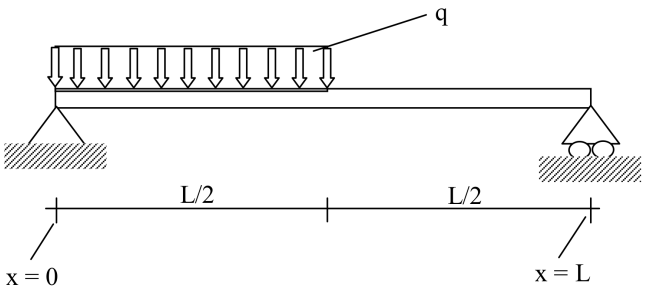
Assignment 03: Functions and modules
Assigned: 22nd October 2019
Due: 06th November 2019 at 5 PM

Note: Please upload your solution as an ipynb file to the Canvas page.

The purpose of this assignment is to develop your skills in writing functions and modules.

1. Write a function to solve for the deflection of a beam supporting a uniformly distributed load q to half its length as shown below. The beam could be both simply supported or fixed end. Assume a default argument of simply supported. Write a Python function using conditional statements (if/else) to compute the deflection at any location x along the length of the beam.



For a given loading, the deflection of a simply supported beam $\delta(x)$ is:

$$\delta(x) = \frac{qx}{384EI}(9L^3 - 24Lx^2 + 16x^3) \qquad 0 \leq x \leq \frac{L}{2}$$

and

$$\delta(x) = \frac{qL}{384EI}(8x^3 - 24Lx^2 + 17L^2x - L^3) \qquad \frac{L}{2} \leq x \leq L$$

For a given loading, the deflection of a fixed-end beam $\delta(x)$ is:

$$\delta(x) = \frac{qx}{384EI}(11L^2x - 26Lx^2 + 16x^3) \qquad 0 \leq x \leq \frac{L}{2}$$

and

$$\delta(x) = \frac{qL}{384EI}(-L^3 + 8L^2x - 13Lx^2 + 6x^3) \qquad \frac{L}{2} \leq x \leq L$$

Note that $x < 0$ and $x > L$ are invalid locations. Use the following values for the various parameters involved in the above expressions:

$$q = 4000\ lb/ft$$
$$L = 20\ ft$$
$$EI = 1.2x10^8\ lb.ft^2$$

Using these values, obtain the deflection at 3 locations: $x = L/4, L/2, 3L/4$ for both simply supported and fixed end beams.

2. Create a user defined function called `bisection` that implements the Bisection Method for computing the roots of equations. The equation that is to be solved should be defined in a separate user defined function called `fn` and this function should be used within the bisection function. Use the bisection function within a Python script to find a root of an equation (see below). The arguments for the bisection function should be the initial guesses for $x0$ and $x1$. The returned values from the bisection function should be the identified root, the number of iterations it took to obtain the root, and the relative error.

   Your bisection function should stop iterating on the root when the approximate relative error (ε_a) is less than 0.01%.

   Your program should contain extensive error checking. At a minimum, it must include the following features:

   (a) If the initial guesses are for an invalid range, then your program should return an error message.

   (b) If the number of iterations it takes to converge to a solution exceeds 300, your program should produce an error message indicating that the solution could not be found within the specified number of iterations.

   When debugging your program it is important to solve a problem for which you know the answer. Use the simple polynomial for the accelerating car (given below):

   $$f(t) = 0.5 * a * t^2 + v_0 * t - d$$

   Specify, $a = 0.6m/s^2$, $v_0 = 4.5m/s$, and $d = 50m$ and find the root at $t \approx 7.4s$.

3. Develop a Python module called `utmath` that computes the following.

   (a) `utmath.pi`

   (b) `utmath.e`

   (c) `utmath.max(a, b)`

   (d) `utmath.min(a, b)`

   (e) `utmath.abs(x)`

   (f) `utmath.sin(x, tolerance)` (using an approximate solution for sin with a default argument of $1e-6$ of tolerance)

(g) `utmath.cos(x, tolerance)` (using an approximate solution for cos with a default argument of $1e-3$ of tolerance)

Using the defined modules compute the following:

(a) `utmath.pi * 5**2`

(b) `print(utmath.e)`

(c) `utmath.max(5, 3)`

(d) `utmath.min(-3, -7)`

(e) `utmath.abs(-5)`

(f) `utmath.sin(0.5)`

(g) `utmath.sin(0.5, 1e-8)`

(h) `utmath.cos(1.5)`

(i) `utmath.cos(1.5, 1e-6)`