Errata for Python and Algorithmic Thinking for the Complete Beginner Second Edition

5.2 What is a Constant?

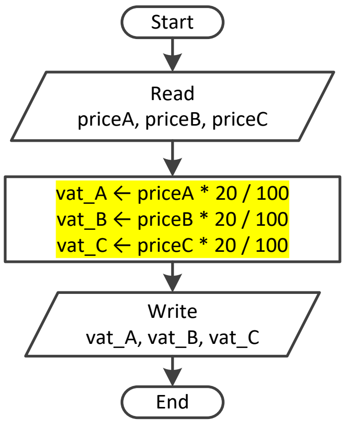

Start
Read
priceA, priceB, priceC
vat_A ← priceA * 20 / 100
vat_B ← priceB * 20 / 100
vat_C ← priceC * 20 / 100
Write
vat_A, vat_B, vat_C
End

Figure 5–1 Calculating the 20% VAT for three products without the use of a constant

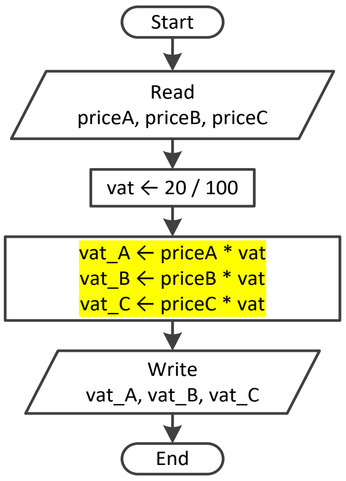


Figure 5–2 Calculating the 20% VAT for three products using a variable, `vat`

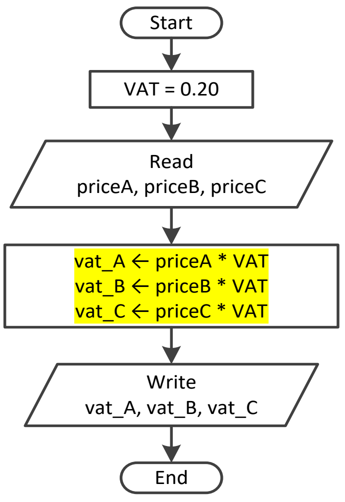


Figure 5-3 Calculating the 20% VAT for three products using a constant, `VAT`

7.8 Review Exercises

3. Match each element from the first column with one element from the second column.

Operation	Result
i. 1 / 2.0	a. 100
ii. 1.0 / 2 * 2	b. 0.25
iii. 0 % 10 * 10	c. 0
iv. 10 % 2 + 7	d. 0.5
	e. 7
	f. 1.0

10.2 Review Exercises

6. Regarding the previous exercise, which of the following results output statements are correct? Which one would you choose to display the volume of the sphere on the user's screen, and why?
    a. `print(V)`
    b. `print(V cubic meters)`
    c. `print(V, cubic meters)`
    d. `print("The volume of the sphere is:" V)`
    e. `print("The volume of the sphere is:", V)`
    f. `print("The volume of the sphere is:", V, cubic meters)`
    g. `print("The volume of the sphere is:", V, "cubic meters")`

14.6 Review Exercises

3. Write a Python program that prompts the user to enter his or her name and then creates a secret password consisting of three letters (in lowercase) randomly picked up from his or her name, and a random four-digit number. For example, if the user enters "Vassilis Bouras" a secret password can probably be one of "sar1359" or "vbs7281" or "bor1459". Space characters are not allowed in the secret password.
4. Write a Python program that prompts the user to enter a three-digit integer and then reverses it. For example, if the user enters the number 375, the number 573 must be displayed. Solve this exercise without using the integer quotient (//) and the integer remainder (%) operators.

15.9 How to Negate Boolean Expressions

For example, if the original Boolean expression is

```
x > 5 and y == 3
```

the negated Boolean expression becomes

```
x <= 5 or y != 3
```

However, there is a small detail that you should be careful with. If both `and` and `or` operators co-exist in a complex Boolean expression, then the expressions that use the `or` operators in the negated Boolean expression must be enclosed in parentheses, in order to preserve the initial order of precedence. For example, if the original Boolean expression is

```
x >= 5 and x <= 10 or y == 3
```

the negated Boolean expression must be

```
(x < 5 or x > 10) and y != 3
```

✎ *If you forget to enclose the expression* `x < 5 or x > 10` *in parentheses, since the* `and` *operator has a higher precedence than the* `or` *operator, the expression* `x > 10 and y != 3` *is evaluated first, which is wrong of course!*

23.3 Review Exercises

4. The following Python program (not code fragment)

```
a = 5
total = total + a
print(total)
```

satisfies the property of effectiveness.

24.4 Review Exercises

14. The following Python program (not code fragment)

```
while True:
    print("Hello")
    i += 1
    if i > 10: break
```

satisfies the property of effectiveness.

25.5 Review Exercises

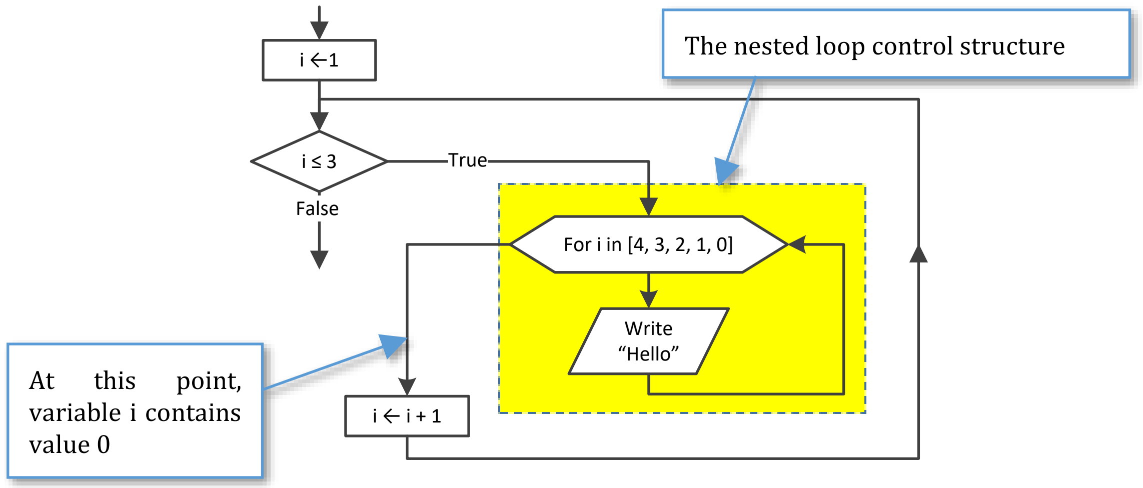
18. Write a Python program that prompts the user to enter a real and an integer and then displays the result of the first number raised to the power of the second number, without using either the exponentiation operator (**) or the built-in `pow()` function of Python.

26.2-2 Counting the Total Number of Iterations

Find the number of times message "Hello" is displayed.

```
i = 1
while i <= 3:
    for i in range(4, -1, -1):
        print("Hello")
    i += 1
```

Solution



The nested loop control structure
i ←1
i ≤ 3
True
False
For i in [4, 3, 2, 1, 0]
Write "Hello"
i ← i + 1
At this point, variable i contains value 0

27.3 The "Ultimate" Rule

You needed a Python program that lets the user enter three positive numbers, not four!

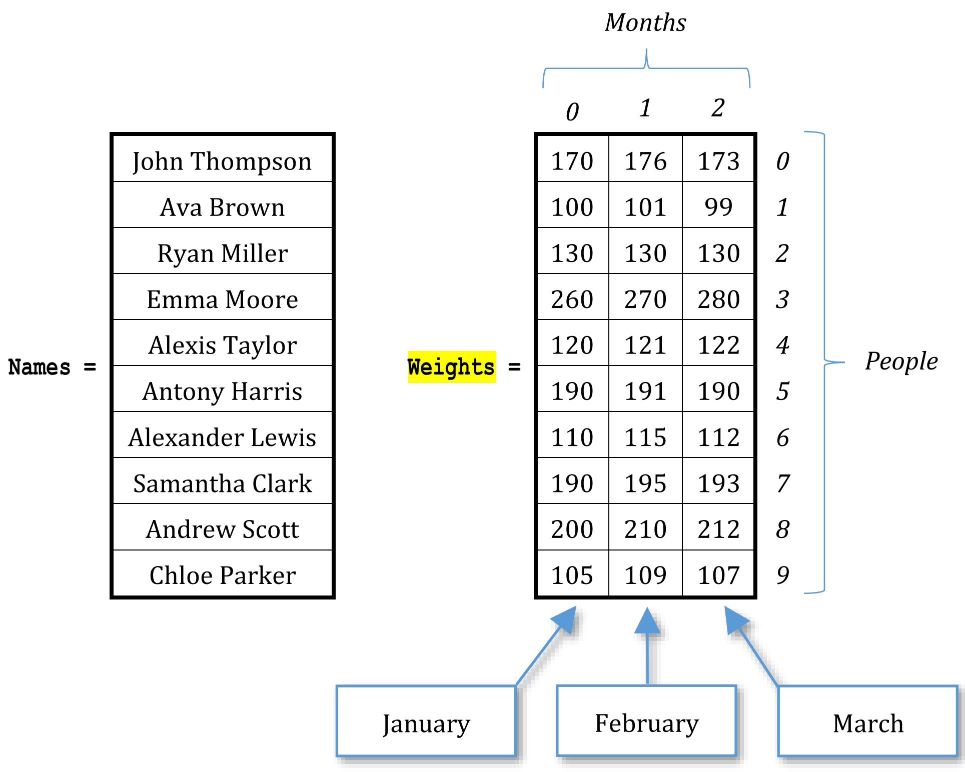
Exercise 30.12-1 Assigning a Value to a Non-Existing Key

Keep in mind though, if `indian_tribes` were actually a list, the last statement would certainly throw an error. Take a look at the following code fragment

```
indian_tribes = ["Navajo", "Cherokee", "Sioux"]
indian_tribes[3] = "Apache"
```

Exercise 30.2-3 Designing Lists

In this exercise, you need a one-dimensional list for names, and a two-dimensional list for people's weights.



Names
John Thompson
Ava Brown
Ryan Miller
Emma Moore
Alexis Taylor
Antony Harris
Alexander Lewis
Samantha Clark
Andrew Scott
Chloe Parker

	0	1	2
0	170	176	173
1	100	101	99
2	130	130	130
3	260	270	280
4	120	121	122
5	190	191	190
6	110	115	112
7	190	195	193
8	200	210	212
9	105	109	107

30.4 How to Get Values from a One-Dimensional List

A negative *step* returns a list in reverse order

```
gods = ["Ares", "Hera", "Aphrodite", "Hermes"]
print(gods[::-1])     #It displays: ['Hermes', 'Aphrodite', 'Hera', 'Ares']
```

30.7 How to Add User-Entered Values to a One-Dimensional List

There is nothing new here. Instead of reading a value from the keyboard and assigning that value to a variable, you can directly assign that value to a specific list element. The next code fragment prompts the user to enter the names of four people, and assigns them to the elements at index positions 0, 1, 2, and 3, of the list `names`.

30.13 How to Iterate Through a Dictionary

Second Approach

Unfortunately, this approach cannot be used to alter the values of the elements of a dictionary. For example, if you want to double the values of all elements of the dictionary `salaries`, you can**not** do the following:

```
salaries = { "Project Manager": 83000,
             "Software Engineer": 81000,
             "Network Engineer": 64000,
             "Systems Administrator": 61000,
             "Software Developer": 70000
           }

for title, salary in salaries.items():
    salary *= 2
```

30.15 Review Questions: True/False

35. If list `b` contains 30 elements (arithmetic values), the following code fragment doubles the values of all of its elements.

```
for i in range(29, -1, -1):
    b[i] = b[i] * 2
```

30.16 Review Questions: Multiple Choice

10. If list `b` contains 30 elements (arithmetic values), the following code fragment

```
for i in range(29, 0, -1):
    b[i] = b[i] * 2
```

a. doubles the values of some of its elements.
b. doubles the values of all of its elements.
c. none of the above

31.4 How to Add User-Entered Values to a Two-Dimensional List

Just as in one-dimensional lists, instead of reading a value entered from the keyboard and assigning that value to a variable, you can directly assign that value to a specific list element. The next code fragment creates a two-dimensional list `names`, prompts the user to enter six values, and assigns those values to the elements of the list.

31.9 Review Exercises

4 Try, without using a trace table, to determine the values that the list will contain when the following code fragment is executed. Do this for three different executions. The corresponding input values are: (i) 5, (ii) 9, and (iii) 3.

```
a = [ [None] * 3 for i in range(2) ]
x = int(input())
for i in range(2):
    for j in range(3):
        a[i][j] = (x + i) * j
```

5. Try, without using a trace table, to determine the values that the list will contain when the following code fragment is executed. Do this for three different executions. The corresponding input values are: (i) 13, (ii) 10, and (iii) 8.

```
a = [ [None] * 3 for i in range(2) ]
x = int(input())
for i in range(2):
    for j in range(3):
        if j < x % 4:
            a[i][j] = (x + i) * j
        else:
            a[i][j] = (x + j) * i + 3
```

32.3 Processing Each Column Individually

First Approach – Creating an auxiliary list

Now, nesting this code fragment in a for-loop that iterates for all columns results in the following.

```
total = [None] * COLUMNS
for j in range(COLUMNS):
    total[j] = 0
    for i in range(ROWS):
        total[j] += b[i][j]
```

Second Approach – Just find it and process it.

This approach uses no auxiliary list; it just calculates and directly processes the sum. The code fragment is as follows.

```
for j in range(COLUMNS):
    total = 0
    for i in range(ROWS):
        total += b[i][j]

    process total
```

Accordingly, the following code fragment calculates and displays the average value of each column.

```
for j in range(COLUMNS):
    total = 0
    for i in range(ROWS):
        total += b[i][j]
    print(total / ROWS)
```

Exercise 32.4-1 Finding the Average Value of Two Grades

file_32_4_1

```
STUDENTS = 20

names = [None] * STUDENTS
grades_lesson1 = [None] * STUDENTS
grades_lesson2 = [None] * STUDENTS

for i in range(STUDENTS):
    names[i] = input("Enter student name No" + str(i + 1) + ": ")
    grades_lesson1[i] = int(input("Enter grade for lesson 1: "))
    grades_lesson2[i] = int(input("Enter grade for lesson 2: "))

#Calculate the average grade for each student
#and display the names of those who are greater than 89
for i in range(STUDENTS):
    total = grades_lesson1[i] + grades_lesson2[i]
    average = total / 2.0
    if average > 89:
        print(names[i])
```

Exercise 33.3-7 Finding the Minimum and the Maximum Value of Each Row

file_33_3_7b

```
ROWS = 30
COLUMNS = 20

b = [ [None] * COLUMNS for i in range(ROWS) ]

for i in range(ROWS):
    for j in range(COLUMNS):
        b[i][j] = float(input())

for i in range(ROWS):
    minimum = b[i][0]
    maximum = b[i][0]
    for j in range(1, COLUMNS):
        if b[i][j] < minimum:
            minimum = b[i][j]
        if b[i][j] > maximum:
            maximum = b[i][j]

    print(minimum, maximum)
```

Exercise 33.4-11 The Three Worst Elapsed Times

Using the “from inner to outer” method, the next code fragment sorts only the first row (row index 0) of the two-dimensional list `elapsed_times` in descending order using the insertion sort algorithm. Assume variable `i` contains the value 0.

```
for m in range(1, LAPS):
    element = elapsed_times[i][m]
    n = m
    while n > 0 and elapsed_times[i][n - 1] < element:
        elapsed_times[i][n] = elapsed_times[i][n - 1]
        n -= 1
    elapsed_times[i][n] = element
```

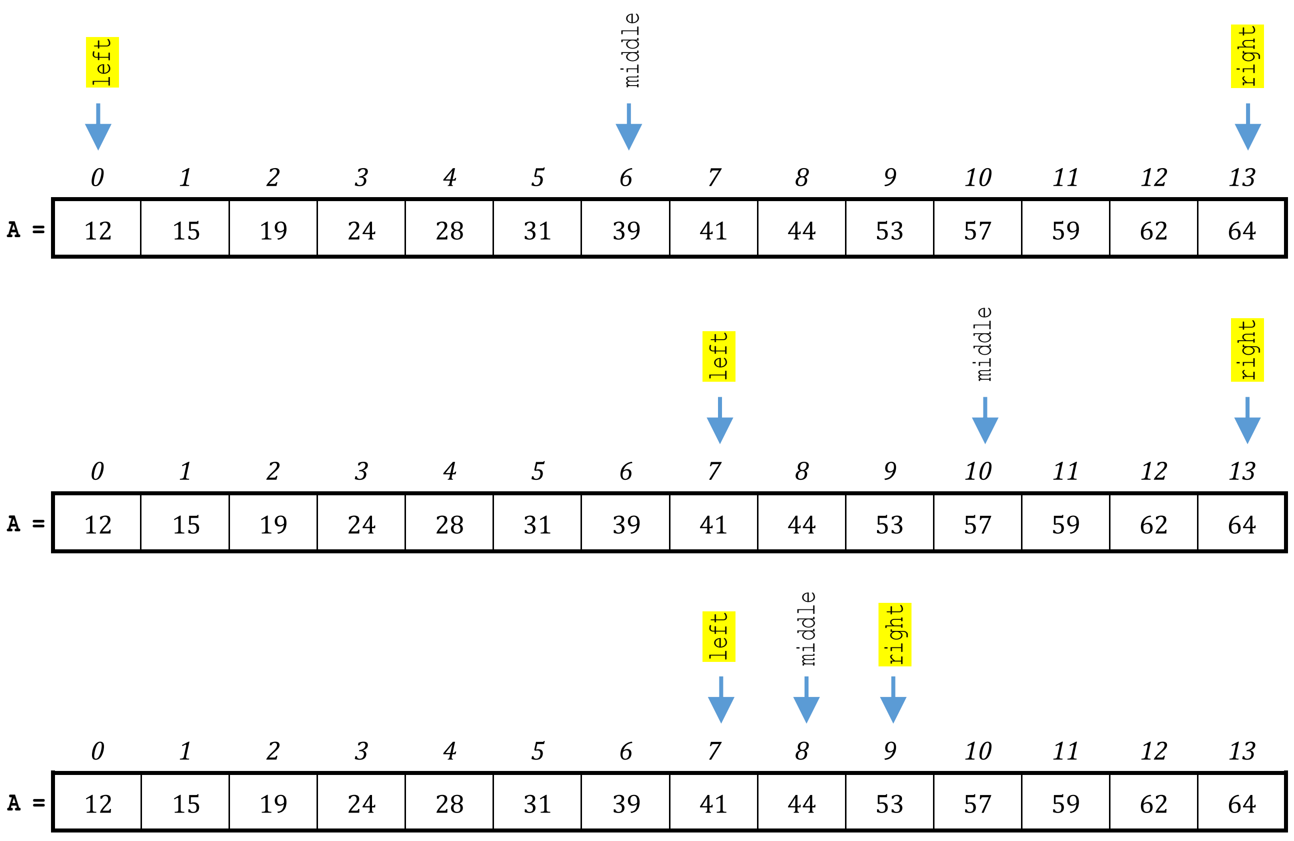
Now, in order to sort all rows, you need to nest this code fragment in a for-loop that iterates for all of them, as follows.

```
for i in range(CARS):
    for m in range(1, LAPS):
        element = elapsed_times[i][m]
        n = m
        while n > 0 and elapsed_times[i][n - 1] < element:
            elapsed_times[i][n] = elapsed_times[i][n - 1]
            n -= 1
        elapsed_times[i][n] = element
```

file_33_4_11

```
…
…
        while n > 0 and elapsed_times[i][n - 1] < element:
```

Exercise 33.5-9 The Binary Search Algorithm – Searching in a Sorted One-Dimensional List

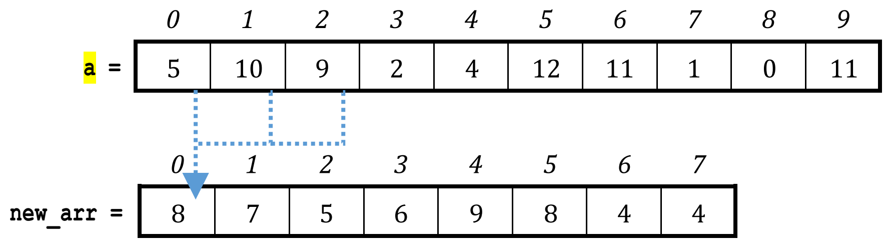


Exercise 33.1-1 Creating a List that Contains the Average Values of its Neighboring Elements

Write a Python program that lets the user enter 100 positive numerical values into a list. Then, the program must create a new list of 98 elements. This new list must contain, in each position the average value of the three elements that exist in the current and the next two positions of the given list.

Solution

Let's try to understand this exercise through an example using 10 elements.



List `new_arr` is the new list that is created. In list `new_arr`, the element at position 0 is the average value of the elements in the current and the next two positions of list a; that is, (5 + 10 + 9) / 3 = 8. The element at position 1 is the average value of the elements in the current and the next two positions of list a; that is, (10 + 9 + 2) / 3 = 7, and so on.

Exercise 33.1-5 Creating Two Lists – Separating Positive from Negative Values

✎ *Note that the lists* `pos` *and* `neg` *contain a total number of* `pos_index` *and* `neg_index` *elements respectively. This is why the two last loop control structures iterate until variable* `i` *reaches values* `pos_index - 1` *and* `neg_index - 1`*, respectively, and not until* `ELEMENTS - 1`*, as you may mistakenly expect.* ~~*Obviously the sum of* `pos_index + neg_index` *equals to* `ELEMENTS`*.*~~

Exercise 33.4-9 Sorting One-Dimensional Lists While Preserving the Relationship with a Second List

Write a Python program that prompts the user to enter the total number of kWh consumed each month for a period of one year. It then displays each number of KWh consumed (in descending order) along with the name of the corresponding month. Use the selection sort algorithm.

Exercise 33.6-4 Display from Highest to Lowest Grades by Student, and in Alphabetical Order

There are 10 students and each one of them has received his or her grades for five lessons. Write a Python program that prompts a teacher to enter the name of each student and his or her grades for all lessons.

35.2 How to Make a Call to a Function

Now, suppose that you want to calculate a result using the following expression

$$y = x^3 + \frac{1}{x}$$

36.6 The Scope of a Variable

The next code fragment declares a global variable `x`, two local variables `x` and `y` within the void function `display_values()`, and one local variable `y` within the void function `display_other_values()`. Keep in mind that the global variable `x` and the local variable `x` are two different variables!

Exercise 37.2-3 Progressive Rates and Electricity Consumption

The LAV Electricity Company charges subscribers for their electricity consumption according to the following table (monthly rates for domestic accounts).

Kilowatt-hours (kWh)	USD per kWh
kWh ≤ 400	\$0.08
401 ≤ kWh ≤ 1500	\$0.22
1501 ≤ kWh ≤ 2000	\$0.35
2001 ≤ kWh	\$0.50

38.8 Class Inheritance

If you want a class to inherit the class `SchoolMember`, it must be defined as follows

```
class Name(SchoolMember):
    def __init__(self, name, age [, …]):
        #Call the constructor of the class SchoolMember
        SchoolMember.__init__(self, name, age)
        Define additional fields for this class

        Additional statement or block of statements


    Define additional methods and/or properties for this class
```

38.10 Review Exercises

3. Do the following
   i. In the class `Pet` of the previous exercise
      a. alter the constructor to accept initial values for the instance fields `kind` and `legs_number` through its formal argument list.

   b. add a property named `kind`. It will be used to get and set the value of the private field `kind`. The setter must throw an error when the field is set to an empty value.
   c. add a property named `legs_number`. It will be used to get and set the value of the private field `legs_number`. The setter must throw an error when the field is set to a negative value.

ii. Write a Python program that creates one instance of the class `Pets` (for example, a dog) and then calls both of its methods. Then try to set erroneous values for properties `kind` and `legs_number` and see what happens.