Note about the code:

Python:

struct.pack('<H', arraySize) is a function from the **struct** module in Python that packs the **arraySize** variable into a binary representation as a little-endian unsigned short (**<H**). It converts the numerical value **arraySize** into a sequence of bytes.

Let's break down the expression:

- **struct.pack()** is the function that packs values into a binary representation according to a specified format.
- **'<H'** is the format string specifying the packing format. **<** indicates little-endian byte order, and **H** indicates an unsigned short integer.
- **arraySize** is the value being packed into the binary representation.

The resulting packed binary data will be a sequence of bytes representing the **arraySize** value. The length of the resulting packed data will depend on the format specifier used.

In the context of the code snippet, the line **ser.write(struct.pack('<H', arraySize))** packs the value of **arraySize** as an unsigned short into a binary representation and writes it to the serial port using **ser.write()**. It allows the Arduino code to correctly interpret and extract the array size from the received bytes.

struct.pack('<' + 'f' * arraySize, *numbers) is another usage of the **struct.pack()** function to pack multiple float values into a binary representation. It dynamically constructs the format string based on the number of elements in the **numbers** array.

Let's break down the expression:

- **struct.pack()** is the function that packs values into a binary representation according to a specified format.
- **'<' + 'f' * arraySize** is the format string being constructed dynamically. It concatenates the little-endian byte order specifier **<** with the **'f'** format specifier for floats repeated **arraySize** times. This creates a format string that represents **arraySize** number of float values.
- ***numbers** is the unpacking operator (*) used to pass the elements of the **numbers** list as individual arguments to **struct.pack()**.