`Assignment 5 --- QMB 3311 --- Spring 2023 --- Due:  March 26, 2023`

Instructions:

Complete this assignment within the space on your *private* GitHub repo in a folder called `Assignment_05`. In this folder, save your answers to a file called `my_CES_A5.py` and `my_logit_A5.py` in the course repository. When you are finished, submit your files to your repository and upload the link to Webcourses.

For all of the exercises, use your examples to test the functions you defined. Since the examples are all contained within the docstrings of your functions, you can use the `doctest.testmod()` function within the doctest module to test your functions automatically.

Don't worry about false alarms: if there are some "failures" that are only different in the smaller decimal places, then your function is good enough. It is much more important that your function runs without throwing an error.

1. Create the following functions from your previous assignments into a two different modules:
   - For Module `my_CES_A5.py`, copy these previous functions:
     - `CES_utility()` from Assignment 2.
     - `CES_utility_valid()` from Assignment 3.
     - `CES_utility_in_budget()` from Assignment 3.
   - For Module `my_logit_A5.py`, copy these previous functions:
     - `logit()` from Assignment 3.
     - `logit_like()` from Assignment 3.
     - `logit_like_sum()` from Assignment 4.

2. Follow the function design recipe to define functions for all of the following parts. For each function, create three examples to test your functions.
   (a) Write a *helper* function `logit_d_i(x_i,k)` that helps you calculate the term d_i in the gradient vector. The formula for `logit_di()` is

   $$d_i = \begin{cases} 1, & \text{if } k = 0, \\ x_i, & \text{if } k = 1, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

   where x_i is a number. It will be a single observation of a list of covariates `x` when `logit_d_i()` is used in another function. Add this function your module `my_logit_A5.py`.

   (b) Write another helper function `logit_dLi_dbk(y_i,x_i, beta_0, beta_1)` that helps you calculate an individual term in the sum of the gradient vector. The formula for `logit_dLi_dbk()` is is

   $$\frac{\partial}{\partial \beta_k} L_i(y_i, x_i; \beta_0, \beta_1) = \begin{cases} d_i(1 - \ell(x_i; \beta_0, \beta_1)), & \text{if } y_i = 1, \\ d_i(-\ell(x_i; \beta_0, \beta_1)), & \text{if } y_i = 0, \\ \text{undefined} & \text{otherwise} \end{cases}$$

where y_i and x_i is a single pair of observations from a list of observations in y and x that can be used in another function. Add this function your module my_logit_A5.py.

For a two-by-two matrix A, the inverse can be obtained with the expression

(c) Write a function CESdemand_calc(r, p_x, p_x, w) that returns a list of two values [x_star,y_star] that achieve the maximum value of CES_utility(), subject to the budget constraint that the consumer's basket of goods should cost no more than their wealth w: $p_x x + p_y y \leq w$. That is, given p_x and p_y, these values maximize the function CESutility_in_budget, without returning a value of None. A senior analyst used calculus to find the optimal values of x^* and y^*:

$$x^* = \frac{p_x^{\frac{1}{r-1}}}{p_x^{\frac{r}{r-1}} + p_y^{\frac{r}{r-1}}} \times w \text{ and } y^* = \frac{p_y^{\frac{1}{r-1}}}{p_x^{\frac{r}{r-1}} + p_y^{\frac{r}{r-1}}} \times w$$

Add this function to your module my_CES_A5.py.

(d) Now write a function that finds values of x and y that maximize CESutility_in_budget(x, y, r, p_x, p_y, w) for given r, p_x, p_y, and w. Write a function definition max_CES_xy(x_min, x_max, y_min, y_max, step, r, p_x, p_y, w) as follows:

1. Find the values of x and y by evaluating CESutility_in_budget(x, y, r, p_x, p_y, w) over every combination of of (x, y) in two lists.
2. Create lists x_list and y_list from ranges $x = x^{min}, \ldots, x^{max}$ and $y = y^{min}, \ldots, y^{max}$, where the neighboring values of x or y are separated by distance step. The np.arange() function is useful for this.
3. Initialize the maximized value with max_CES = float(''-inf'').
4. Loop over the index numbers i and j corresponding to lists x_list and y_list.
   (a) For each pair of i and j, extract the value x_list[i] and y_list[j]
   (b) For each pair of i and j, evaluate CESutility_in_budget(x, y, r, p_x,p_y, w)
   (c) If the value is higher than max_CES, record the new i_max = i and j_max = j and update the newest value of the max_CES. If CESutility_in_budget returns None, make no changes and move on to the next values.
5. After the loops, return [x_list[i_max], y_list[j_max]].
6. Verify that the result matches the values from the previous question (up to accuracy step). You can use the same test cases as you did in the previous question. Add this function to your module my_CES_A5.py