Peak Finder

Write a subprogram `findpeaks` that receives in input a list of real values representing a sequence, a signal, a line, and identifies peaks, that can be defined as a point in the sequence of values such that:

- the value is higher than the adjacent ones (of a plateau, the first point is identified as a peak)
- the first and last items of the sequence cannot be peaks

A plateau is a sequence of points at the same height, higher than the preceding and **following** points.

To introduce flexibility, the subprogram receives a second parameter `mph` to eventually specify that the height of a point has to be greater than a given threshold `mph` (minimum peak height) for it to be considered a peak. When the parameter is set to `None`, all peaks are to be computed and returned.

The subprogram returns a list of values corresponding to the index of the elements in the sequence corresponding to peaks.

To have an idea of what it is expected, consider these use cases:

- input (values): `[5,1,1.4,1,0.9,1,1,1.1,1,0.9,1,1.1,1,1,0.9,1,1,1.1,5,5,1,1,1.1,0.9,1,1.1,1,1,0.9,1,1.1,1,1,1.1,1,0.8,0.9,1,1.2,0.9,1,1,1.1,1.2,1,1.5,1,1.1,0.9,8,3,2,1,1,1,0.9,1,1,3,2.6,7,3,3.2,2,1,1,0.8,4,4,2,2.5,1,1,1]`
- input (mph): `None`
- output: `[2, 5, 7, 11, 15, 18, 22, 25, 30, 33, 38, 40, 43, 45, 47, 49, 56, 58, 60, 62, 67, 70]`

- input (values): `[5,1,1.4,1,0.9,1,1,1.1,1,0.9,1,1.1,1,1,0.9,1,1,1.1,5,5,1,1,1.1,0.9,1,1.1,1,1,0.9,1,1.1,1,1,1.1,1,0.8,0.9,1,1.2,0.9,1,1,1.1,1.2,1,1.5,1,1.1,0.9,8,3,2,1,1,1,0.9,1,1,3,2.6,7,3,3.2,2,1,1,0.8,4,4,2,2.5,1,1,1]`
- input (mph): `1.2`
- output: `[2, 18, 38, 43, 45, 49, 58, 60, 62, 67, 70]`

- input (values): `[-5,-1,-1.4,-1,-0.9,-1,-1,-1.1,-1,-0.9,-1,-1.1,-1,-1,-0.9,-1,-1,-1.1,-5,-5,-1,-1,-1.1,-0.9,-1,-1.1,-1,-1,-0.9,-1,-1.1,-1,-1,-1.1,-1,-0.8,-0.9,-1,-1.2,-0.1,-1,-1,-1.1,-1.2,-1,-1.5,-1,-1.1,-0.9,-8,-3,-2,-1,-1,-1,-0.9,-1,-1,-3,-2.6,-7,-3,-3.2,-2,-1,-1,-0.8,-4,-4,-2,-2.5,-1,-0.0,-1]`
- input (mph): `None`
- output: `[1, 4, 9, 14, 20, 23, 28, 31, 35, 39, 44, 46, 48, 55, 59, 61, 66, 69, 72]`