ASSESSMENT 1 PART A PRACTICAL 4

Weighting: 6% (this is the fourth of five practical assessments totalling 30%).

Due Date: Sunday 11:59 pm, end of Week 4.

Introduction: Exploring vectorisation

Purpose

The purpose of this assessment is for you to demonstrate your ability to apply what you have learnt each week in the creation of a program to solve a problem.

Your tasks

Q1. This question requires you to demonstrate your ability to create arrays and functions. Don't forget to include markdown cells describing your work in your Jupyter Notebook.

Write two functions to calculate the cube:

1) A function that takes a NumPy array of integer numbers and returns a NumPy array of the cube of each the array elements. Your function must use the NumPy universal function power(). Note you must NOT use for loops to access NumPy array elements.
2) A function that takes a Python sequence (range) of integer numbers and returns a python list of the cube of each of the range elements using the Python pow() function.

Create two variables:

1) narray a NumPy array of 100,000 integer numbers
2) nrange a Python range of 100,000 integer numbers.

Call each of your functions showing the result (check they work as expected.)

Q2. Jupyter Notebook, through Python, can give you information about the amount of computer processor time your code takes to run: using the magic command %timeit to determine how much time a cell takes to run. The format is:

%timeit functionName(parameters)

For example, if you wanted to find out how long the NumPy mean() function took to work out the mean value of array A, you could run:

%timeit np.mean(A)

Use %timeit to determine how much faster your function using NumPy arrays is versus your function that uses a sequence. Include your conclusions in your Jupyter Notebook in a markdown cell with an explanation of why Q1 did not allow loops to be used in the function.

Q3. This question requires you to manipulate arrays using slicing, splitting and/or array functions (e.g. append, row_stack, column_stack):

a) Generate a 1D array of the values from 1 to 20. Use a single line of code to convert the array to a 2D array of 4 rows by 5 columns.
b) Look up the NumPy array functions and find the function that will revert your array back to a 1D array. Apply this to your array.
c) Create the following two NumPy arrays: A and B

Array: A

1	2
3	4

Array: B

5	6	7
8	9	10

Create a third NumPy array where the values are the multiplication of the values of array A and B for each row, column position. In positions where one of the arrays does not have a value to use, copy the single value from the other array.

For example, your result array for the example above would be:

Array: result

5	12	7
24	36	10

As you are working with NumPy arrays, your solution should not use for loops

Directions

Submit your assessment via online submission, as a Jupyter Notebook, providing your responses to ques-tions in this Assessment brief. This assessment will be written in Python.

Save your notebook and upload it through the assignment link. Your tutor will assess your work based on the rubric. Make sure you have a look at the rubric so you understand what your tutor will be looking for.

Requirements

You must submit your Jupyter notebook file (containing your Python code) through the Canvas submission link.

Rubric

Assessment Criteria	Level of performance					
Q1	**High Distinction (5/5)** Both functions match specification. No loop for NumPy array. Correct result.	**Distinction (4.2/5)** Both functions match specification. Correct results. Does not meet HD due to use of loop for NumPy array.	**Credit (3.7/5)** Both functions meet specification, but some other aspects of question are missing or incorrect (functions are not called correctly).	**Pass (3.2/5)** At least one function meets specification and returns correct result.	**Fail 0/5)** No functions meet specification.	**/5**
Q2				**Pass (2/2)** Correct timing and explanation provided for both NumPy and sequence.	**Fail 0/2)** Timings missing or incorrect.	**/2**

Q3	**High Distinction (5/5)** Approach works for any dimension arrays, without using loops.	**Distinction (4.2/5)** Approach works for the given example without using loops.	**Credit (3.7/5)** Approach works any example, but uses loops.	**Pass (3.2/5)** Partial solution that demonstrates correct use of splitting and joining arrays.	**Fail 0/5)** Approach does not work.	**/5**
Notebook	**High Distinction (5/5)** Meets Distinction criteria plus provides correct explanation of results (time difference reason and explanation of why loops were restricted).	**Distinction (4.2/5)** All problems include a header related to what data is being generated and how.	**Credit (3.7/5)** All problems include a header related to what data is being generated and a minimal description of how (for example, a list instead of text describing the process).	**Pass (3.2/5)** All problems include a header and a description of the data but process for generating is unclear.	**Fail 0/5)** No headers or description of the process.	**/5**
TOTAL						**/17**