Chapter 6 Reading Questions

David Letscher

Saint Louis University

CSCI 1300: Introduction to Object-Oriented Programming

Reading Quiz

Most difficult questions

Consider the short Python program:

```
message = 'Hello!'
message.lower()
```

What is the value of message after it runs?

Reading Quiz

Most difficult questions

Consider the short Python program:

```
message = 'Hello!'
message.lower()
```

What is the value of message after it runs?

83% 'hello!'

14% 'Hello!'

3% **'hello'**

Reading Quiz

Most difficult questions

Consider the short Python program:

```
message = 'Hello!'
message.lower()
```

What is the value of message after it runs?

83% 'hello!'

14% 'Hello!' **Correct answer**

3% **'hello'**

Reading Quiz

Most difficult questions

What is the output when the following Python program runs:

```
word = 'Duality'
word[1] = 'Q'
print(word)
```

Reading Quiz

Most difficult questions

What is the output when the following Python program runs:

```
word = 'Duality'
word[1] = 'Q'
print(word)
```

86% The program stops with an error

14% DQality

Reading Quiz

Most difficult questions

What is the output when the following Python program runs:

```
word = 'Duality'
word[1] = 'Q'
print(word)
```

86% `The program stops with an error` **Correct answer**

14% `DQality`

String Immutability

String are immutable, so they cannot be changed. The various methods of the class all return new things leaving the original unchanged. For example,

```
message = 'Hello!'
newMessage = message.lower()
```

message is still equal to 'Hello!' and newMessage is equal to 'hello!'.

Reading Quiz

Most difficult questions

What is the value of letters after the following program runs:

```
friends = ['Amy', 'Bill', 'Carol']
friends[1] = ['Doug']
```

Reading Quiz

Most difficult questions

What is the value of letters after the following program runs:

```
friends = ['Amy', 'Bill', 'Carol']
friends[1] = ['Doug']
```

93% ['Amy', 'Doug', 'Carol']

4% ['Doug', 'Bill', 'Carol']

4% The program stops with an error message.

Reading Quiz

Most difficult questions

What is the value of letters after the following program runs:

```
friends = ['Amy', 'Bill', 'Carol']
friends[1] = ['Doug']
```

93% ['Amy', 'Doug', 'Carol'] **Correct answer**

4% ['Doug', 'Bill', 'Carol']

4% The program stops with an error message.

Reading Quiz

Most difficult questions

What is the output of the following program?

```
friends = ['Bill', 'Carol', 'Amy']
print(friends.sort())
```

Reading Quiz

Most difficult questions

What is the output of the following program?

```
friends = ['Bill', 'Carol', 'Amy']
print(friends.sort())
```

72% None

28% ['Amy', 'Bill', 'Carol']

Reading Quiz

Most difficult questions

What is the output of the following program?

```
friends = ['Bill', 'Carol', 'Amy']
print(friends.sort())
```

72% None **Correct answer**

28% ['Amy', 'Bill', 'Carol']

Lists are mutable so their state can change. Many of the methods change the state of the list, e.g. append, insert, pop, remove. Some of these methods return something others do not.

Reading Quiz

Most difficult questions

What is the result of the following:

```
friends = ['Bill', 'Doug', 'Carol', 'Amy']
friends.sort().reverse()
```

Reading Quiz

Most difficult questions

What is the result of the following:

```
friends = ['Bill', 'Doug', 'Carol', 'Amy']
friends.sort().reverse()
```

90% The program exists with an error

7% friends has value ['Doug', 'Carol', 'Bill', 'Amy']

3% friends has value ['Amy', 'Carol', 'Doug', 'Bill']

Reading Quiz

Most difficult questions

What is the result of the following:

```
friends = ['Bill', 'Doug', 'Carol', 'Amy']
friends.sort().reverse()
```

- 90% The program exists with an error **Correct answer**
- 7% friends has value ['Doug', 'Carol', ' Bill ', 'Amy']
- 3% friends has value ['Amy', 'Carol', 'Doug', ' Bill ']

Sort is a mutator that sorts the list and returns nothing. So calling reverse on the result is invalid.

Reading Quiz

Most difficult questions

What is the value of friends after the following is executed:

```
friends = ['Amy', 'Carol', 'Doug']
friends.insert(2, 'Bill')
```

Reading Quiz

Most difficult questions

What is the value of friends after the following is executed:

```
friends = ['Amy', 'Carol', 'Doug']
friends.insert(2, 'Bill')
```

72% ['Amy', 'Carol', 'Bill', 'Doug']

28% ['Amy', 'Carol', 'Doug', 'Bill']

Reading Quiz

Most difficult questions

What is the value of friends after the following is executed:

```
friends = ['Amy', 'Carol', 'Doug']
friends.insert(2, 'Bill')
```

72% ['Amy', 'Carol', 'Bill', 'Doug'] **Correct answer**

28% ['Amy', 'Carol', 'Doug', 'Bill']

List/string/etc indices start at 0. Insert places the new entry in the specified index.

Reading Quiz

Most difficult questions

What is the output of

```
print('Helps'[:-1])
```

Reading Quiz

Most difficult questions

What is the output of

```
print('Helps'[:-1])
```

70% `Help`

17% `'Help'`

3% It quits with an error.

Reading Quiz

Most difficult questions

What is the output of

```
print('Helps'[:-1])
```

70% `Help` **Correct answer**

17% `'Help'`

3% It quits with an error.

Recall the quotes are used in the literal form to indicate that the data is a string. They are not printed.

Indices

Why do they start at 0?

The count how many spots after the beginning (the offset).

Indices

Why do they start at 0?

The count how many spots after the beginning (the offset).

For example groceries = ['Milk', 'Eggs', 'Cheese']

Index	Item
0	'Milk'
1	'Eggs'
2	'Cheese'

Indices

Why do they start at 0?

The count how many spots after the beginning (the offset).

For example groceries = ['Milk', 'Eggs', 'Cheese']

Index	Item
0	'Milk'
1	'Eggs'
2	'Cheese'

len(groceries) returns the length of the list (which is 3). Note that this length is always one more then the index of the last element of the list.

Practice

What is the result of?

```
groceries = ['Milk', 'Eggs', 'Cheese']
groceries.insert(1, 'Apples')
print(groceries.pop(2))
print(groceries)
```

Slicing

Examples

names[2:5]	items with indices 2,3 and 4

Slicing

Examples

names[2:5]	items with indices 2,3 and 4
names[2:]	items with indices 2, 3, ... up to the last element of the list

Slicing

Examples

names[2:5]	items with indices 2,3 and 4
names[2:]	items with indices 2, 3, ... up to the last element of the list
names[:len(names)]	all items

Slicing

Examples

names[2:5]	items with indices 2,3 and 4
names[2:]	items with indices 2, 3, ... up to the last element of the list
names[:len(names)]	all items
names[:len(names)−1]	all but the last item

Slicing

Examples

names[2:5]	items with indices 2,3 and 4
names[2:]	items with indices 2, 3, ... up to the last element of the list
names[:len(names)]	all items
names[:len(names)−1]	all but the last item
names[:−1]	all but the last item

Slicing

Examples

names[2:5]	items with indices 2,3 and 4
names[2:]	items with indices 2, 3, ... up to the last element of the list
names[:len(names)]	all items
names[:len(names)−1]	all but the last item
names[:−1]	all but the last item
names[1:7:2]	

Slicing

Examples

names[2:5]	items with indices 2,3 and 4
names[2:]	items with indices 2, 3, ... up to the last element of the list
names[:len(names)]	all items
names[:len(names)−1]	all but the last item
names[:−1]	all but the last item
names[1:7:2]	items with indices 1, 3 and 5

Slicing

Examples

names[2:5]	items with indices 2,3 and 4
names[2:]	items with indices 2, 3, ... up to the last element of the list
names[:len(names)]	all items
names[:len(names)−1]	all but the last item
names[:−1]	all but the last item
names[1:7:2]	items with indices 1, 3 and 5
names[7:1:−2]	

Slicing

Examples

names[2:5]	items with indices 2,3 and 4
names[2:]	items with indices 2, 3, ... up to the last element of the list
names[:len(names)]	all items
names[:len(names)−1]	all but the last item
names[:−1]	all but the last item
names[1:7:2]	items with indices 1, 3 and 5
names[7:1:−2]	items with indices 7, 5 and 3

Slicing

Examples

names[2:5]	items with indices 2,3 and 4
names[2:]	items with indices 2, 3, ... up to the last element of the list
names[:len(names)]	all items
names[:len(names)−1]	all but the last item
names[:−1]	all but the last item
names[1:7:2]	items with indices 1, 3 and 5
names[7:1:−2]	items with indices 7, 5 and 3
names[::−1]	

Slicing

Examples

names[2:5]	items with indices 2,3 and 4
names[2:]	items with indices 2, 3, ... up to the last element of the list
names[:len(names)]	all items
names[:len(names)−1]	all but the last item
names[:−1]	all but the last item
names[1:7:2]	items with indices 1, 3 and 5
names[7:1:−2]	items with indices 7, 5 and 3
names[::−1]	reversed list

Splits

Examples

Splitting:

```
items = 'apples, milk, bread, cheese'
print(items.split(', '))
```

Result:

```
['apples', 'milk', 'bread', 'cheese']
```

Splits

Examples

Splitting:

```
items = 'apples, milk, bread, cheese'
print(items.split(', '))
```

Result:

```
['apples', 'milk', 'bread', 'cheese']
```

Splitting with no parameter:

```
items = 'How    are you?   '
print(items.split())
```

Result:

```
['How', 'are', 'you?']
```

Joins

Examples

Join:

```
items = ['apples', 'milk', 'bread', 'cheese']
print(', '.join(items))
```

Result:

```
apples, milk, bread, cheese
```

Joins

Examples

Join:

```
items = ['apples', 'milk', 'bread', 'cheese']
print(', '.join(items))
```

Result:

```
apples, milk, bread, cheese
```

Joining using empty string:

```
letters = ['H', 'e', 'l', 'l', 'o']
print(''.join(letters)
```

Result:

```
Hello
```

Challenge

Suppose you have a list groceries containing what you want to buy at the store. Print each item on a separate line.
For example, if groceries = ['Milk', 'Eggs', 'Cheese'] the output would be:

```
Milk
Eggs
Cheese
```

Challenge

Suppose you have a list groceries containing what you want to buy at the store. Print each item on a separate line.
For example, if groceries = ['Milk', 'Eggs', 'Cheese'] the output would be:

```
Milk
Eggs
Cheese
```

A solution

```
print('\n'.join(groceries))
```

Challenge

A little harder

Now try to get it to print a begging message and indent the items with two spaces. So the output for this example would be:

```
You need buy:
  Milk
  Eggs
  Cheese
```

Challenge

A little harder

Now try to get it to print a begging message and indent the items with two spaces. So the output for this example would be:

```
You need buy:
  Milk
  Eggs
  Cheese
```

A solution

```
print('You need to buy\n  ' + '\n
'.join(groceries))
```