Quiet room	40

Write a program that reads a sound level in decibels from the user. If the user enters a decibel level that matches one of the noises in the table then your program should display a message containing only that noise. If the user enters a number of decibels between the noises listed then your program should display a message indicating which noises the level is between. Ensure that your program also generates reasonable output for a value smaller than the quietest noise in the table, and for a value larger than the loudest noise in the table.

Exercise 40: Name that Triangle

(*Solved—20 Lines*)

A triangle can be classified based on the lengths of its sides as equilateral, isosceles or scalene. All 3 sides of an equilateral triangle have the same length. An isosceles triangle has two sides that are the same length, and a third side that is a different length. If all of the sides have different lengths then the triangle is scalene.

Write a program that reads the lengths of 3 sides of a triangle from the user. Display a message indicating the type of the triangle.

Exercise 41: Note To Frequency

(*Solved—39 Lines*)

The following table lists an octave of music notes, beginning with middle C, along with their frequencies.

Note	Frequency (Hz)
C4	261.63
D4	293.66
E4	329.63
F4	349.23
G4	392.00
A4	440.00
B4	493.88

Begin by writing a program that reads the name of a note from the user and displays the note's frequency. Your program should support all of the notes listed previously.

Once you have your program working correctly for the notes listed previously you should add support for all of the notes from C0 to C8. While this could be done by adding many additional cases to your if statement, such a solution is cumbersome, inelegant and unacceptable for the purposes of this exercise. Instead, you should exploit the relationship between notes in adjacent octaves. In particular, the frequency of any note in octave n is half the frequency of the corresponding note in octave $n+1$. By using this relationship, you should be able to add support for the additional notes without adding additional cases to your if statement.

Hint: To complete this exercise you will need to extract individual characters from the two-character note name so that you can work with the letter and the octave number separately. Once you have separated the parts, compute the frequency of the note in the fourth octave using the data in the table above. Then divide the frequency by 2^{4-x}, where x is the octave number entered by the user. This will halve or double the frequency the correct number of times.

Exercise 42: Frequency To Note

(*Solved—40 Lines*)

In the previous question you converted from note name to frequency. In this question you will write a program that reverses that process. Begin by reading a frequency from the user. If the frequency is within one Hertz of a value listed in the table in the previous question then report the name of the note. Otherwise report that the frequency does not correspond to a known note. In this exercise you only need to consider the notes listed in the table. There is no need to consider notes from other octaves.

Exercise 43: Faces on Money

(*31 Lines*)

It is common for images of a country's previous leaders, or other individuals of historical significance, to appear on its money. The individuals that appear on banknotes in the United States are listed in Table 2.1.

Write a program that begins by reading the denomination of a banknote from the user. Then your program should display the name of the individual that appears on the

Table 2.1 Individuals that
appear on Banknotes

Individual	Amount
George Washington	$1
Thomas Jefferson	$2
Abraham Lincoln	$5
Alexander Hamilton	$10
Andrew Jackson	$20
Ulysses S. Grant	$50
Benjamin Franklin	$100

banknote of the entered amount. An appropriate error message should be displayed if no such note exists.

> While two dollar banknotes are rarely seen in circulation in the United States, they are legal tender that can be spent just like any other denomination. The United States has also issued banknotes in denominations of $500, $1,000, $5,000, and $10,000 for public use. However, high denomination banknotes have not been printed since 1945 and were officially discontinued in 1969. As a result, we will not consider them in this exercise.

Exercise 44: Date to Holiday Name

(*18 Lines*)

Canada has three national holidays which fall on the same dates each year.

| Holiday	Date
New year's day	January 1
Canada day	July 1
Christmas day	December 25

Write a program that reads a month and day from the user. If the month and day match one of the holidays listed previously then your program should display the holiday's name. Otherwise your program should indicate that the entered month and day do not correspond to a fixed-date holiday.

> Canada has two additional national holidays, Good Friday and Labour Day, whose dates vary from year to year. There are also numerous provincial and territorial holidays, some of which have fixed dates, and some of which have variable dates. We will not consider any of these additional holidays in this exercise.

Exercise 45: What Color is that Square?

(22 Lines)

Positions on a chess board are identified by a letter and a number. The letter identifies the column, while the number identifies the row, as shown below:
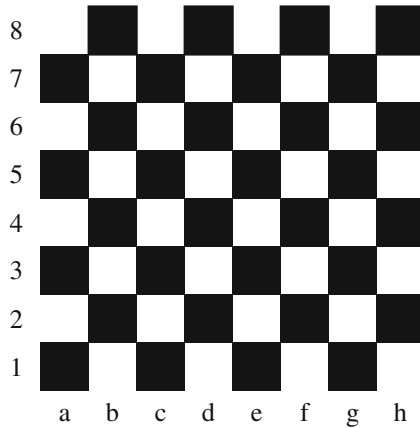
Write a program that reads a position from the user. Use an if statement to determine if the column begins with a black square or a white square. Then use modular arithmetic to report the color of the square in that row. For example, if the user enters a1 then your program should report that the square is black. If the user enters d5 then your program should report that the square is white. Your program may assume that a valid position will always be entered. It does not need to perform any error checking.

Exercise 46: Season from Month and Day

(Solved—40 Lines)

The year is divided into four seasons: spring, summer, fall and winter. While the exact dates that the seasons change vary a little bit from year to year because of the way that the calendar is constructed, we will use the following dates for this exercise:

Season	First day
Spring	March 20
Summer	June 21
Fall	September 22
Winter	December 21

Create a program that reads a month and day from the user. The user will enter the name of the month as a string, followed by the day within the month as an integer. Then your program should display the season associated with the date that was entered.

Exercise 47: Birth Date to Astrological Sign

(47 Lines)

The horoscopes commonly reported in newspapers use the position of the sun at the time of one's birth to try and predict the future. This system of astrology divides the year into twelve zodiac signs, as outline in the table below:

Zodiac sign	Date range
Capricorn	December 22 to January 19
Aquarius	January 20 to February 18
Pisces	February 19 to March 20
Aries	March 21 to April 19
Taurus	April 20 to May 20
Gemini	May 21 to June 20
Cancer	June 21 to July 22
Leo	July 23 to August 22
Virgo	August 23 to September 22
Libra	September 23 to October 22
Scorpio	October 23 to November 21
Sagittarius	November 22 to December 21

Write a program that asks the user to enter his or her month and day of birth. Then your program should report the user's zodiac sign as part of an appropriate output message.

Exercise 48: Chinese Zodiac

(Solved—35 Lines)

The Chinese zodiac assigns animals to years in a 12 year cycle. One 12 year cycle is shown in the table below. The pattern repeats from there, with 2012 being another year of the dragon, and 1999 being another year of the hare.

Year	Animal
2000	Dragon
2001	Snake
2002	Horse
2003	Sheep
2004	Monkey
2005	Rooster
2006	Dog
2007	Pig
2008	Rat
2009	Ox
2010	Tiger
2011	Hare

Write a program that reads a year from the user and displays the animal associated with that year. Your program should work correctly for any year greater than or equal to zero, not just the ones listed in the table.

Exercise 49: Richter Scale

(30 Lines)

The following table contains earthquake magnitude ranges on the Richter scale and their descriptors:

Magnitude	Descriptor
Less than 2.0	Micro
2.0 to less than 3.0	Very minor
3.0 to less than 4.0	Minor
4.0 to less than 5.0	Light
5.0 to less than 6.0	Moderate
6.0 to less than 7.0	Strong
7.0 to less than 8.0	Major
8.0 to less than 10.0	Great
10.0 or more	Meteoric

Write a program that reads a magnitude from the user and displays the appropriate descriptor as part of a meaningful message. For example, if the user enters 5.5 then your program should indicate that a magnitude 5.5 earthquake is considered to be a moderate earthquake.

Exercise 50: Roots of a Quadratic Function

(24 Lines)

A univariate quadratic function has the form $f(x) = ax^2 + bx + c$, where a, b and c are constants, and a is non-zero. The roots of a quadratic function can be found by finding the values of x that satisfy the quadratic equation $ax^2 + bx + c = 0$. A quadratic function may have 0, 1 or 2 real roots. These roots can be computed using the quadratic formula, shown below:

$$root = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

The portion of the expression under the square root sign is called the discriminant. If the discriminant is negative then the quadratic equation does not have any real roots. If the discriminant is 0, then the equation has one real root. Otherwise the equation has two real roots, and the expression must be evaluated twice, once using a plus sign, and once using a minus sign, when computing the numerator.

Write a program that computes the real roots of a quadratic function. Your program should begin by prompting the user for the values of a, b and c. Then it should display a message indicating the number of real roots, along with the values of the real roots (if any).

Exercise 51: Letter Grade to Grade Points

(Solved—52 Lines)

At a particular university, letter grades are mapped to grade points in the following manner:

Letter	Grade points
A+	4.0
A	4.0
A−	3.7
B+	3.3
B	3.0
B−	2.7
C+	2.3
C	2.0
C−	1.7
D+	1.3
D	1.0
F	0

Write a program that begins by reading a letter grade from the user. Then your program should compute and display the equivalent number of grade points. Ensure

that your program generates an appropriate error message if the user enters an invalid
letter grade.

Exercise 52: Grade Points to Letter Grade

(47 Lines)

In the previous exercise you created a program that converts a letter grade into the
equivalent number of grade points. In this exercise you will create a program that
reverses the process and converts from a grade point value entered by the user to a
letter grade. Ensure that your program handles grade point values that fall between
letter grades. These should be rounded to the closest letter grade. Your program
should report A+ for a 4.0 (or greater) grade point average.

Exercise 53: Assessing Employees

(Solved—28 Lines)

At a particular company, employees are rated at the end of each year. The rating scale
begins at 0.0, with higher values indicating better performance and resulting in larger
raises. The value awarded to an employee is either 0.0, 0.4, or 0.6 or more. Values
between 0.0 and 0.4, and between 0.4 and 0.6 are never used. The meaning associated
with each rating is shown in the following table. The amount of an employee's raise
is $2400.00 multiplied by their rating.

Rating	Meaning
0.0	Unacceptable performance
0.4	Acceptable performance
0.6 or more	Meritorious performance

Write a program that reads a rating from the user and indicates whether the perfor-
mance was unacceptable, acceptable or meritorious. The amount of the employee's
raise should also be reported. Your program should display an appropriate error
message if an invalid rating is entered.

Exercise 54: Wavelengths of Visible Light

(38 Lines)

The wavelength of visible light ranges from 380 to 750 nanometers (nm). While the
spectrum is continuous, it is often divided into 6 colors as shown below:

Color	Wavelength (nm)
Violet	380 to less than 450
Blue	450 to less than 495
Green	495 to less than 570
Yellow	570 to less than 590
Orange	590 to less than 620
Red	620 to 750

Write a program that reads a wavelength from the user and reports its color. Display an appropriate error message if the wavelength entered by the user is outside of the visible spectrum.

Exercise 55: Frequency to Name

(31 Lines)

Electromagnetic radiation can be classified into one of 7 categories according to its frequency, as shown in the table below:

Name	Frequency range (Hz)
Radio waves	Less than 3×10^9
Microwaves	3×10^9 to less than 3×10^{12}
Infrared light	3×10^{12} to less than 4.3×10^{14}
Visible light	4.3×10^{14} to less than 7.5×10^{14}
Ultraviolet light	7.5×10^{14} to less than 3×10^{17}
X-rays	3×10^{17} to less than 3×10^{19}
Gamma rays	3×10^{19} or more

Write a program that reads the frequency of the radiation from the user and displays the appropriate name.

Exercise 56: Cell Phone Bill

(44 Lines)

A particular cell phone plan includes 50 minutes of air time and 50 text messages for $15.00 a month. Each additional minute of air time costs $0.25, while additional text messages cost $0.15 each. All cell phone bills include an additional charge of $0.44 to support 911 call centers, and the entire bill (including the 911 charge) is subject to 5 percent sales tax.

Write a program that reads the number of minutes and text messages used in a month from the user. Display the base charge, additional minutes charge (if any), additional text message charge (if any), the 911 fee, tax and total bill amount. Only display the additional minute and text message charges if the user incurred costs in these categories. Ensure that all of the charges are displayed using 2 decimal places.

Exercise 57: Is it a Leap Year?

(*Solved—22 Lines*)

Most years have 365 days. However, the time required for the Earth to orbit the Sun is actually slightly more than that. As a result, an extra day, February 29, is included in some years to correct for this difference. Such years are referred to as leap years. The rules for determining whether or not a year is a leap year follow:

- Any year that is divisible by 400 is a leap year.
- Of the remaining years, any year that is divisible by 100 is **not** a leap year.
- Of the remaining years, any year that is divisible by 4 is a leap year.
- All other years are **not** leap years.

Write a program that reads a year from the user and displays a message indicating whether or not it is a leap year.

Exercise 58: Next Day

(*50 Lines*)

Write a program that reads a date from the user and computes its immediate successor. For example, if the user enters values that represent 2013-11-18 then your program should display a message indicating that the day immediately after 2013-11-18 is 2013-11-19. If the user enters values that represent 2013-11-30 then the program should indicate that the next day is 2013-12-01. If the user enters values that represent 2013-12-31 then the program should indicate that the next day is 2014-01-01. The date will be entered in numeric form with three separate input statements; one for the year, one for the month, and one for the day. Ensure that your program works correctly for leap years.

Exercise 59: Is a License Plate Valid?

(*Solved—28 Lines*)

In a particular jurisdiction, older license plates consist of three uppercase letters followed by three numbers. When all of the license plates following that pattern had

been used, the format was changed to four numbers followed by three uppercase letters.

Write a program that begins by reading a string of characters from the user. Then your program should display a message indicating whether the characters are valid for an older style license plate or a newer style license plate. Your program should display an appropriate message if the string entered by the user is not valid for either style of license plate.

Exercise 60: Roulette Payouts

(*Solved—45 Lines*)

A roulette wheel has 38 spaces on it. Of these spaces, 18 are black, 18 are red, and two are green. The green spaces are numbered 0 and 00. The red spaces are numbered 1, 3, 5, 7, 9, 12, 14, 16, 18, 19, 21, 23, 25, 27, 30 32, 34 and 36. The remaining integers between 1 and 36 are used to number the black spaces.

Many different bets can be placed in roulette. We will only consider the following subset of them in this exercise:

- Single number (1 to 36, 0, or 00)
- Red versus Black
- Odd versus Even (Note that 0 and 00 do **not** pay out for even)
- 1 to 18 versus 19 to 36

Write a program that simulates a spin of a roulette wheel by using Python's random number generator. Display the number that was selected and all of the bets that must be payed. For example, if 13 is selected then your program should display:

```
The spin resulted in 13...
Pay 13
Pay Black
Pay Odd
Pay 1 to 18
```

If the simulation results in 0 or 00 then your program should display `Pay 0` or `Pay 00` without any further output.

Loop Exercises

3

The exercises that appear in this chapter should all be completed using loops. In some cases the exercise specifies what type of loop to use. In other cases you must make this decision yourself. Some of the exercises can be completed easily with both `for` loops and `while` loops. Other exercises are much better suited to one type of loop than the other. In addition, some of the exercises require multiple loops. When multiple loops are involved, one loop might need to be nested inside the other. Carefully consider your choice of loops as you design your solution to each problem.

Exercise 61: Average

(26 Lines)

In this exercise you will create a program that computes the average of a collection of values entered by the user. The user will enter 0 as a sentinel value to indicate that no further values will be provided. Your program should display an appropriate error message if the first value entered by the user is 0.

> Hint: Because the 0 marks the end of the input it should **not** be included in the average.

Exercise 62: Discount Table

(18 Lines)

A particular retailer is having a 60 percent off sale on a variety of discontinued products. The retailer would like to help its customers determine the reduced price of the merchandise by having a printed discount table on the shelf that shows the

original prices and the prices after the discount has been applied. Write a program that uses a loop to generate this table, showing the original price, the discount amount, and the new price for purchases of $4.95, $9.95, $14.95, $19.95 and $24.95. Ensure that the discount amounts and the new prices are rounded to 2 decimal places when they are displayed.

Exercise 63: Temperature Conversion Table

(*22 Lines*)

Write a program that displays a temperature conversion table for degrees Celsius and degrees Fahrenheit. The table should include rows for all temperatures between 0 and 100 degrees Celsius that are multiples of 10 degrees Celsius. Include appropriate headings on your columns. The formula for converting between degrees Celsius and degrees Fahrenheit can be found on the internet.

Exercise 64: No More Pennies

(*Solved—38 Lines*)

February 4, 2013 was the last day that pennies were distributed by the Royal Canadian Mint. Now that pennies have been phased out retailers must adjust totals so that they are multiples of 5 cents when they are paid for with cash (credit card and debit card transactions continue to be charged to the penny). While retailers have some freedom in how they do this, most choose to round to the closest nickel.

Write a program that reads prices from the user until a blank line is entered. Display the total cost of all the entered items on one line, followed by the amount due if the customer pays with cash on a second line. The amount due for a cash payment should be rounded to the nearest nickel. One way to compute the cash payment amount is to begin by determining how many pennies would be needed to pay the total. Then compute the remainder when this number of pennies is divided by 5. Finally, adjust the total down if the remainder is less than 2.5. Otherwise adjust the total up.

Exercise 65: Compute the Perimeter of a Polygon

(*Solved—42 Lines*)

Write a program that computes the perimeter of a polygon. Begin by reading the x and y values for the first point on the perimeter of the polygon from the user. Then continue reading pairs of x and y values until the user enters a blank line for the

x-coordinate. Each time you read an additional coordinate you should compute the distance to the previous point and add it to the perimeter. When a blank line is entered for the x-coordinate your program should add the distance from the last point back to the first point to the perimeter. Then it should display the total perimeter. Sample input and output is shown below, with user input shown in bold:

```
Enter the x part of the coordinate: 0
Enter the y part of the coordinate: 0
Enter the x part of the coordinate: (blank to quit): 1
Enter the y part of the coordinate: 0
Enter the x part of the coordinate: (blank to quit): 0
Enter the y part of the coordinate: 1
Enter the x part of the coordinate: (blank to quit):
The perimeter of that polygon is 3.414213562373095
```

Exercise 66: Compute a Grade Point Average

(62 Lines)

Exercise 51 included a table that shows the conversion from letter grades to grade points at a particular academic institution. In this exercise you will compute the grade point average of an arbitrary number of letter grades entered by the user. The user will enter a blank line to indicate that all of the grades have been provided. For example, if the user enters A, followed by C+, followed by B, followed by a blank line then your program should report a grade point average of 3.1.

You may find your solution to Exercise 51 helpful when completing this exercise. Your program does not need to do any error checking. It can assume that each value entered by the user will always be a valid letter grade or a blank line.

Exercise 67: Admission Price

(Solved—38 Lines)

A particular zoo determines the price of admission based on the age of the guest. Guests 2 years of age and less are admitted without charge. Children between 3 and 12 years of age cost $14.00. Seniors aged 65 years and over cost $18.00. Admission for all other guests is $23.00.

Create a program that begins by reading the ages of all of the guests in a group from the user, with one age entered on each line. The user will enter a blank line to indicate that there are no more guests in the group. Then your program should display the admission cost for the group with an appropriate message. The cost should be displayed using two decimal places.

Exercise 68: Parity Bits

(*Solved—25 Lines*)

A parity bit is a simple mechanism for detecting errors in data transmitted over an unreliable connection such as a telephone line. The basic idea is that an additional bit is transmitted after each group of 8 bits so that a single bit error in the transmission can be detected.

Parity bits can be computed for either even parity or odd parity. If even parity is selected then the parity bit that is transmitted is chosen so that the total number of one bits transmitted (8 bits of data plus the parity bit) is even. When odd parity is selected the parity bit is chosen so that the total number of one bits transmitted is odd.

Write a program that computes the parity bit for groups of 8 bits entered by the user using even parity. Your program should read strings containing 8 bits until the user enters a blank line. After each string is entered by the user your program should display a clear message indicating whether the parity bit should be 0 or 1. Display an appropriate error message if the user enters something other than 8 bits.

> Hint: You should read the input from the user as a string. Then you can use the `count` method to help you determine the number of zeros and ones in the string. Information about the `count` method is available online.

Exercise 69: Approximate π

(*23 Lines*)

The value of π can be approximated by the following infinite series:

$$\pi \approx 3 + \frac{4}{2 \times 3 \times 4} - \frac{4}{4 \times 5 \times 6} + \frac{4}{6 \times 7 \times 8} - \frac{4}{8 \times 9 \times 10} + \frac{4}{10 \times 11 \times 12} - \cdots$$

Write a program that displays 15 approximations of π. The first approximation should make use of only the first term from the infinite series. Each additional approximation displayed by your program should include one more term in the series, making it a better approximation of π than any of the approximations displayed previously.

Exercise 70: Caesar Cipher

(*Solved—35 Lines*)

One of the first known examples of encryption was used by Julius Caesar. Caesar needed to provide written instructions to his generals, but he didn't want his enemies

to learn his plans if the message slipped into their hands. As result, he developed what later became known as the Caesar Cipher.

The idea behind this cipher is simple (and as a result, it provides no protection against modern code breaking techniques). Each letter in the original message is shifted by 3 places. As a result, A becomes D, B becomes E, C becomes F, D becomes G, etc. The last three letters in the alphabet are wrapped around to the beginning: X becomes A, Y becomes B and Z becomes C. Non-letter characters are not modified by the cipher.

Write a program that implements a Caesar cipher. Allow the user to supply the message and the shift amount, and then display the shifted message. Ensure that your program encodes both uppercase and lowercase letters. Your program should also support negative shift values so that it can be used both to encode messages and decode messages.

Exercise 71: Square Root

(*14 Lines*)

Write a program that implements Newton's method to compute and display the square root of a number entered by the user. The algorithm for Newton's method follows:

Read *x* from the user
Initialize *guess* to *x*/2
While *guess* is not good enough **do**
    Update *guess* to be the average of *guess* and *x*/*guess*

When this algorithm completes, *guess* contains an approximation of the square root. The quality of the approximation depends on how you define "good enough". In the author's solution, *guess* was considered good enough when the absolute value of the difference between *guess* $*$ *guess* and *x* was less than or equal to 10^{-12}.

Exercise 72: Is a String a Palindrome?

(*Solved—23 Lines*)

A string is a palindrome if it is identical forward and backward. For example "anna", "civic", "level" and "hannah" are all examples of palindromic words. Write a program that reads a string from the user and uses a loop to determines whether or not it is a palindrome. Display the result, including a meaningful output message.

Exercise 73: Multiple Word Palindromes

(35 Lines)

There are numerous phrases that are palindromes when spacing is ignored. Examples include "go dog", "flee to me remote elf" and "some men interpret nine memos", among many others. Extend your solution to Exercise 72 so that it ignores spacing while determining whether or not a string is a palindrome. For an additional challenge, extend your solution so that is also ignores punctuation marks and treats uppercase and lowercase letters as equivalent.

Exercise 74: Multiplication Table

(Solved—18 Lines)

In this exercise you will create a program that displays a multiplication table that shows the products of all combinations of integers from 1 times 1 up to and including 10 times 10. Your multiplication table should include a row of labels across the top of it containing the numbers 1 through 10. It should also include labels down the left side consisting of the numbers 1 through 10. The expected output from the program is shown below:

```
         1    2    3    4    5    6    7    8    9   10
    1    1    2    3    4    5    6    7    8    9   10
    2    2    4    6    8   10   12   14   16   18   20
    3    3    6    9   12   15   18   21   24   27   30
    4    4    8   12   16   20   24   28   32   36   40
    5    5   10   15   20   25   30   35   40   45   50
    6    6   12   18   24   30   36   42   48   54   60
    7    7   14   21   28   35   42   49   56   63   70
    8    8   16   24   32   40   48   56   64   72   80
    9    9   18   27   36   45   54   63   72   81   90
   10   10   20   30   40   50   60   70   80   90  100
```

When completing this exercise you will probably find it helpful to be able to print out a value without moving down to the next line. This can be accomplished by added `end=""` as the last parameter to your print statement. For example, `print("A")` will display the letter A and then move down to the next line. The statement `print("A", end="")` will display the letter A without moving down to the next line, causing the next print statement to display its result on the same line as the letter A.

Exercise 75: Greatest Common Divisor

(*Solved—17 Lines*)

The greatest common divisor of two positive integers, *n* and *m*, is the largest number, *d*, which divides evenly into both *n* and *m*. There are several algorithms that can be used to solve this problem, including:

> Initialize *d* to the smaller of *m* and *n*.
> **While** *d* does not evenly divide *m* or *d* does not evenly divide *n* **do**
> > Decrease the value of *d* by 1
>
> Report *d* as the greatest common divisor of *n* and *m*

Write a program that reads two positive integers from the user and uses this algorithm to determine and report their greatest common divisor.

Exercise 76: Prime Factors

(*27 Lines*)

The prime factorization of an integer, *n*, can be determined using the following steps:

> Initialize *factor* to two
> **While** *factor* is less than or equal to *n* **do**
> > **If** *n* is evenly divisible by *factor* **then**
> > > Conclude that *factor* is a factor of *n*
> > > Divide *n* by *factor* using integer division
> >
> > **Else**
> > > Increase *factor* by one

Write a program that reads an integer from the user. If the value entered by the user is less than 2 then your program should display an appropriate error message. Otherwise your program should display the prime numbers that can be multiplied together to compute *n*, with one factor appearing on each line. For example:

```
Enter an integer (2 or greater): 72
The prime factors of 72 are:
2
2
2
3
3
```

Exercise 77: Binary to Decimal

(18 Lines)

Write a program that converts a binary (base 2) number to decimal (base 10). Your program should begin by reading the binary number from the user as a string. Then it should compute the equivalent decimal number by processing each digit in the binary number. Finally, your program should display the equivalent decimal number with an appropriate message.

Exercise 78: Decimal to Binary

(Solved—26 Lines)

Write a program that converts a decimal (base 10) number to binary (base 2). Read the decimal number from the user as an integer and then use the division algorithm shown below to perform the conversion. When the algorithm completes, *result* contains the binary representation of the number. Display the result, along with an appropriate message.

> Let *result* be an empty string
> Let q represent the number to convert
> **repeat**
>   Set r equal to the remainder when q is divided by 2
>   Convert r to a string and add it to the beginning of *result*
>   Divide q by 2, discarding any remainder, and store the result back into q
> **until** q is 0

Exercise 79: Maximum Integer

(Solved—34 Lines)

This exercise examines the process of identifying the maximum value in a collection of integers. Each of the integers will be randomly selected from the numbers between 1 and 100. The collection of integers may contain duplicate values, and some of the integers between 1 and 100 may not be present.

Take a moment and think about how you would handle this problem on paper. Many people would check each integer in sequence and ask themself if the number that they are currently considering is larger than the largest number that they have seen previously. If it is, then they forget the previous maximum number and remember the current number as the new maximum number. This is a reasonable approach, and will result in the correct answer when the process is performed carefully. If you were performing this task, how many times would you expect to need to update the maximum value and remember a new number?

While we can answer the question posed at the end of the previous paragraph using probability theory, we are going to explore it by simulating the situation. Create a program that begins by selecting a random integer between 1 and 100. Save this integer as the maximum number encountered so far. After the initial integer has been selected, generate 99 additional random integers between 1 and 100. Check each integer as it is generated to see if it is larger than the maximum number encountered so far. If it is then your program should update the maximum number encountered and count the fact that you performed an update. Display each integer after you generate it. Include a notation with those integers which represent a new maximum.

After you have displayed 100 integers your program should display the maximum value encountered, along with the number of times the maximum value was updated during the process. Partial output for the program is shown below, with… representing the remaining integers that your program will display. Run your program several times. Is the number of updates performed on the maximum value what you expected?

```
30
74 <== Update
58
17
40
37
13
34
46
52
80 <== Update
37
97 <== Update
45
55
73
...

The maximum value found was 100
The maximum value was updated 5 times
```

Exercise 80: Coin Flip Simulation

(47 Lines)

What's the minimum number of times you have to flip a coin before you can have three consecutive flips that result in the same outcome (either all three are heads or all three are tails)? What's the maximum number of flips that might be needed? How

many flips are needed on average? In this exercise we will explore these questions
by creating a program that simulates several series of coin flips.

Create a program that uses Python's random number generator to simulate flipping
a coin several times. The simulated coin should be fair, meaning that the probability
of heads is equal to the probability of tails. Your program should flip simulated
coins until either 3 consecutive heads of 3 consecutive tails occur. Display an H each
time the outcome is heads, and a T each time the outcome is tails, with all of the
outcomes shown on the same line. Then display the number of flips needed to reach
3 consecutive flips with the same outcome. When your program is run it should
perform the simulation 10 times and report the average number of flips needed.
Sample output is shown below:

```
H T T T (4 flips)
H H T T H T H T T H H T H T T H T T T (19 flips)
T T T (3 flips)
T H H H (4 flips)
H H H (3 flips)
T H T T H T H H T T H H T H T H H H (18 flips)
H T T H H H (6 flips)
T H T T T (5 flips)
T T H T T H T H T H H H (12 flips)
T H T T T (5 flips)
On average, 7.9 flips were needed.
```