can print them.

Could you simply store each value in a separate variable? If you know that there are ten values, then you could store the values in ten variables `value1`, `value2`, `value3`, ..., `value10`. However, such a sequence of variables is not very practical to use. You would have to write quite a bit of code ten times, once for each of the variables. In Python, a **list** is a much better choice for storing a sequence of values.

A list is a container that stores a sequence of values.

Here we create a list and specify the initial values that are to be stored in the new list (see Figure 1):

```
values = [32, 54, 67.5, 29, 35, 80, 115, 44.5, 100, 65] # A list with ten elements
```

The square brackets indicate that we are creating a list. The items are stored in the order they are provided. You will want to store the list in a variable so that you can access it later.

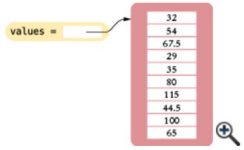


Figure 1 A List of Size 10

SELF CHECK

• **1.** Which of the following statements correctly creates a list with five elements?

- [ ] `grades = {85, 97, 74, 90, 83}`
- [x] `grades = [85, 97, 74, 90, 83]`
- [ ] `grades = (85, 97, 74, 90, 83)`
- [ ] `grades = [5]`

One correct, 0 errors, 100%

• **2.** A list is a ____ that stores a sequence of values.

- [x] container
- [ ] string
- [ ] variable
- [ ] function

One correct, 0 errors, 100%

6.1

Albert Nguyen
CS131-47853 MW 06:00 PM – 08:30 PM
Chapter 6 Homework – 6.1 – 6.7

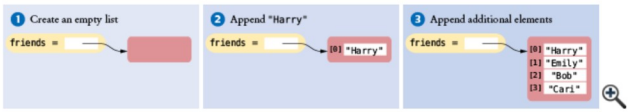


Figure 4 Appending Elements to a List

SELF CHECK

• **1.** Insert the missing statement in the following code fragment, which is designed to initialize the list with the values 1, 2, 3, ..., 100.

```
numbers = []
for k in range(100) :
    ____________
```

- [ ] `numbers = [k]`
- [ ] `numbers.append(k)`
- [ ] `number = [k + 1]`
- [x] `numbers.append(k + 1)`

One correct, 0 errors, 100%

• **2.** Complete the program below that builds and prints a list with the even integer values 2 through 20.

evenvalues.py

```
values = []

# Use a loop to add the even values 2, 4, .., 20 to the values list.
for even in range(1, 21) :
   if even % 2 == 0 :
      values.append(even)
print(values)
```

CodeCheck Reset

Running evenvalues.py

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

pass

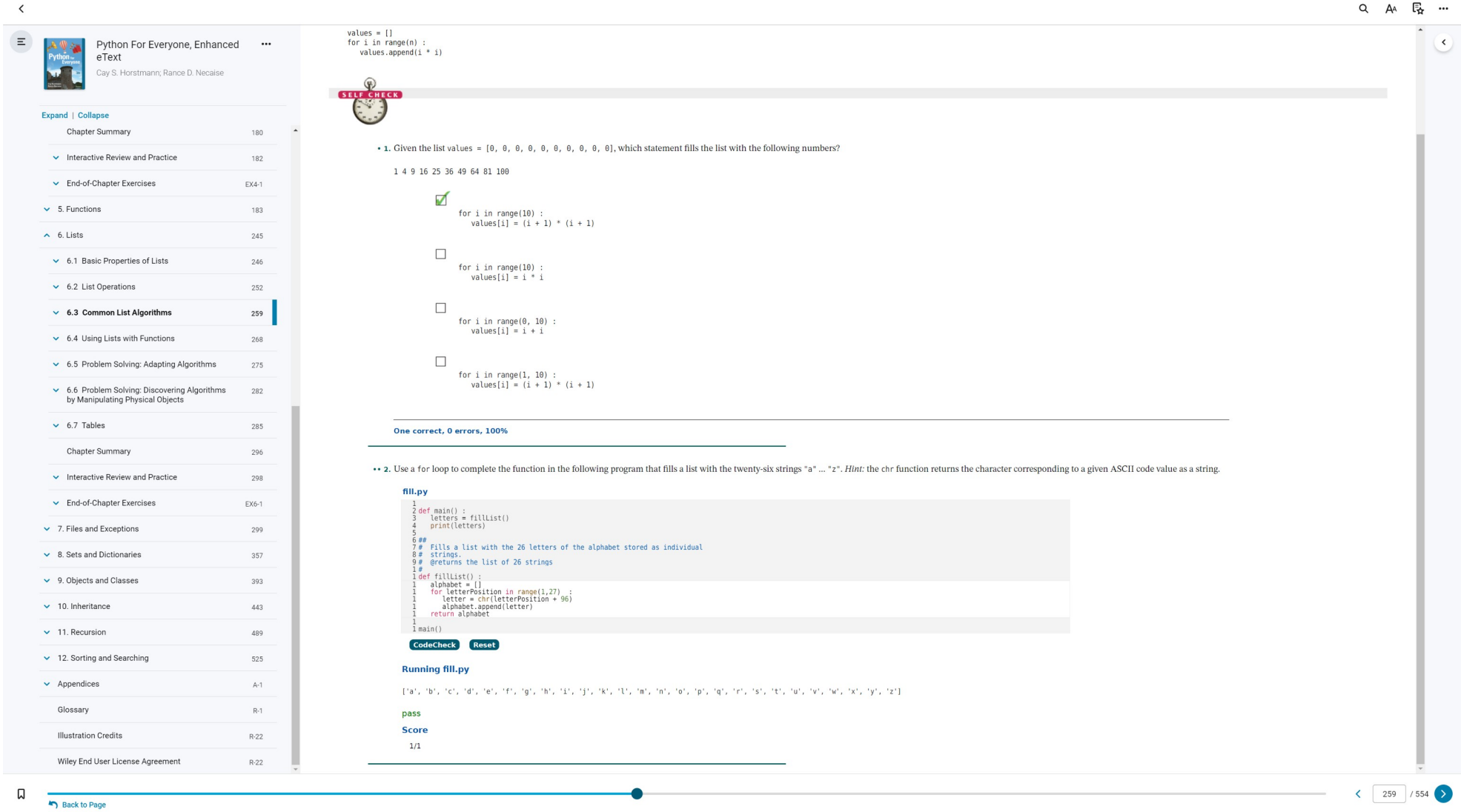
Score

1/1

Back to Page

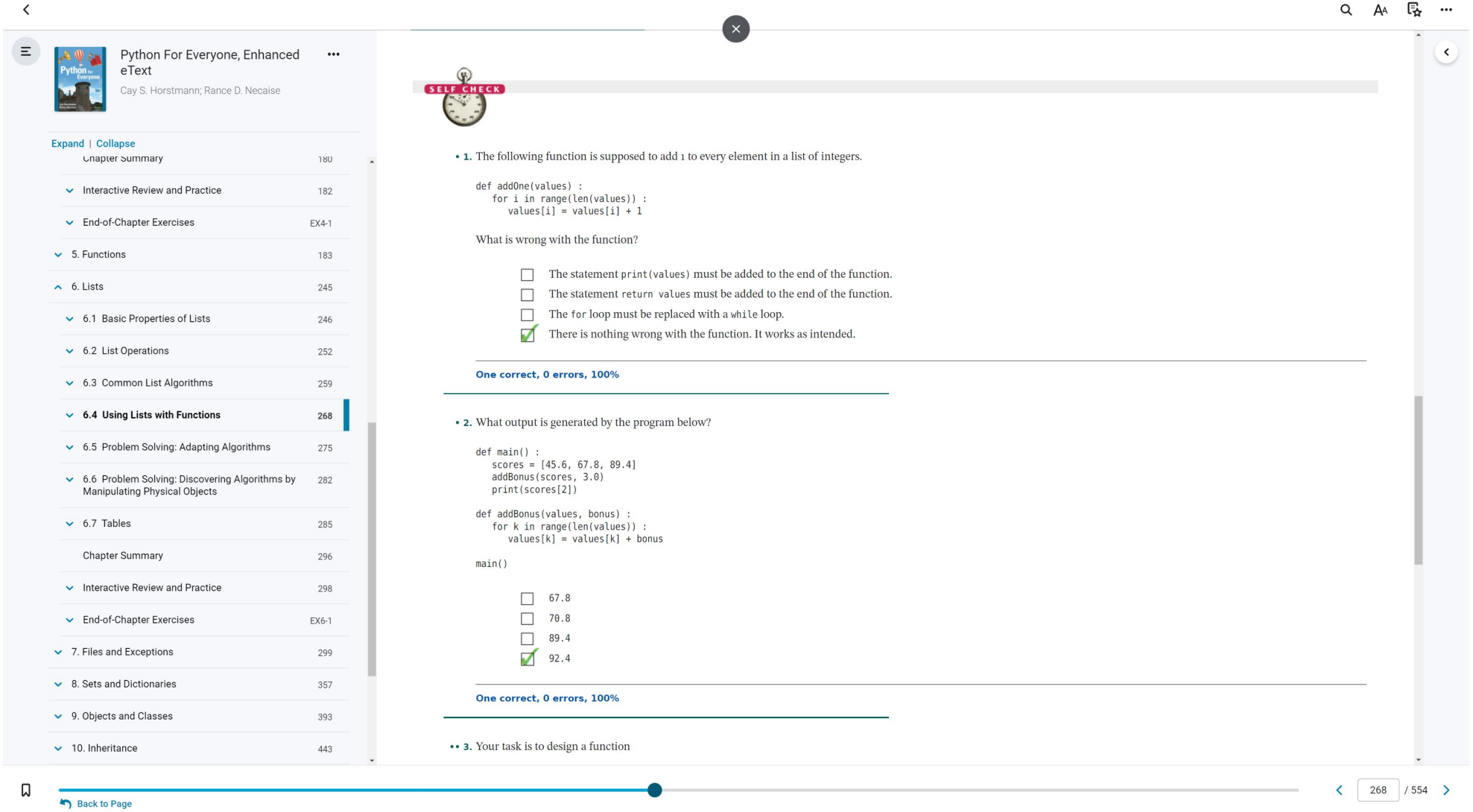
6.2

Albert Nguyen
CS131-47853 MW 06:00 PM – 08:30 PM
Chapter 6 Homework – 6.1 – 6.7

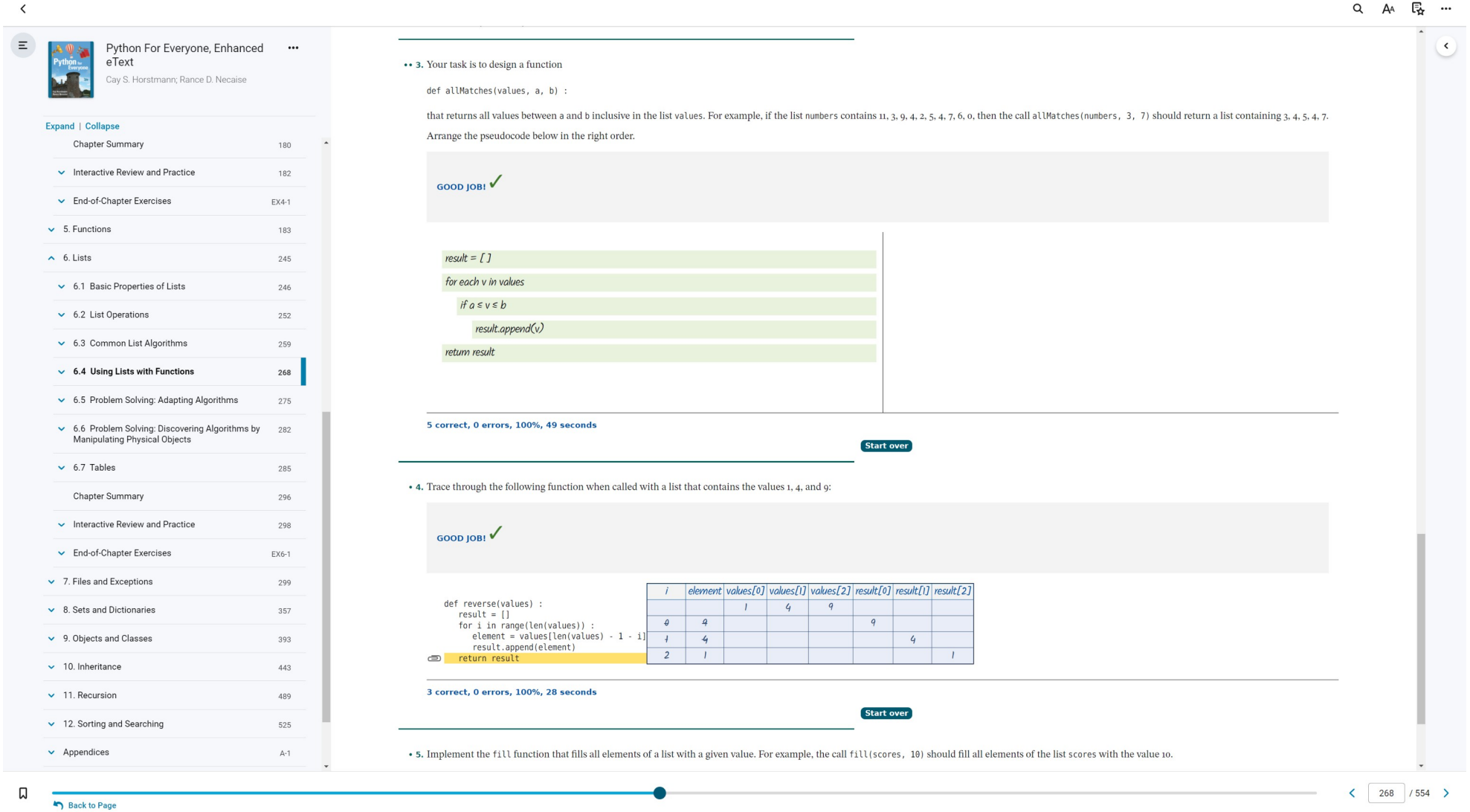


6.3

Albert Nguyen
CS131-47853 MW 06:00 PM – 08:30 PM
Chapter 6 Homework – 6.1 – 6.7



6.4 (p.1)



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

Expand | Collapse

•• 3. Your task is to design a function

```
def allMatches(values, a, b) :
```

that returns all values between a and b inclusive in the list values. For example, if the list numbers contains 11, 3, 9, 4, 2, 5, 4, 7, 6, 0, then the call allMatches(numbers, 3, 7) should return a list containing 3, 4, 5, 4, 7.

Arrange the pseudocode below in the right order.

GOOD JOB! ✓

```
result = [ ]
for each v in values
   if a ≤ v ≤ b
      result.append(v)
return result
```

5 correct, 0 errors, 100%, 49 seconds

Start over

• 4. Trace through the following function when called with a list that contains the values 1, 4, and 9:

GOOD JOB! ✓

```
def reverse(values) :
   result = []
   for i in range(len(values)) :
      element = values[len(values) - 1 - i]
      result.append(element)
   return result
```

i	element	values[0]	values[1]	values[2]	result[0]	result[1]	result[2]
		1	4	9			
0	4				9		
1	4					4	
2	1						1

3 correct, 0 errors, 100%, 28 seconds

Start over

• 5. Implement the fill function that fills all elements of a list with a given value. For example, the call fill(scores, 10) should fill all elements of the list scores with the value 10.

Back to Page

268 / 554

6.4 (p.2)

3 correct, 0 errors, 100%, 28 seconds

Start over

• **5.** Implement the `fill` function that fills all elements of a list with a given value. For example, the call `fill(scores, 10)` should fill all elements of the list `scores` with the value 10.

filllist.py

```
##
#  Fills all elements of a list with a given value.
#  @param data the list whose elements are to be changed
#  @param value the value placed in each element of the list
#
def fill(data, value) :
   list = []
   for index in range(len(data)) :
      data[index] = value
   return list
```

CodeCheck Reset

Testers

Running Tester.py

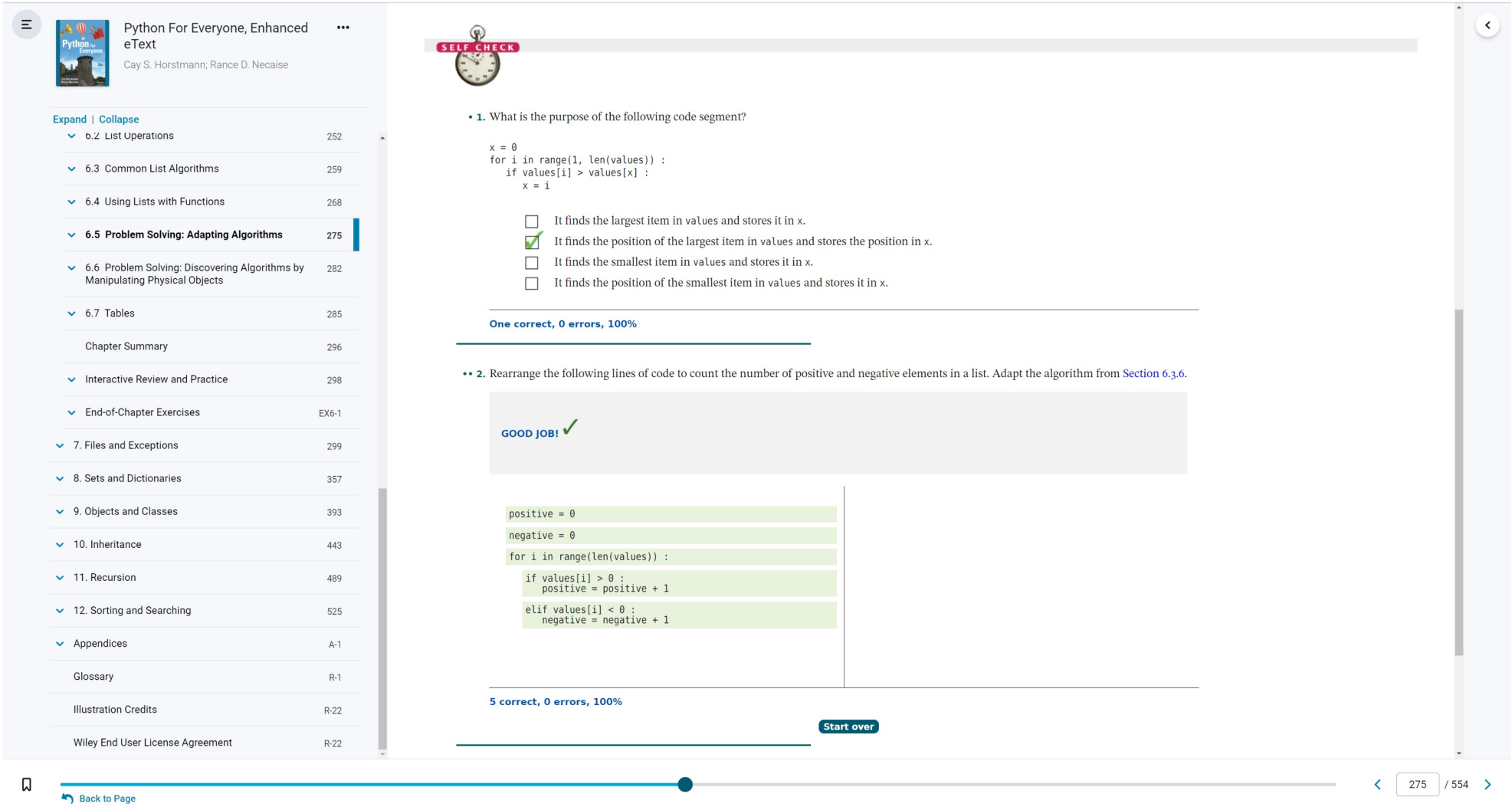
pass pass pass

```
fill( [17, 25, 5, 30, 100, 96, 48, 5, 14, 30] , 10)
[10, 10, 10, 10, 10, 10, 10, 10, 10, 10]
Expected: [10, 10, 10, 10, 10, 10, 10, 10, 10, 10]
fill( [17, 25, 5, 30, 100, 96, 48, 5, 14, 30] , 25)
[25, 25, 25, 25, 25, 25, 25, 25, 25, 25]
Expected: [25, 25, 25, 25, 25, 25, 25, 25, 25, 25]
fill( [8, 6, 9, 3, 5] , 0)
[0, 0, 0, 0, 0]
Expected: [0, 0, 0, 0, 0]
```

Score

3/3

Albert Nguyen
CS131-47853 MW 06:00 PM – 08:30 PM
Chapter 6 Homework – 6.1 – 6.7



Python For Everyone, Enhanced eText

Cay S. Horstmann; Rance D. Necaise

•• 3. Modify the algorithm in Section 6.3.3 and the code below so that it prints all *positive* values in a list, separated by commas.

printpositives.py

```
values = [1, -2, 3, 4]
positiveValues = []
for i in range(len(values)) :
   if values[i] > 0 :
      positiveValues.append(values[i])
positiveValuesString = str(positiveValues)[1:-1]
print(positiveValuesString)
print()
```

CodeCheck Reset

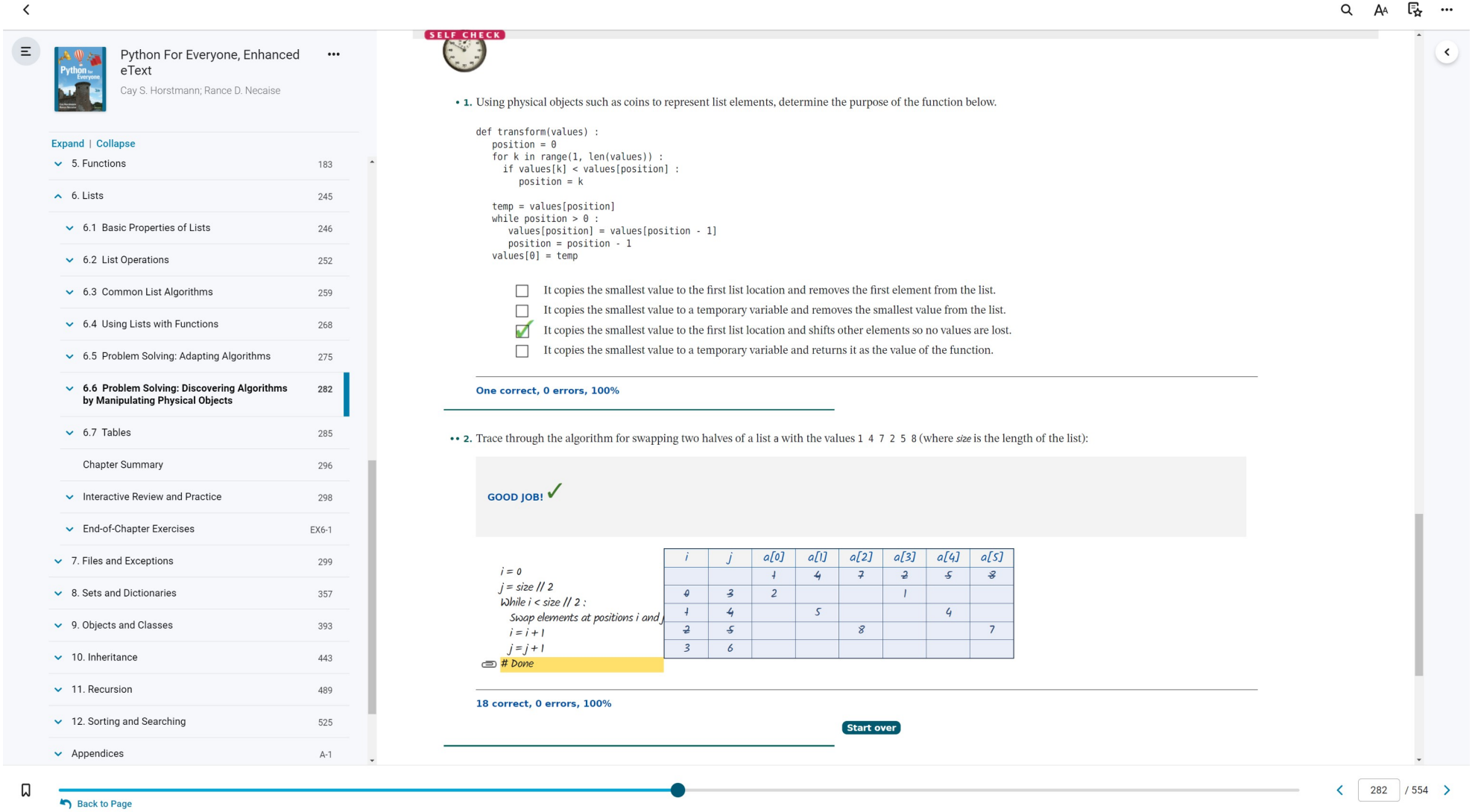
Running program with substitutions

	values	Actual	Expected
pass	[1, -2, 3, 4]	1, 3, 4	1, 3, 4
pass	[17, -25, 5, 30, -100, 96, -48, 5, -14, 30]	17, 5, 30, 96, 5, 30	17, 5, 30, 96, 5, 30
pass	[-5, 18, -25, -28]	18	18

Score

3/3

Albert Nguyen
CS131-47853 MW 06:00 PM – 08:30 PM
Chapter 6 Homework – 6.1 – 6.7



6.6 (p.1)

Python For Everyone, Enhanced eText

Cay S. Horstmann; Rance D. Necaise

Expand | Collapse

2	5			8			7
3	6						

18 correct, 0 errors, 100%

Start over

• **3.** Take out some coins and simulate the following pseudocode, using two paper clips to indicate the positions for *i* and *j*, and where *size* is the length of the list.

```
i = 0
j = size - 1
While i < size :
   Swap elements at positions i and j
   i = i + 1
   j = j - 1
```

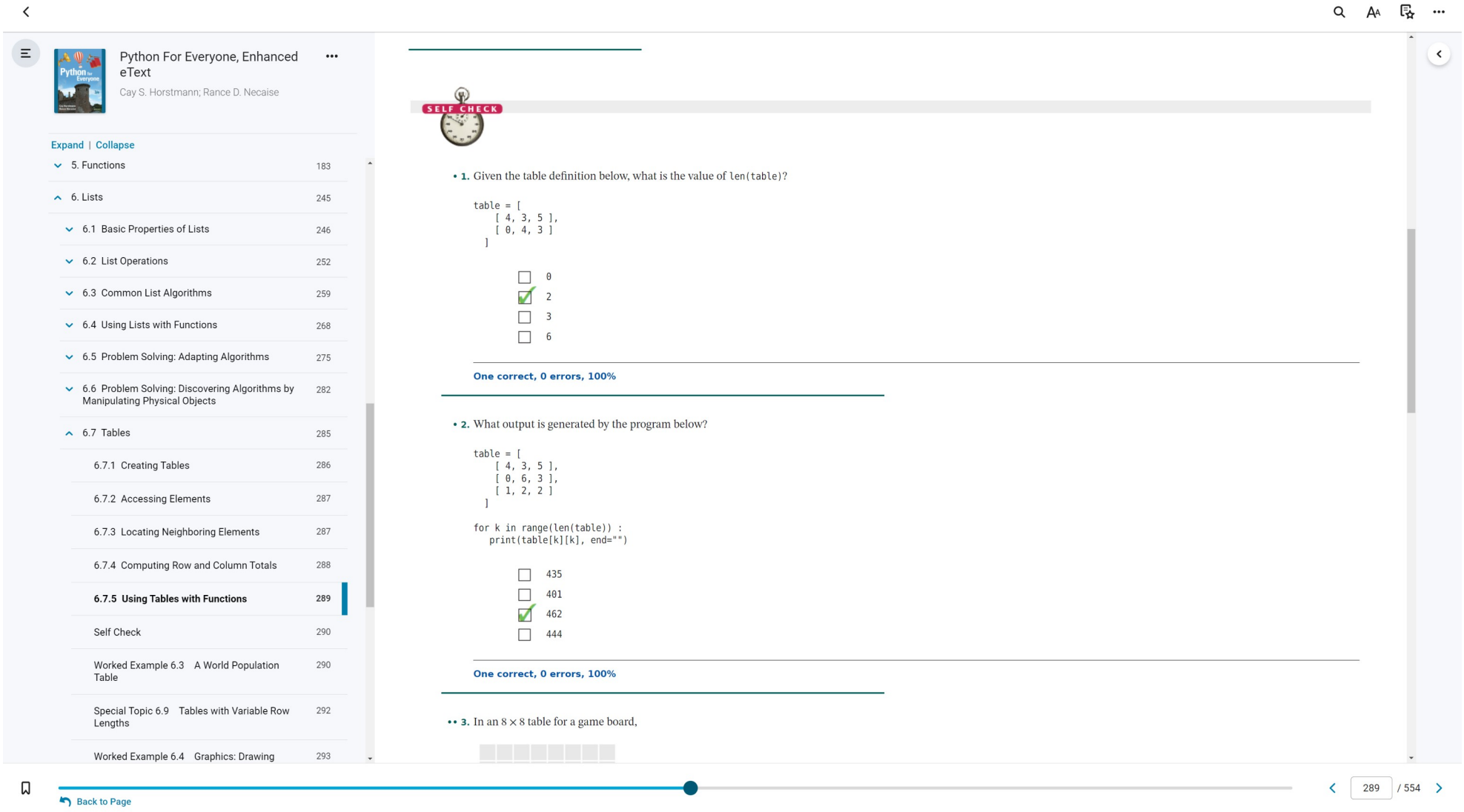
What does the algorithm do?

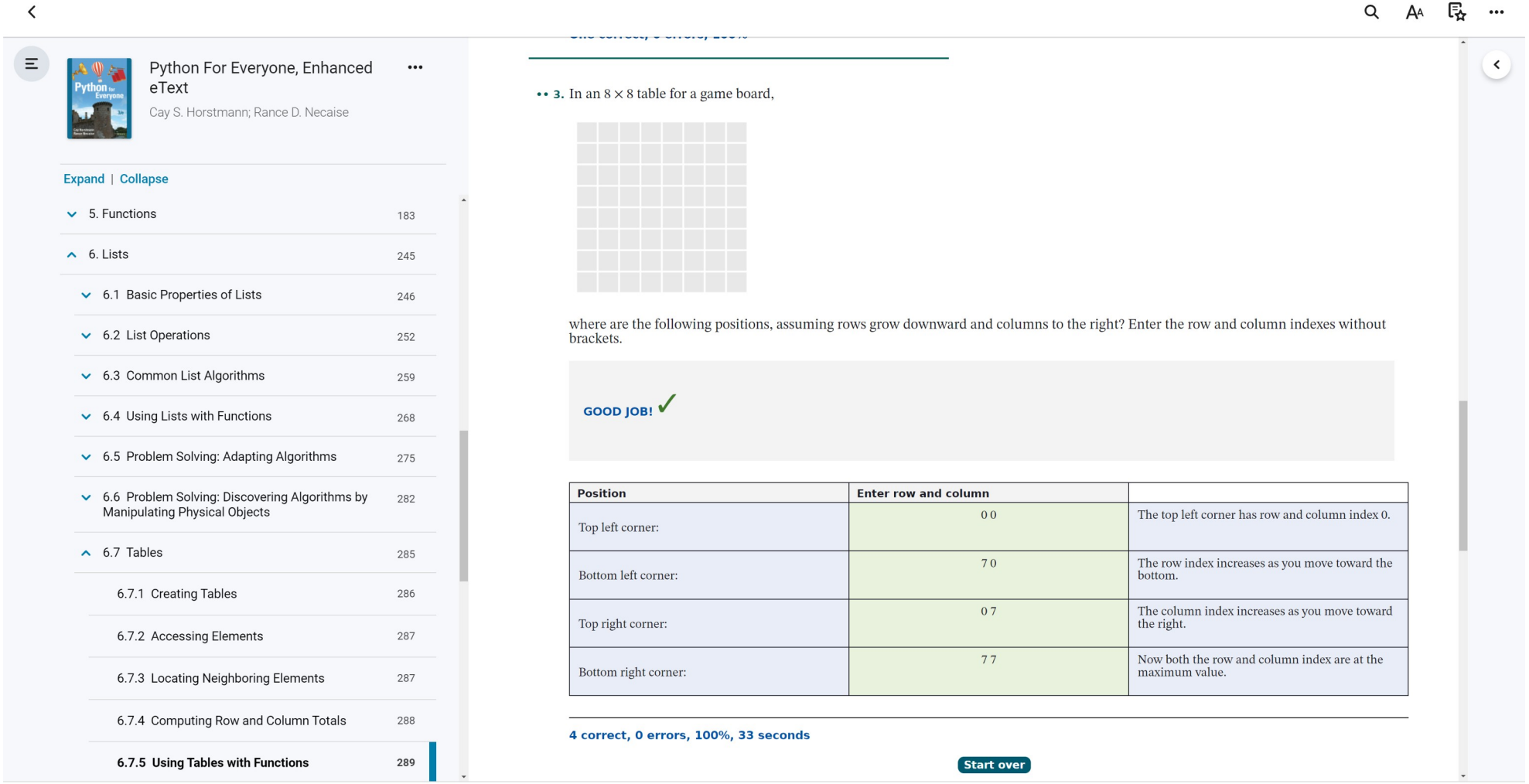
- ☐ Copy the first half of the list to the second half.
- ☐ Copy the second half of the list to the first half.
- ☐ Swap the first and second half of the list.
- ☑ Nothing.

The first and second half are swapped, then they are swapped again. After the algorithm has completed, the list has exactly the same contents as before.

One correct, 0 errors, 100%

Albert Nguyen
CS131-47853 MW 06:00 PM – 08:30 PM
Chapter 6 Homework – 6.1 – 6.7





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

Expand | Collapse

•• 3. In an 8 × 8 table for a game board,

where are the following positions, assuming rows grow downward and columns to the right? Enter the row and column indexes without brackets.

GOOD JOB! ✓

Position	Enter row and column	
Top left corner:	0 0	The top left corner has row and column index 0.
Bottom left corner:	7 0	The row index increases as you move toward the bottom.
Top right corner:	0 7	The column index increases as you move toward the right.
Bottom right corner:	7 7	Now both the row and column index are at the maximum value.

4 correct, 0 errors, 100%, 33 seconds

Start over

Back to Page

289 / 554

6.7 (p.2)

Albert Nguyen
CS131-47853 MW 06:00 PM – 08:30 PM
Chapter 6 Homework – 6.1 – 6.7