Finding root of an odd degree polynomial

In this exercise, we will implement a bisection-based algorithm that approximately finds a root of an odd degree polynomial $p(t) := p_0 + p_1 t + p_2 t^2 + \ldots + p_n t^n$ (for odd n).

Exercise 1
Write a function `evaluate_polynomial(p, t)` that takes a list `p` of coefficients of the polynomial (each element in this list is of type `float`), and an argument `t` of type `float`. Your function should return a value $p[0] + p[1] * t + p[2] * t^2 + \ldots + p[n] * t^n$.

Exercise 2
Write a function `bound(p)` that takes as an argument a list `p` of coefficients with the last element $p[n] > 0$ where $n = len(p) - 1$ is the index of the last element of the list. The function should return $\sum_{i=0}^{n} |p[i]|/p[n]$.

Hint: you can use a built-in `abs(x)` function that returns an absolute value of its argument (or you can implement it yourself).

Discussion
Note that for an odd-degree polynomial p with $p_n > 0$ (and $n = 2k+1$), if we define $B = \sum_i |p_i|/p_n$, we have $p(B) > 0$ and $p(-B) < 0$. I provide a proof of this fact below, you do not need to read it to solve the problem.

Expanding definition of p, we have

$$\begin{aligned} p(B) &= p_n B^n + \sum_{i=0}^{n-1} p_i B^i \\ &\geq p_n B^n - \sum_{i=0}^{n-1} |p_i| B^i \end{aligned}$$

Since $B = \sum_i |p_i|/p_n \geq |p_n|/p_n = 1$, for $i \leq n-1$ we have $B^i \leq B^{n-1}$, and hence

$$p(B) \geq p_n B^n - \sum_{i=0}^{n-1} |p_i| B^{n-1}$$

Since $\sum_{i=0}^{n-1} |p_i| < B p_n$ by definition of B, we have

$$p_n B^n - \sum_{i=0}^{n-1} |p_i| B^{n-1} > p_n B^n - p_n B^n = 0.$$

Combining those inequalities yields $p(B) > 0$. Similar calculation shows that $p(-B) < 0$.

Exercise 3

Write a function `approximate_root(p, precision)`, that takes as an input a list of coefficients `p` (again, assuming that `p[n] > 0`), and a floating point value `precision`. The function should find a value $\tilde{x}$ close to a root of the polynomial x, using the binary search algorithm. That is, the value $\tilde{x}$ returned by the function `approximate_root` should be such that there is some x with $p(x) = 0$, and $|x - \tilde{x}| < \mathrm{precision}$.

To this end, your function should keep two endpoints a, b of an interval (satisfying $a < b$, initially $a := -B$ and $b := B$), s.t. at all times we have $p(a) < 0$ and $p(b) > 0$. Note that, whenever we have a pair $a < b$, such that $p(a) < 0$ and $p(b) > 0$, by continuity there is some $x \in [a, b]$ with $p(x) = 0$.

In a loop your functionf should compute the midpoint $m := (a + b)/2$, and check if $p(m) > 0$ (use the function `evaluate_polynomial` from the first exercise to do this). Depending on the result of the comparison, update either a or b, halving the length of the interval. The loop should keep running as long as $|a - b| > \mathrm{precision}$. After the loop has terminated, it is guaranteed that $|a - b| < \mathrm{precision}$, $p(a) < 0$ and $p(b) > 0$. Your function can now return any element in the interval $[a, b]$ (for example a, b, or $(a + b)/2$.