This assignment is due **July 26, 2024**. You may work in teams of up to four people. If you work in teams, make sure you include everyone's name on the assignment and there is only one submission per team. Please submit one html or pdf file. Show all your work. Use as few Python packages as possible to complete the coding portion of this assignment. Email your solutions to **charles.tutoring@gmail.com**.

1. The gradient descent algorithm shown below finds a local minimum of the function f given an initial guess x_0. The value h is used to find the numerical derivative approximation g. The learning rate η controls the step size taken in each iteration. A value of η too small can lead to slow convergence, while a value of η too large can cause divergence. The algorithm terminates when either the absolute difference between successive iterations is smaller than a tolerance α (indicating convergence), larger than a threshold β (indicating divergence), or when a maximum number of iterations N is reached.

Input: f, x_0, h, η, α, β, N
def *g(x)*:
  return $\frac{f(x+h)-f(x-h)}{2h}$;
for $i \leftarrow 1$ *to* N **do**
  $x \leftarrow x_0 - \eta g(x_0)$;
  if $|x - x_0| < \alpha$ **then**
    return x;
  else if $|x - x_0| > \beta$ **then**
    return NaN;
  else
    $x_0 \leftarrow x$;
  end
end
return NaN;

(a) Write a function that implements this algorithm in Python. Set respective default arguments for h, η, α, β, and N of 10^{-2}, 10^{-3}, 10^{-6}, 10^{6}, and 100000.

For parts (b), (c), and (d), use $f(x) = 3x^4 - 16x^3 + 6x^2 + 72x + 5$.

(b) Pick a few values of x_0 and use your function from (a) to find the local minima of f.

(c) Repeat (b) with $-f$ instead of f to find the local maxima of f.

(d) Verify your work by using `plot` and `scatter` in `matplotlib.pyplot` to plot f and your local extrema, respectively.

2. We want to estimate the area under the curve $y = e^{-\sqrt{x}}$ from $x = 1$ to $x = 100$.

(a) Approximate the area using a Riemann sum with

$$P = \left(1, 1 + \frac{99}{n}, 1 + \frac{99 \cdot 2}{n}, \ldots, 1 + \frac{99 \cdot (n-1)}{n}, 100\right).$$

Find the height of each rectangle using the left, middle, and right points of each subinterval determined by consecutive entries of P. Calculate the results for $n = 10,\ 25,\ 50,\ 75$, and 100 rectangles. Save your results in a `pandas` data frame.

(b) In part (a), the Riemann sum was calculated using a uniform partition, which is not the most efficient way to approximate the area. This is because $e^{-\sqrt{x}} \approx 0$ for $x \gg 0$. The approximation will be better if we sample more small values of x and fewer large values. With this reasoning in mind repeat (a) but with

$$P = \left(1, 100^{1/n}, 100^{2/n}, \ldots, 100^{(n-1)/n}, 100\right).$$

(c) Use analytic techniques to find the exact area

$$\int_1^{100} e^{-\sqrt{x}}\, dx.$$

(d) Using `matplotlib.pyplot`, graph your results from parts (a) and (b) via the function `scatter`. Use `subplots` to create three subplots, one for left endpoints, midpoints, and right endpoints. In each subplot, draw a dashed horizontal red line using `axhline` to denote your result from part (c). Make sure to add labels and legends so that it is easy to identify all of your subplots' features.

3. You have graduated from the MFE program and landed yourself a nice job pricing annuities! For each of the following perpetuities (i.e. annuities with never-ending payments) find the price, or prove that the result would not converge. Derive your solutions analytically when possible. Assume one payment is made at the end of each year and a constant annual discount rate of 7%. Let us say that it is January 1, 2024, for simplicity.

(a) Payments of \$10.

(b) Payments of the numerical value of the year, e.g. the perpetuity would pay \$2024 this year.

(c) A payment of \$1 this year, an increase of 10% per year for the next four years, and an increase of 5% per year after that. That is, a payment sequence of

$$\$1, \$1.10, \$1.10^2, \$1.10^3, \$1.10^4, \$1.10^4 \times 1.05, \$1.10^4 \times 1.05^2, \ldots.$$

(d) Payments of $\$k^k$ in the k-th year, where 2024 corresponds to $k = 1$.