Due Wednesday 17 January 2018

If you want to get assistance setting up your system, please complete the “set-up assistance survey” available via the link that is now included in the related announcement.

1. Read Ch. 1 and “play along” with the introduction to Python and numpy. Note that you may need to add parentheses around things in print statements (which is a major change between Python 2.7 and 3.6. Also beware of integer division…)
2. Compute and store an array of n values equally spaced over the interval [0,2*Pi]:
    1. Write a function using a for() loop to compute the values and store them in a Numpy array.
    2. Use Numpy’s linspace() function to perform the same task.
    3. Plot the values as a function of index number for n=11 using matplotlib.
3. Write functions to perform the following computations with 1D arrays of values. Create a Numpy array to store your output (i.e., do not worry about doing “in-place” operations where you write the result into one of the input arrays), and be sure to include comments to describe the inputs and outputs of your functions:
    1. Scalar multiplication
    2. Component-wise addition
    3. Evaluate the linear function y = c*x + d (which combines parts 1a and 1b)
    4. Component-wise multiplication
    5. Inner product
    6. Euclidean norm
4. Using Numba, create parallel versions of your functions from problem 2. Identify parts that cannot be immediately parallelized and write serial implementations as needed.
5. Write and execute a program to test your functions as follows:
    5. Create input arrays u and v of length n.
       Start with n=5.
        1. Set each entry in v equal to 1.
        2. Set each entry in u to 1/(n-1), then reset the first entry to 1 (remember that means u[0]=1).

- Compute z=-u and the norm of u+z. Inspect and verify your results.

1. Compute the dot product (inner product) of u and v. Inspect and verify your results.
2. Create a “reversed dot” product in which you sum the contributions in reverse order. Inspect and verify your results when computing the dot product of u and v using this new function.

1. Repeat part (b.iv and b.v) with longer arrays, e.g. n=10,000,000. At what length does the result produce something unexpected? Does the order of summation in the dot product affect the results? Why or why not?

Submit your source file via Canvas. **Write your main() function to include print() statements, so that executing main() writes your responses to the terminal.**