Computer Science I for majors by James Tam | Return to the course web page

CPSC 231: Mini-Assignment 3

Due Monday June 5 at 5 PM

New Concepts to be applied for the assignment

- Defining functions
- Parameter passing
- Function return values

Description:

JT's hint: knowing your terminology and having a deep understanding of concepts is always important before starting an assignment. It is imperative that you complete this prerequisite work for this assignment otherwise you will not understand the requirements below.

Write a program that consists of 4 functions:

1) **`start()`**: the starting execution point of the program. All instructions not defined in the other functions (save for the initial call to start) must be contained in the body of this function. *Worth 1 GPA*.

2) **`getInput()`**: the first function called by `start()`. Prompts and gets from the user for a number (which can include a fractional component). This number is then returned back to the caller (which is the start function). *Worth 1 GPA*

3) **`double()`**: the second function called by `start()`. Takes as an input/parameter the number returned by `getInput()` and it returns back to `start()` this number doubled. *Worth 1 GPA*

3) **`display()`**: the third function called by `start()`. Takes as an input/parameter the number returned by `double()` and displays it to the console (text-based output). *Worth 1 GPA*

Summary of functions to define:

Function name	Inputs/parameters	Return values	Responsibility of the function
`getInput()`	None	Float (numeric assumed)	Prompt and get a number from that user, return that number back to the caller of the function
`double()`	Float (numeric assumed)	Float (numeric assumed)	Take a number as a parameter, return the double of that number
`display()`	Float (numeric assumed)	None	Take a number as a parameter, display the number to the console (text-output)
`start()`	None	None	Starting execution point of the program, contains the calls to the other 3 functions

You can still practice applying good style in your solution as well as writing documentation. Unlike the full assignments you will be just graded on program functionality for the mini-assignments.

D2L configuration:

- Multiple submissions are possible for each assignment: You can and should submit many times before the due date. D2L will simply overwrite previous submissions with newer ones.
- **Important**! Multiple files can be submitted for each assignment. I am allowing you to submit multiple files for each assignment so **don't archive/compress multiple files using a utility such as zip**. However, this means that **TAs will only mark the latest versions** of each file submitted via D2L. Even if the version of a document that you want marked has been uploaded into D2L if it isn't the latest version then you will only get marks for the latest version. (It's unfair to have the TAs check versions or to remark assignments because marking is enough work as-is).

Marking

- Assignments will be marked by your tutorial instructor (the "Teaching Assistant" or "TA"). When you have questions about marking this is the first person that you should be directing your questions towards. If you still have question after you have talked to your TA, then you can talk to your course (lecture) instructor.
- Here is the [marking checklist] that specifies the grade breakdown for specific parts of the assignment. Besides seeing your grade point in D2L you can also see the detailed feedback that your TA will enter for each student. You can access the grading sheet in D2L under `Assessments->Dropbox` and then clicking on the appropriate assignment link. If you still cannot find the grading sheet then here is a [help link]

Points to keep in mind:

1. **Due time:** All assignments are due at 5 **PM** on the due dates listed on the course web page. Late assignments or components of assignments will not be accepted for marking without approval for an extension beforehand. Alternate submission mechanisms (non exhaustive list of examples: email, uploads to cloud-based systems such as Google drive, time-stamps, TA memories) cannot be used as alternatives if you have forgotten to submit work or otherwise have not properly submitted into D2L. **Only files submitted into D2L by the due date is what will be marked**, everything else will be awarded no credit.
2. **Extensions** may be granted for reasonable cases by the course instructor with the receipt of the appropriate documentation (e.g., a doctor's note). Typical examples of reasonable cases for an extension include: illness or a death in the family. Example cases where extensions will not be granted include situations that are typical of student life: having multiple due dates, work commitments etc. Tutorial instructors (TA's) will not be able to provide extension on their own and must receive permission from the course instructor first.
3. **Method of submission:** You are to submit your assignment using D2L [help link]. Make sure that you [check the contents of your submitted files] (e.g., is the file okay or was it corrupted, is it the correct version etc.). It's your responsibility to do this! (Make sure that you submit your assignment with enough time before it comes due for you to do a check).
4. **Identifying information**: All assignments should include contact information (full name, student ID number and tutorial section) at the very top of your program in the class where the '`start()`' function resides (or the starting execution point for assignments that don't require functions to be defined). (Note other documentation is also required for most full assignments).
5. **Collaboration**: Assignments must reflect individual work; group work is not allowed in this class nor can you copy the work of others. For more detailed information as to what constitutes academic misconduct (i.e., cheating) for this course please read the following [link].

6. **Execution**: programs must run on the computer science network running Python 3.x. If you write you code in the lab and work remotely using a remote login program such as Putty or SSH. If you choose to install Python on your own computer then it is your responsibility to ensure that your program will run properly here. It's not recommended that you use an IDE for writing your programs but if you use one then make sure that you submit your program in the form of text "`.py`" file or files.

7. **Use of pre-created Python libraries**: unless otherwise told you are to write the code yourself and not use any pre-created functions. For this assignment the usual acceptable functions include: `print()`, `input()` and the 'conversion' functions such as `int()`, `float()`, `str()`. Look at the particular assignment description for a list of other classes that you are allowed to use and still get credit in an assignment submission.