



	Python For Everyone, Enhanced eText	...
	Cay S. Horstmann; Rance D. Necaise	
Expand Collapse		
	Chapter Summary	180
↳	Interactive Review and Practice	182
↳	End-of-Chapter Exercises	EX4-1
↳	5. Functions	183
↳	6. Lists	245
↳	6.1 Basic Properties of Lists	246
↳	6.2 List Operations	252
↳	6.3 Common List Algorithms	259
↳	6.4 Using Lists with Functions	268
↳	6.5 Problem Solving: Adapting Algorithms	275
↳	6.6 Problem Solving: Discovering Algorithms by Manipulating Physical Objects	282
↳	6.7 Tables	285
	Chapter Summary	296
↳	Interactive Review and Practice	298
↳	End-of-Chapter Exercises	EX6-1
↳	7. Files and Exceptions	299
↳	8. Sets and Dictionaries	357
↳	9. Objects and Classes	393
↳	10. Inheritance	443

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.

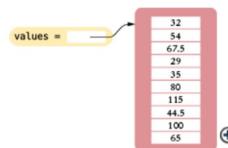


Figure 1 A List of Size 10



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

- grades = {85, 97, 74, 90, 83}
- 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.

- container
- string
- variable
- function

One correct, 0 errors, 100%



[Back to Page](#)



	Python For Everyone, Enhanced eText	...
	Cay S. Horstmann; Rance D. Necaise	
Expand Collapse		
5. Functions	183	
6. Lists	245	
6.1 Basic Properties of Lists	246	
6.2 List Operations	252	
6.3 Common List Algorithms	259	
6.4 Using Lists with Functions	268	
6.5 Problem Solving: Adapting Algorithms	275	
6.6 Problem Solving: Discovering Algorithms by Manipulating Physical Objects	282	
6.7 Tables	285	
Chapter Summary	296	
Interactive Review and Practice	298	
End-of-Chapter Exercises	EX-1	
7. Files and Exceptions	299	
8. Sets and Dictionaries	357	
9. Objects and Classes	393	
10. Inheritance	443	
11. Recursion	489	
12. Sorting and Searching	525	
Appendices	A-1	

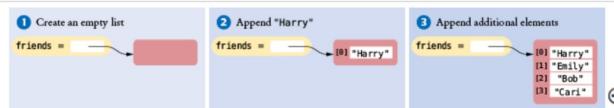


Figure 4 Appending Elements to a List



- 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]
- 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

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

[CodeCheck](#) [Reset](#)

Running evenvalues.py

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

pass

Score

1/1



[Back to Page](#)



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

[Expand](#) | [Collapse](#)

Chapter Summary	180
Interactive Review and Practice	182
End-of-Chapter Exercises	EX4-1
5. Functions	183
6. Lists	245
6.1 Basic Properties of Lists	246
6.2 List Operations	252
6.3 Common List Algorithms	259
6.4 Using Lists with Functions	268
6.5 Problem Solving: Adapting Algorithms	275
6.6 Problem Solving: Discovering Algorithms by Manipulating Physical Objects	282
6.7 Tables	285
Chapter Summary	296
Interactive Review and Practice	298
End-of-Chapter Exercises	EX6-1
7. Files and Exceptions	299
8. Sets and Dictionaries	357
9. Objects and Classes	393
10. Inheritance	443
11. Recursion	489
12. Sorting and Searching	525
Appendices	A-1
Glossary	R-1
Illustration Credits	R-22
Wiley End User License Agreement	R-22

values = []
 for i in range(n) :
 values.append(i * i)

SELF CHECK

1. Given the list values = [0, 0, 0, 0, 0, 0, 0, 0, 0], which statement fills the list with the following numbers?
 1 4 9 16 25 36 49 64 81 100

for i in range(10) :
 values[i] = (i + 1) * (i + 1)

for i in range(10) :
 values[i] = i * i

for i in range(0, 10) :
 values[i] = i + 1

for i in range(1, 10) :
 values[i] = (i + 1) * (i + 1)

One correct, 0 errors, 100%

2. Use a for loop to complete the function in the following program that fills a list with the twenty-six strings "a" ... "z". Hint: the chr function returns the character corresponding to a given ASCII code value as a string.

```
fill.py
1 def main():
2     letters = fillList()
3     print(letters)
4
5 #
6 # Fills a list with the 26 letters of the alphabet stored as individual
7 # strings.
8 # @return the list of 26 strings
9 #
10 def fillList():
11     alphabet = []
12     for letterPosition in range(1,27):
13         letter = chr(letterPosition + 96)
14         alphabet.append(letter)
15     return alphabet
16
17 main()
```

CodeCheck **Reset**

Running fill.py

```
[ 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z' ]
```

pass

Score

1/1



	Python For Everyone, Enhanced eText	...
Cay S. Horstmann; Rance D. Necaise		
Expand Collapse		
Chapter Summary 180		
Interactive Review and Practice 182		
End-of-Chapter Exercises EX4-1		
5. Functions 183		
6. Lists 245		
6.1 Basic Properties of Lists 246		
6.2 List Operations 252		
6.3 Common List Algorithms 259		
6.4 Using Lists with Functions 268		
6.5 Problem Solving: Adapting Algorithms 275		
6.6 Problem Solving: Discovering Algorithms by Manipulating Physical Objects 282		
6.7 Tables 285		
Chapter Summary 296		
Interactive Review and Practice 298		
End-of-Chapter Exercises EX6-1		
7. Files and Exceptions 299		
8. Sets and Dictionaries 357		
9. Objects and Classes 393		
10. Inheritance 443		



- 1. The following function is supposed to add 1 to every element in a list of integers.

```
def addOne(values) :  
    for i in range(len(values)) :  
        values[i] = values[i] + 1
```

What is wrong with the function?

- The statement `print(values)` must be added to the end of the function.
- The statement `return values` must be added to the end of the function.
- The `for` loop must be replaced with a `while` loop.
- There is nothing wrong with the function. It works as intended.

One correct, 0 errors, 100%

- 2. What output is generated by the program below?

```
def main() :  
    scores = [45.6, 67.8, 89.4]  
    addBonus(scores, 3.0)  
    print(scores[2])  
  
def addBonus(values, bonus) :  
    for k in range(len(values)) :  
        values[k] = values[k] + bonus  
  
main()
```

- 67.8
- 70.8
- 89.4
- 92.4

One correct, 0 errors, 100%

- *** 3. Your task is to design a function



Python For Everyone, Enhanced eText

Cay S. Horstmann, Rance D. Necaise

Expand | Collapse

Chapter Summary	180
Interactive Review and Practice	182
End-of-Chapter Exercises	EX4-1
5. Functions	183
6. Lists	245
6.1 Basic Properties of Lists	246
6.2 List Operations	252
6.3 Common List Algorithms	259
6.4 Using Lists with Functions	268
6.5 Problem Solving: Adapting Algorithms	275
6.6 Problem Solving: Discovering Algorithms by Manipulating Physical Objects	282
6.7 Tables	285
Chapter Summary	296
Interactive Review and Practice	298
End-of-Chapter Exercises	EX6-1
7. Files and Exceptions	299
8. Sets and Dictionaries	357
9. Objects and Classes	393
10. Inheritance	443
11. Recursion	489
12. Sorting and Searching	525
Appendices	A-1

- 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 [1, 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]
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.



Python For Everyone, Enhanced eText

Cay S. Horstmann; Rance D. Necaise

Expand | Collapse

▲ 6. Lists	245
▼ 6.1 Basic Properties of Lists	246
▼ 6.2 List Operations	252
▼ 6.3 Common List Algorithms	259
▼ 6.4 Using Lists with Functions	268
▼ 6.5 Problem Solving: Adapting Algorithms	275
▼ 6.6 Problem Solving: Discovering Algorithms by Manipulating Physical Objects	282
▼ 6.7 Tables	285
Chapter Summary	296
▼ Interactive Review and Practice	298
▼ End-of-Chapter Exercises	EX6-1

3 CORRECT, 0 ERRORS, 100%, 20 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

```
1 ##  
2 # Fills all elements of a list with a given value.  
3 # @param data the list whose elements are to be changed  
4 # @param value the value placed in each element of the list  
5 #  
6 def fill(data, value) :  
7     list = []  
8     for index in range(len(data)) :  
9         data[index] = value  
10    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]  
Expected: [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]  
Expected: [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



Back to Page



	Python For Everyone, Enhanced eText	...
	Cay S. Horstmann; Rance D. Necaise	
Expand Collapse		
↳	6.2 List Operations	252
↳	6.3 Common List Algorithms	259
↳	6.4 Using Lists with Functions	268
↳	6.5 Problem Solving: Adapting Algorithms	275
↳	6.6 Problem Solving: Discovering Algorithms by Manipulating Physical Objects	282
↳	6.7 Tables	285
	Chapter Summary	296
↳	Interactive Review and Practice	298
↳	End-of-Chapter Exercises	EX6-1
↳	7. Files and Exceptions	299
↳	8. Sets and Dictionaries	357
↳	9. Objects and Classes	393
↳	10. Inheritance	443
↳	11. Recursion	489
↳	12. Sorting and Searching	525
↳	Appendices	A-1
	Glossary	R-1
	Illustration Credits	R-22
	Wiley End User License Agreement	R-22



- 1. What is the purpose of the following code segment?

```
x = 0
for i in range(1, len(values)) :
    if values[i] > values[x] :
        x = i
```

- It finds the largest item in values and stores it in x.
- It finds the position of the largest item in values and stores the position in x.
- It finds the smallest item in values and stores it in x.
- It finds the position of the smallest item in values and stores it in x.

One correct, 0 errors, 100%

- 2. Rearrange the following lines of code to count the number of positive and negative elements in a list. Adapt the algorithm from [Section 6.3.6](#).

GOOD JOB! ✓

```
positive = 0
negative = 0
for i in range(len(values)) :
    if values[i] > 0 :
        positive = positive + 1
    elif values[i] < 0 :
        negative = negative + 1
```

5 correct, 0 errors, 100%

Start over



Python For Everyone, Enhanced eText

Cay S. Horstmann; Rance D. Necaise

Expand | Collapse

6.2 List Operations 252

6.3 Common List Algorithms 259

6.4 Using Lists with Functions 268

6.5 Problem Solving: Adapting Algorithms 275

6.6 Problem Solving: Discovering Algorithms by Manipulating Physical Objects 282

6.7 Tables 285

Chapter Summary 296

Interactive Review and Practice 298

End-of-Chapter Exercises EX6-1

7. Files and Exceptions 299

8. Sets and Dictionaries 357

- 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

```
1 values = [1, -2, 3, 4]
2 positiveValues = []
3 for i in range(len(values)) :
4     if values[i] > 0 :
5         positiveValues.append(values[i])
6 positiveValuesString = str(positiveValues)[1:-1]
7 print(positiveValuesString)
8 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



Back to Page



	Python For Everyone, Enhanced eText	...
Cay S. Horstmann; Rance D. Necaise		
Expand Collapse		
5. Functions	183	
6. Lists	245	
6.1 Basic Properties of Lists	246	
6.2 List Operations	252	
6.3 Common List Algorithms	259	
6.4 Using Lists with Functions	268	
6.5 Problem Solving: Adapting Algorithms	275	
6.6 Problem Solving: Discovering Algorithms by Manipulating Physical Objects	282	
6.7 Tables	285	
Chapter Summary	296	
Interactive Review and Practice	298	
End-of-Chapter Exercises	EX6-1	
7. Files and Exceptions	299	
8. Sets and Dictionaries	357	
9. Objects and Classes	393	
10. Inheritance	443	
11. Recursion	489	
12. Sorting and Searching	525	
Appendices	A-1	



- 1. Using physical objects such as coins to represent list elements, determine the purpose of the function below.

```
def transform(values) :  
    position = 0  
    for k in range(1, len(values)) :  
        if values[k] < values[position] :  
            position = k  
  
    temp = values[position]  
    while position > 0 :  
        values[position] = values[position - 1]  
        position = position - 1  
    values[0] = temp
```

- It copies the smallest value to the first list location and removes the first element from the list.
- It copies the smallest value to a temporary variable and removes the smallest value from the list.
- It copies the smallest value to the first list location and shifts other elements so no values are lost.
- It copies the smallest value to a temporary variable and returns it as the value of the function.

One correct, 0 errors, 100%

- 2. Trace through the algorithm for swapping two halves of a list *a* with the values 1 4 7 2 5 8 (where *size* is the length of the list):

GOOD JOB! ✓

```
i = 0  
j = size // 2  
while i < size // 2 :  
    Swap elements at positions i and j  
    i = i + 1  
    j = j + 1  
# Done
```

<i>i</i>	<i>j</i>	<i>a[0]</i>	<i>a[1]</i>	<i>a[2]</i>	<i>a[3]</i>	<i>a[4]</i>	<i>a[5]</i>
		1	4	7	2	5	8
0	3	2			1		
1	4		5			4	
2	5			8			7
3	6						

18 correct, 0 errors, 100%

[Start over](#)



Python For Everyone, Enhanced eText

Cay S. Horstmann; Rance D. Necaise

Expand | Collapse

5. Functions	183
6. Lists	245
6.1 Basic Properties of Lists	246
6.2 List Operations	252
6.3 Common List Algorithms	259
6.4 Using Lists with Functions	268
6.5 Problem Solving: Adapting Algorithms	275
6.6 Problem Solving: Discovering Algorithms by Manipulating Physical Objects	282
6.7 Tables	285
Chapter Summary	296
Interactive Review and Practice	298

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%



Back to Page

Python For Everyone, Enhanced eText

Cay S. Horstmann; Rance D. Necaise

Expand | Collapse

- 5. Functions 183
- 6. Lists 245
 - 6.1 Basic Properties of Lists 246
 - 6.2 List Operations 252
 - 6.3 Common List Algorithms 259
 - 6.4 Using Lists with Functions 268
 - 6.5 Problem Solving: Adapting Algorithms 275
 - 6.6 Problem Solving: Discovering Algorithms by Manipulating Physical Objects 282
- 6.7 Tables 285
 - 6.7.1 Creating Tables 286
 - 6.7.2 Accessing Elements 287
 - 6.7.3 Locating Neighboring Elements 287
 - 6.7.4 Computing Row and Column Totals 288
 - 6.7.5 Using Tables with Functions 289**
 - Self Check 290
 - Worked Example 6.3 A World Population Table 290
 - Special Topic 6.9 Tables with Variable Row Lengths 292
 - Worked Example 6.4 Graphics: Drawing 293

SELF CHECK

1. Given the table definition below, what is the value of len(table)?

```
table = [
    [ 4, 3, 5 ],
    [ 0, 4, 3 ]
]
```

0
 2
 3
 6

One correct, 0 errors, 100%

2. What output is generated by the program below?

```
table = [
    [ 4, 3, 5 ],
    [ 0, 6, 3 ],
    [ 1, 2, 2 ]
]

for k in range(len(table)) :
    print(table[k][k], end="")
```

435
 401
 462
 444

One correct, 0 errors, 100%

3. In an 8×8 table for a game board,

Back to Page

289 / 554

< Q A ⌂ ...

Python For Everyone, Enhanced eText

Cay S. Horstmann; Rance D. Necaise

Expand | Collapse

- 5. Functions 183
- 6. Lists 245
- 6.1 Basic Properties of Lists 246
- 6.2 List Operations 252
- 6.3 Common List Algorithms 259
- 6.4 Using Lists with Functions 268
- 6.5 Problem Solving: Adapting Algorithms 275
- 6.6 Problem Solving: Discovering Algorithms by Manipulating Physical Objects 282
- 6.7 Tables 285
 - 6.7.1 Creating Tables 286
 - 6.7.2 Accessing Elements 287
 - 6.7.3 Locating Neighboring Elements 287
 - 6.7.4 Computing Row and Column Totals 288
 - 6.7.5 Using Tables with Functions 289

•• 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 >

Python For Everyone, Enhanced eText

Cay S. Horstmann; Rance D. Necaise

Expand | Collapse

- 5. Functions 183
- 6. Lists 245
 - 6.1 Basic Properties of Lists 246
 - 6.2 List Operations 252
 - 6.3 Common List Algorithms 259
 - 6.4 Using Lists with Functions 268
 - 6.5 Problem Solving: Adapting Algorithms 275
 - 6.6 Problem Solving: Discovering Algorithms by Manipulating Physical Objects 282
 - 6.7 Tables 285
 - 6.7.1 Creating Tables 286
 - 6.7.2 Accessing Elements 287
 - 6.7.3 Locating Neighboring Elements 287
 - 6.7.4 Computing Row and Column Totals 288
 - 6.7.5 Using Tables with Functions 289
 - Self Check 290
 - Worked Example 6.3 A World Population Table 290
 - Special Topic 6.9 Tables with Variable Row Lengths 292
 - Worked Example 6.4 Graphics: Drawing Regular Polygons 293
 - Chapter Summary 296
 - Interactive Review and Practice 298

• 4. Consider an 8×8 table for a board game. Initialize the board so that zeroes and ones alternate, as on a checkerboard:

```
0 1 0 1 0 1 0 1  
1 0 1 0 1 0 1 0  
0 1 0 1 0 1 0 1  
1 0 1 0 1 0 1 0  
0 1 0 1 0 1 0 1  
1 0 1 0 1 0 1 0  
0 1 0 1 0 1 0 1  
1 0 1 0 1 0 1 0
```

Rearrange the following lines of code to produce a solution. Assume the table used to represent the board has already been created. Not all lines are useful.

GOOD JOB! ✓

```
for i in range(8) :  
    for j in range(8) :  
        board[i][j] = (i + j) % 2
```

```
board[i][j] = i % 2 + j % 2  
board[i][j] = j % 2  
board[i][j] = i % 2  
for i in range(64) :
```

3 correct, 0 errors, 100%, 27 seconds

Start over

• 5. Write a function that fills a specific column of a table with a given value. Complete this code:

```
fillcol.py
```

```
1##  
2# Fills a given column of a table with a given value.  
3# @param values the table  
4# @param column the index of the column to fill  
5# @param fillValue the value to fill it with  
6#  
7def fillColumn(values, column, fillValue):  
8    for row in values:  
9        row[column] = fillValue
```

CodeCheck Reset

Testers

Running fillcolTester.py

```
pass pass
```

matrix:
[[1, 0, 0], [0, 1, 0], [1, 0, 0], [0, 1, 0]]
fillColumn(matrix, 0, 2);
[[2, 0, 0], [2, 1, 0], [2, 0, 0], [2, 1, 0]]
Expected: [[2, 0, 0], [2, 1, 0], [2, 0, 0], [2, 1, 0]]
fillColumn(matrix, 2, 3);
[[2, 0, 3], [2, 1, 3], [2, 0, 3], [2, 1, 3]]
Expected: [[2, 0, 3], [2, 1, 3], [2, 0, 3], [2, 1, 3]]

Score

2/2

Back to Page / 289 / 554