CSC108H Lab 2

Objectives

• Explore module math using help and dir

• Use type bool

• Practice submitting an assignment

• Write if statements

`math` Module

As you learned in lecture, Python provides a large number of useful functions, many of which are stored in separate modules. In this section, we will explore the `math` module. We have provided you with a few tasks to complete, and you should use functions from the math module to do so.

You should use `dir(math)` to find out which functions math contains, and `help(math.function)` (where function is replaced by the name of a function from `math`) will provide an English description of a function. `help(math)` will provide English descriptions of all the functions in `math`.

To begin, import math. In the shell, write code to do the following:

• Take the floor and ceiling of 56.4. • Calculate the square root of the logarithm of 244. • Calculate the base 2 logarithm of 256. • Calculate the absolute value of -34.84. • Calculate 2π.

Type `bool` and the Poisonous Potions Problem

Boolean logic employs two values, True and False, and three basic Boolean operators, `and`, `or`, and `not`. If a and b are Boolean variables:

- `not a` is True iff a is False
- `a and b` is True iff a is True and b is True
- `a or b` is True iff at least one of a and b is True

Similarly, "`a and b and c`" will evaluate to True only if all three variables have the value True and "`a or b or c`" will be True if any one or more of the three variables is True.

This part of your lab is about a logic puzzle involving four potions. We use four boolean variables, `p1` to `p4`, to represent the potions; a True value indicates that the potion is poisonous, and a False value indicates that it is not.

One of the potions is poisonous. We won't tell you which, but here are five hints:

1. At least one of the four potions is poisonous.
2. p2 is not poisonous.
3. At least one of p1 and p3 is poisonous.
4. At least one of p3 and p4 is not poisonous.
5. Exactly one of the potions p1, p3, and p4 is poisonous.

The potions puzzle can be represented in Python. Download `puzzle.py` from the Labs folder on the course website. The purpose of this program is to test the user's hypothesis about the answer to the puzzle. Look at the main block. For each potion, it asks the user whether they think it is poisonous or not. Then it calls function solution to find out if the user is correct, that is, to find out if their hypothesis satisfies all 5 hints.

Function `solution` makes calls to five "helper" functions, `hint1` to `hint5`. Each one checks whether the hypothesis satisfies that one hint. For example, if you call `hint1(True, True, False, True)`, you are testing the guess that p1, p2, and p4 are poisonous, and p3 is not. This satisfies hint 1, so function `hint1` will return True.

`hint1(p1, p2, p3, p4)`	Return True if one or more of the potions is poisonous and False otherwise.
`hint2(p2)`	Return True if potion p2 is not poisonous, and False otherwise.
`hint3(p1, p3)`	Return True if least one of potions p1 and p3 is poisonous, and False otherwise.
`hint4(p3, p4)`	Return True if at least one of potions p3 and p4 is not poisonous, and False otherwise.
`hint5(p1, p3, p4)`	Return True if exactly one of the given potions is poisonous, and False otherwise.

Function `hint1` has already been written, but the others have not. (As a result the program will crash if you run it.) Write the other four functions according to the descriptions in the table on the previous page, using `hint1` as an example of what the code should look like. Add the functions to `puzzle.py`, switching driver and navigator after each one.

Once you have added the rest of the hint functions, run the program and try to solve the puzzle. If the program gives you the "sorry" message, either your solution really is incorrect or one of the hint functions is wrong!

Once you have found the solution, show your TA your working puzzle code and then proceed to the next section.

Practice submitting an assignment

The assignments in this course will be submitted electronically using the MarkUs submission system. MarkUs can be found at:

https://stanley.cdf.toronto.edu/markus/csc108-2011-01/

To help you learn how to submit electronically, you are going to declare your partnership to MarkUs and then submit your file `puzzle.py` as if it were an assignment. Here's what to do:

1. Log in: One of you should go to the MarkUs website and log in, using your cdf account name and password.

2. Go to the assignment: Once you log in, you will see a list of the course assignments. So far, that includes only the fake assignment called LabPractice. Click on LabPractice to see a page of information about the assignment. It includes the fact that you don't have a "group" (in this case just a partner) yet.

3. Invite your partner: Under Group Information, click Create to indicate that you want to create a group rather than work alone. Then click Invite. In the box that pops up, type your partner's cdf username. You have now invited them to be your official partner (just for this LabPractice exercise).

4. Switch users: Now log out (look in the upper-right corner of the window) and let your partner log in.

5. Accept the invitation: Now that you, the second student in the pair, are logged in, click on assignment LabPractice and your personal version of the assignment information will appear. It includes the fact that you have been invited to be a partner. Under Group Information, click Join to accept the invitation.

6. Submit work: Now the two of you are ready to submit your work. Click on the submissions tab (either partner can do this). You'll see that you are "required" to submit `puzzle.py`. Click Add a New File, click Browse to choose it, and then click Submit. You're done!

You can check that you submitted the right file by clicking on a file name in MarkUs and looking at its contents.

This is the only time that you will need to submit your lab work electronically and what you submit will not be marked; this is just for practice submitting. Also, the partnership that you declare will not carry over to the real assignments. If you want to work with the same person again you can, but you will declare that separately for each assignment.

Switch roles: s1 drives and s2 navigates: s1 should also practice submitting `puzzle.py`.

if statements

Download the file ifs.py. In it you will see two function definitions:

`leap(year):`

checks to see if a year is a leap year. The rules for a leap year are: any year divisible by 4 is a leap year except years that end in 00. If a year ends in 00, then it is a leap year if when you remove the 00, the remaining part of the year is divisible by 4. Otherwise, it is not.

Example: 1995, 1998 and 1700 are not leap years. 1600, 1996 and 2004 are.

Write the body of the function so that it returns True if the year is a leap year and False otherwise. The function assumes that parameter `year` is a positive integer.

`fizzbuzz():`

prints the numbers from 1 to 100 in order, on separate lines. There is already a `while` loop in the function doing this.

Your task is to modify the code inside the `while` loop so that:

- if the current number is divisible by 3, instead of it the function prints "Fizz"
- if the current number is divisible by 5, instead of it the function prints "Buzz"
- if the current number is divisible both by 3 and by 5, instead of it the function print "FizzBuzz"

The FizzBuzz test is actually a very popular interview technique for IT jobs. Interviewers will often ask programmers to solve this problem on paper in a job interview. Staggeringly, only 1 in 200 applicants for programming jobs can do it. So, if you solve this correctly, you could be in the top 0.5% of programming job applicants!

You can run `ifs.py` in Wing and call `leap()` and `fizzbuzz()` from the Python shell in Wing to test them.

Show the TA your work, and you're done! See you next week!