CodeWrite
Simon Fraser University
Welcome to CodeWrite

ASSIGNMENT #6.
READ THE WHOLE DESCRIPTION, INSTRUCTIONS AND REQUIREMENTS

The required exercises were created to encourage solving the questions using while loops, and possibly including two conditions. You could solve the questions using for loops, or while loops with a single condition and using break in some cases, but you miss the opportunity to practice with while loops. You may want to attempt more than one solution (within Codewrite or also outside Codewrite).

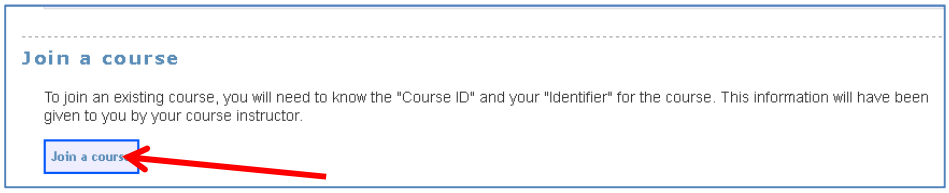
First, Optional DEADLINE:
If you want that your invented question is considered to be included in the Midterm
Submit by Sunday July 10, 5:00 PM -- NOTICE THE TIME

Second, required DEADLINE:
Wednesday 13 , 5:00 PM -- NOTICE THE TIME

- **solving the REQUIRED** (and a possible BONUS) exercises (solutions will be posted immediately after the deadline)
- **inventing your own questions (of type "PEER")**
- **solving PEER** questions

You need to join and work on the **Codewrite (CW) course #24**: "CMPT 120 2016-2 Assignment 6".

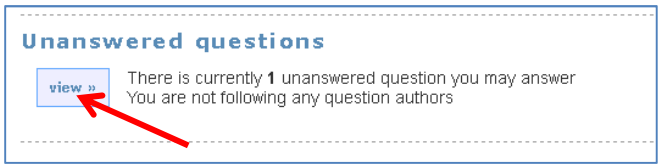
To join a new course in CW, If you did the previous CW assignments, just login to CW and scroll down until you find the option:



If you did not work with Codewrite before, you need to register. Follow the instructions as in Assignment 4.1, but this time registering to the Codewrite course #24.

WHAT YOU NEED TO DO

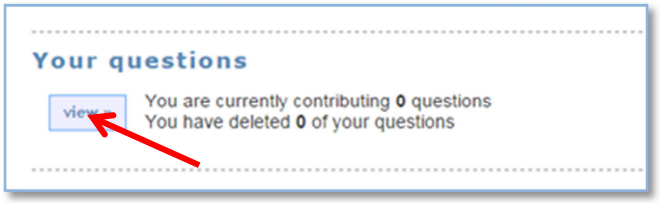
1) **SOLVING REQUIRED (and maybe a BONUS) QUESTIONS**
   Everyone will need to solve the 6 (six) unanswered questions. The question description starts with (REQ) and the topic is REQUIRED – You may list only those questions by choosing topic REQUIRED).
   You may want to solve a BONUS question. The question description starts with (BONUS) and the topic is BONUS – You may list only this question by choosing topic BONUS).



Similarly to the previous Codewrite assignment,

One exercise (i.e. one function) is considered solved if it is **CORRECT FOR ALL THE TEST CASES** (all PASS).

2) INVENTING YOUR OWN 2 (TWO) QUESTIONS (with **topic PEER**) .



Detailed guided instructions to invent questions and requirements for inventing are included below. You may invent more than 2 questions, but points will be given for inventing up to 2 questions. Make sure that you include the topic PEER when you invent them.

3) SOLVE 5 PEER INVENTED PROBLEMS (with **topic PEER)**

Solve 5 (five) questions of topic: PEER. Choose interesting questions! You can certainly solve more for practice, but only 5 PEER questions will be considered for points.

Marking scheme:

Everyone will receive points for just working in this CW assignment (3 points) , 2 points for each of the 6 instructor defined exercises (topic REQUIRED) correctly solved , 2 points for each correctly invented and solved question, and 2 points for correctly solving (up to 5) PEER questions. Additionally you may get 3 extra points if you correctly solve the BONUS question.

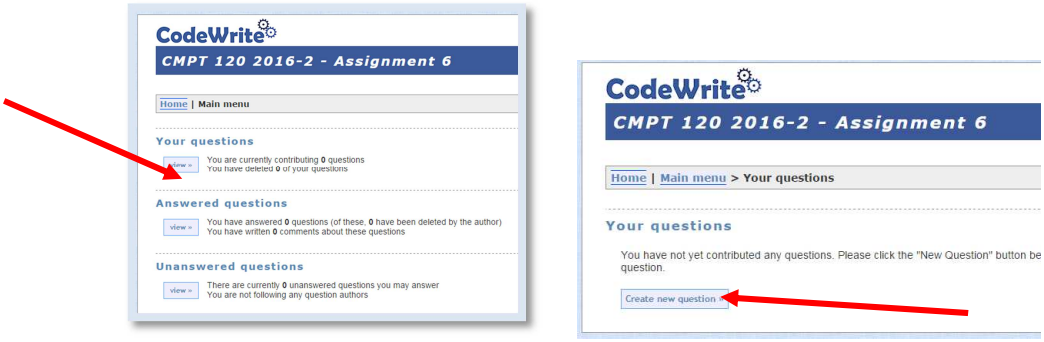
Given the marking scheme, the maximum is 29 points, however you may get up to 32 points. (Note: (The so called "score" in Codewrite is not used in the points calculations).

Maximum considered: 3 + 2*6 + 2*2 + 2*5 = 29 points
Maximum points including bonus: 29+3 = 32 points

This assignment is worth 3% of the whole course.

GUIDED INSTRUCTIONS AND REQUIREMENTS - INVENTING YOUR OWN QUESTIONS

Follow the link: "view >>" under **"Your questions"** , and then **Create New Question**



The questions that you create, just like the ones you worked on with the previous assignments, will require a single function to be written to solve some task. It is up to you what the task is. Make the question relevant to the course! Not too easy, not too complex! You need to decide:

- what the name of the function should be
- how many input parameters it should have and the type of the parameters
- what the result should be
- create the **function as productive**, that is, so that it **returns** a value, and so that the printing is done in the test case, when calling or invoking the function outside the function definition, as it was done in previous Codewrite assignments.

Function description:

- make the **function name** clear of its purpose (but not too long) and mention it early in the description, so that it is visible in the list of questions available to answer
- you may want to provide a name to each of the **input parameter/s**. Name the parameters so that it is clear what their role is (but it is not a too long name). Avoid naming variables using keywords or functions in the language, such as str or max
- provide a clear description of what your function is supposed to do – this will help other students who choose to answer your exercise.

Your solution

- Provide your solution . Notice that there is some help/ description in Codewrite referring to a strongly typed language reminding you to include types (such as int). Such is not the case for Python, therefore you only have to describe in English what the type of the parameters and the result is, but you should not include types as part of the solution code.

Test cases

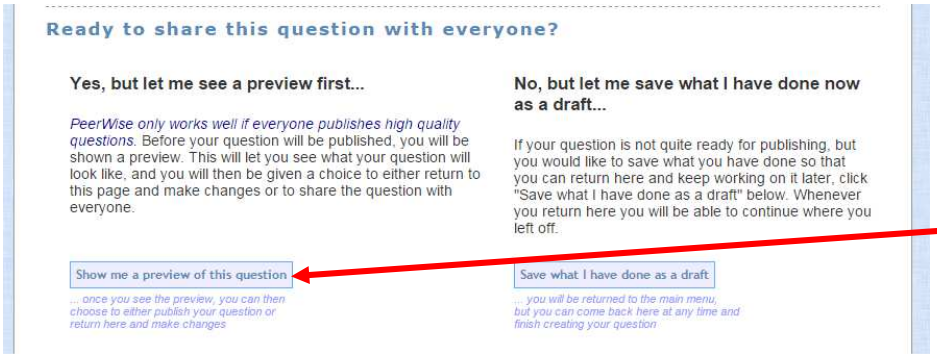
- Define several test cases (you need to at least provide 5). The first test case will be provided as example to the reader, a specific last case will be the hidden case.

- Have the function **return** the resulting value. Have the test case do the print.
- The expected output should be exactly what will be printed (from the calling of the test case)
- Make sure that you include tests cases which cover **general situations and also limit cases**.
- Limit cases are to be expected as input to the function, but may include extreme or unusual values, such as the value 0 or 1 or the null string or a string with one character or a negative number, etc. **All the cases need to be reasonable, and agree with the description** of your function. For example, if you indicate in the description that the function assumes that the input values are always positive you should not create a test case with a negative input value.

Topics associated to your function

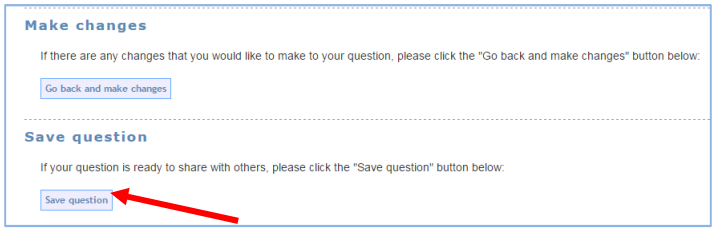
- Make sure to include the topic PEER.
- Example of topics: String manipulations, numeric calculations, repetition
- Consider defining problems needing for loops ,while loops, if statements, a combination of repetition and if statements, accumulation, flags, etc.
- Do not yet include problems requiring nested loops.

Submitting your question



IMPORTANT: Continue here to preview what you created so far and to have your solution be checked

If all the test cases pass you will be able to save the question or go back to do changes. Once saved you are done creating the question.



Debugging the code and test cases

After you "preview" your exercise details and provide a solution, you should "save " your question. Only when all the test cases pass the system will allow you to save a question.

NOTE: If your code does not work for all test cases it may be because you need to revise the solution code and/ or the test cases. Polish your exercise until you can correctly save the question.

Further things to keep in mind when creating/authoring an exercise:

- **Please put effort into clearly defining your exercises**, and defining a good set of test cases. The only feedback other students will get about whether their code is correct is whether or not it passes all of your defined test cases.
- Do not write exercises that require material that has not been covered in class. It can be frustrating for others if your exercises are too difficult to solve – instead try to write exercises that will be helpful practice for others.
- Describe clearly in words what the inputs to your function are, and what the outputs should be. The challenge for another student should be in trying to write the code to solve your exercise – not in trying to work out what the problem is asking. The BEST problems are ones with fairly simple problem statements, but which require effort when coding.
- Avoid spelling and grammar mistakes.
- When needing quotes (to surround strings) in your code solution , do not copy paste from a Word document. The curved quotes ' " ' " from Word documents are not recognized by Python and hence they are not recognized by CodeWrite either (nor in IDLE, for that matter).
- Don't rush creating your exercise - writing a good exercise takes time!

CAREFUL

- **DO NOT USE** the character '&' nor the character '#' in your test cases. Codewrite produces an error in these cases.
- **DO NOT** OPEN MORE THAN ONE CODEWRITE COURSE AT A TIME IN DIFFERENT TABS OR WINDOWS. One user can only work with Codewrite one course at a time.

End of assignment #6 description.