NILO NEY COUTINHO MENEZES

PYTHON

FROM SCRATCH

PROGRAMMING FOR ABSOLUTE BEGINNERS WITH PYTHON

EXERCISES

1st edition - updated on Aug 2, 2025

LEARN STEP-BY-STEP HOW TO PROGRAM

novatec

LogiKraft

PYTHON
FROM SCRATCH

EXERCISES

1st edition - updated on Aug 2, 2025

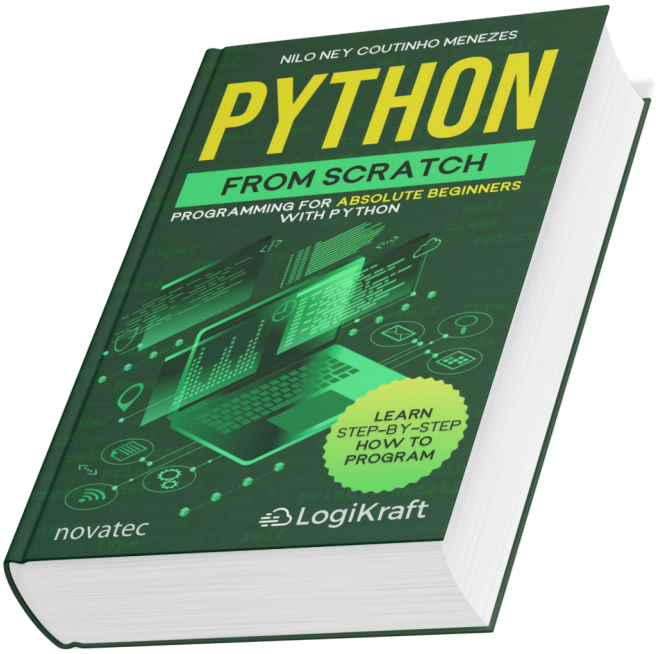
Nilo Ney Coutinho Menezes

questions@pythonfromscratch.com
Telegram: https://t.me/pythonfromscratchbook
Website: https://pythonfromscratch.com

Contents

About the book

This book is aimed at beginners in programming. The basic concepts of programming, such as expressions, variables, loops, decisions, lists, dictionaries, sets, functions, files, classes, objects, databases with SQLite 3, regular expressions, and graphical interfaces with tkinter are presented one by one with examples and exercises. The work aims to explore computer programming as a day-to-day tool. It can be read during an introductory computer programming course and used as a study guide for self-learners. To fully benefit from the content, only basic computer knowledge, such as typing texts, opening and saving files, is sufficient. All software used in the book can be downloaded for free and runs on Windows, Linux, and macOS.



https://pythonfromscratch.com/wheretobuy.html

Introduction

This document was created to provide all the exercises from the book in a single file. The book's website can be accessed at https://pythonfromscratch.com or via the QR code below:

Book website: https://www.pythonfromscratch.com

Chapter 02

2025-07-31

Chapter 02

Exercise 02-01

Convert the following mathematical expressions so that they can be calculated using the Python interpreter.

10 + 20 × 30

42 ÷ 30

(94 + 2) × 6 - 1

Exercise 02-02

Type the following expression in the interpreter:

10 % 3 * 10 ** 2 + 1 - 10 * 4 / 2

Try to solve the same calculation using only pencil and paper. Notice how important the priority of operations is.

Exercise 02-03

Make a program that displays your name on the screen.

Exercise 02-04

Write a program that displays the result of 2a × 3b, where a is 3 and b is 5.

Exercise 02-05

Write a program that calculates the sum of three variables and prints the result on the screen.

Exercise 02-06

Modify Program 2.2 so that it calculates a 15% increase for a salary of $750.

Exercise 02-07

Using the properties of division and multiplication, try to understand how these results are the same:

0.2 * 6 + 8 * 0.3 + 7 * 0.5 = (20 * 6 + 8 * 30 + 7 * 50) / 100

Chapter 03

Exercise 03-01

Complete the following table, marking integer or floating-point depending on the number presented.

Exercise 03-02

Complete the following table, answering True or False. Consider a = 4, b = 10, c = 5.0, d = 1, and f = 5.

Exercise 03-03

Complete the following table using a = True, b = False, and c = True.

Exercise 03-04

Write an expression to determine whether a person should pay tax. Consider that people whose salary is greater than $1,200.00 pay taxes.

Exercise 03-05

Calculate the result of the expression A > B and C or D, using the values in the following table.

```
A  B C     D      Result

1  2 True  False
```

```
10 3 False False

5  1 True  True
```

Exercise 03-06

Write an expression that will be used to decide whether a student is approved. To be approved, all student averages (arithmetic mean) must be greater than or equal to 7 (consider 10 the maximum grade). Consider that the student only takes three subjects and that the grade for each one is stored in the following variables: grade1, grade2, and grade3.

Exercise 03-07

Make a program that asks for two integer numbers. Print the sum of these two numbers on the screen.

Exercise 03-08

Write a program that reads a value in meters and displays it converted to millimeters.

Exercise 03-09

Write a program that reads the user's number of days, hours, minutes, and seconds. Calculate the total in seconds.

Exercise 03-10

Make a program that calculates a pay raise. It must request the amount of the salary and the percentage of the raise. Display the amount of the raise and the new salary.

Exercise 03-11

Make a program that asks for the price of a commodity and the discount percentage. Display the discount amount and the price to pay.

Exercise 03-12

Write a program that calculates the time of a car trip. Ask the distance to cover and the average speed expected for the trip.

Exercise 03-13

Write a program that converts a temperature entered in °C to °F. The formula for this conversion is:

Exercise 03-14

Write a program that asks the number of kilometers traveled by a rental car and the number of days the vehicle was rented. Calculate the price to pay, knowing that the car costs $60 a day and $0.15 per km driven.

Exercise 03-15

Write a program to calculate a smoker's lifespan reduction. Ask how many cigarettes a day they smoke and how many years they have smoked. Consider that a smoker loses 10 minutes of life with each cigarette and calculate how many days of life the smoker will lose. Display the total in days.

Chapter 04

Exercise 04-01

Analyze Program 4.1. What happens if the first and second values are the same? Explain.

Exercise 04-02

Write a program that asks the speed of a user's car. If it exceeds 80 km/h, display a message stating that the user has been fined. In this case, show the amount of the fine, charging $5 per km above 80 km/h.

Exercise 04-03

Write a program that reads three numbers and prints the largest and the smallest.

Exercise 04-04

Write a program that asks for the employee's salary and calculates the amount of the rise. For salaries above $1,250, calculate a raise of 10%. For those equal or lower, 15%.

Exercise 04-05

Run Program 4.5 and try some values. Check that the results are the same as in Program 4.2.

Exercise 04-06

Write a program that asks the distance a passenger wishes to cover in kilometers. Calculate the ticket price, charging $0.50 per km for trips up to 200 km and $0.45 for longer trips.

Exercise 04-07

Analyze Program 4.3. Does using else in that program make sense? Explain your answer.

Exercise 04-08

Rewrite Program 4.4 and calculate the Bye operator account using else.

Exercise 04-09

Trace Program 4.8. Compare your result to that shown in Table 4.2.

Exercise 04-10

Write a program that reads two numbers and asks what operation you want to perform. You must be able to calculate sum (+), subtraction (-), multiplication (*), and division (/). Display the result of the requested operation.

Exercise 04-11

Write a program to approve a bank loan for the purchase of a home. The program must ask the price of the house to buy, the salary, and the number of years to pay.

The amount of the monthly installment cannot exceed 30% of the salary. Calculate the installment as the amount of the house to be purchased divided by the number of months to pay.

Exercise 04-12

Write a program that calculates the price to pay for electricity. Ask the amount of kWh consumed and the type of installation: R for residential, I for industrial, and C for commercial. Calculate the price to pay according to the following table.

Exercise 04-13

In the following program, invert the if and else lines, negating the condition. Add the necessary lines to make it work in Python.

```
 if a > b:

    print("a is greater than b")

else:

    print("b is greater than a")
```

Exercise 04-14

Rewrite the following program with if-elif-else. Add the necessary lines to make it work in Python.

```
if a < 10:

    print("a is less than 10")

if a >= 10 and a < 20:

    print("a is greater than 10 and less than 20")

if a >= 20:

    print("a is greater than 20")
```

Exercise 04-15

Rewrite the following program with if-elif-else.

```
time = int(input("Enter the current time:"))

if time < 12:

    print("Good morning!")

if time >= 12 and time <= 18:

   print("Good afternoon!")

if time >= 18:

    print("Good evening!")
```

Exercise 04-16

Correct the following program:

```
score = input("Enter your exam scores:")

if score < 4:
```

```
    print("Unfortunately you didn't pass")

if score < 7:

    print("You need to resit the exam")

if score > 7:

    print("You passed the year")
```

Chapter 05

Exercise 05-01

Modify the program to display numbers from 1 to 100.

Exercise 05-02

Modify the program to display numbers from 50 to 100.

Exercise 05-03

Make a program to write the countdown of a rocket launch. The program must print 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0, and Fire! on the screen.

Exercise 05-04

Modify the previous program to print from 1 to the number entered by the user, this time producing only odd numbers.

Exercise 05-05

Rewrite the previous program to write the first 10 multiples of 3.

Exercise 05-06

Change the previous program to display the results in the same format as a multiplication table: 2 x 1 = 2, 2 x 2 = 4,...

Exercise 05-07

Modify the previous program so that the user must input the beginning and end of the multiplication table, instead of starting and ending with 1 and 10.

Exercise 05-08

Write a program that reads two numbers. Print the result of multiplying the first by the second. Use only the addition and subtraction operators to calculate the result. Remember that we can understand the multiplication of two numbers as successive sums of one of them (e.g., $4 \times 5 = 5 + 5 + 5 + 5 = 4 + 4 + 4 + 4 + 4$).

Exercise 05-09

Write a program that reads two numbers. Print the result of dividing the first by the second. Use only the addition and subtraction operators to calculate the result. Remember that the quotient of dividing two numbers is the number of times we can subtract the divisor from the dividend. For example, $20 \div 4 = 5$ since we can subtract 4 five times from 20.

Exercise 05-10

Modify the previous program so it accepts answers with uppercase and lowercase letters for all questions.

Exercise 05-11

Write a program that asks for a savings account's initial deposit and interest rate. Display month-by-month values for the first 24 months. Write the total interest earned for the period.

Exercise 05-12

Amend the previous program to ask for the monthly amount deposited. This amount will be deposited at the beginning of each month. Include the deposit amount when calculating interest for the following month.

Exercise 05-13

Write a program that asks for the initial amount of debt and the monthly interest. Also, ask for the monthly amount that will be paid. Print the number of months it will take for the debt to be repaid, the full amount that will be paid, and the total interest that will be paid.

Exercise 05-14

Write a program that reads integers input by the user. The program must read the numbers until the user enters 0 (zero). At the end of the execution, display how many numbers were entered, their sum, and their arithmetic mean.

Exercise 05-15

Write a program to control a small cash register. You must ask the user to enter the product code and the purchase quantity. Use the following code table to obtain the price of each product:

```
Code Price

1    0.50

2    1.00

3    4.00

5    7.00

9    8.00
```

Your program should display the total of the purchases after the user enters 0. Any other code should generate the “Invalid Code” error message.

Exercise 05-16

Run Program 5.1 for the following values: 501, 745, 384, 2, 7, and 1.

Exercise 05-17

What happens if we enter 0 (zero) in the amount to be paid?

Exercise 05-18

Modify the program so it also works with $100 bills.

Exercise 05-19

Modify the program to accept decimal values and count coins of $0.01, $0.05, $0.10, and $0.50.

Exercise 05-20

What happens if we type 0.001 in the previous program? If it doesn't work, change the program to fix the issue.

Exercise 05-21

Rewrite Program 5.1 to continue running until the value entered is 0. Use nested loops.

Exercise 05-22

Write a program that displays a list of options (menu): addition, subtraction, division, multiplication, and exit. Print the multiplication table for the chosen operation. Repeat until the exit option is chosen.

Exercise 05-23

Write a program that reads a number and checks whether it is a prime number. To check, calculate the rest of the division of the number by 2 and then by all the odd numbers up to the number read. If the remainder of one of these divisions equals zero, the number is not prime. Note that 0 and 1 are not prime and that 2 is the only even prime number.

Exercise 05-24

Modify the previous program to read a number n. Print the first n prime numbers.

Exercise 05-25

Write a program that calculates the square root of a number, using Newton's method to get an approximate result. Since n is the number to obtain the square root, consider the base b = 2. Calculate p using the formula p = (b + (n/b)) / 2. Now, calculate the square of p. At each step, do b = p and recalculate p using the formula presented. Stop when the absolute difference between n and the square of p is less than 0.0001.

Exercise 05-26

Write a program that calculates the rest of the integer division between two numbers. Use only the addition and subtraction operations to calculate the result.

Exercise 05-27-a

Write a program that checks whether a number is palindromic. A number is palindromic if it remains the same if its digits are reversed. Examples include 454 and 10501.

Exercise 05-27-b

Write a program that checks whether a number is palindromic. A number is palindromic if it remains the same if its digits are reversed. Examples include 454 and 10501.

Chapter 06

Exercise 06-01

Modify Program 6.2 to read seven grades instead of five.

Exercise 06-02

Make a program that reads two lists and generates a third one with the elements of the first two lists.

Exercise 06-03

Make a program that goes through two lists and generates a third one without repeated elements.

Exercise 06-04

Modify the first example (Program 6.7) to perform the same task without using the variable found. Tip: Look at the while exit condition.

Exercise 06-05

Modify the example to search for two values. Instead of just p, read another value v that will also be searched. In the printout, indicate which of the two values was found first.

Exercise 06-06

Modify the Exercise 6.5 program to search for p and v throughout the list and inform the user of the position in which p and the position in which v were found.

Exercise 06-07

Modify Program 6.6 using for. Explain why every while cannot be turned into a for.

Exercise 06-08

Change Program 6.9 to print the smallest element in the list.

Exercise 06-09

The temperature for Mons, Belgium, was stored in the list T = [-10, -8, 0, 1, 2, 5, -2, -4]. Make a program that prints the lowest, highest, and average temperatures.

Exercise 06-10

Modify Program 6.11 to show how many tickets were sold in each room. Use a list that is the same size as the number of rooms and count the number of tickets sold in each room using its elements as counters. Print the total sales at the end of the program on the screen.

Exercise 06-11

Modify Program 6.11 to ask for the number of rooms and the number of available seats in each.

Exercise 06-12

What happens when the list is already ordered? Trace Program 6.18 but with the list L = [1, 2, 3, 4, 5].

Exercise 06-13

What happens when two values are the same? Trace Program 6.18 but with the list L = [3, 3, 1, 5, 4].

Exercise 06-14

Modify Program 6.18 to sort the list in descending order. L = [1, 2, 3, 4, 5] must be ordered as L = [5, 4, 3, 2, 1].

Exercise 06-15

What happens when we don't check that the list is empty before calling the pop method?

Exercise 06-16

Change Program 6.19 so that you can work with several commands entered at once. Currently, only one command can be entered at a time. Please change it to consider the operation as a string. For example, AAASSSX would mean three new customer arrivals, three services, and finally, the exit from the program.

Exercise 06-17

Modify the program to work with two lines. To make your job easier, consider command S for serving line 1 and T for servicing line 2. The same for the arrival of customers: A for line 1 and B for line 2.

Exercise 06-18

Make a program that reads an expression with parentheses. Using stacks, verify that the parentheses have been opened and closed in the correct order. Example:

(()) OK

()())(OK

()) Error

You can add elements to the stack whenever you find an open parenthesis and unstack it whenever you find a closed one. When unstacking, make sure that the top of the stack is an open parenthesis. If the expression is correct, your stack will be empty at the end.

Exercise 06-19

Change Program 6.22 to request the product and quantity sold from the user. Check if the product name entered exists in the dictionary and only then carry out the stock operation.

Exercise 06-20-a

Write a program that generates a dictionary, where each key is a character and its value is the number of that character found in a sentence input by the user.

Example: “The mouse” → \{“T”: 1, “h”: 1, “e”: 2, ’ “: 1, ’m”: 1, “o”: 1, “u”: 1, “s”: 1}

Exercise 06-20-b

Write a program that generates a dictionary, where each key is a character and its value is the number of that character found in a sentence input by the user.

Example: “The mouse” → \{“T”: 1, “h”: 1, “e”: 2, ’ “: 1, ’m”: 1, “o”: 1, “u”: 1, “s”: 1}

Exercise 06-21

Write a program that compares two lists. Using operations with sets, print:

Exercise 06-22

Write a program that compares two lists. Consider the first list as the initial version and the second as the version after changes. Using operations with sets, your program should print the list of modifications between these two versions, listing:

- The elements that haven't changed

- The new elements

- The elements that were removed

Chapter 07

Exercise 07-01

Write a program that reads two strings. Check that the second occurs inside the first one and print the starting position.

1st string: AABBEFAATT

2nd string: BE

Result: BE found in position 3 of AABBEFAATT

Exercise 07-02

Write a program that reads two strings and generates a third with the characters common to the two strings read.

1st string: AAACTBF

2nd string: CBT

Result: CBT

The order of the characters in the result string is not important, but it must contain all the letters common to both.

Exercise 07-03

Write a program that reads two strings and generates a third one with the characters that appear in only one string.

1st string: CTA

2nd string: ABC

3rd string: BT

The order of the characters in the third string is not important.

Exercise 07-04

Write a program that reads a string and prints how many times each character appears in that string.

String: TTAAC

Result:

T: 2x

A: 2x

C: 1x

Exercise 07-05

Write a program that reads two strings and generates a third one in which the characters of the second are removed from the first.

1st string: AATTGGAA

2nd string: TG

3rd string: AAAA

Exercise 07-06

Write a program that reads three strings. The first string is your source string. The second one has the characters that will be replaced by the ones in the third string. Your program should create a fourth string, the resulting string, which is the first string with the characters of the second replaced by the ones in the third.

1st string: AATTCGAA

2nd string: TG

3rd string: AC

Result: AAAACCAA

Exercise 07-07

Write a program that asks the user to type a phrase and print out how many vowels it contains. Don't consider uppercase and lowercase letters to be different. Example: A phrase like "The house" should print three "eou".

Exercise 07-08

Write a program to display all the words in a sentence. Consider that a word ends with a blank space or when the string ends. Example: "The mouse gnawed at the clothes" should print 6.

Exercise 07-09

Modify the hangman game (Program 7.2) to write the secret word in case the player loses.

Exercise 07-10

Modify Program 7.2 to use a list of words. At the beginning, ask for a number and calculate the index of the word to be used using the formula: index = (number * 776) % len(word_list).

Exercise 07-11

Modify Program 7.2 to use lists of strings to draw the hangman doll. You can use a list for each row and organize them into a list of lists. Instead of controlling when to print each part, draw on those lists, replacing the element to be drawn.

Example:

>>> line = list("X------") >>> line ["X", "-", "-", "-", "-", "-", "-"]

>>> line[6] = "|" >>> line ["X", "-", "-", "-", "-", "-", "|"]

>>> "".join(line) "X-----|"

Exercise 07-12

Write a tic-tac-toe game for two players. The game should ask you where you want to play and switch between players. With each move, check if the position is free. Also, check when a player has won the match. A tic-tac-toe game can be seen as a list of three elements, each element being another list with three elements.

Game example:

```
 X | O |

---+---+---

   | X | X

---+---+---

   |   | O
```

Each position can be viewed as a number. Below is an example of the positions mapped to the same position on your numeric keypad.

```
 7 | 8 | 9

---+---+---

 4 | 5 | 6

---+---+---

 1 | 2 | 3
```

Chapter 08

Exercise 08-01

Write a function that returns the greater of two numbers.

Expected values: maximum(5, 6) == 6

maximum(2, 1) == 2

maximum(7, 7) == 7

Exercise 08-02

Write a function that takes two numbers and returns True if the first number is a multiple of the second.

Expected values: multiple(8, 4) == True

multiple(7, 3) == False

multiple(5, 5) == True

Exercise 08-03

Write a function that takes the length of the side of a square and returns its area (A = side2).

Expected values:

square_area(4) == 16

square_area(9) == 81

Exercise 08-04

Write a function that takes the base and height of a triangle and returns its area (A = (base x height) / 2).

Expected values: triangle_area(6, 9) == 27 triangle_area(5, 8) == 20

Exercise 08-05

Rewrite the function of Program 8.1 to use the list search methods (seen in Chapter 7).

Exercise 08-06

Rewrite Program 8.2 to use for instead of while.

Exercise 08-07

Define a recursive function that calculates the greatest common divisor (G.C.D.) between two numbers a and b, where a > b.

Where

l c m (a , b) = \{ | a × b | g c d (a , b) [.tml-eqn]##

can be written in Python as: a % b.

Exercise 08-08

Using the gcd function defined in the previous exercise, define a function to calculate the least common multiple (LCM) between two numbers.

Where |a × b| can be written in Python as: abs(a * b).

Exercise 08-09

Trace Program 8.6 and compare your result with the one presented.

Exercise 08-10

Rewrite the function for calculating the Fibonacci sequence without recursion.

Exercise 08-11

Write a function to validate a string variable. This function takes the string, the minimum and maximum number of characters, as parameters. Return True if the string size is between the maximum and minimum values; otherwise, return False.

Exercise 08-12

Write a function that takes a string and a list. The function must compare the string passed with the elements of the list, which is also passed as a parameter. Return True if the string is found within the list; otherwise, return False.

Exercise 08-13-a

Write a function that receives a string with the valid options to accept (each option is a letter). Convert valid options to lowercase letters. Use input to read an option, convert the value to lowercase letters, and verify that the option is valid. In the case of an invalid option, the function must ask the user to re-enter another option.

Exercise 08-13-b

Write a function that receives a string with the valid options to accept (each option is a letter). Convert valid options to lowercase letters. Use input to read an option, convert the value to lowercase letters, and verify that the option is valid. In the case of an invalid option, the function must ask the user to re-enter another option.

Exercise 08-14

Change Program 8.22 so that the user has three chances of getting the number right. The program terminates if the user finds the right number or makes three mistakes.

Exercise 08-15

Change Program 7.2, the hangman game. Choose the word to guess using random numbers.

Exercise 08-16

Modify the alien game. Create a variable that represents the player's life, starting with 100 points. The game ends when you find the alien or you run out of life (⇐0). With each mistake, your life is decreased by a random value between 5 and 20 points, representing an attack by the alien. You can remove the part of the game's code that limits the number of attempts and let only the player's or alien's life decide when the match ends. Show how much life the player has left before guessing the next tree number.

Exercise 08-17

Improve the program from the previous exercise by asking the player for the desired difficulty level. In easy mode, life starts at 100 points, and the alien can do between 5 and 20 points of damage. In normal mode, life begins at 80 points, and the alien can cause damage between 10 and 25 points. In hard mode, on the other hand, life begins at 75, and the alien causes damage between 20 and 30 points. Add messages and special characters to make the game more fun.

Exercise 08-18

Modify Program 8.26 to receive two optional parameters. One is to indicate the character to print before the number, with white space being the default value. The second optional parameter is how many characters to add per level, with 2 as the default value.

Exercise 08-19

Write a generator capable of generating the sequence of prime numbers.

Exercise 08-20

Write a generator capable of generating a sequence with the factorial from 1 to n, where n is passed as a parameter to the generator.

Exercise 08-21

Write a function that generates numbers like Python's range function, but the last number is included in the interval. This function takes three parameters, and its behavior changes if we pass one, two, or three parameters. Call it a myrange.

Examples: list(myrange(1)) [0, 1]

list(myrange(1, 10)) [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

list(myrange(0, 10, 2)) [0, 2, 4, 6, 8, 10]

You may have noticed that, unlike range, the myrange function considers the end of the interval as closed; the last number is part of the range.

Exercise 08-22

Modify the calculator program that uses partial to support two more operations: root for square root and power for exponentiation.

Chapter 09

Exercise 09-01

Write a program that takes the name of a file from the command line and prints every line in that file.

Exercise 09-02

Modify the program from Exercise 9.1 to receive two more parameters for printing: the start line and the end line. The program must print only the lines between these values (including the start and end lines).

Exercise 09-03

Create a program that reads the files evens.txt and odds.txt and creates a single file evensandodds.txt with all the lines from the other two files in numerical order.

Exercise 09-04

Create a program that takes the names of two files as command line parameters and generates an output file with the lines from the first followed by the lines from the second file. The name of the output file can also be passed as a parameter on the command line.

Exercise 09-05

Create a program that reverses the order of the lines in the evens.txt file. The first line must contain the largest number, and the last line must contain the smallest number.

Exercise 09-06

Modify Program 9.5 to print the = symbol 40 times if = is the first character in the line. Also, add the option to stop printing until you press the Enter key each time a line starts with . (dot).

Exercise 09-07

Create a program that reads a text file and generates a paginated output file. Each line must not contain more than 76 characters. Each page should have a maximum of 60 lines. The last line of each page should include the number of the current page and the name of the original file.

Exercise 09-08

Modify the program from Exercise 9.7 to receive the number of characters per line and the number of lines per page from the command line.

Exercise 09-09

Create a program that prints a list of files. The file names will be passed in the command line. You should open and print them one by one.

Exercise 09-10

Create a program that receives a list of file names and generates one large output file containing all other files.

Exercise 09-11

Create a program that reads a file and creates a dictionary where each key is a word and each value is the number of occurrences in the file.

Exercise 09-12

Modify the Exercise 9.11 program to also record the row and column of each occurrence of the word in the file. To do this, use lists with the values of each word, saving the row and column of each occurrence.

Exercise 09-13

Create a program that prints the lines of a file. This program must receive three parameters via the command line: the file's name, the starting line, and the last line to print.

Exercise 09-14

Create a program that reads a text file and eliminates repeated spaces between words and at the end of lines. The output file must also not have more than one repeated blank line.

Exercise 09-15

Refer to Program 7.2, the hangman game. Use a text editor to generate a file with a word written on each line. Modify the program to load (read) the list of words from the text file. Also, try asking for the player's name and generating a file with the number of correct answers for the five best players.

Exercise 09-16

Explain how the name and phone fields are stored in the output file.

Exercise 09-17

Change Program 9.6 to display the phonebook size in the main menu. Consider the number of names it contains as its size.

Exercise 09-18

What happens if your name or phone number contains the character used as a separator in your content? Explain the problem and propose a solution.

Exercise 09-19

Change the list_all function so that it also displays the position of each element.

Exercise 09-20

Add the option to sort the list by name in the main menu.

Exercise 09-21

Using the update and delete functions, ask the user to confirm the change and deletion of a name before performing the operation itself.

Exercise 09-22

When reading or writing a new phonebook, verify that the current phonebook has already been saved. You can use a variable to control when the phonebook was changed (new, updated, deleted) and reset that value when loaded or saved.

Exercise 09-23

Change the program to load the last phonebook when initializing. Tip: Use another file to store the filename.

Exercise 09-24

What happens to the phonebook if a loading or saving error occurs? Explain.

Exercise 09-25

Change the ask_name and ask_telephone functions to receive an optional parameter. If this parameter is passed, use it as the value returned if the data entry is empty.

Exercise 09-26

Change the program to verify the repetition of names. Generate an error message to avoid having two phonebook entries with the same name. You should check this before adding or updating names.

Exercise 09-27

Modify the program to control each person's birthday and email address.

Exercise 09-28

Modify the program to register multiple phones for the same person. This also allows you to register the type of telephone number: cell phone, landline, home, or work.

Exercise 09-29

Modify Program 9.8 to use the p element for movie titles instead of h2.

Exercise 09-30

Modify Program 9.8 to generate an HTML list using the ul and li elements. Every element in the list must be inside the ul element and inside an li element. Here's an example:

```
<ul\><li>Item1</li><li>Item2</li><li>Item3</li></ul>
```

Exercise 09-31

Create a program that corrects Program 9.9 to verify that z exists and is a directory.

Exercise 09-32

Modify Program 9.9 to receive the file name or directory to be verified through the command line. Print if it exists and if it's a file or a directory.

Exercise 09-33

Create a program that generates an HTML page with links to all the jpg and png files found in a directory entered on the command line.

Exercise 09-34

Revisit Program 7.2, the hangman game. Modify it to use time functions to record the duration of the matches.

Exercise 09-35

Using the os.walk function, create an HTML page with the name and size of each file from a directory passed in the command line, including all its subdirectories.

Exercise 09-36

Using the os.walk function, create a program that calculates the space occupied per directory, generating an HTML page with the results.

Exercise 09-37

Write a program that reads a student's name and four grades. The program must write the data into a file using the JSON format.

Exercise 09-38

Modify the previous program to read the same file, allowing you to add more data. If the same name is entered twice, update the data using the new entry.

Tip: You may have to use a list of dictionaries to hold multiple entries.

Exercise 09-39

Modify Program 9.6, the phonebook. Have it read and write the phonebook in a file in JSON format.

Exercise 09-40

Modify the visualize.py program to print only the first 512 bytes of the file.

Exercise 09-41

Change the visualize.py program to receive the maximum number of bytes to print and how many bytes per line as command line parameters.

Exercise 09-42

Modify Program 9.20 so that it uses the name of the image to be generated from the command line.

Exercise 09-43

Modify the program from the previous exercise to receive a second parameter with the file name that contains the drawing. The goal is to read the drawing from that file.

Exercise 09-44

Modify the previous program to receive a third parameter with the color conversion table in JSON format.

Chapter 10

Exercise 10-01

Add size and brand attributes to the Television class. Create two Television objects and assign them different sizes and brands. Then, print the value of those attributes to confirm the independence of the values of each instance (object).

Exercise 10-02

Currently, the Television class initializes the channel with 2. Modify the Television class to receive the initial channel in its constructor as an optional parameter.

Exercise 10-03

Modify the Television class so that, if we ask to change the channel down beyond the minimum, it goes to the maximum channel, and vice versa.

Example: >>> tv = Television(2, 10) >>> tv.change_channel_down() >>> tv.channel 10

>>> tv.change_channel_up() >>> tv.channel 2

Exercise 10-04

Using what we learned with functions, modify the Television class constructor so that channel_min and channel_max are optional parameters, where channel_min defaults to 2 and channel_max defaults to 14.

Exercise 10-05

Using the Television class modified in the previous exercise, create two instances (objects), specifying the channel_min and channel_max value by name.

Exercise 10-06

Modify the Television class so that the mute_channel_up and change_channel_down methods return the channel after the change.

Exercise 10-07

Change the Television class to only accept the commands to change channels if turned on.

Exercise 10-08

Change the program to produce a message stating the user has an insufficient account balance if the user attempts to withdraw more than the available balance.

Exercise 10-09

Modify the summary method of the Account class to display the name and phone number of each client.

Exercise 10-10

Create a new account with Joao and Jose as clients and a balance of $500.

Exercise 10-11

Create classes representing states and cities. Each state has a name, an acronym, and cities. Each city has a name and population. Write a test program to create three states with a few cities. Display the population of each state as the sum of the population of its cities.

Exercise 10-12

Modify the Account and SpecialAccount classes so that the withdrawal operation returns True if the withdrawal was made and False if the withdrawal failed.

Exercise 10-13

Change the SpecialAccount class so that your statement shows the limit and the total available for withdrawal.

Exercise 10-14

Observe the withdrawal method from the Account and SpecialAccount classes. Modify the withdrawal method of the Account class so that the possibility of withdrawal is verified by a new method, replacing the current condition. This new method should return True if the withdrawal can be performed or False if not. Modify the SpecialAccount class to work with this new method. Check whether you need to change the SpecialAccount withdrawal method or just the new method created to verify the possibility of withdrawing.

Exercise 10-15

Modify the UniqueList class to override the UserList.extend method. extend works like append but takes a list as a parameter. Check the type of each element in the list before adding it.

Chapter 11

Exercise 11-01

Make a program that creates the prices.db database with the price table to store a list of sales prices for products. The table must contain the name of the product and its respective price. The program must also insert some data for testing.

Exercise 11-02

Make a program to list all the prices in the prices.db database.

Exercise 11-03

Write a program that performs queries in the prices.db database created in Exercise 11.1. The program must ask for the product's name and list its price.

Exercise 11-04

Modify the Exercise 11.3 program to ask for two values and list all the products with prices between those two values.

Exercise 11-05

Write a program that increases the price of all products from the prices.db database by 10%.

Exercise 11-06

Write a program that asks for the product's name and a new price. Using the prices.

db database, update the product price with the same name in the database.

Chapter 12

Exercise 12-01

Modify the previous program to recognize letter sequences. A letter is a character between A and Z or between a and z (you must account for uppercase and lowercase letters). Ignore accented characters. Print a list with the strings of letters found.

Exercise 12-02

Using the pattern_check function, rewrite the function that parsed the numbers in the entry ABC431DEF901C431203FXEW9 .

Exercise 12-03

Using the pattern_check function, write a function that detects a date in the format dd/mm/yy where dd is the day, mm the month, and yy the year. The function should only detect the date pattern and it does not need to verify whether the date is valid.

Exercise 12-04

Using the pattern_check function, write a function that detects a value in dollars in the format $999.99, where 9 represents any digit. The first number can have one or more digits, but the second part (cents) must have a maximum of two digits.

Exercise 12-05

Create a sequence function that receives qmax and qmin. It should work similarly to number but call the sequence function. It should also work when qmin is 0 when the sequence is optional.

Exercise 12-06

Create a function using pattern_check that validates cell phone numbers. A cell phone has 9 digits after the LDC (for example, (92)99812-1103).

Exercise 12-07

Write a program that validates user data entry. It will validate ISBNs (International Standard Book Numbers), which are used to identify books worldwide. The program must accept ISBNs in the following format: ISBN 999-9-99-999999-9, where each 9 represents a digit. The ISBN letters are optional and be case-insensitive. The space between N and the first number is mandatory if the ISBN is present. Require the dashes as in the example, verifying the correct number of digits.

Valid values: ISBN 123-3-12-123456-1 Isbn 123-3-12-123456-1 123-3-12-123456-1

Invalid values: ISBN123-3-12-123456-1 ISBN 1233-12123456-1

Exercise 12-08

Write a program that validates user data entry. It will validate ISSNs (International Standard Serial Numbers), which are used to identify periodical publications, like

magazines. The program must accept ISSNs in the following format: ISSN 9999-9999, where each 9 represents a digit. Require the dash at the end, verifying the correct number of digits. The word ISSN is optional and must be accepted regardless of the case. If the word ISSN is specified, the space between it and the number is required.

Valid values : ISSN 1234-4567 1234-4567

Invalid values: ISSN 123-4567 IssN11112-22

Exercise 12-09

Write a program that validates user data entry. Try to find a valid ISBN or ISSN, as defined in previous exercises. Display a message stating whether the number is a valid ISBN or ISSN (identify which one in your response).

Exercise 12-10

Write a function that accepts prices in dollars. The program must ignore white spaces and accept values prefixed with $ or not. The user must enter correctly formatted values with a comma separating the thousands and a period separating the cents. If the user types cents, they must have two digits.

Valid values: $500 $500 $500.10 $7,312.10

Invalid values: $500\.1

500\.1$ 500.1 7312.10$

The function must return the entered value converted to float or generate a

ValueError exception if the value entered is invalid.

Chapter 13

Exercise 13-01

Modify Program 13.7 to save and load the drawing in JSON format. You can traverse the objects on the canvas and save the chosen shape, coordinates, and colors.

Exercise 13-02

Modify the previous program and save the image in SVG format. The SVG format is a text file that follows a well-defined format.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg width="190mm" height="63mm" viewBox="0 0 190 63" version="1.1"
id="svg1"
    xmlns="http://www.w3.org/2000/svg" xmlns:svg="http://www.w3.org/2000/
svg">
    <g id="layer1">
        <rect style="fill:#cccccc;stroke:#000000;stroke-width:0.264583"
id="rect1" width="42.451897" height="29.063223" x="14.858164"
y="16.001099" />
        <ellipse style="fill:#cccccc;stroke:#000000;stroke-width:0.264583"
id="path1" cx="93.067612" cy="18.940079" rx="15.837823" ry="12.082462" />
        <path style="fill:none;stroke:#000000;stroke-width:0.264583px;
```

```
stroke-linecap:butt;stroke-linejoin:miter;stroke-opacity:1" d="m

172.90984,16.654206 -54.69763,37.5536" id="path2" />

    </g>

</svg>
```

This draws the image shown in Figure 13.14.

Tip: mm — millimeters. We have the rectangle in rect.

The coordinates are inside the style attribute. fill is the internal filling color. strike the outline color. We have our oval in the ellipse element. cx, cy are the coordinates of the center of the ellipse. rx and ry are the horizontal and vertical distance from the center (radius).

The line is in the path element, property d. In d, we have a mini language in which m means to move, followed by the coordinates x1, y1 (without spaces) of the first point and the displacement (x2, y2) after that as the second coordinate. The line is then drawn from (x1, y1) to (x1 + x2, y1 + y2). Try altering the file in a text editor and changing the values before writing the program.

You can view an SVG file in any web browser, so don't forget to reload the page whenever you save your changes. Once you have a clear understanding of how to manipulate these values, write the Python program. You can search for the specification in SVG format on the internet, starting with the Wikipedia website:

https://en.wikipedia.org/wiki/SVG and the Mozilla Foundation: https://developer.mozilla.org/en-US/docs/Web/SVG/Tutorial.

Exercise 13-03

Modify the Window.ok code (Program 13.12) to validate dates when editing or adding new links. Display an error message if the date is invalid.

Exercise 13-04

Modify the Window.ok code (Program 13.12) not to accept blank URLs. Display an error message if the URL is invalid (blank).

Exercise 13-05

Modify the Window.ok code (Program 13.12) to accept only URLs starting with http:// or https://.

Exercise 13-06

Modify Program 13.13 to load and save data from a database. Modify the SiteManager class or create another one that allows you to switch the JSON storage for a database like SQLite.