CS2302 Data Structures

Spring 2023

Arrays – Exercise 1

Write the following functions. See the commented out part of the program for expected results. Also, make sure your functions don't modify the array (if any received as parameter.

1. Write the function *is_square(A)* that receives an array *A* and determines if it is square. An array is square if it has two dimensions and they are the same size (that is, it has the same number of rows and columns).
2. The numpy function *np.max(A)* returns the maximum value in array *A*. Write the function *max_loop(A)* that uses a loop to find the maximum value in 1D array *A*.
3. The numpy function *np.argmax(A)* returns the index where the maximum value in array *A* is stored. Write the function *argmax_loop(A)* that uses a loop to find the argmax in 1D array *A*.
4. The numpy function *np.sum(A)* returns the sum of the elements in array *A*. Write the function *sum_loop(A)* that uses a loop to compute the sum of the elements in 1D array *A*.
5. Write the function *sum_row(A,n)* that receives a 2D array of integers *A* and a positive integer *n* and returns the sum of row *n* in *A*. You may use built-in numpy functions.
6. Write the function *sum_column(A,n)* that receives a 2D array of integers *A* and a positive integer *n*, and returns the sum of column *n* in *A*. You may use built-in numpy functions.
7. Write the function *first_n_rows(A,n)* that receives a 2D array *A* and an integer *n*, where *n>0*, and returns a 2D array containing the first *n* rows of *A* (or all of *A* if *A* has less than *n* rows).
8. Write the function *first_n_colums(A,n)* that receives a 2D array *A* and an integer *n*, where *n>0*, and returns a 2D array containing the first *n* columns of *A* (or all of *A* if *A* has less than *n* columns).
9. Write the function *last_n_rows(A,n)* that receives a 2D array *A* and an integer *n*, where *n>0*, and returns a 2D array containing the last *n* rows of *A* (or all of *A* if *A* has less than *n* rows).
10. Write the function *last_n_colums(A,n)* that receives a 2D array *A* and an integer *n*, where *n>0*, and returns a 2D array containing the last *n* columns of *A* (or all of *A* if *A* has less than *n* columns).
11. Write the function *all_n(n,r,c)* that receives integers *n*, *r*, and *c* and, where *r>0* and *c>0*, and returns a 2D array with *r* rows and *c* columns where every element is equal to *n*.
12. Write the function *diagonal(A)* that receives a square array A and returns a 1D array containing the elements in the diagonal of *A* (that is *[A[0,0], A[1,1],* ...).
13. Write the function *count_digits(A)* that counts the number of times each of the numbers 0,1,…,9 appears in an array. The function should receive an array *A* of any dimensionality and return a 1D array of length 10, where the first element in the array is the number of times 0 appears in *A*, the second element is the number of times 1 appears in *A*, and so on. Do this by having an array of counters *count,*then iterate over all elements of *A,* and for every element, check if it is a single digit, that is, it is greater or equal to 0 and less or equal to 9, if the element is a digit, add one to the corresponding counter, otherwise do nothing.