Lab 2: Expressions and Control Structures

Due at 11:59pm on 01/28/2015.

Starter Files

Download lab02.zip. Inside the archive, you will find starter files for the questions in this lab, along with a copy of the OK autograder.

Submission

By the end of this lab, you should have submitted the lab with . You may submit more than once before the deadline; only the final submission will be graded.

- To receive credit for this lab, you must complete Questions 6, 9, and 10 in lab02.py and submit through OK.
- Questions 1, 2, 3, 4, 5, 7, and 8 (What Would Python Print?) are designed to help introduce concepts and test your understanding.
- Questions 11, 12, and 13 are optional . It is recommended that you complete these problems on your own time.

Table of Contents

Using Python

When running a Python file, you can use on the command line to inspect your code further. Here are a few that will come in handy. If you want to learn more about other Python flags, take a look at the documentation.

- Using no flags will run the code in the file you provide and return you to the command line.
-
- : The option runs your Python script, then opens an interactive session.
-
- : Runs doctests in a particular file. Doctests are marked by triple quotes () and are usually located within functions.
-

Using OK

In 61A, we use a program called OK for autograding labs, homeworks, and projects. You should have downloaded ok at the start of this lab. You can use ok to run doctests for a specified function. For example,

You can also use the option: if an error occurs, an interactive interpreter will open, allowing you to enter Python commands to test out your program:

By default, only errors will show up. You can use the option to show all tests, including successful ones:

Finally, when you have finished all the quesitons in lab02.py, you can submit the assignment using the option:

Expressions

Question 1: What would Python print?

Think about what the output of each of the following expressions will be. Then type them into Python to verify your answers!

Primitive Expressions

A requires only a single evaluation step: use the literal value directly. For example, numbers, names, and strings are all primitive expressions.

Call Expressions

A call expression applies a function, which may or may not accept arguments. The call expression evaluates to the function's return value.

The syntax of a function call:

Every call expression requires a set of parentheses delimiting its comma-separated operands.

To evaluate a function call:

1. Evaluate the operator, and then the operands (from left to right).
2. Apply the operator to the operands (the values of the operands).

If an operand is a nested call expression, then these two steps are applied to that operand in order to evaluate it.

Question 2: What Would Python Print?

Division

Let's compare the different division-related operators in Python:

- •True Division (decimal division) with the operator:
-
-
-
-
-
-
- •Floor Division (integer division) with the operator:
-
-
-
-
-
-
- •Modulo (similar to a remainder) with the operator:
-
-
-
-
-
-

•One useful technique involving the operator is for checking whether a number is divisible by another number :

•

Pure and Non-Pure Functions

1.Pure functions have no side effects – they only produce a return value. They will always evaluate to the same result, given the same argument value(s).
2.Non-pure functions produce side effects, such as printing to your terminal or returning different outputs on different invocations of the function (non-determinism).

Later in the semester, we will expand on the notion of a pure function versus a non-pure function.

Question 3: What Would Python Print?

Boolean operators

Question 4: What Would Python Print?

What would Python print? Try to figure it out before you type it into the interpreter!

Boolean order of operations

What do you think the following expression evaluates to?

It turns out that Python interprets that expression in the following way:

Using parentheses can be helpful to understand how a program will behave.

Boolean operators, like arithmetic operators, have an order of operation:

- has the highest priority
-
- has the lowest priority

Short-circuit operators

In Python, and are examples of . Consider the following code:

Notice that if we just evaluate , Python will raise an error, stopping evaluation altogether! However, the original line of code will not cause any errors — in fact, it will evaluate to . This is made possible due to short-circuiting, which works as follows:

- For statements, Python will go left to right until it runs into the first value that is false-y — then it will immediately evaluate to that value. If all of the values are truth-y, it returns the last value.
- For statements, Python will go left to right until it runs into the first value that is truth-y — then it will immediately evaluate to that value. If all of the values are false-y, it returns the last value.

Informally, false-y values are things that are "empty". The false-y values we have learned about so far are , , , and (the empty string).

Question 5: What Would Python Print?

Question 6: Fix the Bug

The following snippet of code doesn't work! Figure out what is wrong and fix the bugs.

You can test your solution by using OK:

If statements

Question 7: What Would Python Print?

While loops

Question 8: What Would Python Print?

Question 9: Factor This II

Define a function which takes in a number, , and prints out all of the numbers that divide evenly. For example, the factors of 20 are 20, 10, 5, 4, 2, 1.

You can test your solution by using OK:

Question 10: Fibonacci

The Fibonacci sequence is a famous sequence in mathematics. The first element in the sequence is 0 and the second element is 1. The th element is defined as .

Implement the function, which takes an integer and returns the th Fibonacci number. Use a loop in your solution.

You can test your solution by using OK:

Error messages

By now, you've probably seen a couple of error messages. Even though they might look intimidating, error messages are actually very helpful in debugging code. The following are some common types of errors (found at the bottom of an error message):

- •: Indicates that your code contains improper syntax (e.g. missing a colon after an statement).
- •: Indicates that your code contains improper indentation (e.g. inconsistent indentation of a function body)
- •: Indicates an attempted operation on incompatible types (e.g. trying to add a function and an int)
- •: Indicates an attempted division by zero.

Using these descriptions of error messages, you should be able to get a better idea of what went wrong with your code. You can often Google unknown error messages to see what similar mistakes others have made to help you debug your own code.

For example:

Notice that the last line of the error message tells us exactly what we did wrong - we gave 2 arguments when it only takes in 1 argument. In general, the last line is the most helpful. Here's a link to an extremely helpful Debugging Guide written by Albert Wu. It is highly recommended that you read this in its entirety! Pay particular attention to the section called "Error Types" (the other sections are fairly involved but will be useful in the larger projects).

Extra Questions

Questions in this section are not required for submission. However, we encourage you to try them out on your own time for extra practice.

Question 11: Disneyland Discounts

Disneyland is having a special where they give discounts for grandparents accompanying their grandchildren. Help Disneyland figure out when the discount should be given. Define a function that takes two numbers as input (representing the two ages) and returns if one of them is a senior citizen (age 65 or above) and the other is a child (age 12 or below). You should not use in your solution.

You can test your solution by using OK:

Question 12: Factor This

Define a function that checks whether its first argument is a factor of its second argument. We will assume that is not a factor of any number but any non-zero number is a factor of . You should not use in your solution.

You can test your solution by using OK:

Question 13: Factorials

Let's write a function , which is a "falling" factorial that takes two arguments, and , and returns the product of consecutive numbers, starting from and working downwards.

You can test your solution by using OK: