Assignment Guidelines

- This assignment covers material in Module 01.
- Submission details:
  - Solutions to these questions must be placed in files a01q1.py, a01q2.py, a01q3.py, and a01q4.py.
  - You must be using Python 3 or higher. Do NOT use Python 2.
  - Download the interface file from the course Web page to ensure that all function names are spelled correctly and each function has the correct number and order of parameters.
  - All solutions must be submitted to MarkUs. No solutions will be accepted through email, even if you are having issues with MarkUs.
  - Verify using MarkUs and your basic test results that your files were properly submitted and are readable on MarkUs.
  - For full style marks, your program must follow the Python section of the CS116 Style Guide.
  - Be sure to review the Academic Integrity policy on the Assignments page
- Download the testing module from the course web page. Include import check in each solution file.
  - When a function produces a floating point value, you must use check.within for your testing. Unless told otherwise, you may use a tolerance of 0.00001 in your tests.
- Restrictions:
  - Do not import any modules other than math and check.
  - Do not use Python constructs from later modules (e.g. loops and lists). Do not use any other Python functions not discussed in class or explicitly allowed elsewhere. See the allowable functions post on Piazza. You are always allowed to define your own helper functions, as long as they meet the assignment restrictions.
  - While you may use global constants in your solutions, do not use global variables for anything other than testing.
  - Read each question carefully for additional restrictions.

The solutions you submit must be entirely your own work. Do not look up either full or partial solutions on the Internet or in printed sources.
How many cell phones, placed end-to-end, does it take to go to the moon? To travel around the equator? To go from Waterloo to Toronto? Well, this is something we can figure out...

1. Write a Python function \texttt{how\_many\_phones} that consumes two positive floating point numbers: phone\_length (measured in cm) and distance (measured in km), and returns a natural number corresponding to the minimum number of cell phones of length phone\_length cm that must be needed to be placed end-to-end to cover distance km. For example, \texttt{how\_many\_phones(14.7, 1.6)} => 10885, so 10885 Blackberry Priv phones, which are 14.7 cm long, must be placed end-to-end down University Avenue to cover the 1.6 km distance from the University of Waterloo to Wilfrid Laurier University.

   You will find the function \texttt{math.ceil} useful.

\textit{From Statistics to Python} ...

2. In probability theory, the \textbf{normal distribution} is an incredibly important function, which is commonly referred to as the bell curve. The equation of this function is given as:

   \[
   f(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi} \sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}
   \]

   Where \(\sigma\) is the standard deviation, \(\mu\) is the mean, and \(\pi\) and \(e\) are the well-known constants. Write a Python function called \texttt{normal\_distribution} which consumes 3 positive, floating point numbers, namely \(x\), \texttt{mean} and \texttt{std\_dev}, and produces the corresponding value associated with the normal distribution. Do NOT round your answer! You must use \texttt{check\_within} with a tolerance value of 0.00001. You may find the \texttt{math} module very helpful for this question. For example:

   \texttt{normal\_distribution(3, 5, 2)} => 0.12098536225957168

   Note: You do not have to get 0.12098536225957168 exactly as in the example due to different machine precisions.

The following function will help you calculate your final grade in the course, assuming you pass the weighted average of the midterm and final ...

3. Write a Python function \texttt{basic\_grade}, that consumes five parameters: the first four parameters are floating point values between 0 and 100 (inclusive), corresponding to (in order) your grades on the assignments, the midterm exam, the final exam, and the clicker questions asked in class. The final parameter is a natural number between 0 and 12 (inclusive) corresponding to the number of weeks you attended a tutorial.

   The function returns the \textbf{basic grade} (rounded to an integer using the built-in Python function \texttt{round}), calculated using the grading scheme on slide 4 of Module 01:
• 20% Assignments
• 30% Midterm exam
• 45% Final exam
• 5% Participation

For this question, we are ignoring the requirement on the weighted exam average.

The participation grade is determined by converting the clicker grade to a mark out of 5, and then adding 0.1 mark for each tutorial attended, to a maximum of 5 marks total.

For example, \( \text{basic\_grade}(60.0, 75.8, 90.0, 55.5, 9) \Rightarrow 79 \)

From floor tiling to Python ...

4. For this question, you will be required to determine the minimum number of identical tiles all with the same orientation (see examples below) that are required to cover the floor of a rectangular room. Any excess from a tile exceeding the floor area is discarded and cannot be reused. Write a Python function called \( \text{min\_tiles} \) which consumes 4 positive integers, \( \text{room\_width, room\_length, tile\_width, tile\_length} \), and produces the minimum number of tiles required to completely cover the floor of the room. See examples below for clarification.

Examples
• If the room is 4 x 4, and the tiles are 2 x 2, then you need exactly 4 tiles to cover the floor. Hence, \( \text{min\_tiles}(4, 4, 2, 2) \Rightarrow 4 \)

• If the room is 4 x 4 and the tiles are 3 x 3, then you still need 4 tiles to cover the floor (but we would simply discard the excess pieces which are cut off): \( \text{min\_tiles}(4, 4, 3, 3) \Rightarrow 4 \)

The yellow represents the room, and the white squares are the tiles. As you can see, it takes 4 tiles to cover the room, but there is excess tile left over.

• Note: rectangular (non-square) tiles can either be oriented vertically or horizontally, but never both in the same room. So, if the floor is 3 x 4, and the tiles are 1 x 3, then in one direction it would take 6 tiles to cover the floor (discarding excess pieces), but in the other
direction it would only take 4 tiles to cover the floor. You should produce the minimum in this case, which would be 4. Hence \( \text{min\_tiles}(3, 4, 1, 3) \Rightarrow 4 \)

The yellow represents the room, and the white squares are the tiles. As you can see, it takes 6 tiles to cover the room in the first diagram, but it only takes 4 tiles to cover the room in the second diagram.