Introduction to Python

Style Requirements

Code can be complicated -- especially to the humans reading it. The Python community has established coding standards as one of the keys to writing *clear*, *readable* code. **Use of the below coding standards in your solutions is required.**

Style Standards

1. Variable names: all lowercase, with underscores between any words

   Please do not use **camelCase**, but instead **words** or **words_with_underscores**.

2. Use 4-space indents, no tabs

   Most of us will not have to worry about this because PyCharm inserts 4 spaces whenever we use the [Tab] key. If you are using a different editor, you must configure it to do the same. See me for details.

3. One space on either side of operators

```
var = aa + bb     # note one space on either side of = and +
```

4. No space between function name and argument list (parentheses)

```
string_length = len('hello')           # not len ('hello')
```

5. Use single blank lines to separate single-spaced "paragraphs" ("steps") in your code.

   Think of your code as an essay with "paragraphs", reading like a story:

```
current_year = 2021
u_name = input('please enter your name: ')
u_age = input('please enter your age: ')

ui_age = int(u_age)
uc_born = current_year - ui_age

print(f'Hi {u_name}, you were born in {uc_born}')
```

   Each "paragraph" is a "step" your code takes. Like a paragraph, it clarifies that this part of the code is a separate "thought".

   Also, please do not use more than one blank line between "paragraphs".

Code Clarity

6. Use descriptive variable names

Variable names like **xx**, **var1**, **number**, etc. force the reader to remember what values they represent, but if we use names like **line_count**, **user_guess** or **fname**, the reader can more easily recall where they came from and what they are for.

(It is true that I use non-descriptive variable names in my examples, but this is done to clarify the difference between my variables and Python's function and method names.)

7. Use **f"** strings to combine values with strings.

The **f"** string allows us to embed values (numbers or strings) into display strings. Please do not use concatenation or commas to add spaces.

f" strings are easier to write and read. Compare this busy print statement:

```
name = 'Holden'
age = 29

print('My name is ' + name + ' and my age is ' + str(age) + '.')
```

to this one:

```
name = 'Holden'
age = 29

print(f'My name is {name} and my age is {age}')
```

8. Comment your code ***only*** when needed (for clarity).

Python is very readable, which means that we don't usually need comments. But some code statements are less clear and need a comment describing what they do.

However, you must not include unnecessary or obvious comments in your solutions. They may be effectively used during the development process, but they should be removed before submitting.

9. Remove test code and development code before submitting a solution.

During the development process you may create code statements (print statements, experiments, earlier drafts, etc.) that your final program does not need. These must be removed before submitting, otherwise they will clutter the code or make its purpose less clear.

Code Organization

10. Place a "docstring" at the top of your code.

    This should go right at the top of your program and can include any information you think necessary -- usually starting with name, purpose, author and date:

```
"""
    solution_1.2.py -- calculate a tip based on user input

    Author:  David Blaikie (dbb212@nyu.edu)
    Last Revised:  6/1/2021

"""
```

11. Avoid repetition (DRY principle)

    Don't **R**epeat **Y**ourself: if you find yourself issuing the same statement or group of statements more than once, ask whether the repetition can be eliminated.

    When you have finished your program, go over it once more to look for repetitions.