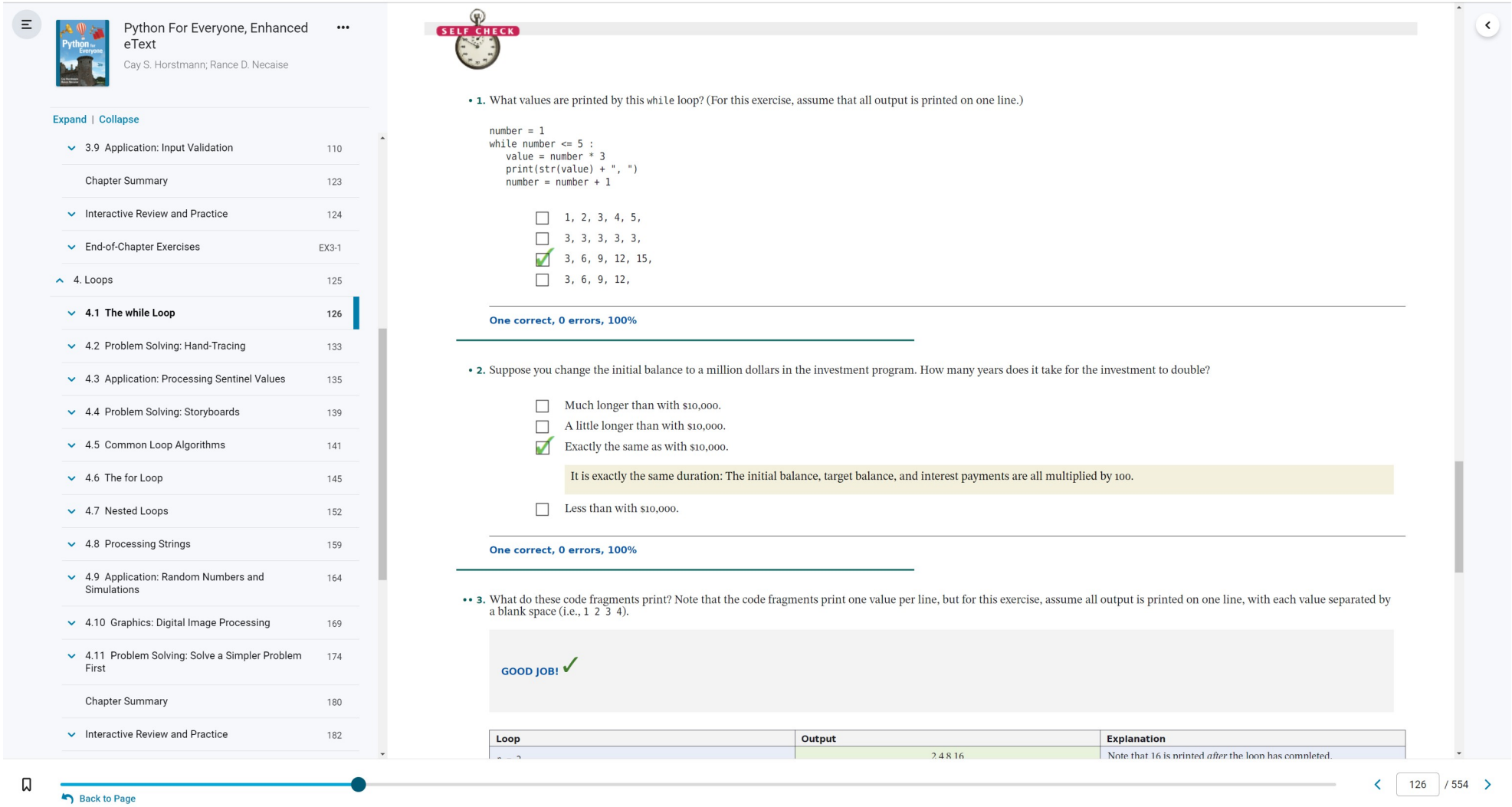


Python For Everyone, Enhanced eText
Cay S. Horstmann; Rance D. Necaise

Expand | Collapse

SELF CHECK

• **1.** What values are printed by this `while` loop? (For this exercise, assume that all output is printed on one line.)

```
number = 1
while number <= 5 :
   value = number * 3
   print(str(value) + ", ")
   number = number + 1
```

- ☐ 1, 2, 3, 4, 5,
- ☐ 3, 3, 3, 3, 3,
- ☑ 3, 6, 9, 12, 15,
- ☐ 3, 6, 9, 12,

One correct, 0 errors, 100%

• **2.** Suppose you change the initial balance to a million dollars in the investment program. How many years does it take for the investment to double?

- ☐ Much longer than with $10,000.
- ☐ A little longer than with $10,000.
- ☑ Exactly the same as with $10,000.

  It is exactly the same duration: The initial balance, target balance, and interest payments are all multiplied by 100.

- ☐ Less than with $10,000.

One correct, 0 errors, 100%

•• **3.** What do these code fragments print? Note that the code fragments print one value per line, but for this exercise, assume all output is printed on one line, with each value separated by a blank space (i.e., `1 2 3 4`).

GOOD JOB! ✓

Loop	Output	Explanation
	2 4 8 16	Note that 16 is printed *after* the loop has completed.

Back to Page

126 / 554

•• **3.** What do these code fragments print? Note that the code fragments print one value per line, but for this exercise, assume all output is printed on one line, with each value separated by a blank space (i.e., `1 2 3 4`).

GOOD JOB! ✓

Loop	Output	Explanation

`n = 2`
`while n < 10 :`
`  print(n)`
`  n = 2 * n`
`print(n)`
	2 4 8 16	Note that 16 is printed *after* the loop has completed.

`n = 2`
`while n < 10 :`
`  n = 2 * n`
`  print(n)`
`print(n)`
	4 8 16 16	In the third iteration, `n` is set to 16, and that value is printed. It is printed again after the loop has completed.

`n = 1`
`while n < 100 :`
`  print(n)`
`  n = 10 * n`
	1 10	`n` is set to 100 in the third loop iteration. The loop exits because the condition `n < 100` is no longer true.

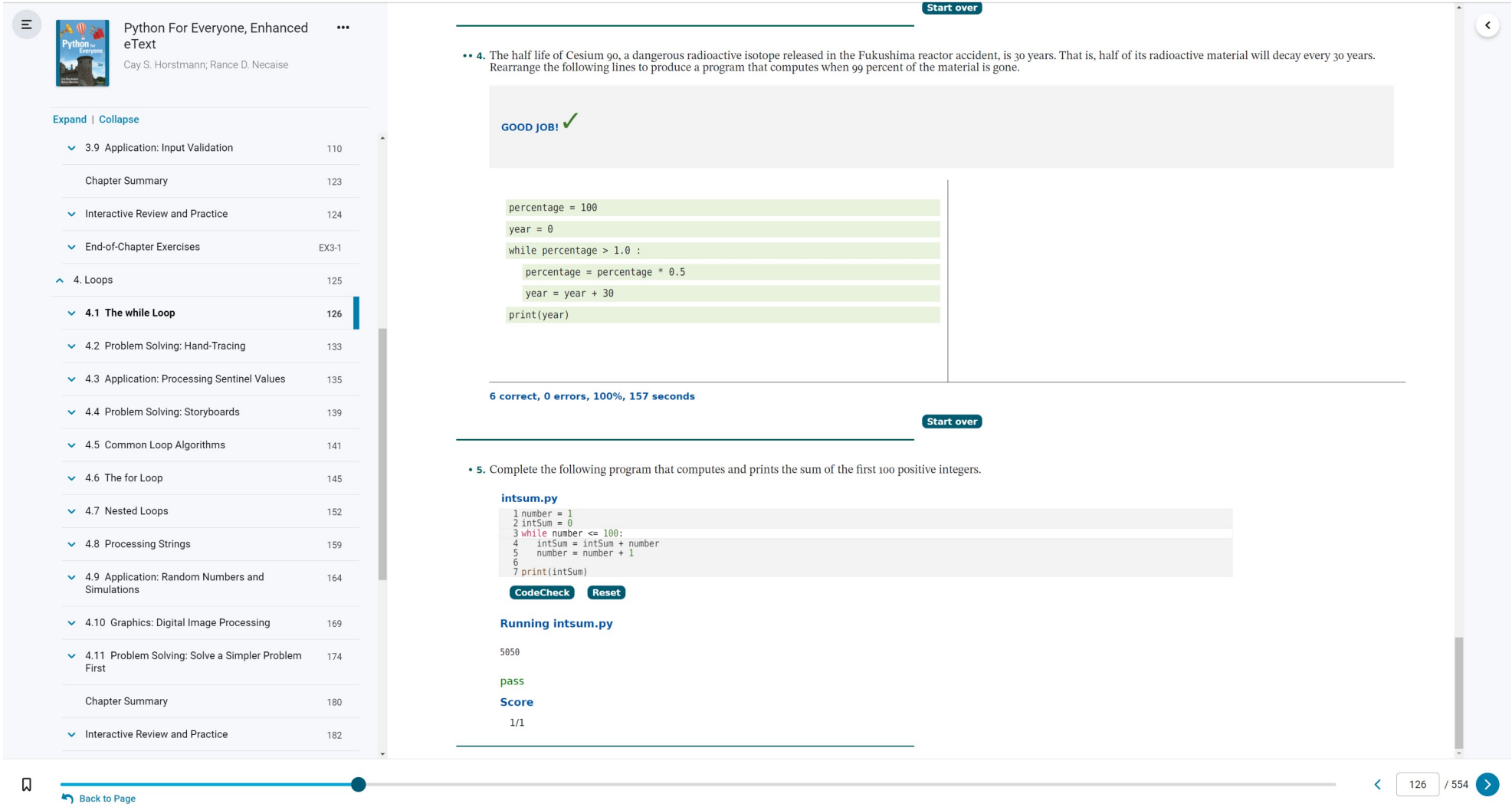
`n = 1`
`while n <= 100 :`
`  print(n)`
`  n = 10 * n`
	1 10 100	`n` is set to 100 in the third loop iteration. The loop executes one more time because the condition `n <= 100` is still true. When `n` is set to 1000, the loop exits.

`n = 1`
`while n >= 100 :`
`  print(n)`
`  n = 10 * n`
`print(n)`
	1	This loop is never entered because the condition `n >= 100` is false at the outset. Thus, `n` still has a value of 1 when it is printed.

5 correct, 0 errors, 100%, 31 seconds

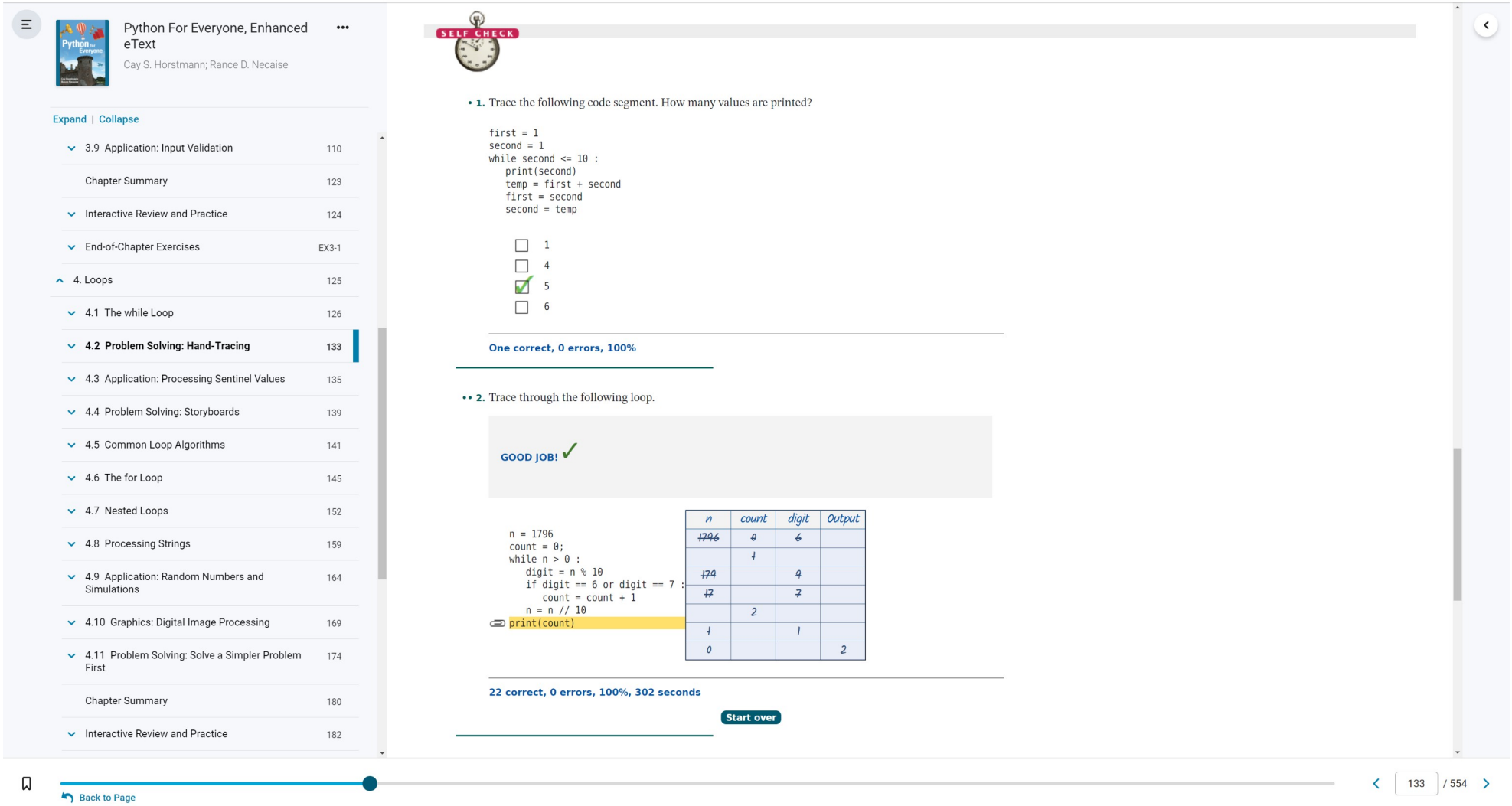
Start over

•• **4.** The half life of Cesium 90, a dangerous radioactive isotope released in the Fukushima reactor accident, is 30 years. That is, half of its radioactive material will decay every 30 years. Rearrange the following lines to produce a program that computes when 99 percent of the material is gone.

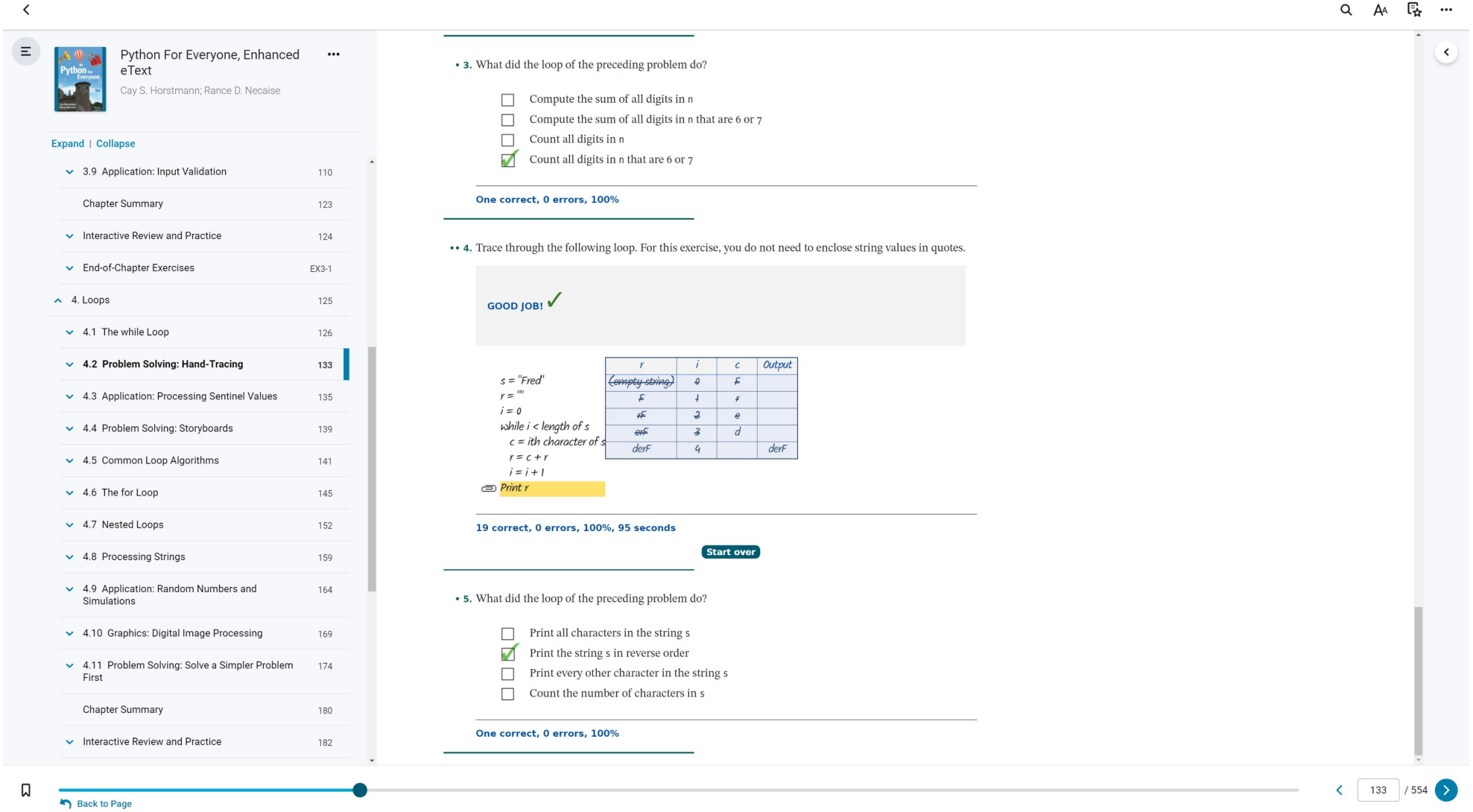

Python For Everyone, Enhanced eText
Cay S. Horstmann; Rance D. Necaise
Expand | Collapse
3.9 Application: Input Validation 110
Chapter Summary 123
Interactive Review and Practice 124
End-of-Chapter Exercises EX3-1
4. Loops 125
4.1 The while Loop 126
4.2 Problem Solving: Hand-Tracing 133
4.3 Application: Processing Sentinel Values 135
4.4 Problem Solving: Storyboards 139
4.5 Common Loop Algorithms 141
4.6 The for Loop 145
4.7 Nested Loops 152
4.8 Processing Strings 159
4.9 Application: Random Numbers and Simulations 164
4.10 Graphics: Digital Image Processing 169
4.11 Problem Solving: Solve a Simpler Problem First 174
Chapter Summary 180
Interactive Review and Practice 182
Start over
•• 4. The half life of Cesium 90, a dangerous radioactive isotope released in the Fukushima reactor accident, is 30 years. That is, half of its radioactive material will decay every 30 years. Rearrange the following lines to produce a program that computes when 99 percent of the material is gone.
GOOD JOB!
percentage = 100
year = 0
while percentage > 1.0 :
percentage = percentage * 0.5
year = year + 30
print(year)
6 correct, 0 errors, 100%, 157 seconds
Start over
• 5. Complete the following program that computes and prints the sum of the first 100 positive integers.
intsum.py
1 number = 1
2 intSum = 0
3 while number <= 100:
4 intSum = intSum + number
5 number = number + 1
6
7 print(intSum)
CodeCheck Reset
Running intsum.py
5050
pass
Score
1/1
Back to Page
126 / 554

4.1 (p.3)

Albert Nguyen
CS131-47853 MW 06:00 PM – 08:30 PM
Chapter 4 Homework – 4.1 – 4.9, 4.11



4.2 (p.1)



• **3.** What did the loop of the preceding problem do?

- ☐ Compute the sum of all digits in n
- ☐ Compute the sum of all digits in n that are 6 or 7
- ☐ Count all digits in n
- ☑ Count all digits in n that are 6 or 7

One correct, 0 errors, 100%

•• **4.** Trace through the following loop. For this exercise, you do not need to enclose string values in quotes.

GOOD JOB! ✓

```
s = "Fred"
r = ""
i = 0
while i < length of s
   c = ith character of s
   r = c + r
   i = i + 1
Print r
```

r	i	c	Output
~~(empty string)~~	~~0~~	~~F~~	
~~F~~	~~1~~	~~r~~	
~~rF~~	~~2~~	~~e~~	
~~erF~~	~~3~~	d	
derF	4		derF

19 correct, 0 errors, 100%, 95 seconds

Start over

• **5.** What did the loop of the preceding problem do?

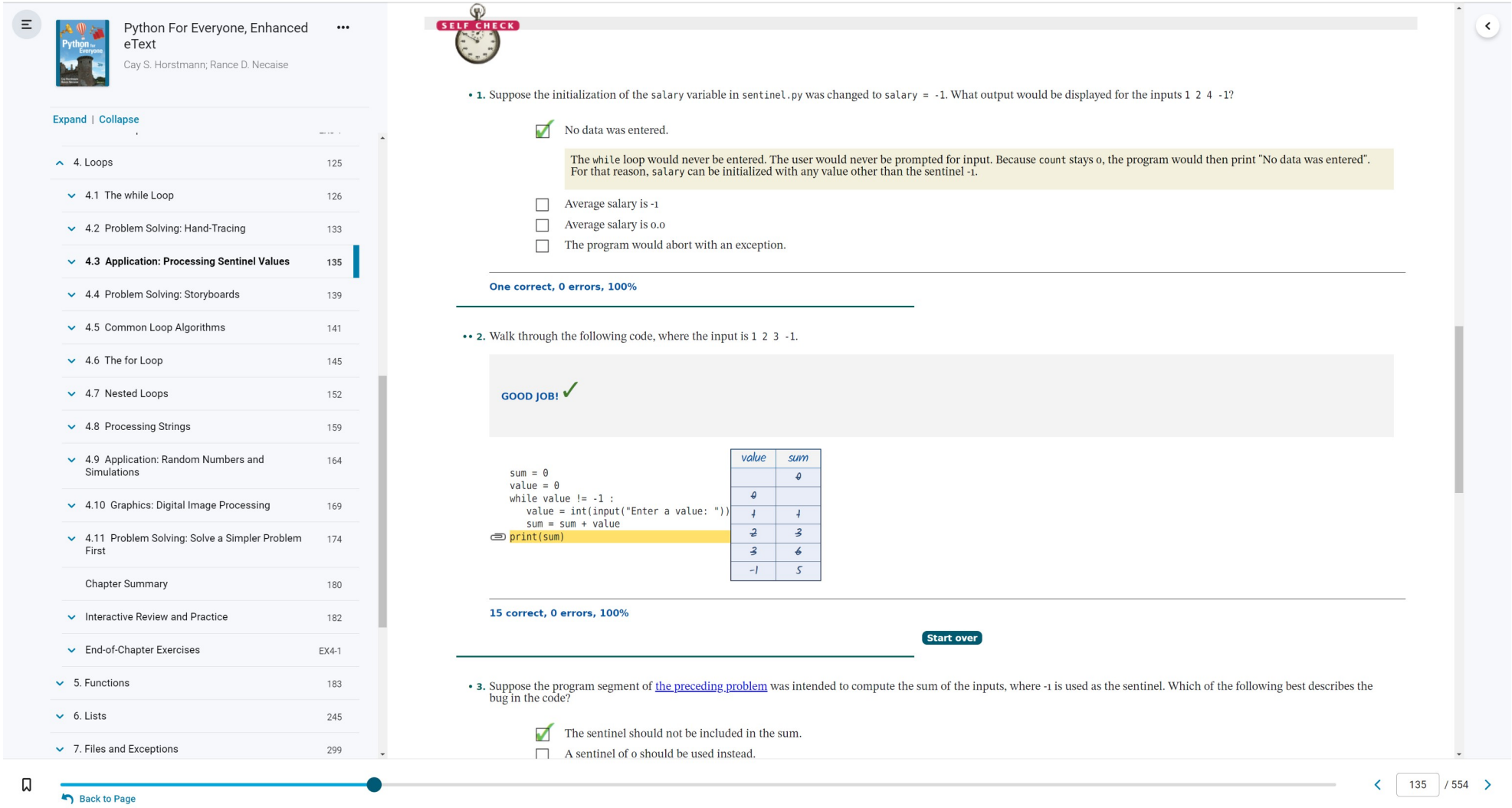
- ☐ Print all characters in the string s
- ☑ Print the string s in reverse order
- ☐ Print every other character in the string s
- ☐ Count the number of characters in s

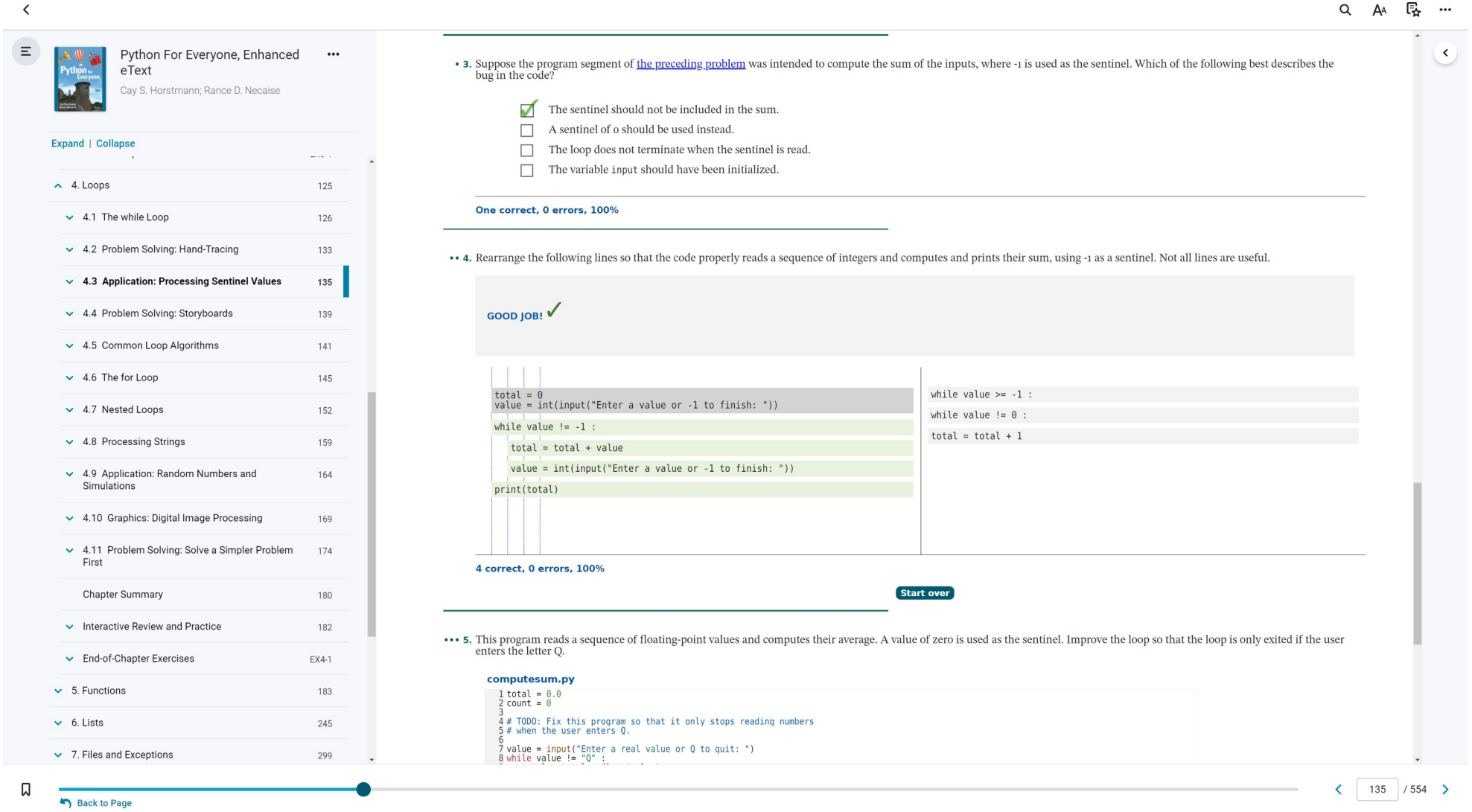
One correct, 0 errors, 100%

Back to Page

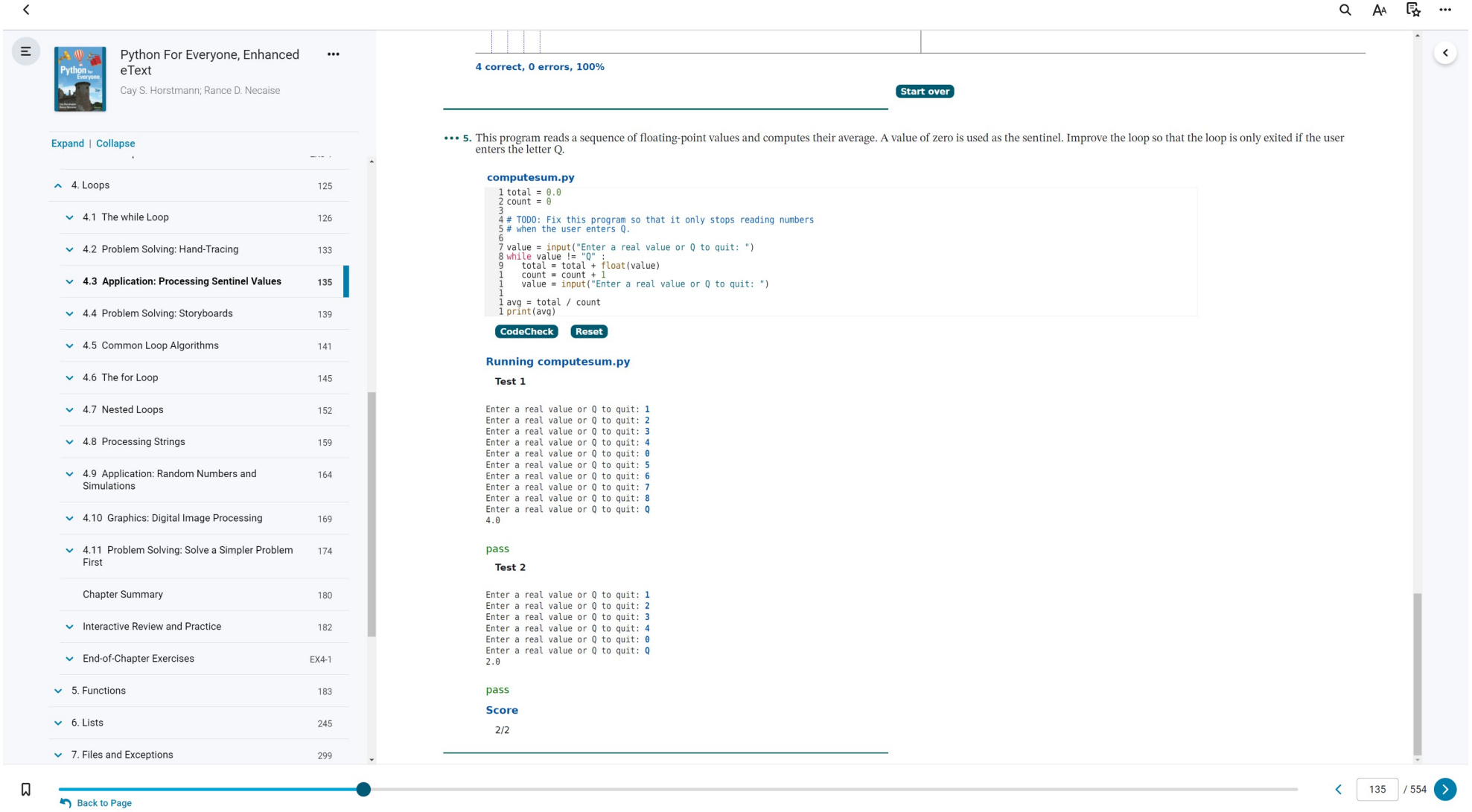
133 / 554

Albert Nguyen
CS131-47853 MW 06:00 PM – 08:30 PM
Chapter 4 Homework – 4.1 – 4.9, 4.11

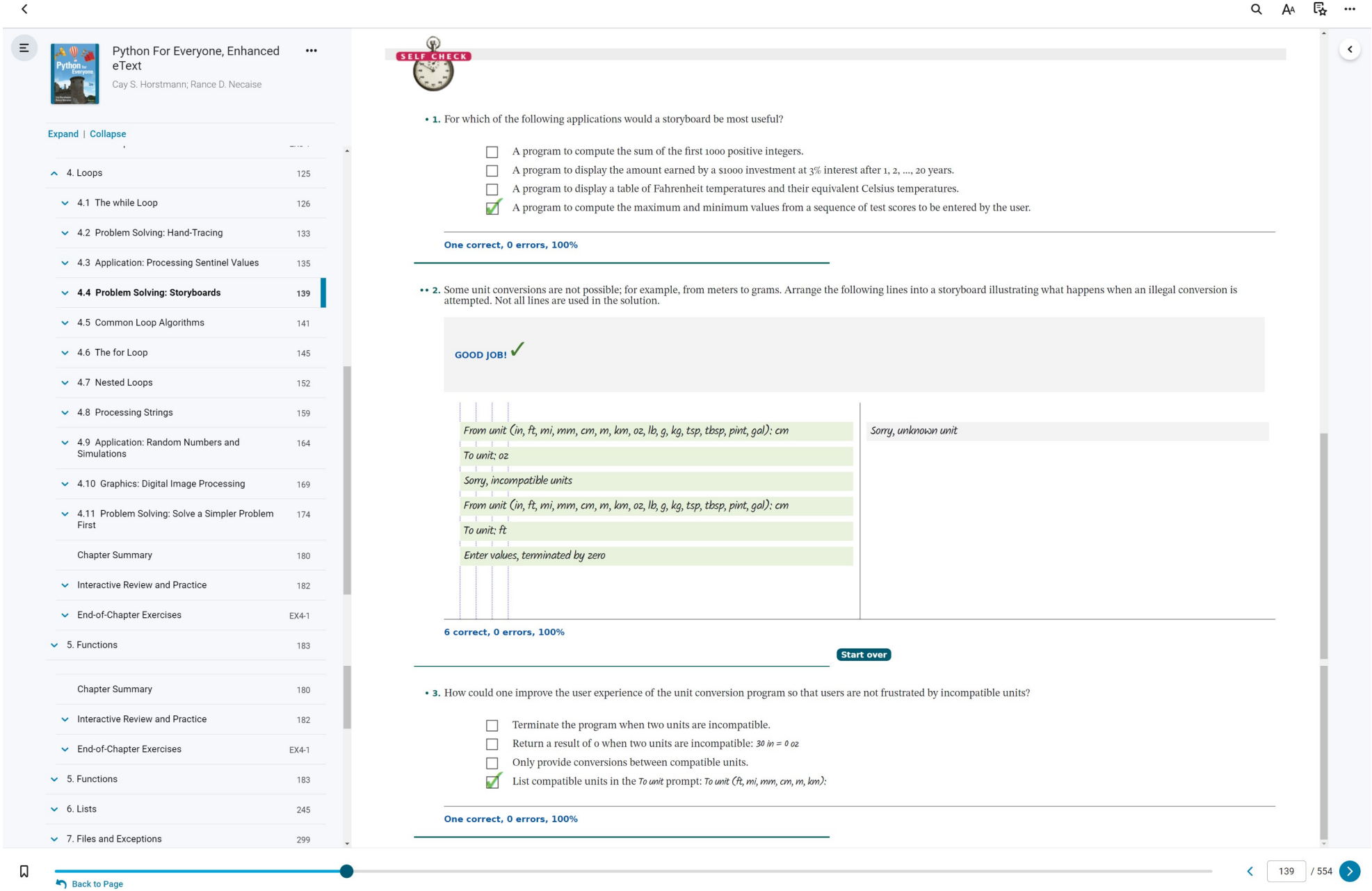



Python For Everyone, Enhanced eText
Cay S. Horstmann; Rance D. Necaise
Expand | Collapse
4. Loops 125
4.1 The while Loop 126
4.2 Problem Solving: Hand-Tracing 133
4.3 Application: Processing Sentinel Values 135
4.4 Problem Solving: Storyboards 139
4.5 Common Loop Algorithms 141
4.6 The for Loop 145
4.7 Nested Loops 152
4.8 Processing Strings 159
4.9 Application: Random Numbers and Simulations 164
4.10 Graphics: Digital Image Processing 169
4.11 Problem Solving: Solve a Simpler Problem First 174
Chapter Summary 180
Interactive Review and Practice 182
End-of-Chapter Exercises EX4-1
5. Functions 183
6. Lists 245
7. Files and Exceptions 299
• 3. Suppose the program segment of the preceding problem was intended to compute the sum of the inputs, where -1 is used as the sentinel. Which of the following best describes the bug in the code?
The sentinel should not be included in the sum.
A sentinel of 0 should be used instead.
The loop does not terminate when the sentinel is read.
The variable input should have been initialized.
One correct, 0 errors, 100%
•• 4. Rearrange the following lines so that the code properly reads a sequence of integers and computes and prints their sum, using -1 as a sentinel. Not all lines are useful.
GOOD JOB!
total = 0
value = int(input("Enter a value or -1 to finish: "))
while value != -1 :
total = total + value
value = int(input("Enter a value or -1 to finish: "))
print(total)
while value >= -1 :
while value != 0 :
total = total + 1
4 correct, 0 errors, 100%
Start over
••• 5. This program reads a sequence of floating-point values and computes their average. A value of zero is used as the sentinel. Improve the loop so that the loop is only exited if the user enters the letter Q.
computesum.py
1 total = 0.0
2 count = 0
3
4 # TODO: Fix this program so that it only stops reading numbers
5 # when the user enters Q.
6
7 value = input("Enter a real value or Q to quit: ")
8 while value != "Q" :
Back to Page
135 / 554

4.3 (p.2)

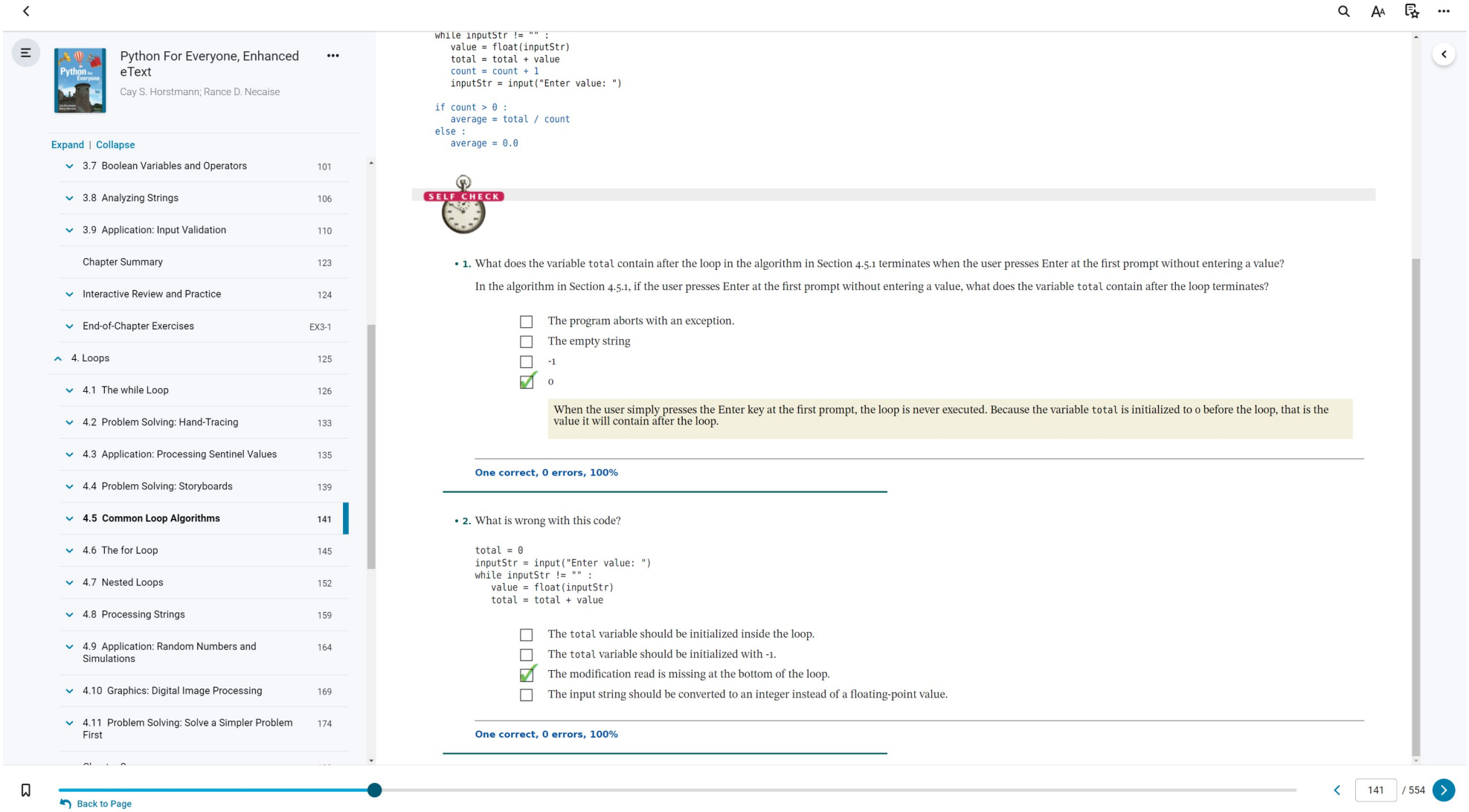

Python For Everyone, Enhanced eText
Cay S. Horstmann; Rance D. Necaise
Expand | Collapse
4. Loops 125
4.1 The while Loop 126
4.2 Problem Solving: Hand-Tracing 133
4.3 Application: Processing Sentinel Values 135
4.4 Problem Solving: Storyboards 139
4.5 Common Loop Algorithms 141
4.6 The for Loop 145
4.7 Nested Loops 152
4.8 Processing Strings 159
4.9 Application: Random Numbers and Simulations 164
4.10 Graphics: Digital Image Processing 169
4.11 Problem Solving: Solve a Simpler Problem First 174
Chapter Summary 180
Interactive Review and Practice 182
End-of-Chapter Exercises EX4-1
5. Functions 183
6. Lists 245
7. Files and Exceptions 299
4 correct, 0 errors, 100%
Start over
5. This program reads a sequence of floating-point values and computes their average. A value of zero is used as the sentinel. Improve the loop so that the loop is only exited if the user enters the letter Q.
computesum.py
total = 0.0
count = 0
# TODO: Fix this program so that it only stops reading numbers
# when the user enters Q.
value = input("Enter a real value or Q to quit: ")
while value != "Q" :
total = total + float(value)
count = count + 1
value = input("Enter a real value or Q to quit: ")
avg = total / count
print(avg)
CodeCheck
Reset
Running computesum.py
Test 1
Enter a real value or Q to quit: 1
Enter a real value or Q to quit: 2
Enter a real value or Q to quit: 3
Enter a real value or Q to quit: 4
Enter a real value or Q to quit: 0
Enter a real value or Q to quit: 5
Enter a real value or Q to quit: 6
Enter a real value or Q to quit: 7
Enter a real value or Q to quit: 8
Enter a real value or Q to quit: Q
4.0
pass
Test 2
Enter a real value or Q to quit: 1
Enter a real value or Q to quit: 2
Enter a real value or Q to quit: 3
Enter a real value or Q to quit: 4
Enter a real value or Q to quit: 0
Enter a real value or Q to quit: Q
2.0
pass
Score
2/2
Back to Page
135 / 554

4.3 (p.3)


Python For Everyone, Enhanced eText
Cay S. Horstmann; Rance D. Necaise
Expand | Collapse
4. Loops 125
4.1 The while Loop 126
4.2 Problem Solving: Hand-Tracing 133
4.3 Application: Processing Sentinel Values 135
4.4 Problem Solving: Storyboards 139
4.5 Common Loop Algorithms 141
4.6 The for Loop 145
4.7 Nested Loops 152
4.8 Processing Strings 159
4.9 Application: Random Numbers and Simulations 164
4.10 Graphics: Digital Image Processing 169
4.11 Problem Solving: Solve a Simpler Problem First 174
Chapter Summary 180
Interactive Review and Practice 182
End-of-Chapter Exercises EX4-1
5. Functions 183
Chapter Summary 180
Interactive Review and Practice 182
End-of-Chapter Exercises EX4-1
5. Functions 183
6. Lists 245
7. Files and Exceptions 299
SELF CHECK
• 1. For which of the following applications would a storyboard be most useful?
A program to compute the sum of the first 1000 positive integers.
A program to display the amount earned by a $1000 investment at 3% interest after 1, 2, ..., 20 years.
A program to display a table of Fahrenheit temperatures and their equivalent Celsius temperatures.
A program to compute the maximum and minimum values from a sequence of test scores to be entered by the user.
One correct, 0 errors, 100%
•• 2. Some unit conversions are not possible; for example, from meters to grams. Arrange the following lines into a storyboard illustrating what happens when an illegal conversion is attempted. Not all lines are used in the solution.
GOOD JOB!
From unit (in, ft, mi, mm, cm, m, km, oz, lb, g, kg, tsp, tbsp, pint, gal): cm
To unit: oz
Sorry, incompatible units
From unit (in, ft, mi, mm, cm, m, km, oz, lb, g, kg, tsp, tbsp, pint, gal): cm
To unit: ft
Enter values, terminated by zero
Sorry, unknown unit
6 correct, 0 errors, 100%
Start over
• 3. How could one improve the user experience of the unit conversion program so that users are not frustrated by incompatible units?
Terminate the program when two units are incompatible.
Return a result of 0 when two units are incompatible: 30 in = 0 oz
Only provide conversions between compatible units.
List compatible units in the To unit prompt: To unit (ft, mi, mm, cm, m, km):
One correct, 0 errors, 100%
Back to Page
139 / 554

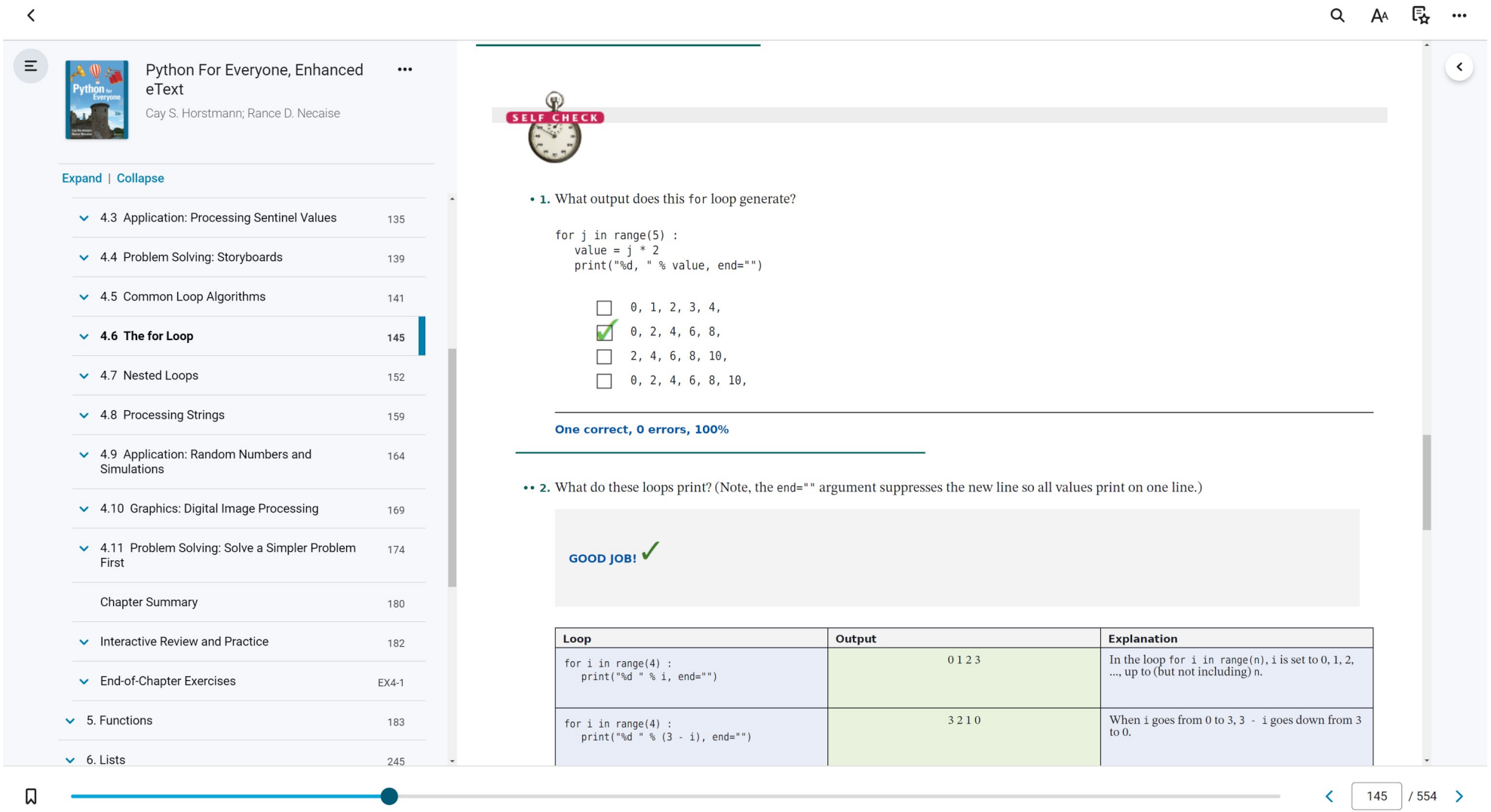
4.4

Albert Nguyen
CS131-47853 MW 06:00 PM – 08:30 PM
Chapter 4 Homework – 4.1 – 4.9, 4.11



4.5

Albert Nguyen
CS131-47853 MW 06:00 PM – 08:30 PM
Chapter 4 Homework – 4.1 – 4.9, 4.11



•• 2. What do these loops print? (Note, the `end=""` argument suppresses the new line so all values print on one line.)

GOOD JOB! ✓

Loop	Output	Explanation
`for i in range(4) :` `  print("%d " % i, end="")`	0 1 2 3	In the loop `for i in range(n)`, `i` is set to 0, 1, 2, ..., up to (but not including) `n`.
`for i in range(4) :` `  print("%d " % (3 - i), end="")`	3 2 1 0	When `i` goes from 0 to 3, `3 - i` goes down from 3 to 0.

`name = "Fred"`
`for ch in name :`
`  print(ch, end="")`
	Fred	The `for` loop is used to iterate over containers, one element at a time.

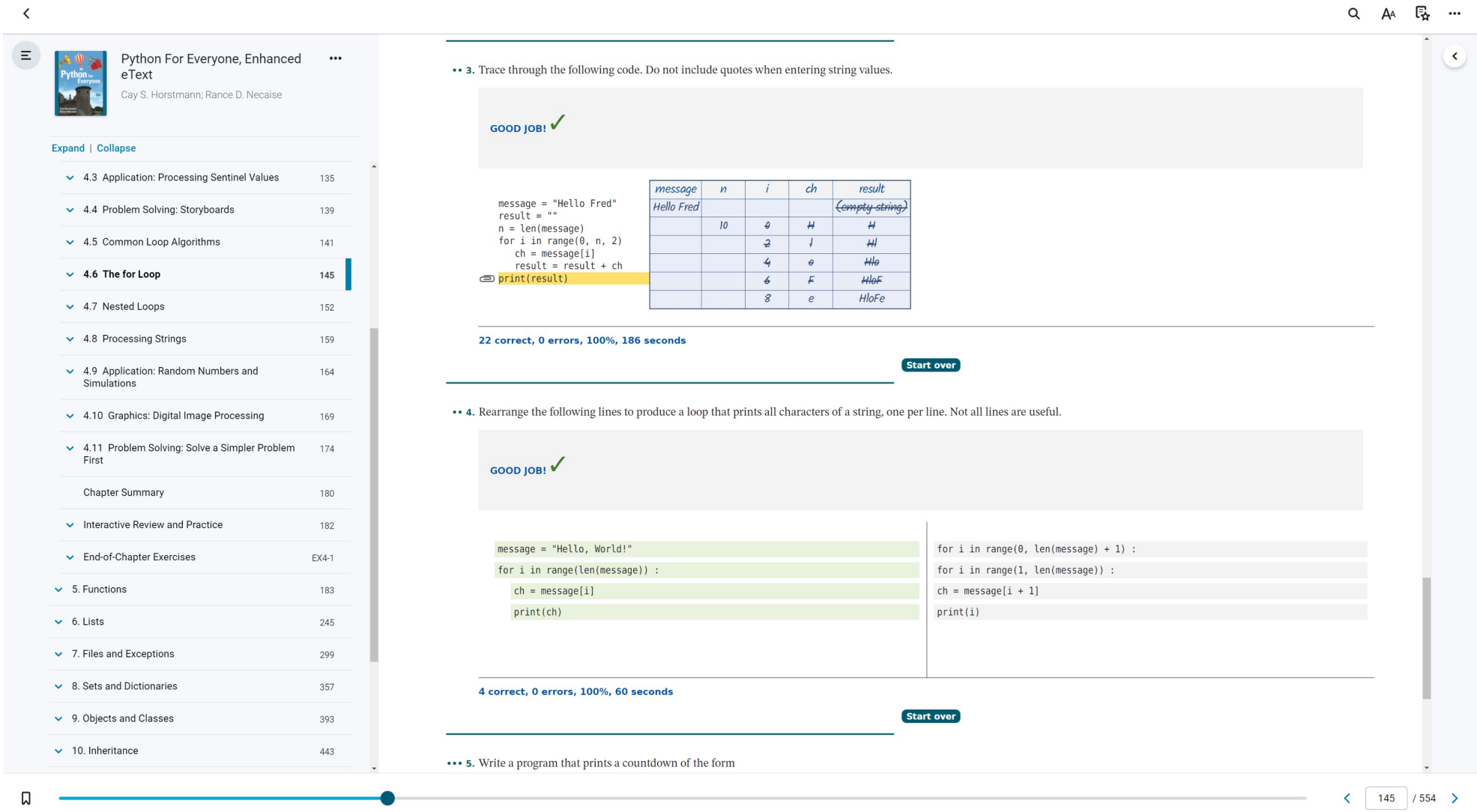
`name = "Fred"`
`for i in range(len(name) - 1, -1, -1) :`
`  print(name[i], end="")`
	derF	This is one way of traversing the characters backwards.

`for i in range(0, 8, 2) :` `  print("%d " % i, end="")`	0 2 4 6	When passing three arguments to the `range` function, the third argument is the step value. This is how you implement a step size of 2.
`for i in range(3, 0, -1) :` `  print("%d " % i, end="")`	3 2 1	This is how you step backwards.

6 correct, 0 errors, 100%, 225 seconds

Start over

•• 3. Trace through the following code. Do not include quotes when entering string values.



Python For Everyone, Enhanced eText
Cay S. Horstmann; Rance D. Necaise

Expand | Collapse

•• 3. Trace through the following code. Do not include quotes when entering string values.

GOOD JOB! ✓

```
message = "Hello Fred"
result = ""
n = len(message)
for i in range(0, n, 2)
   ch = message[i]
   result = result + ch
print(result)
```

message	n	i	ch	result
Hello Fred				~~(empty string)~~
	10	~~0~~	~~H~~	~~H~~
		~~2~~	~~l~~	~~Hl~~
		~~4~~	~~o~~	~~Hlo~~
		~~6~~	~~F~~	~~HloF~~
		8	e	HloFe

22 correct, 0 errors, 100%, 186 seconds

Start over

•• 4. Rearrange the following lines to produce a loop that prints all characters of a string, one per line. Not all lines are useful.

GOOD JOB! ✓

```
message = "Hello, World!"
for i in range(len(message)) :
   ch = message[i]
   print(ch)
```

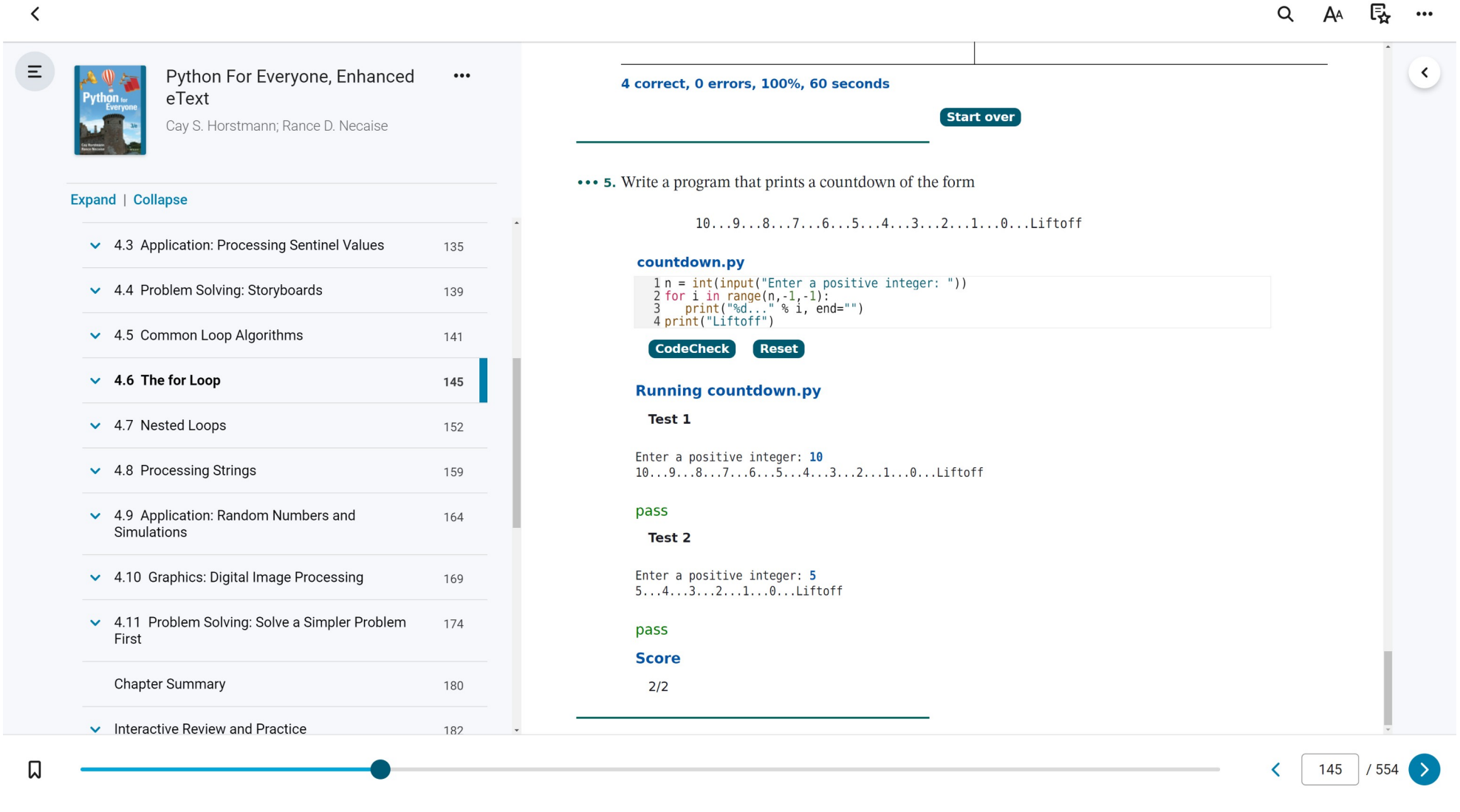
```
for i in range(0, len(message) + 1) :
for i in range(1, len(message)) :
ch = message[i + 1]
print(i)
```

4 correct, 0 errors, 100%, 60 seconds

Start over

••• 5. Write a program that prints a countdown of the form

145 / 554



Python For Everyone, Enhanced eText

Cay S. Horstmann; Rance D. Necaise

Expand | Collapse

4 correct, 0 errors, 100%, 60 seconds

Start over

••• 5. Write a program that prints a countdown of the form

```
10...9...8...7...6...5...4...3...2...1...0...Liftoff
```

countdown.py

```
1 n = int(input("Enter a positive integer: "))
2 for i in range(n,-1,-1):
3    print("%d..." % i, end="")
4 print("Liftoff")
```

CodeCheck Reset

Running countdown.py

Test 1

```
Enter a positive integer: 10
10...9...8...7...6...5...4...3...2...1...0...Liftoff
```

pass

Test 2

```
Enter a positive integer: 5
5...4...3...2...1...0...Liftoff
```

pass

Score

2/2

Albert Nguyen
CS131-47853 MW 06:00 PM – 08:30 PM
Chapter 4 Homework – 4.1 – 4.9, 4.11

SELF CHECK

• 1. What output do these nested for loops generate?

```
for j in range(1, 4) :
   for k in range(2, 4) :
      print("%d, " % k, end="")
```

- [ ] 1, 2, 3, 1, 2, 3,
- [ ] 2, 3, 4, 2, 3, 4,
- [ ] 2, 3, 2, 3,
- [x] 2, 3, 2, 3, 2, 3,

One correct, 0 errors, 100%

•• 2. Trace through the following code:

GOOD JOB! ✓

```
for n in range(2, 4) :
   p = 1
   for i in range(1, 4) :
      p = p * n
      print("%2d" % p, end="")
   print()
print()
```

n	i	p
~~2~~		~~1~~
	~~1~~	~~2~~
	~~2~~	~~4~~
	~~3~~	~~8~~
3		~~1~~
	~~1~~	~~3~~
	~~2~~	~~9~~
	3	27

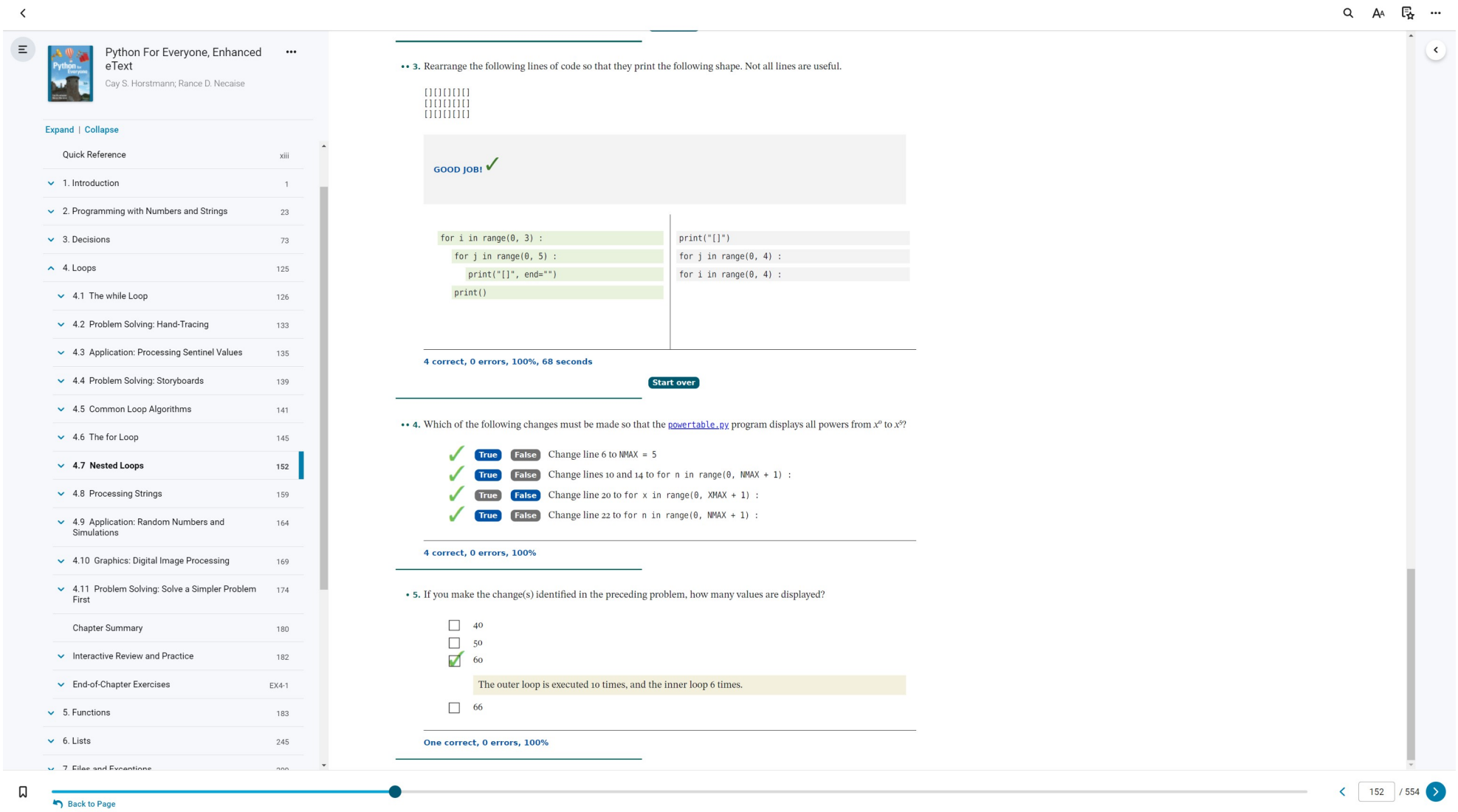
18 correct, 0 errors, 100%, 130 seconds

Start over

•• 3. Rearrange the following lines of code so that they print the following shape. Not all lines are useful.

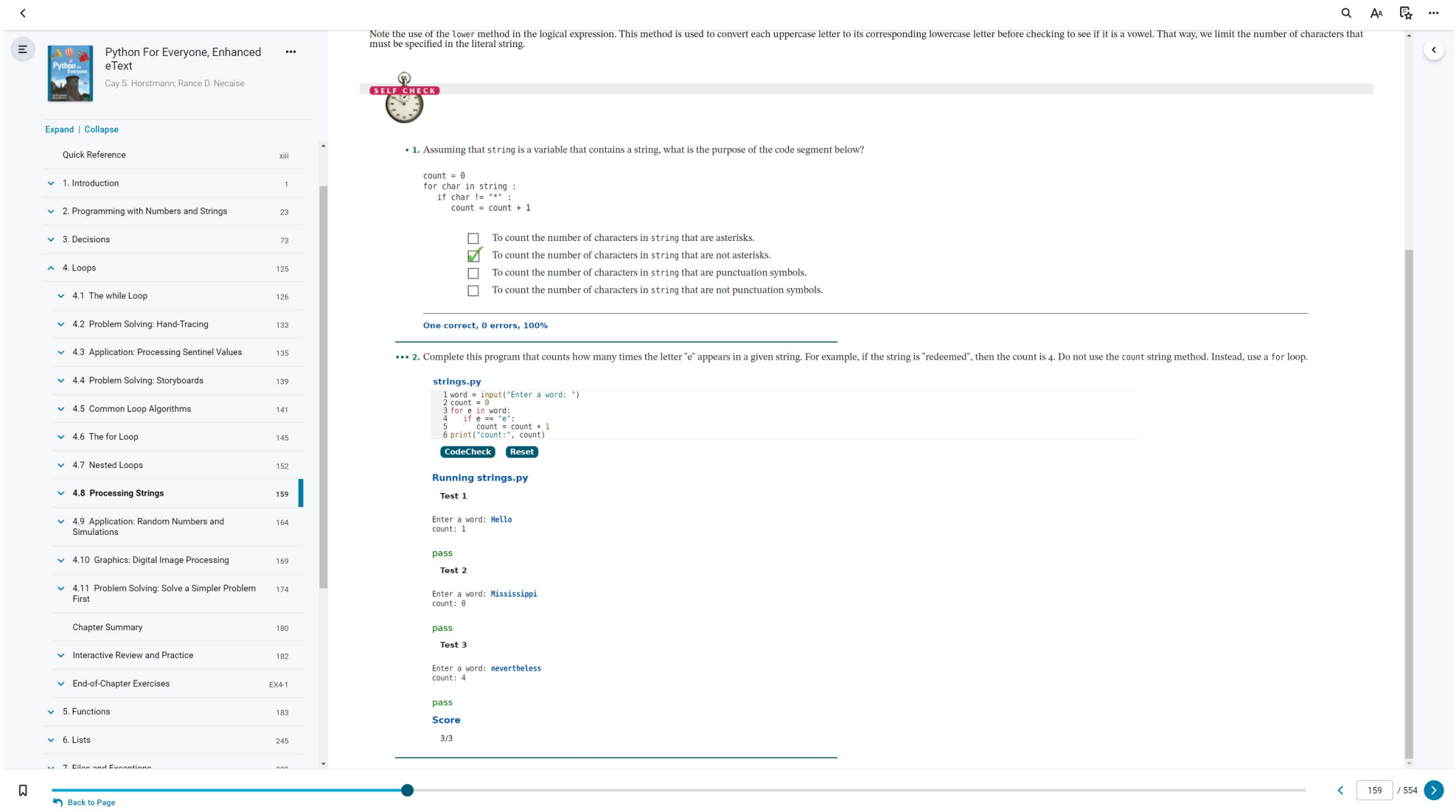
```
[][][][][]
[][][][][]
[][][][][]
```


4.7 (p.1)

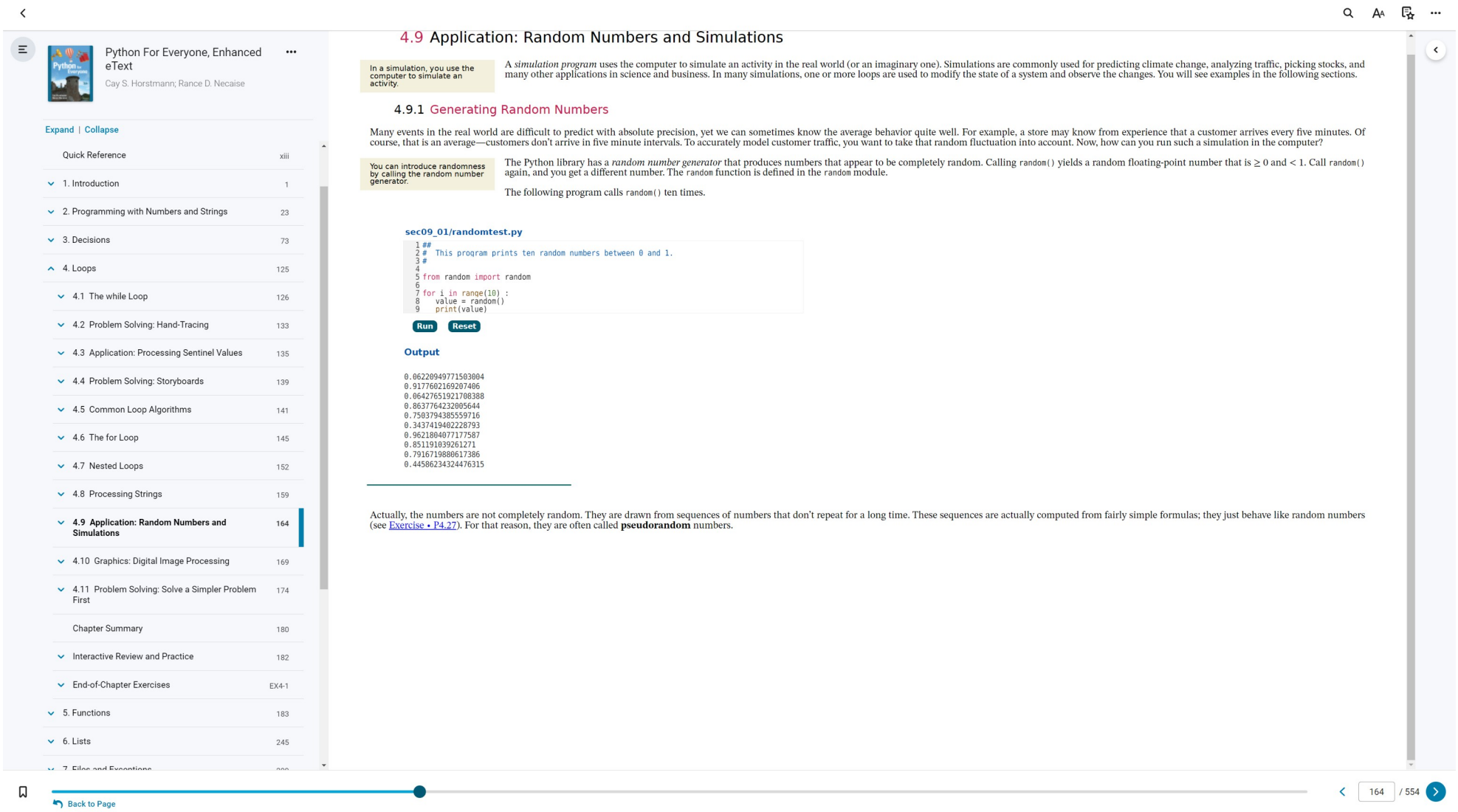

Python For Everyone, Enhanced eText
Cay S. Horstmann; Rance D. Necaise
Expand | Collapse
Quick Reference xiii
1. Introduction 1
2. Programming with Numbers and Strings 23
3. Decisions 73
4. Loops 125
4.1 The while Loop 126
4.2 Problem Solving: Hand-Tracing 133
4.3 Application: Processing Sentinel Values 135
4.4 Problem Solving: Storyboards 139
4.5 Common Loop Algorithms 141
4.6 The for Loop 145
4.7 Nested Loops 152
4.8 Processing Strings 159
4.9 Application: Random Numbers and Simulations 164
4.10 Graphics: Digital Image Processing 169
4.11 Problem Solving: Solve a Simpler Problem First 174
Chapter Summary 180
Interactive Review and Practice 182
End-of-Chapter Exercises EX4-1
5. Functions 183
6. Lists 245
•• 3. Rearrange the following lines of code so that they print the following shape. Not all lines are useful.
[][][][][]
[][][][][]
[][][][][]
GOOD JOB!
for i in range(0, 3) :
for j in range(0, 5) :
print("[]", end="")
print()
print("[]")
for j in range(0, 4) :
for i in range(0, 4) :
4 correct, 0 errors, 100%, 68 seconds
Start over
•• 4. Which of the following changes must be made so that the powertable.py program displays all powers from x^0 to x^5?
True False Change line 6 to NMAX = 5
True False Change lines 10 and 14 to for n in range(0, NMAX + 1) :
True False Change line 20 to for x in range(0, XMAX + 1) :
True False Change line 22 to for n in range(0, NMAX + 1) :
4 correct, 0 errors, 100%
• 5. If you make the change(s) identified in the preceding problem, how many values are displayed?
40
50
60
The outer loop is executed 10 times, and the inner loop 6 times.
66
One correct, 0 errors, 100%
Back to Page
152 / 554

4.7 (p.2)

Albert Nguyen
CS131-47853 MW 06:00 PM – 08:30 PM
Chapter 4 Homework – 4.1 – 4.9, 4.11



Albert Nguyen
CS131-47853 MW 06:00 PM – 08:30 PM
Chapter 4 Homework – 4.1 – 4.9, 4.11



Python For Everyone, Enhanced eText
Cay S. Horstmann; Rance D. Necaise

Expand | Collapse

4.9 Application: Random Numbers and Simulations

In a simulation, you use the computer to simulate an activity.

A *simulation program* uses the computer to simulate an activity in the real world (or an imaginary one). Simulations are commonly used for predicting climate change, analyzing traffic, picking stocks, and many other applications in science and business. In many simulations, one or more loops are used to modify the state of a system and observe the changes. You will see examples in the following sections.

4.9.1 Generating Random Numbers

Many events in the real world are difficult to predict with absolute precision, yet we can sometimes know the average behavior quite well. For example, a store may know from experience that a customer arrives every five minutes. Of course, that is an average—customers don't arrive in five minute intervals. To accurately model customer traffic, you want to take that random fluctuation into account. Now, how can you run such a simulation in the computer?

You can introduce randomness by calling the random number generator.

The Python library has a *random number generator* that produces numbers that appear to be completely random. Calling `random()` yields a random floating-point number that is ≥ 0 and < 1. Call `random()` again, and you get a different number. The `random` function is defined in the `random` module.

The following program calls `random()` ten times.

sec09_01/randomtest.py

```
##
#  This program prints ten random numbers between 0 and 1.
#

from random import random

for i in range(10) :
   value = random()
   print(value)
```

Run Reset

Output

```
0.062209497715030004
0.9177602169207406
0.06427651921708388
0.8637764232005644
0.7503794385559716
0.3437419402228793
0.9621804077177587
0.851191039261271
0.7916719880617386
0.44586234324476315
```

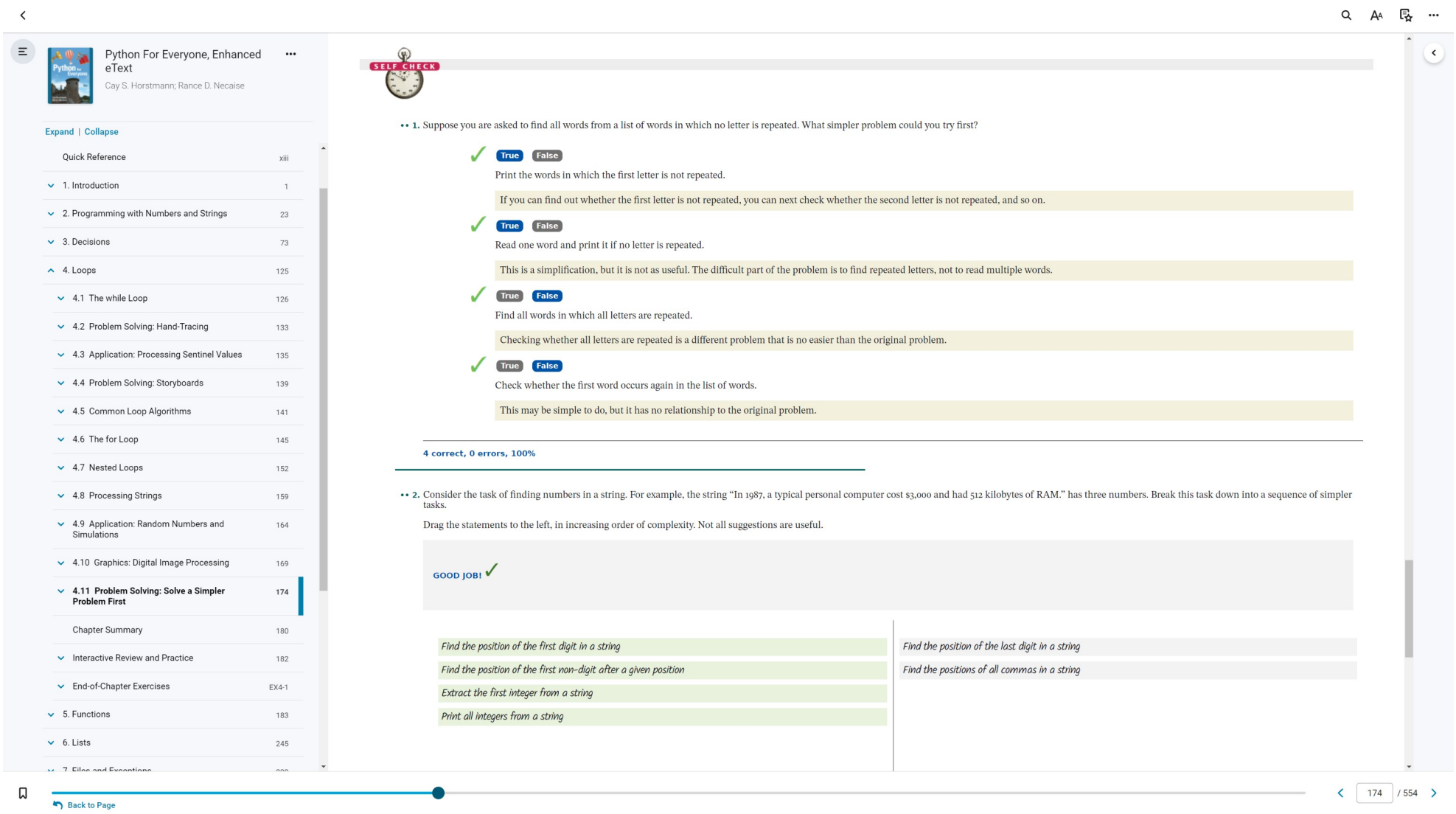
Actually, the numbers are not completely random. They are drawn from sequences of numbers that don't repeat for a long time. These sequences are actually computed from fairly simple formulas; they just behave like random numbers (see Exercise • P4.27). For that reason, they are often called **pseudorandom** numbers.

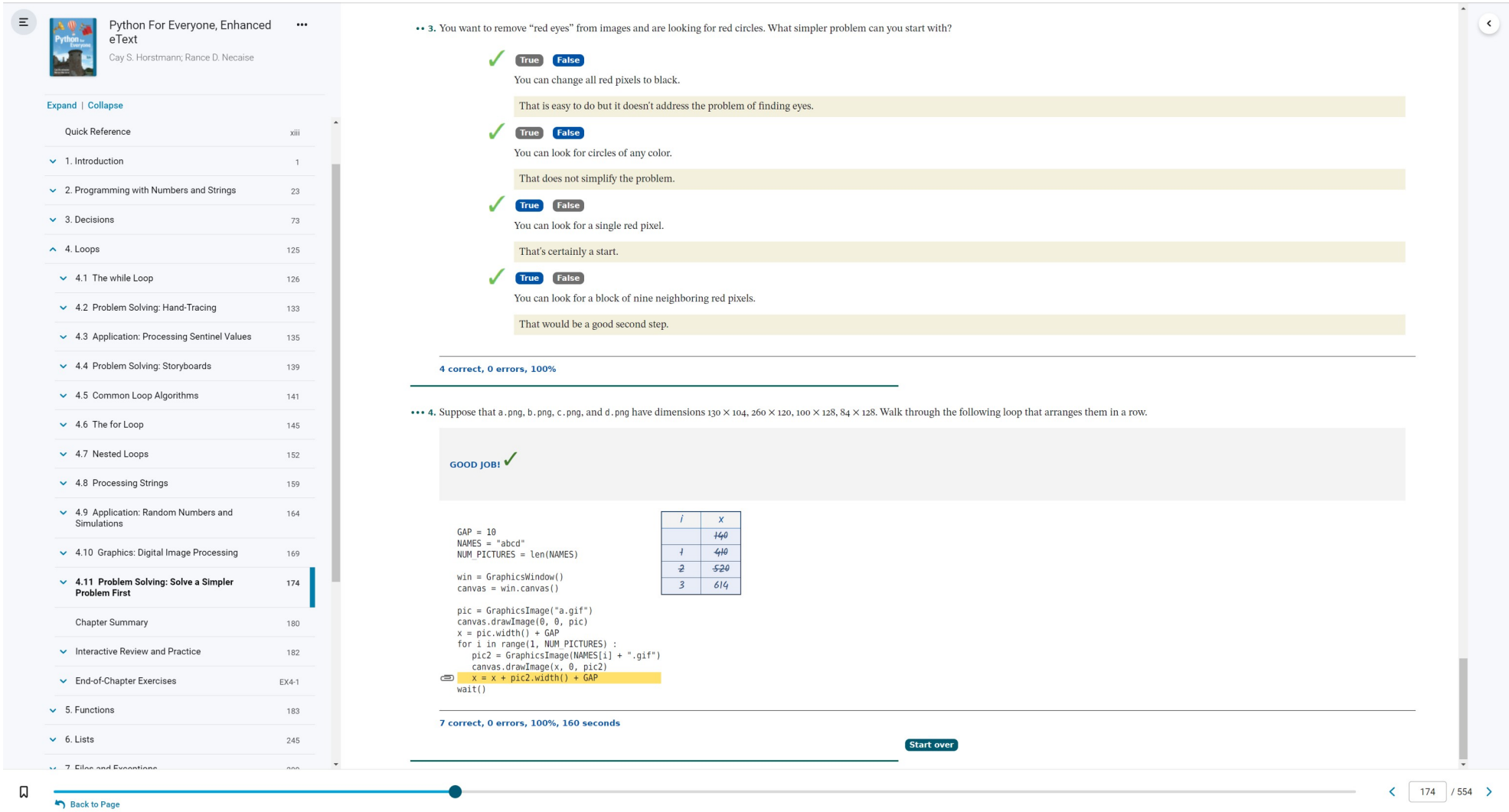
Back to Page

164 / 554

4.9

Albert Nguyen
CS131-47853 MW 06:00 PM – 08:30 PM
Chapter 4 Homework – 4.1 – 4.9, 4.11





Python For Everyone, Enhanced eText
Cay S. Horstmann; Rance D. Necaise

Expand | Collapse

•• 3. You want to remove "red eyes" from images and are looking for red circles. What simpler problem can you start with?

✓ True False
You can change all red pixels to black.
That is easy to do but it doesn't address the problem of finding eyes.

✓ True False
You can look for circles of any color.
That does not simplify the problem.

✓ True False
You can look for a single red pixel.
That's certainly a start.

✓ True False
You can look for a block of nine neighboring red pixels.
That would be a good second step.

4 correct, 0 errors, 100%

••• 4. Suppose that a.png, b.png, c.png, and d.png have dimensions 130 × 104, 260 × 120, 100 × 128, 84 × 128. Walk through the following loop that arranges them in a row.

GOOD JOB! ✓

```
GAP = 10
NAMES = "abcd"
NUM_PICTURES = len(NAMES)

win = GraphicsWindow()
canvas = win.canvas()

pic = GraphicsImage("a.gif")
canvas.drawImage(0, 0, pic)
x = pic.width() + GAP
for i in range(1, NUM_PICTURES) :
   pic2 = GraphicsImage(NAMES[i] + ".gif")
   canvas.drawImage(x, 0, pic2)
   x = x + pic2.width() + GAP
wait()
```

i	x
	~~140~~
~~1~~	~~410~~
~~2~~	~~520~~
3	614

7 correct, 0 errors, 100%, 160 seconds

Start over

Back to Page

174 / 554