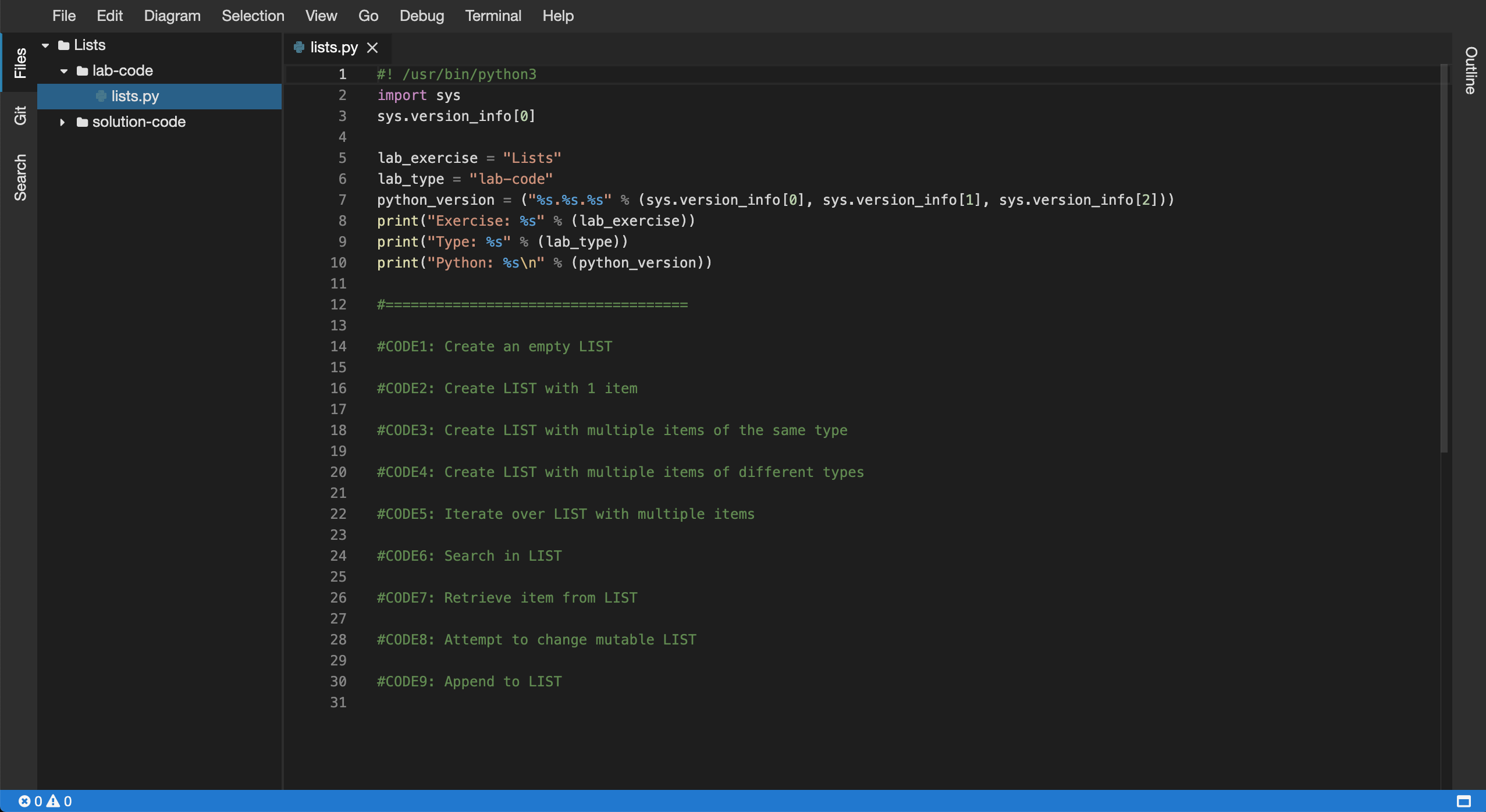
Lists

Open the lists.py source file within the editor. Take some time to review the uncompleted code within this file to understand the intended design:



5. Next, you will be required to complete the code in the following source files:

* lists.py

6. Within the editor ensure that the lists.py file now has focus.

7. Replace the //Code1: comment, create an empty LIST:

[**Copy code**](https://app.qa.com/lab/coding-python-collection-types/exercise-2-lists/?context_id=637&context_resource=lp)

#CODE1: Create an empty LIST

list1 = []

print("CODE1:")

print(f"list1 = {list1}")

print(f"data type = {type(list1)}")

print(f"length = {len(list1)}")

print()

8. Replace the //Code2: comment, create LIST with 1 item:

[**Copy code**](https://app.qa.com/lab/coding-python-collection-types/exercise-2-lists/?context_id=637&context_resource=lp)

#CODE2: Create LIST with 1 item

list2 = ["cloudacademy"]

print("CODE2:")

print(f"list2 = {list2}")

print(f"data type = {type(list2)}")

print(f"length = {len(list2)}")

print()

9. Replace the //Code3: comment, create LIST with multiple items of the same type

[**Copy code**](https://app.qa.com/lab/coding-python-collection-types/exercise-2-lists/?context_id=637&context_resource=lp)

#CODE3: Create LIST with multiple items of the same type

list3 = [1, 2, 3, 4, 5]

print("CODE3:")

print(f"list3 = {list3}")

print(f"data type = {type(list3)}")

print(f"length = {len(list3)}")

print()

10. Replace the //Code4: comment, create LIST with multiple items of different types

[**Copy code**](https://app.qa.com/lab/coding-python-collection-types/exercise-2-lists/?context_id=637&context_resource=lp)

#CODE4: Create LIST with multiple items of different types

list4 = ["cloud", "academy", 1, True, False]

print("CODE4:")

print(f"list4 = {list4}")

print(f"data type = {type(list4)}")

print(f"length = {len(list4)}")

print()

11. Replace the //Code5: comment, iterate over LIST with multiple items

Note: Ensure the indentation within the following code block is maintained within the editor view when you perform the copy and paste operation.

[**Copy code**](https://app.qa.com/lab/coding-python-collection-types/exercise-2-lists/?context_id=637&context_resource=lp)

#CODE5: Iterate over LIST with multiple items

print("CODE5:")

for item in list4:

print(item)

print()

12. Replace the //Code6: comment, search within the LIST:

[**Copy code**](https://app.qa.com/lab/coding-python-collection-types/exercise-2-lists/?context_id=637&context_resource=lp)

#CODE6: Search in LIST

print("CODE6:")

print ("cloud" in list4)

print ("blah" in list4)

print()

13. Replace the //Code7: comment, retrieve item from LIST:

[**Copy code**](https://app.qa.com/lab/coding-python-collection-types/exercise-2-lists/?context_id=637&context_resource=lp)

#CODE7: Retrieve item from LIST

print("CODE7:")

item0 = list4[0]

item1 = list4[1]

print(f"item0 = {item0}")

print(f"item1 = {item1}")

print()

14. Replace the //Code8: comment, change mutable LIST:

[**Copy code**](https://app.qa.com/lab/coding-python-collection-types/exercise-2-lists/?context_id=637&context_resource=lp)

#CODE8: Change mutable LIST

print("CODE8:")

list4[0] = "possible!!"

print(f"list4 = {list4}")

print(f"data type = {type(list4)}")

print(f"length = {len(list4)}")

print()

15. Replace the //Code9: comment, append to LIST:

[**Copy code**](https://app.qa.com/lab/coding-python-collection-types/exercise-2-lists/?context_id=637&context_resource=lp)

#CODE9: Append to LIST

print("CODE9:")

list4.append("new item")

print(f"list4 = {list4}")

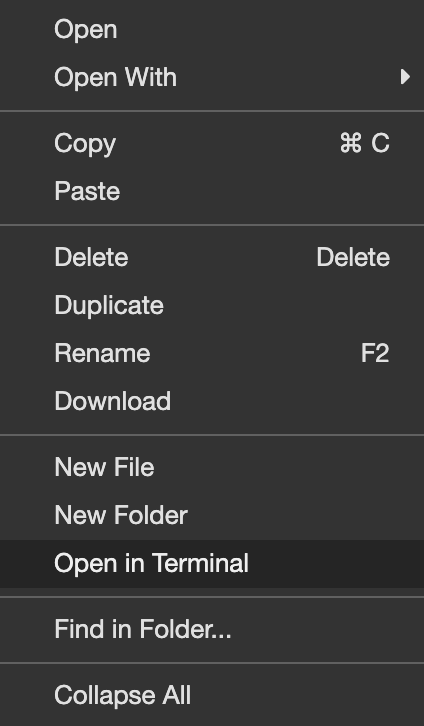
print(f"data type = {type(list4)}")

print(f"length = {len(list4)}")

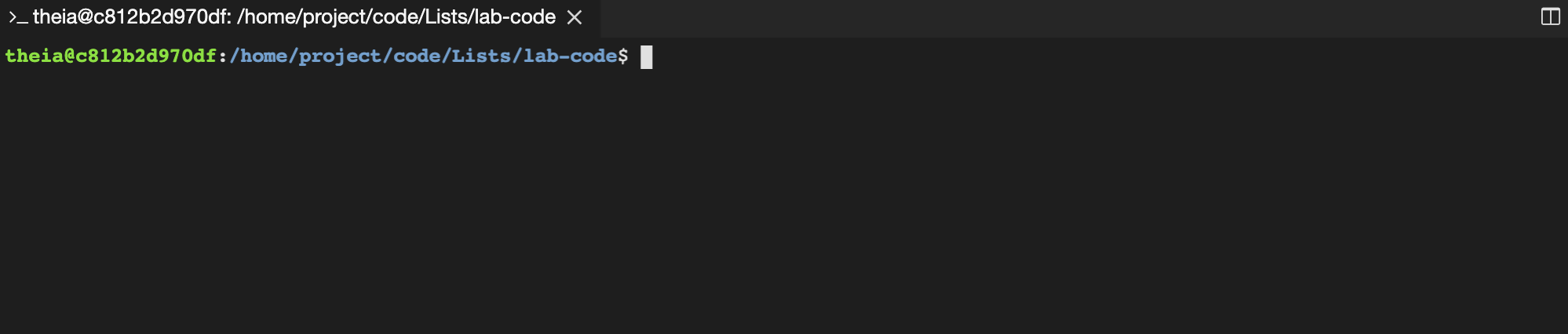
print()

16. Ok the lists.py Python script is now ready to be executed by the Python interpreter. To do so you will need to use the embedded terminal to launch Python.

17. Within the Files tree view (lefthand side menu), select the lab-code folder and right click and select the Open in Terminal option:

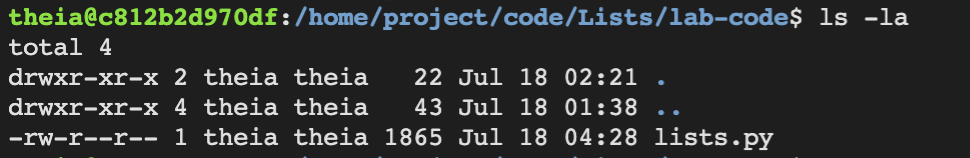


18. Access to the embedded terminal is now provided in the bottom pane of the current view:

  
19. Peform a directory listing on the current directory to ensure the presence of the lists.py file, like so:

[**Copy code**](https://app.qa.com/lab/coding-python-collection-types/exercise-2-lists/?context_id=637&context_resource=lp)

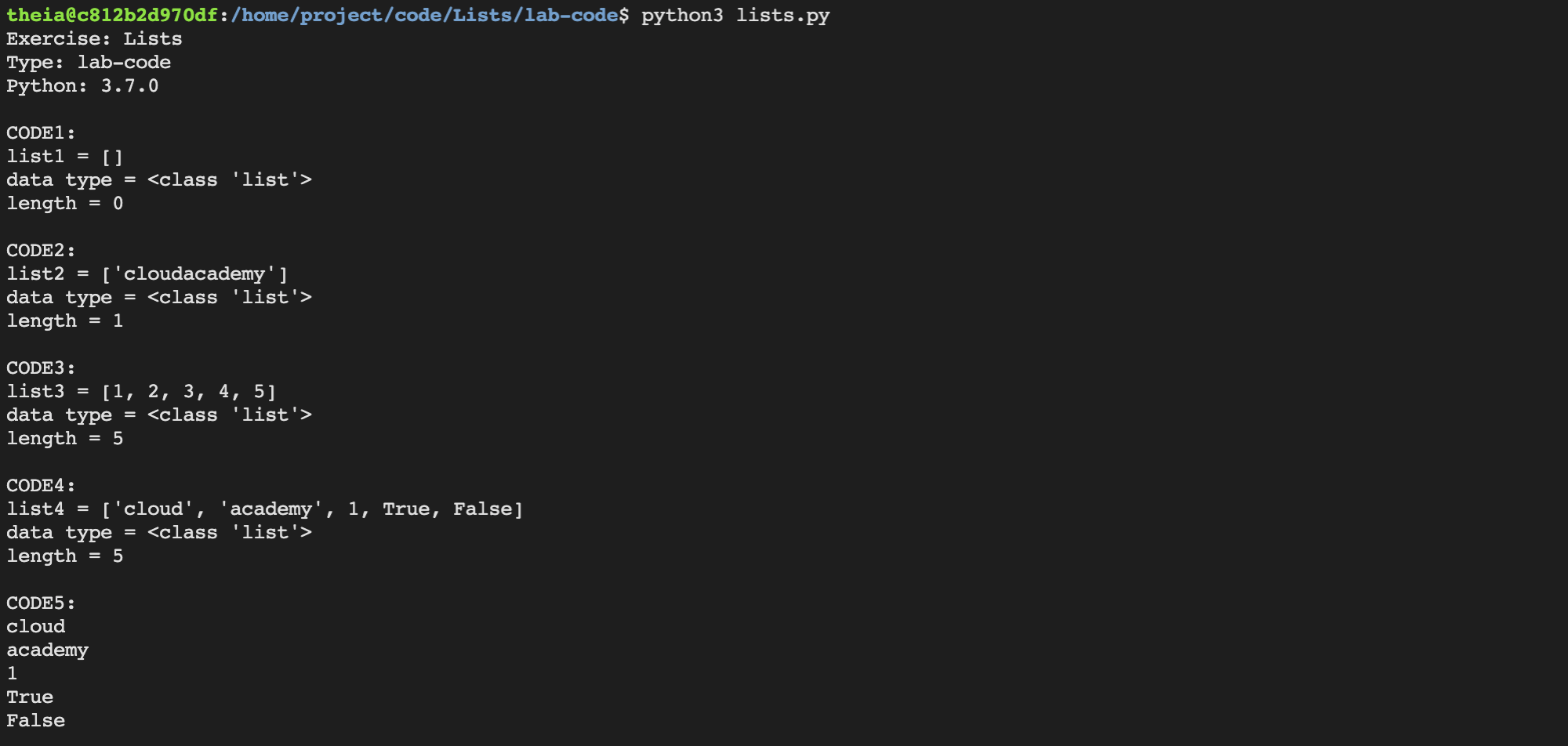
ls -la

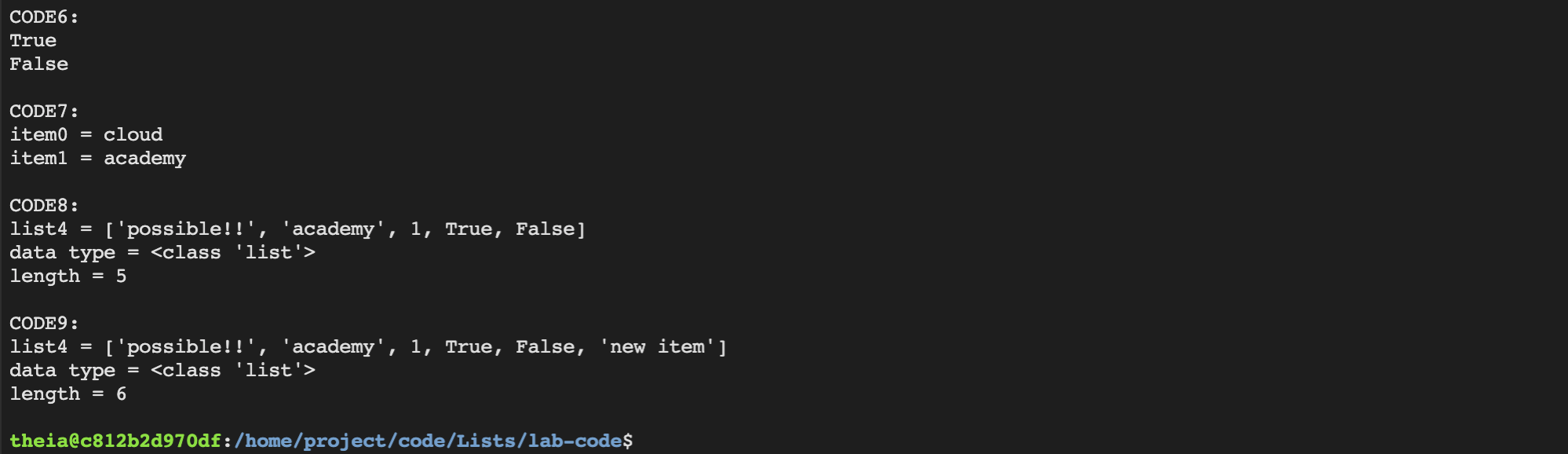


20. Let's now execute the updated lists.py script using the python3 command. Within the terminal enter the following command:

[**Copy code**](https://app.qa.com/lab/coding-python-collection-types/exercise-2-lists/?context_id=637&context_resource=lp)

python3 lists.py





21. Great! Your lists.py script has executed successfully! This demonstrates how to work with Lists.

22. Try updating the lists.py script and then re-executing using the same steps above. See the following documentation for ideas:

[https://docs.python.org/3.3/tutorial/datastructures.html#more-on-lists](https://docs.python.org/3.3/tutorial/datastructures.html" \l "more-on-lists)

### 

### Summary

In this Lab Step, you opened the **Lists** Workspace and then updated the lists.py file to store and manipulate data using Lists. Next you used the embedded terminal to launch and debug the lists.py script file.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Solution Code

#! /usr/bin/python3

import sys

sys.version\_info[0]

lab\_exercise = "Lists"

lab\_type = "solution-code"

python\_version = ("%s.%s.%s" % (sys.version\_info[0], sys.version\_info[1], sys.version\_info[2]))

print("Exercise: %s" % (lab\_exercise))

print("Type: %s" % (lab\_type))

print("Python: %s\n" % (python\_version))

#====================================

#CODE1: Create an empty LIST

list1 = []

print("CODE1:")

print(f"list1 = {list1}")

print(f"data type = {type(list1)}")

print(f"length = {len(list1)}")

print()

#CODE2: Create LIST with 1 item

list2 = ["cloudacademy"]

print("CODE2:")

print(f"list2 = {list2}")

print(f"data type = {type(list2)}")

print(f"length = {len(list2)}")

print()

#CODE3: Create LIST with multiple items of the same type

list3 = [1, 2, 3, 4, 5]

print("CODE3:")

print(f"list3 = {list3}")

print(f"data type = {type(list3)}")

print(f"length = {len(list3)}")

print()

#CODE4: Create LIST with multiple items of different types

list4 = ["cloud", "academy", 1, True, False]

print("CODE4:")

print(f"list4 = {list4}")

print(f"data type = {type(list4)}")

print(f"length = {len(list4)}")

print()

#CODE5: Iterate over LIST with multiple items

print("CODE5:")

for item in list4:

print(item)

print()

#CODE6: Search in LIST

print("CODE6:")

print ("cloud" in list4)

print ("blah" in list4)

print()

#CODE7: Retrieve item from LIST

print("CODE7:")

item0 = list4[0]

item1 = list4[1]

print(f"item0 = {item0}")

print(f"item1 = {item1}")

print()

#CODE8: Change mutable LIST

print("CODE8:")

list4[0] = "possible!!"

print(f"list4 = {list4}")

print(f"data type = {type(list4)}")

print(f"length = {len(list4)}")

print()

#CODE9: Append to LIST

print("CODE9:")

list4.append("new item")

print(f"list4 = {list4}")

print(f"data type = {type(list4)}")

print(f"length = {len(list4)}")

print()

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

output console

python3 lists.py

Exercise: Lists

Type: solution-code

Python: 3.7.0

CODE1:

list1 = []

data type = <class 'list'>

length = 0

CODE2:

list2 = ['cloudacademy']

data type = <class 'list'>

length = 1

CODE3:

list3 = [1, 2, 3, 4, 5]

data type = <class 'list'>

length = 5

CODE4:

list4 = ['cloud', 'academy', 1, True, False]

data type = <class 'list'>

length = 5

CODE5:

cloud

academy

1

True

False

CODE6:

True

False

CODE7:

item0 = cloud

item1 = academy

CODE8:

list4 = ['possible!!', 'academy', 1, True, False]

data type = <class 'list'>

length = 5

CODE9:

list4 = ['possible!!', 'academy', 1, True, False, 'new item']

data type = <class 'list'>

length = 6

theia@production-session-93142-6dd54db79c-z8dtq:/home/project/code/Lists/solution-code$