ECP 3004: Python for Business Analytics
Department of Economics
College of Business
University of Central Florida
Spring 2021

Midterm Examination

Due Monday, March 8, 2021 at 5:59 PM
in your GitHub repository

Instructions:

Complete this examination within the space on your *private* GitHub repo (not a fork of the course repo ECP3004S21!) in a folder called `midterm_exam`. In this folder, save your answers to Questions 1 and 2 in a file called `my_midterm_module.py`, following the sample script in the folder `midterm_exam` in the course repository. When you are finished, submit it by uploading your files to your GitHub repo using any one of the approaches outlined in Question 3. As this is an examination, you are NOT free to discuss your approach to each question with your classmates and you must upload your own work.

Question 1:

Follow the function design recipe to define functions for all of the following Exercises. For each function, create three examples to test your functions. Record the definitions in the sample script `my_midterm_module.py`. Together, the function definitions will form a module called `my_midterm_module` that you can read in and test using another script.

Under time constraints, the emphasis is on creating function definitions and testing with examples. You are not responsible for error handling and can assume that the user will follow the preconditions and type contract. Refer to the partial docstrings that are already included in `my_midterm_module.py`.

Exercise 1 In economic problems, consumers must make a purchase decision within a budget constraint. Suppose a consumer buys some quantities of two goods x and y, with prices p_x and p_y, and has wealth w to spend. The consumer's total expenditure is $xp_x + yp_y$. Write a function `in_budget(x, y, p_x, p_y, w)` that returns a boolean indicator of whether the consumer's expenditure is less than or equal to w. No error messages are necessary.

Exercise 2 Now write a function that calculates the consumer's optimal bundle of goods, that is, the amounts x^* and y^* that are within budget and maximize utility $u(x, y) = x^\alpha y^{1-\alpha}$. Using calculus, you can show that the following choices maximize the consumer's utility:

$$x^* = \frac{\alpha}{p_x} w \quad \text{and} \quad y^* = \frac{1-\alpha}{p_y} w.$$

Write a function that returns a list with x^* and y^* called `calc_bundle(p_x, p_y, w, alpha)`.

Exercise 3 Now let's verify the derivation of the above solution. Start by writing a function for the expenditure on y, given a chosen amount x^*. The consumer will want to spend the entire budget, so that $x^*p_x + y^*p_y = w$ or

$$y^* = \frac{w - x^*p_x}{p_y}.$$

Call this function `y_solve(x_star, p_x, p_y, w)`.

Exercise 4 Now write a function that calculates the consumer's optimal bundle of goods that maximizes utility $u(x, y) = x^\alpha y^{1-\alpha}$ and is within budget. Do this by looping over candidate values of x^*. Set y^* within budget using the function `y_solve(x_star, p_x, p_y, w)` within the loop on x, for x from zero to w/p_x in increments of size `step`. Evaluate $u(x, y)$ for each candidate value of x^* and find the highest value `max_util`. Write a function that returns a list with x^* and y^* called `one_loop_bundle(p_x, p_y, w, alpha, step)`.

Exercise 5 Write a function `util_in_budget(x, y, p_x, p_y, w, alpha)` that calculates the consumer's utility:

$$u(x, y) = x^\alpha y^{1-\alpha}$$

when x^* and y^* are within budget and $x^* \geq 0$ and $y^* \geq 0$ and $0 \leq \alpha \leq 1$ and returns zero otherwise. No error messages are necessary. Note that you can use any function from previous Assignments as a starting point.

Exercise 6 Now find values of `x_star` and `y_star` that maximize `util_in_budget(x, y, p_x, p_y, w, alpha)` by searching over both `x_star` and `y_star` independently. Write a function `two_loop_bundle(p_x, p_y, w, alpha, step)` as follows:

a) Find these values by evaluating `util_in_budget(x, y, p_x, p_y, w, alpha)` over every combination of of (x^*, y^*) in two lists.

b) Create lists `x_star_list` and `y_star_list` from ranges $x^* = 0, \ldots, w/p_x$ and $y^* = 0, \ldots, w/p_y$, where the neighboring values of x^* or y^* are separated by distance `step`.

c) Start with `max_util = 0`. Loop over the index numbers `i` and `j`, corresponding to lists `x_star_list` and `y_star_list`.

d) For each pair of `i` and `j`, extract the value `x_star_list[i]` and `y_star_list[j]`.

e) For each pair of `i` and `j`, evaluate `util_in_budget(x, y, p_x, p_y, w, alpha)`. If it is higher than `max_util`, record the new `i_max = i` and `j_max = j` and update the newest value of `max_util`.

f) After the loops, return `[ x_star_list[i_max], y_star_list[j_max] ]`

Question 2:

For all of the Exercises in Question 1, use your examples to test the functions you defined. Since the examples are all contained within the docstrings of your functions, use the `doctest.testmod()` function within the `doctest` module to test your functions automatically at the bottom of your module.

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.

Question 3:

Push your completed files to your GitHub repository following one of these three methods.

Method 1: In a Browser

Upload your code to your GitHub repo using the interface in a browser.

1. Browse to your `assignment_0X` folder in your repository ("X" corresponds to Assignment X.).
2. Click on the "Add file" button and select "Upload files" from the drop-down menu.
3. Revise the generic message "Added files via upload" to leave a more specific message. You can also add a description of what you are uploading in the field marked "Add an optional extended description..."
4. Press "'Commit changes," leaving the button set to "Commit directly to the `main` branch."

Method 2: With GitHub Desktop

Upload your code to your GitHub repo using the interface in GitHub Desktop.

1. Save your file within the folder in your repository in GitHub Desktop.
2. When you see the changes in GitHub Desktop, add a description of the changes you are making in the bottom left panel.
3. Press the button "Commit to main" to commit those changes.
4. Press the button "Push origin" to push the changes to the online repository. After this step, the changes should be visible on a browser, after refreshing the page.

Method 3: At the Command Line

Push your code directly to the repository from the command line in a terminal window, such as GitBash on a Windows machine or Terminal on a Mac.

1. Open GitBash or Terminal and navigate to the folder inside your local copy of your git repo containing your assignments. Any easy way to do this is to right-click and open GitBash within the folder in Explorer. A better way is to navigate with UNIX commands, such as `cd`.
2. Enter `git add .` to stage all of your files to commit to your repo. You can enter `git add my_filename.ext` to add files one at a time, such as `my_functions.py` in this Assignment.
3. Enter `git commit -m "Describe your changes here"`, with an appropriate description, to commit the changes. This packages all the `add`ed changes into a single unit and stages them to `push` to your online repo.
4. Enter `git push origin main` to push the changes to the online repository. After this step, the changes should be visible on a browser, after refreshing the page.