Review of "groupAnagrams" method

Correctness
The code seems correct and does what it is supposed to do, which is grouping the input list of strings into their anagram groups.

Efficiency
The time complexity of the code is O(N*MlogM), where N is the number of strings in the input list and M is the maximum length of a string. This is because for each string, it sorts the characters in the string using the built-in sorted function which takes MlogM time complexity. This operation is done N times. Therefore, the overall time complexity is N*MlogM. This solution is a reasonably efficient one.

Style
The code follows PEP 8 style guide for Python code, which is great. However, there are some minor issues with the code style.
* There should be two blank lines before the `ob1` object creation line.
* The variable name `x` is not descriptive. A better name would be `sorted_string`.
* The indentation level for the `groupAnagrams` method definition should be increased by 4 spaces.

Documentation
There is no documentation provided for the code, such as a brief explanation of the purpose of the method and what it returns. Adding docstrings to the method would help other developers to understand the code more quickly and avoid the need to read the code.

Positive Aspects
* The code is clear and easy to understand.
* The code follows PEP 8 style guide.

Suggestions for Improvement
* Add docstrings to the `groupAnagrams` method.
* Use a more descriptive variable name instead of `x`.
* Add two blank lines before creating the `ob1` object.
* Increase the indentation level for the `groupAnagrams` method definition by 4 spaces.

Overall, this is a good implementation of the group anagrams problem. With the minor improvements suggested, the code will be even better. Great job!