2 Decision Making

The programs that you worked with in Chap. 1 were strictly sequential. Each program's statements were executed in sequence, starting from the beginning of the program and continuing, without interruption, to its end. While sequential execution of every statement in a program can be used to solve some small exercises, it is not sufficient to solve most interesting problems.

Decision making constructs allow programs to contain statements that may or may not be executed when the program runs. Execution still begins at the top of the program and progresses toward the bottom, but some statements that are present in the program may be skipped. This allows programs to perform different tasks for different input values and greatly increases the variety of problems that a Python program can solve.

2.1 If Statements

Python programs make decisions using `if` statements. An `if` statement includes a *condition* and one or more statements that form the *body* of the `if` statement. When an `if` statement is executed, its condition is evaluated to determine whether or not the statements in its body will execute. If the condition evaluates to `True` then the body of the `if` statement executes, followed by the rest of the statements in the program. If the `if` statement's condition evaluates to `False` then the body of the `if` statement is skipped and execution continues at the first line after the body of the `if` statement.

The condition on an `if` statement can be an arbitrarily complex expression that evaluates to either `True` or `False`. Such an expression is called a Boolean expression, named after George Boole (1815–1864), who was a pioneer in formal logic. An `if` statement's condition often includes a relational operator that compares two

B. Stephenson, *The Python Workbook*, Texts in Computer Science,
https://doi.org/10.1007/978-3-030-18873-3_2

values, variables or complex expressions. Python's relational operators are listed below.

Relational Operator	Meaning
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
==	Equal to
!=	Not equal to

The body of an `if` statement consists of one or more statements that must be indented more than the `if` keyword. It ends before the next line that is indented the same amount as (or less than) the `if` keyword. You can choose how many spaces to use when indenting the bodies of your `if` statements. All of the programs presented in this book use two spaces for indenting, but you can use one space, or several spaces, if your prefer.[1]

The following program reads a number from the user, uses two `if` statements to store a message describing the number into the `result` variable, and then displays the message. Each `if` statement's condition uses a relational operator to determine whether or not its body, which is indented, will execute. A colon immediately follows each condition to separate the `if` statement's condition from its body.

```
# Read a number from the user
num = float(input("Enter a number: "))

# Store the appropriate message in result
if num == 0:
  result = "The number was zero"
if num != 0:
  result = "The number was not zero"

# Display the message
print(result)
```

2.2 If-Else Statements

The previous example stored one message into `result` when the number entered by the user was zero, and it stored a different message into `result` when the entered

[1]Most programmers choose to use the same number of spaces each time they indent the body of an `if` statement, though Python does not require this consistency.

number was non-zero. More generally, the conditions on the if statements were constructed so that exactly one of the two if statement bodies would execute. There is no way for both bodies to execute and there is no way for neither body to execute. Such conditions are said to be *mutually exclusive*.

An if-else statement consists of an if part with a condition and a body, and an else part with a body (but no condition). When the statement executes its condition is evaluated. If the condition evaluates to True then the body of the if part executes and the body of the else part is skipped. When the condition evaluates to False the body of the if part is skipped and the body of the else part executes. It is impossible for both bodies to execute, and it is impossible to skip both bodies. As a result, an if-else statement can be used instead of two if statements when one if statement immediately follows the other and the conditions on the if statements are mutually exclusive. Using an if-else statement is preferable because only one condition needs to be written, only one condition needs to be evaluated when the program executes, and only one condition needs to be corrected if a bug is discovered at some point in the future. The program that reports whether or not a value is zero, rewritten so that it uses an if-else statement, is shown below.

```
# Read a number from the user
num = float(input("Enter a number: "))

# Store the appropriate message in result
if num == 0:
  result = "The number was zero"
else:
  result = "The number was not zero"

# Display the message
print(result)
```

When the number entered by the user is zero, the condition on the if-else statement evaluates to True, so the body of the if part of the statement executes and the appropriate message is stored into result. Then the body of the else part of the statement is skipped. When the number is non-zero, the condition on the if-else statement evaluates to False, so the body of the if part of the statement is skipped. Since the body of the if part was skipped, the body of the else part is executed, storing a different message into result. In either case, Python goes on and runs the rest of the program, which displays the message.

2.3 If-Elif-Else Statements

An if-elif-else statement is used to execute exactly one of several alternatives. The statement begins with an if part, followed by one or more elif parts, followed by an else part. All of these parts must include a body that is indented. Each of the if and elif parts must also include a condition that evaluates to either True or False.

When an if-elif-else statement is executed the condition on the if part is evaluated first. If it evaluates to True then the body of the if part is executed and all of the elif and else parts are skipped. But if the if part's condition evaluates to False then its body is skipped and Python goes on and evaluates the condition on the first elif part. If this condition evaluates to True then the body of the first elif part executes and all of the remaining conditions and bodies are skipped. Otherwise Python continues by evaluating the condition on each elif part in sequence. This continues until a condition is found that evaluates to True. Then the body associated with that condition is executed and the remaining elif and else parts are skipped. If Python reaches the else part of the statement (because all of the conditions on the if and elif parts evaluated to False) then it executes the body of the else part.

Let's extend the previous example so that one message is displayed for positive numbers, a different message is displayed for negative numbers, and yet another different message is displayed if the number is zero. While we could solve this problem using a combination of if and/or if-else statements, this problem is well suited to an if-elif-else statement because exactly one of three alternatives must be executed.

```
# Read a number from the user
num = float(input("Enter a number: "))

# Store the appropriate message in result
if num > 0:
  result = "That's a positive number"
elif num < 0:
  result = "That's a negative number"
else:
  result = "That's zero"

# Display the message
print(result)
```

When the user enters a positive number the condition on the if part of the statement evaluates to True so the body of the if part executes. Once the body of the if part has executed, the program continues by executing the print statement on its final line. The bodies of both the elif part and the else part were skipped without evaluating the condition on the elif part of the statement.

When the user enters a negative number the condition on the if part of the statement evaluates to False. Python skips the body of the if part and goes on and evaluates the condition on the elif part of the statement. This condition evaluates to True, so the body of the elif part is executed. Then the else part is skipped and the program continues by executing the print statement.

Finally, when the user enters zero the condition on the if part of the statement evaluates to False, so the body of the if part is skipped and Python goes on and evaluates the condition on the elif part. Its condition also evaluates to False, so Python goes on and executes the body of the else part. Then the final print statement is executed.

Exactly one of an arbitrarily large number of options is executed by an `if-elif-else` statement. The statement begins with an `if` part, followed by as many `elif` parts as needed. The `else` part always appears last and its body only executes when all of the conditions on the `if` and `elif` parts evaluate to `False`.

2.4 If-Elif Statements

The `else` that appears at the end of an `if-elif-else` statement is optional. When the `else` is present, the statement selects *exactly* one of several options. Omitting the `else` selects *at most* one of several options. When an `if-elif` statement is used, none of the bodies execute when all of the conditions evaluate to `False`. Whether one of the bodies executes, or not, the program will continue executing at the first statement after the body of the final `elif` part.

2.5 Nested If Statements

The body of any `if` part, `elif` part or `else` part of any type of `if` statement can contain (almost) any Python statement, including another `if`, `if-else`, `if-elif` or `if-elif-else` statement. When one `if` statement (of any type) appears in the body of another `if` statement (of any type) the `if` statements are said to be *nested*. The following program includes a nested `if` statement.

```
# Read a number from the user
num = float(input("Enter a number: "))

# Store the appropriate message in result
if num > 0:
  # Determine what adjective should be used to describe the number
  adjective = " "
  if num >= 1000000:
    adjective = " really big "
  elif num >= 1000:
    adjective = " big "

  # Store the message for positive numbers including the appropriate adjective
  result = "That's a" + adjective + "positive number"
elif num < 0:
  result = "That's a negative number"
else:
  result = "That's zero"

# Display the message
print(result)
```

This program begins by reading a number from the user. If the number entered by the user is greater than zero then the body of the outer `if` statement is executed. It begins by assigning a string containing one space to `adjective`. Then the inner `if-elif` statement, which is nested inside the outer `if-elif-else` statement, is executed. The inner statement updates `adjective` to `really big` if the entered number is at least 1,000,000 and it updates `adjective` to `big` if the entered number is between 1,000 and 999,999. The final line in the body of the outer `if` part stores the complete message in `result` and then the bodies of the outer `elif` part and the outer `else` part are skipped because the body of the outer `if` part was executed. Finally, the program completes by executing the print statement.

Now consider what happens if the number entered by the user is less than or equal to zero. When this occurs the body of the outer `if` statement is skipped and either the body of the outer `elif` part or the body of the `else` part is executed. Both of these cases store an appropriate message in `result`. Then execution continues with the print statement at the end of the program.

2.6 Boolean Logic

A Boolean expression is an expression that evaluates to either `True` or `False`. The expression can include a wide variety of elements such as the Boolean values `True` and `False`, variables containing Boolean values, relational operators, and calls to functions that return Boolean results. Boolean expressions can also include Boolean operators that combine and manipulate Boolean values. Python includes three Boolean operators: `not`, `and`, and `or`.

The `not` operator reverses the truth of a Boolean expression. If the expression, `x`, which appears to the right of the not operator, evaluates to `True` then `not x` evaluates to `False`. If `x` evaluates to `False` then `not x` evaluates to `True`.

The behavior of any Boolean expression can be described by a *truth table*. A truth table has one column for each distinct variable in the Boolean expression, as well as a column for the expression itself. Each row in the truth table represents one combination of `True` and `False` values for the variables in the expression. A truth table for an expression having n distinct variables has 2^n rows, each of which show the result computed by the expression for a different combination of values. The truth table for the `not` operator, which is applied to a single variable, `x`, has $2^1 = 2$ rows, as shown below.

x	not x
False	True
True	False

The `and` and `or` operators combine two Boolean values to compute a Boolean result. The Boolean expression `x and y` evaluates to `True` if `x` is `True` and `y` is also `True`. If `x` is `False`, or `y` is `False`, or both `x` and `y` are `False` then `x and` y evaluates to False. The truth table for the and operator is shown below. It has $2^2 = 4$ rows because the and operator is applied to two variables.

x	y	x and y
False	False	False
False	True	False
True	False	False
True	True	True

The Boolean expression x or y evaluates to True if x is True, or if y is True, or if both x and y are True. It only evaluates to False if both x and y are False. The truth table for the or operator is shown below:

x	y	x or y
False	False	False
False	True	True
True	False	True
True	True	True

The following Python program uses the or operator to determine whether or not the value entered by the user is one of the first 5 prime numbers. The and and not operators can be used in a similar manner when constructing a complex condition.

```
# Read an integer from the user
x = int(input("Enter an integer: "))

# Determine if it is one of the first 5 primes and report the result
if x == 2 or x == 3 or x == 5 or x == 7 or x == 11:
  print("That's one of the first 5 primes.")
else:
  print("That is not one of the first 5 primes.")
```

2.7 Exercises

The following exercises should be completed using if, if-else, if-elif, and if-elif-else statements together with the concepts that were introduced in Chap. 1. You may also find it helpful to nest an if statement inside the body of another if statement in some of your solutions.

Exercise 35: 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 36: 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 37: 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 38: 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 39: 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 40: Sound Levels

(30 Lines)

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

Noise	Decibel Level
Jackhammer	130 dB
Gas Lawnmower	106 dB
Alarm Clock	70 dB
Quiet Room	40 dB

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 value 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 41: Classifying Triangles

(Solved, 21 Lines)

A triangle can be classified based on the lengths of its sides as equilateral, isosceles or scalene. All three 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 the three sides of a triangle from the user. Then display a message that states the triangle's type.

Exercise 42: 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: You will want to access the characters in the note entered by the user individually when completing this exercise. Begin by separating the letter from the octave. Then compute the frequency for that letter in the fourth octave using the data in the table above. Once you have this frequency you should divide it 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 43: Frequency to Note

(*Solved, 42 Lines*)

In the previous question you converted from a note's name to its 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 corresponding 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 44: 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 below.

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

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 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 45: 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

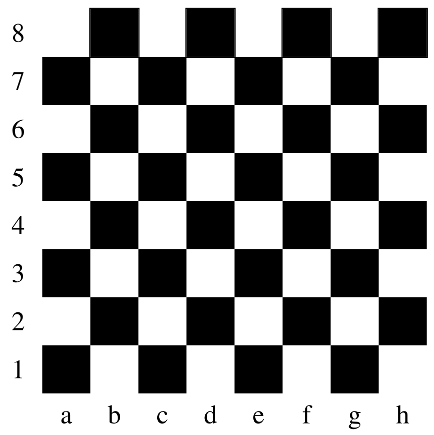
Write a program that reads a month and day from the user. If the month and day match one of the holidays listed previously then your program should display the holiday's name. Otherwise your program should indicate that the entered month and day do not correspond to a fixed-date holiday.

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 46: 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 47: Season from Month and Day

(*Solved, 43 Lines*)

The year is divided into four seasons: spring, summer, fall (or autumn) 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 48: 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 49: Chinese Zodiac

(*Solved, 40 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 50: 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 51: 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. Its roots can be identified by finding the values of x that satisfy the quadratic equation $ax^2 + bx + c = 0$. These values can be computed using the quadratic formula, shown below. A quadratic function may have 0, 1 or 2 real roots.

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

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 52: 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 53: Grade Points to Letter Grade

(*47 Lines*)

In the previous exercise you created a program that converted 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+ if the value entered by the user is 4.0 or more.

Exercise 54: Assessing Employees

(*Solved, 30 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 $2,400.00 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 performance for that rating is 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 55: 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 56: 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 some radiation from the user and displays name of the radiation as part of an appropriate message.

Exercise 57: 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 58: 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 59: 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 2019-11-18 then your program should display a message indicating that the day immediately after 2019-11-18 is 2019-11-19. If the user enters values that represent 2019-11-30 then the program should indicate that the next day is 2019-12-01. If the user enters values that represent 2019-12-31 then the program should indicate that the next day is 2020-01-01. 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 60: What Day of the Week Is January 1?

(*32 Lines*)

The following formula can be used to determine the day of the week for January 1 in a given year:

$$\begin{aligned}\text{day\_of\_the\_week} = (&\text{year} + \text{floor}((\text{year} - 1) / 4) - \text{floor}((\text{year} - 1) / 100) + \\ &\text{floor}((\text{year} - 1) / 400)) \;\%\; 7\end{aligned}$$

The result calculated by this formula is an integer that represents the day of the week. Sunday is represented by 0. The remaining days of the week following in sequence through to Saturday, which is represented by 6.

Use the formula above to write a program that reads a year from the user and reports the day of the week for January 1 of that year. The output from your program should include the full name of the day of the week, not just the integer returned by the formula.

Exercise 61: Is a License Plate Valid?

(*Solved, 28 Lines*)

In a particular jurisdiction, older license plates consist of three uppercase letters followed by three digits. When all of the license plates following that pattern had been used, the format was changed to four digits 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 62: 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.