PRACTICAL 7: Implementing coding practices in Python using PEP8.

- **What is PEP-8?**

Indeed coding and applying logic is the foundation of any programming language but there's also another factor that every coder must keep in mind while coding and that is the coding style. Keeping this in mind, Python maintains a strict way of order and format of scripting.Following this sometimes mandatory and is a great help on the user's end, to understand. Making it easy for others to read code is always a good idea, and adopting a nice coding style helps tremendously for that.

For Python, PEP 8 has emerged as the style guide that most projects adhere to; it promotes a very readable and eye-pleasing coding style. Every Python developer should read it at some point; here are the most important points extracted for you:

1. **Use 4-space indentation and no tabs.**
   The 4 space rule is not always mandatory and can be overruled for continuation line.

2. **Use docstrings:** There are both single and multi-line docstrings that can be used in Python. However, the single line comment fits in one line; triple quotes are used in both cases. These are used to define a particular program or define a particular function.

3. **Wrap lines so that they don't exceed 79 characters:** The Python standard library is conservative and requires limiting lines to 79 characters. The lines can be wrapped using parenthesis, brackets, and braces. They should be used in preference to backslashes.

4. **Uses of regular and updated comments are valuable to both the coders and users:** There are also various types and conditions that if followed can be of great help from programs and users point of view. Comments should form complete sentences. If a comment is a full sentence, its first word should be capitalized, unless it is an identifier that begins with a lower case letter. In short comments, the period at the end can be omitted. In block comments, there are more than one paragraphs and each sentence must end with a period. Block comments and inline comments can be written followed by a single '#'.

5. **Use of trailing commas:** This is not mandatory except while making a tuple.

6. Use Python's default UTF-8 or ASCII encodings and not any fancy encodings, if it is meant for international environment.

7. Use spaces around operators and after commas, but not directly inside bracketing.

8. **Naming Conventions:** There are few naming conventions that should be followed in order to make the program less complex and more readable. At the same time, the naming conventions in Python are a bit of mess, but here are few conventions that can be followed easily. There is an overriding principle that follows that the names that are visible to the user as public parts of API should follow conventions that reflect usage rather than implementation.
   In addition to these few leading or trailing underscores are also considered.
   single_trailing_underscore_: used to avoid conflicts with Python keyword.
   __double_leading_underscore: when naming a class attribute, invokes name mangling.

9. **Characters that should not be used for identifiers:** ‘l’ (lowercase letter el), ‘O’ (uppercase letter oh), or ‘I’ (uppercase letter eye) as single character variable names as these are similar to the numerals one and zero.

10. Don’t use non-ASCII characters in identifiers if there is only the slightest chance people speaking a different language will read or maintain the code.

11. **Name your classes and functions consistently:** The convention is to use Camel Case for classes and lower_case_with_underscores for functions and methods. Always use self as the name for the first method argument.

12. While naming of function of methods always use self for the first argument to instance methods and cls for the first argument to class methods. If a functions argument name matches with reserved words then it can be written with a trailing comma.