

Attention:

1. Please submit your homework electronically through the Blackboard. E-mail submissions will not be accepted.
2. Please submit only one zip file including only Python codes with .py extension. There will be a separate .py file for each question in the zip file. Submissions in other formats will not be accepted. Name your Python codes as *lastname1.py*, *lastname2.py*, and so on.
3. Codes are expected to be clear, legible and compact with minimum number of lines and variables, and proper indentation. Variable names are expected to be appropriate, meaningful and self-explanatory. Codes will be graded based on these criteria. A correctly working code does not guarantee full credit.

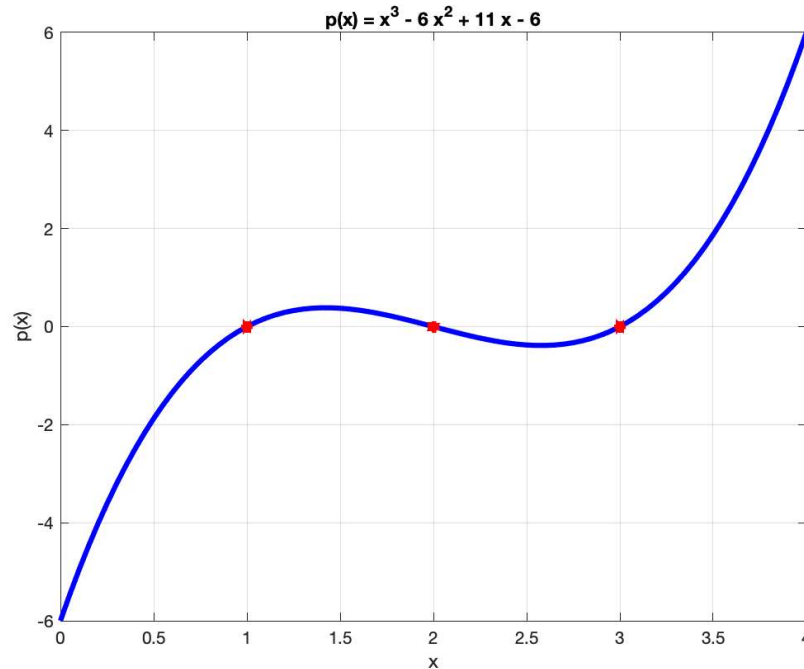
1. (30 points) Consider the following polynomial with a real variable, x ;

$$p(x) = x^3 - 6x^2 + 11x - 6 = (x - 1)(x - 2)(x - 3)$$

which has roots at $x=1$, $x=2$ and $x=3$. The polynomial and its roots are shown in the figure below.

- a. (5 points) Write a Python function for $p(x)$ that takes x as a formal parameter and returns $p(x)$.
- b. (15 points) Write a Python function for root finding using the Bisection search that takes a lower bound, x_l , and an upper bound, x_u , of a search interval as formal parameters, and returns a root, x_1 , of $p(x)$ such that $p(x_1) \sim 0$. x_1 can be any root of $p(x)$. At each iteration, the function is expected to print the **iteration number**, the **new interval midpoint** and the **absolute value of $p(x)$ at the midpoint**. Iterations should stop when the absolute value of $p(x)$ at a new midpoint is less than 10^{-6} . The function will return that midpoint as x_1 .
- c. (10 points) Write a main program that will get the lower and upper bounds, x_l and x_u , from a user using the **input** statement, and call the functions developed earlier as needed to find a root of the polynomial. In order for the

Bisection search to work, the value of $p(x)$ should have different signs at the lower and upper bounds. If the value of $p(x)$ has the same sign at the lower and upper bounds provided by the user, the program should print an error message and quit. Otherwise, it should call the function developed for question 1.b to find a root. The program is supposed to print the root right after locating it.



2. (45 points) Consider the polynomial, $p(x)$, shown in question 1 above.
 - a. (5 points) Copy the function developed for question 1.a to use it in the program for this question as well. In addition, write a Python function for the derivative of $p(x)$ that takes x as a formal parameter and returns the derivative of $p(x)$. Take the derivative of $p(x)$ analytically with respect to x before writing the function.
 - b. (15 points) Write a Python function for root finding using the Newton-Raphson method that takes an initial guess, x_0 , as a formal parameter, and returns the first root, x_1 , of $p(x)$ such that $p(x_1) \sim 0$. x_1 can be any root of $p(x)$. At each iteration, the function is expected to print the **iteration number**, the **new guess** and the **absolute value of $p(x)$ at the guess**. Iterations should stop when the absolute value of $p(x)$ at a new guess is less than 10^{-6} . The function will return that guess as x_1 . The Newton-Raphson iteration that will be implemented for the first root is given as

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

- c. (15 points) Write a Python function for root finding using the Newton-Raphson method that takes an initial guess, x_0 , the first root, x_1 (which will be found by the function developed for question 2.b), as formal parameters, and returns the second root, x_2 , of $p(x)$ such that $p(x_2) \sim 0$. x_2 can be any remaining root of $p(x)$ except x_1 . At each iteration, the function is expected to print the **iteration number**, the **new guess** and the **absolute value of $p(x)$ at the guess**. Iterations should stop when the absolute value of $p(x)$ at a new guess is less than 10^{-6} . The function will return that guess as x_2 . The Newton-Raphson iteration that will be implemented for the second root is given as

$$x_{i+1} \approx x_i - \frac{f(x_i)}{f'(x_i) - \frac{f(x_i)}{x_i - x_1}}$$

- d. (10 points) Write a main program that will get the initial guess, x_0 , from a user using the **input** statement, and call the functions developed earlier as needed to find the first and second roots of the polynomial. It should call the function developed for question 2.b followed by the function developed for question 2.c to find the two roots. The program is supposed to print the first and second roots right after locating them.

3. (25 points) Consider the following function:

$$U_n = \begin{cases} 0, & n = 0, \\ 1, & n = 1, \\ U_{n-1} + U_{n-2} & n > 1. \end{cases}$$

- a. (15 points) Write a **recursive** Python function that gets a nonnegative integer, n , as a formal parameter and returns U_n .
- b. (10 points) Write a main program that gets n as a nonnegative integer from a user using the **input** statement, calls the function developed for question 3.a to compute U_n , and prints n and U_n . Assume that the user enters a proper input, so it is not necessary to check if n is a nonnegative integer.