QMB 3311: Python for Business Analytics
Department of Economics
College of Business
University of Central Florida
Spring 2022

Assignment 6

Due Sunday, April 3, 2021 at 11:59 PM
in your GitHub repository

Instructions:

Complete this assignment within the space of your private GitHub repo in a folder called `assignment_06`. In this folder, save your answers to Questions 1 and 2 in a file called `my_A6_module.py`, following the sample script in the folder `assignment_06` 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. You are free to discuss your approach to each question with your classmates but 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_A6_module.py`

Exercise 1 A *Taylor series* is the sum of a sequence of terms that approximates the value of a function in the neighborhood of a particular point. For the natural logarithm function, $\ln(z)$, the Taylor series expansion around the point $z_0 = 1$ is

$$\ln(z) = \ln(1) + \sum_{k=1}^{\infty} (-1)^{k-1} \frac{1}{k} (z-1)^k,$$

and, since $\ln(1) = 0$, the approximation can be calculated without use of the `math.log` function. Write a function `ln_taylor(z, n)` that calculates the first n terms of this approximation. In crafting your examples, you will find that this approximation of the `math.log` function is more accurate for values of z close to 1 or when a large number of terms n is used.

Exercise 2 Another way to solve for the value of the $\ln(z)$ function is to transform it into a root-finding problem: solve for the value of x such that

$$e^x = z \text{ or } e^x - z = 0,$$

which is true when $x = \ln(z)$, for a given z. Write a function `exp_x_diff(x, z)` that returns the value $e^x - z$.

Exercise 3 Now solve the function $f(x) =$ `exp_x_diff(x, z) == 0` for the root x^*. Write a function `ln_z_bisect(z, a_0, b_0, num_iter)` that calculates the root using the bisection method. Essentially, this will produce an algorithm for calculating the natural logarithm of `z`. Follow the algorithm used in the lecture that recursively splits the interval in half and, in each iteration, assigns the midpoint `m_i` to the endpoint for which `exp_x_diff()` has the same sign as `exp_x_diff(m_i, z)`. It should start with the interval (a_0, b_0) and perform `num_iter` iterations. To guarantee a solution, your function should first check that the signs of $f(x)$ differ at the endpoints, i.e. $f(a_0)f(b_0) < 0$.

Exercise 4 Next, solve for the roots using Newton's method. Before doing this, you will need a function that returns the derivative. Write a function `exp_x_diff_prime(x, z)` $(= f'(x))$ that returns the derivative of `exp_x_diff(x, z)` $(= f(x))$ with respect to `x`. Note that `z` is a constant parameter in this function.

Exercise 5 Now, use Newton's method to find the natural logarithm of z. Use the recurrence relation

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

repeatedly until it reaches a value x_n such that $|f(x_n)| < \epsilon$, where ϵ is a small number represented by `tol`. Write a function `ln_z_newton(z, x0, tol, num_iter)` that solves for the natural logarithm of z using the initial value `x0`, up to level of tolerance `tol`, and a maximum number of iterations `num_iter`. It should print a warning message if it reaches the maximum number of iterations before the `exp_x_diff(x_i, z)` function value is less than `tol`.

Exercise 6 In the next exercise, you will use the fixed-point method to find a root of this function. A *fixed point* of a function $g(x)$ is a value x^* such that $g(x^*) = x^*$. Consider the function

$$g(x) = \frac{1}{2}\left(z - e^x + 2x\right).$$

Note that $x^* = \ln(z)$ is a fixed point of $g(x)$; that is, $g(\ln(z)) = \ln(z)$. Write a function `exp_x_fp_fn(x, z)` that returns the value $g(x)$, for a given value z.

Exercise 7 Finally, use the fixed-point method to find the natural logarithm of z. That is, use the recurrence relation $x_{i+1} = g(x_i)$ repeatedly until it reaches a value x_n such that $|g(x_n) - x_n| < \epsilon$, where ϵ is a small number represented by `tol`. Write this algorithm within a function `ln_z_fixed_pt(z, x0, tol, num_iter)`.

Question 2:

For all of the Exercises in Question 1, use your examples to test the functions you defined. Use the `doctest.testmod()` function within the `doctest` module to test your functions automatically. The test results should print when the script is run but not when the functions in the script are imported as a 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 (the "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 the button "'Commit changes," leaving the buton 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 within the folder referenced 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.