Python Coding Style

Most languages can be written (or more concise, *formatted*) in different styles; some are more readable than others. Making it easy for others to read your code is always a good idea, and adopting a nice coding style helps tremendously for that.

For Python, “Python Enhancement Proposal 8”, also known as “PEP 8”, [see https://www.python.org/dev/peps/pep-0008/] 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:

- Use 4-space indentation, and no tabs.

  4 spaces are a good compromise between small indentation (allows greater nesting depth) and large indentation (easier to read). Tabs introduce confusion, and are best left out.

- Wrap lines so that they don’t exceed 79 characters.

  This helps users with small displays and makes it possible to have several code files side-by-side on larger displays.

- Use blank lines to separate functions and classes, and larger blocks of code inside functions.

- When possible, put comments on a line of their own.

- Use docstrings: string line(s) that occurs as the first statement in a module, function, class, or method definition, enclosed between """.

- Use spaces around operators and after commas, but not directly inside bracketing constructs. Example: a = f(1, 2) + g(3, 4).

- Name your classes and functions consistently; the convention is to use

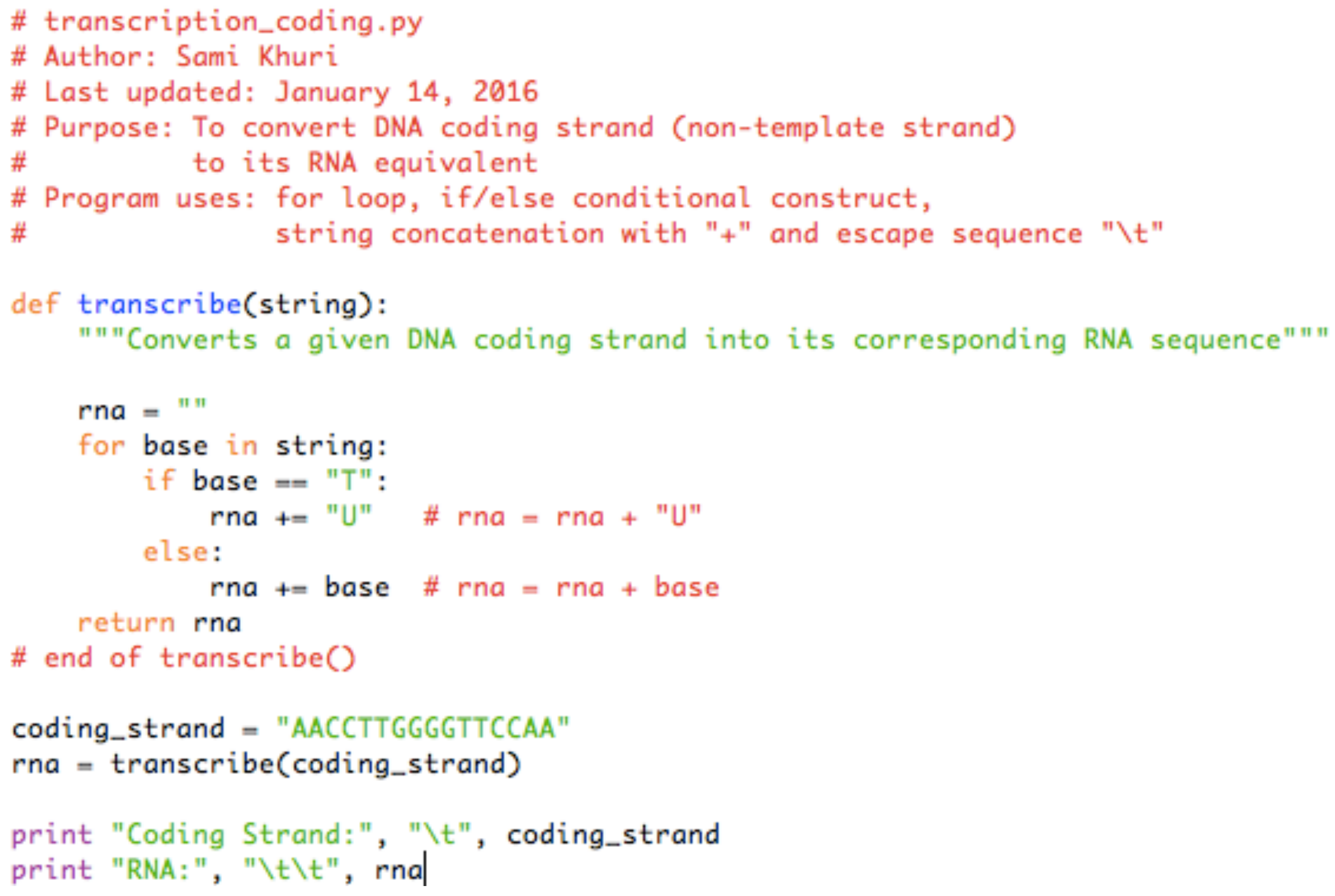
  - CamelCase for classes, and
  - lower_case_with_underscores for functions and methods.

  Always use “self” as the name for the first method argument.
  For more on classes and methods, see “A First Look at Classes” at
  https://docs.python.org/release/2.6.8/tutorial/classes.html#tut-firstclasses

- Don’t use fancy encodings if your code is meant to be used in international environments. Plain ASCII works best in any case.

Python Program Header Comments

We are going to put the following comments at the very top of the program, before even entering Python commands. We shall do that for every single Python program we write.



```
# transcription_coding.py
# Author: Sami Khuri
# Last updated: January 14, 2016
# Purpose: To convert DNA coding strand (non-template strand)
#          to its RNA equivalent
# Program uses: for loop, if/else conditional construct,
#               string concatenation with "+" and escape sequence "\t"

def transcribe(string):
    """Converts a given DNA coding strand into its corresponding RNA sequence"""

    rna = ""
    for base in string:
        if base == "T":
            rna += "U"   # rna = rna + "U"
        else:
            rna += base  # rna = rna + base
    return rna
# end of transcribe()

coding_strand = "AACCTTGGGGTTCCAA"
rna = transcribe(coding_strand)

print "Coding Strand:", "\t", coding_strand
print "RNA:", "\t\t", rna
```

No need to understand the whole program at this point. Just notice the top 8 lines of comments that contain:

1) the name of the program
2) the author of the program
3) the date of the last update of the program
4) the purpose of the program
5) the constructs, data types, etc… the program uses.

Of course one can add more comments, but please have at least these 5 comments, in that order, in every program, big or small, that you write.

Note that the program follows the 4-space indentation mentioned on page 1.