CSC120H Lab 3

1 Objectives

- Practice using boolean values
- Practice writing boolean expressions and conditional statements
- Practice writing functions with conditional if-statements.

2 Driver and navigator

As always, you **must** complete this lab **with a partner**, and you and your partner will take on distinct roles:

driver: The person typing at the keyboard.

navigator: The person watching for mistakes, and thinking ahead.

The rest of these instructions call you `s1` and `s2`. Pick which one is which. `s1` should log in, start up Wing, and be the first driver.

3 Boolean warmup

A student may get into CSC267 (not a real course) if they have a GPA of 2.7 or higher, and if they have taken at least one of CSC108 or CSC120.
Once you answer a question, use the Python shell to check your answer.

Variable `gpa` represents a student's current GPA, and `csc108` and `csc120` are Boolean variables: `csc108` is `True` if and only if the student has taken CSC108, and `csc120` is `True` if and only if the student has taken CSC120.

1. Write an expression, using variables `gpa`, `csc108` and `csc120`, that produces `True` if and only if the student can take CSC267.
2. Now, assume that we don't know what the student's actual GPA is, (we don't have variable `gpa`) but we have a Boolean variable `gpa_okay` that represents whether the student's GPA is at least 2.7.

   Write an expression, using variables `gpa_okay`, `csc108` and `csc120`, that produces `True` if and only if the student can take CSC267.
3. If the student is in 3rd year or higher, they can automatically take CSC267 if their GPA is 3.2 or higher. The variable `year` is an integer representing the year the student is in. Write an expresion using variables `gpa` (the value of the gpa), `csc108`, `csc120`, and `year`, that produces `True` if and only if the student can take CSC267. **You do not need an if-statement for this question.**

4 If statements

In this part of the lab, you will practise writing functions that use boolean values and if-statements.
Create a new Python file `lab3_part4.py` and use the Function Design Recipe to design the functions below. For **every** function, make sure to provide good parameter names, a correct type contract, and a proper docstring with a description and examples. The Function Design Recipe Summary can be found on the Lectures page of our course website, if you need it a reminder on the five steps.
Switch driver and navigator.

- **How long will it take?**
  A common calculation for estimating a runner's marathon time is to take their best half-marathon time, multiply by two, and add ten minutes. This works pretty well unless the marathon course is hilly. If it is hilly, add an extra 20 minutes to the estimate.
  Write the function `marathon_time` that takes a half-marathon time in minutes and whether or not the marathon course is hilly as arguments, and returns the estimated number of minutes the runner will take to run the full marathon.

  Create four test cases (more than just the ones you used in your docstring) for your function, and call your functions and use `print` to print the results of your function calls to the shell. Remember to call `print` **outside** of your function. Make sure you get the values you expect.

Switch driver and navigator.

- **Going out with the gang**
  You are planning an outing with some friends and need to calculate the total price of the tickets. A regular ticket usually costs $3.99 and a student ticket costs only $2.99. If the total number of tickets being ordered (including both non-students and students) is at least 10, then a 10% discount is applied to the order. However, if it is a holiday, the group discount is only 5%.

  Write the function `total_ticket_price` that given three arguments (the number of regular tickets, the number of student tickets and whether or not it is a holiday), computes and returns the total price of the tickets.

  Create four test cases and `print` the result of the function calls to the shell.

Switch driver and navigator.

- **Is it time for tea?**
  Your friend lives in London, England in a timezone that is normally 6 hours ahead of Toronto. But sometimes England and Canada switch to daylight savings time on different dates. If Toronto is on daylight savings time and England isn't, then the time difference is only 5 hours. Conversely if England is on daylight savings time and Toronto isn't, the time change is 7 hours. Of course if both are on daylight savings time, the difference is back to the standard 6 hours.

  Write the function `british_time` whose first argument is a value representing the time in Toronto from 0 to 24 (the time can be a decimal number). The next two arguments are boolean values indicating whether or not Toronto and London are on daylight savings time respectively. Your function should return the time in London, England.

  One last complication is that your function should not return a time greater than 24.0 or less than 0.0
  Suggestion: At first, don't consider this final complication. Run some test cases your where the expected output is already less than 24.0 (in the same day.) Once that is working, switch driver and navigator and add another example where the time in Toronto is so late at night that London is already the next morning. Fix the body of your function to work correctly.

  Create enough test cases in your to cover all the different possibilities of boolean value combinations. Test Toronto times close to 24 as well. `print` your test cases to the shell.

Once you have completed the functions, demonstrate them to your TA. Show your TA your functions and your example calls. If you run out of time and would like to finish the lab on your own, the handout will be posted on the Labs webpage.