Numpy warmup ¶

- For this exercise, for 2d arrays, we will refer to the collection of numbers having the same index on the first axis as **row**, and having the same index on the second axis as **column**
- Submit one file called `numpy_warmup.py` .

1. Generate an array of random integers of shape (6,6) called A with the integers between `[10,20]` .
   A. Create a list of all the (2,2) sub arrays, going from left to right then top to bottom. (For example : (0,0), (0,1),(1,0),(1,1) defines the first subarray). There should 36/4 = 9 subarrays.
   B. Concatenate the list of subarrays along a new axis to get a (9,2,2) array.
   C. Generate another array of random integers of shape (6,6) called B with the integers between `[15,20]` .
   D. Create an array C which is the elementwise maximum of A & B.
   E. Create an array D such that for an element `D[i,j]= A[i,j] if A[i,j] - B[i,j] is divisible by 2, otherwise D[i,j]=B[i,j]` .
   F. Count how many elements in D are equal to A.

> Wrap the previous question in a function warmup_1(). Use comments so that each subpart is easy to check.

2. Create the array `[[1,2,3],[4,5,6],[7,8,9]]` without hardcoding it.
   - Generalize it to a function `consecutive_2d_grid(num)` that returns an array of shape (num,num) where the numbers from 1...num^2 are filled in row-first order (like the example).
   - Add another parameter to `consecutive_2d_grid` called `along` with values "row" or "column", depending on which the numbers are arranged along the rows or the columns. Do not use the ".T" attribute.

> Submit only the final function consecutive_2d_grid

3. Generate an array of shape (5,7) filled with fractional numbers between [-5,5] picked at random.
   A. Find the average of the maximum of each row. (i.e. the average row maximum)
   B. Find the minimum of the maximum of each column (i.e. the minimum column maximum)
   C. Divide each entry of the array by the maximum of its column. That, is the entries at x[3,4] and x[5,4] should be divided by the maximum of the 5th column.
   D. Divide each entry of the array by the sum of its row.
   E. Find the elements which are the maximum in both their row and their column. (Hint : Use argmax - with a bit of thinking you can construct a list of coordinates (as tuples?) for row maximums, and another for column maximums. What's the next step?)

> Wrap the previous question in a function warmup_3(). Use comments so that each subpart is easy to check.

4. Generate a random array between [-5,5] of shape (9,9). Let's call this A.
   A. Create a 1-d random array of 9 numbers between [10,20]. Let's call this B.
   B. Add B to the rows of A - (i.e. if two entries are on the same column, they should add to the same element of B).
   C. Add B to the columns of A. (i.e. same row => same element of B)

D. Create a new array of zeros of shape (3,3) called block_sum. Every element of block_sum should correspond to the sum of the corresponding (3,3) subarray in A. For example, a sudoku puzzle is a 9x9 array is divided into 9 blocks of 3x3 subarrays. (Hint: Slices and loops are enough - no need for anything fancy!)

> Wrap the previous question in a function warmup_4(). Use comments so that each subpart is easy to check.

5. Generate a 1d array of 10 evenly spaced numbers between [-1,1]. Let's call this A. Calculate a (10,10) array where the (i,j)th entry is the product of the ith and jth number in A.

> Wrap the previous question in a function warmup_5().

Tip : These questions might seem confusing at first - they are supposed to be! Take a deep breath, and break the question down into parts!