PHY1112 Lab 5

Data and Recursion – Down the Rabbit Hole

February 6th, 2024

|  |  |  |  |
| --- | --- | --- | --- |
| Part | 1 | 2 | Total |
| Points | 15 | 15 | 30 |
| Score |  |  |  |

Objectives

1. Create a recursive function to calculate factorials.
2. Unit test corner cases and compare with math.factorial().
3. Read in and use a large data set using NumPy.

Part 1: Recursion and testing – Going down the rabbit hole and figuring out how to get back up

1. (5 points) Create a new file named “lab5.py” and create a recursive function named recursive\_factorial() that calculates the factorial

for some integer a. Note that this can be written in a closed form like above as

.

1. (5 points) Run your recursive\_factorial() function for a suitable set of inputs, including edge cases, and work out the math by hand below to show that your function evaluates to the correct answer.
2. (5 points) Write a unit test for your recursive\_factorial() function in a similar format as the example below for the function recursive\_multiplication(). Make sure to follow the guidelines laid out in lecture 8.

Here is an example recursive function that does multiplication for an integer b (a bit different than what was done in lecture:

def recursive\_multiplication(a, b):

if b == 1:

return 0

return a + recursive\_multiplication(a, b - 1)

# the following unit test would suffice to test recursive\_multiplication()

inputs = [(0,0), (1,0), (0,1), (0,10), (3,5), (4,6)]

outputs = [0, 0, 0, 0, 15, 24]

for i in range(len(inputs)):

check\_string = “(“ + str(inputs[i][0]) + “\*” + str(inputs[i][1]) + “)”

check\_value = recursive\_multiplication(inputs[i][0],inputs[i][1])

check\_result = (check\_value == outputs[i])

print(check\_string, “ == ”, str(outputs[i]), “? ”, check\_result, sep=’’)

Here is what would be printed to the screen. We want all the tests to be “True”.

(0\*0) == 0? True

(1\*0) == 0? True

(0\*1) == 0? True

(0\*10) == 0? True

(3\*5) == 15? True

(4\*6) == 24? True

Part 2: Handling large data – NumPy to the rescue!

1. (5 points) Read in the daily high temperature and daily low temperature data in Ottawa during the year 2022 from the file “weather\_data\_lab5.csv” (found on Brightspace) using np.genfromtext(). There should be one year’s worth of weather data in that file, however the only columns of interest are the “Max Temp (°C)” and “Min Temp (°C)” columns.
   * The usecols keyword argument will be helpful here for only grabbing the columns of interest.
2. (10 points) Using vectorization, determine the lowest and highest daily temperatures in Ottawa in 2022, as well as on which day they occurred (between day 0 and 364). Further, determine the largest difference between the daily high and low in any one day, and also on which day that occurred.
   * Note that there are some skipped entries in the data. These skipped entries will have values of NaN, and as such we would want to skip them in our statistics. To skip them, use the np.nanmax(), np.nanmin(), np.nanargmax(), and np.nanargmin() functions.

**Remember to submit this document filled out, along with your python file on Brightspace!**