Working with Numeric Data Types

Lab overview

Python makes it easier to do math. In fact, Python is a popular language among data scientists, who must analyze large amounts of data. In this lab, you will explore the basic data types that are used to store numeric values.

In this lab, you will:

- Use the Python shell
- Use the int data type
- Use the float data type
- Use the complex data type
- Use the bool data type

Estimated completion time

60 minutes

Accessing the AWS Cloud9 IDE

1. Start your lab environment by going to the top of these instructions and choosing **Start Lab**.

   A **Start Lab** panel opens, displaying the lab status.

2. Wait until you see the message *Lab status: ready*, and then close the **Start Lab** panel by choosing the **X**.

3. At the top of these instructions, choose **AWS**.

   The AWS Management Console opens in a new browser tab. The system automatically logs you in.

   > **Note:** If a new browser tab does not open, a banner or icon at the top of your browser typically indicates that your browser is preventing the site from opening pop-up windows. Choose the banner or icon, and choose **Allow pop ups**.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

4. In the AWS Management Console, choose **Services** > **Cloud9**. In the **Your environments** panel, locate the **reStart-python-cloud9** card, and choose **Open IDE**.

   The AWS Cloud9 environment opens.

> **Note:** If a pop-up window opens with the message *.c9/project.settings have been changed on disk*, choose **Discard** to ignore it. Likewise, if a dialog window prompts you to *Show third-party content*, choose **No** to decline.

Creating your Python exercise file

5. From the menu bar, choose **File > New From Template > Python File**.

   This action creates an untitled file.

6. Delete the sample code from the template file.

7. Choose **File > Save As...**, and provide a suitable name for the exercise file (for example, *numeric-data.py*) and save it under the **/home/ec2-user/environment** directory.

Accessing the terminal session

8. In your AWS Cloud9 IDE, choose the **+** icon and select **New Terminal**.

   A terminal session opens.

9. To display the present working directory, enter `pwd`. This command points to **/home/ec2-user/environment**.

10. In this directory, you should also be able to locate the file you created in the previous section.

Exercise 1: Using the Python shell

In the terminal tab, a Python shell can be started by entering the following command:

```
python3
```

The Python shell should look similar to the following example.

```
Python 3.6.12 (default, Aug 31 2020, 18:56:18)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-28)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

The three greater-than symbols (`>>>`) represent the prompt where the user can enter Python commands. In the following activities, you will practice using the Python shell by issuing some numeric commands.

Adding

11. Enter the following input:

```
2 + 2
```

12. Press ENTER.

13. Confirm that you get *4* as output.

Subtracting

14. Enter the following input

```
4 - 2
```

15. Press ENTER.

16. Confirm that you get *2* as output.

Multiplying

To multiply, you use the ***** symbol:

17. Enter the following input:

```
2 + 2
```

18. Press ENTER.

19. Confirm that you get *4* as output.

Dividing

To divide, use the **/** symbol:

20. Enter the following input:

```
4 / 2
```

21. Press ENTER.

22. Confirm that you get *2.0* as output.

Exiting the Python shell

23. To exit the Python shell, enter the following command:

```
quit()
```

Exercise 2: Introducing the int data type

To learn more about data types, you will use some built-in functions. A *function* is a piece of reusable code with a name. You use a function by:

- Calling by its name
- Including a list of one or more inputs called *arguments*, which are enclosed in parentheses

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
17

Python has several built-in functions that you can use to help you write more useful programs.

A collection of functions is called a *library*. Python’s collection of built-in functions is called the *Python Standard Library*.

Editing a Python file

Instead of entering commands one by one in the Python shell, you will edit a text file that contains a sequence of commands:

24. From the navigation pane of the IDE, choose the file that you created in the previous *Creating your Python exercise file* section.
25. In the file, enter the following code:

    ```
    print("Python has three numeric types: int, float, and complex")
    ```

26. To save the file, choose **File > Save**.
27. At the top of the IDE window, choose **Run** (the **Play** button).
28. In the bottom (console) pane of the IDE, confirm that the program prints the message: *Python has three numeric types: int, float, and complex*

    Note: You might need to scroll up to see the console output.

29. In the terminal tab, you can also run the program by entering the following command, where <*lab-python-file-name*> is the name of the file that you created for this lab:

    ```
    python3 <lab-python-file-name>.py
    ```

30. Confirm that the text you wrote is written to standard output.

    ```
    ~ $ python3 <lab-python-file-name>.py
    Python has three numeric types: int, float, and complex
    ```

Creating a variable

A variable is like a labeled box that stores information. You can change the contents of the box, but the label stays the same. In this activity, you will use the variable name *myValue*, but will store different data types in that labeled box.

31. Return to the Python file and on a new line, enter the following code:

    ```
    myValue=1
    ```

32. Use the `print()` function to write the value of the variable to the shell. In the context of programming, *writing* means to add information to the shell.

    ```
    print(myValue)
    ```

33. To get the data type of the variable, use the `type()` built-in function:

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
17

```
print(type(myValue))
```

34. To combine numbers and text, use the `str()` built-in function, which converts an argument into a collection of letters called a *string*. In this instance, you are converting the int (integer) data type into the *string* data type:

```
print(str(myValue) + " is of the data type " + str(type(myValue)))
```

35. Save the file.

36. To run the file, choose **Run**.

37. In the bottom pane of the IDE, confirm that you have the following output:

```
Python has three numeric types: int, float, and complex
1
<class 'int'>
1 is of the data type <class 'int'>
~ $
```

> **Note:** You might need to scroll up to see the output.

Exercise 3: Introducing the float data type

The int data type only stores whole numbers. If you want to store a number with a decimal, like *3.14*, you need a new data type called a *float*.

38. Return to the Python file and on a new line, enter the following code:

```
myValue=3.14
```

39. To write the value of the variable to the shell, use the `print()` function:

```
print(myValue)
```

40. Get the data type of the variable by using the `type()` built-in function:

```
print(type(myValue))
```

41. To combine numbers and text, use the `str()` built-in function:

```
print(str(myValue) + " is of the data type " + str(type(myValue)))
```

42. Save the file.

43. To run the file, choose **Run**.

44. In the bottom pane of the IDE, confirm that you see the following output:

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

```
Python has three numeric types: int, float, and complex
1
<class 'int'>
1 is of the data type <class 'int'>
3.14
<class 'float'>
3.14 is of the data type <class 'float'>
~ $
```

> **Note:** Recall that you might need to scroll up to see the output.

Exercise 4: Introducing the complex data type

In advanced math, an imaginary number is a complex number that can be written as a real number that is multiplied by the imaginary unit *i*. This complex data type is complicated because it must represent a letter and a number, such as *5j*.

45. Return to the Python file and enter the following code:

```
myValue=5j
```

46. Write the value of the variable with the `print()` function:

```
print(myValue)
```

47. Get the data type of the variable with the `type()` function:

```
print(type(myValue))
```

48. To combine numbers and text, use the `str()` built-in function:

```
print(str(myValue) + " is of the data type " + str(type(myValue)))
```

49. Save the file.

50. To run the file, choose **Run**.

51. In the bottom pane of the IDE, confirm that you have the following output:

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
17

```
Python has three numeric types: int, float, and complex
1
<class 'int'>
1 is of the data type <class 'int'>
3.14
<class 'float'>
3.14 is of the data type <class 'float'>
5j
<class 'complex'>
5j is of the data type <class 'complex'>
~ $
```

> **Note:** Recall that you might need to scroll up to see the output.

Exercise 5: Introducing the bool data type

The bool (Boolean) data type comprises the permanent names *True* and *False*, which are represented by the numerals *1* and *0*, where *1* = *True* and *0* = *False*. The bool data type is implemented as a subset of int and is not considered a real data type. However, in some programming languages, it is implemented as a different data type. These exercises call the Python bool a *fake data type*.

52. Return to your text file, and enter the following code:

```
myValue=True
```

53. Write the value of the variable to the shell by using the `print()` function:

```
print(myValue)
```

54. Get the data type of the variable by using the `type()` built-in function:

```
print(type(myValue))
```

55. To combine numbers and text, use the `str()` built-in function:

```
print(str(myValue) + " is of the data type " + str(type(myValue)))
```

56. Save the file.

57. Choose **Run** (the **Play** button).

58. In the bottom pane of the IDE, confirm that it displays the correct output.

59. Return to your **.py** file and enter the following code:

```
myValue=False
```

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
17

60. Use the `print()` function to write the value of the variable to the shell:

```
print(myValue)
```

61. To get the data type of the variable, use the `type()` built-in function:

    ```
    print(type(myValue))
    ```

62. To combine numbers and text, use the `str()` built-in function:

    ```
    print(str(myValue) + " is of the data type " + str(type(myValue)))
    ```

63. Save the file.

64. Choose **Run** (the **Play** button).

65. In the bottom pane of the IDE, confirm that you have the following output:

    ```
    Python has three numeric types: int, float, and complex
    1
    <class 'int'>
    1 is of the data type <class 'int'>
    3.14
    <class 'float'>
    3.14 is of the data type <class 'float'>
    5j
    <class 'complex'>
    5j is of the data type <class 'complex'>
    True
    <class 'bool'>
    True is of the data type <class 'bool'>
    False
    <class 'bool'>
    False is of the data type <class 'bool'>
    ~ $
    ```

Congratulations! You have learned about Python’s three numeric data types: int, float, and complex. Additionally, you were introduced to the Python fake data type that is called *bool*. Note that bool is actually the numerals *0* and *1*, which represent the values of *True* and *False*.___

End Lab

Congratulations! You have completed the lab.

66. Choose **End Lab** at the top of this page, and then select Yes to confirm that you want to end the lab.

    A panel indicates that *DELETE has been initiated... You may close this message box now.*

67. A message *Ended AWS Lab Successfully* is briefly displayed, indicating that the lab has ended.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

17

Additional Resources

For more information about AWS Training and Certification, see https://aws.amazon.com/training/ (https://aws.amazon.com/training/).

Your feedback is welcome and appreciated. If you would like to share any suggestions or corrections, please provide the details in our AWS Training and Certification Contact Form (https://support.aws.amazon.com/#/contacts/aws-training).

○ Yes

○ No

< Rubric: 2 - Numeric Data Types | Points: 0 >

Previous Next

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17