Welcome back from holiday. I will be back in class on Wednesday. Please be kind and courteous to your supply teachers while you work to complete the following expectations.

1. Python Style Guide Questions  
   Read the following Document: https://www.python.org/dev/peps/pep-0008/  
   Answer the questions below  
   Upload your responses to your repository
2. Tic Tac Toe Assignment (From Before the Break)  
   Reference on the GitHub Repository is “Modules D: Python”/”TicTacToe Assignment”

Please share ideas with your group members  
Be prepared to demo your game on Wednesday

Style Guide Questions

1. What is the purpose of a programming language style guide like PEP 08?

Pep 08 is intended to improve the readability of code and make it consistent across the wide spectrum of Python code

1. Provide some examples of when it would be ok to be inconsistent with the PEP 08 style guide.

* When applying the guideline would make the code less readable, even for someone who is used to reading code that follows this PEP.
* To be consistent with surrounding code that also breaks it (maybe for historic reasons) -- although this is also an opportunity to clean up someone else's mess (in true XP style).
* Because the code in question predates the introduction of the guideline and there is no other reason to be modifying that code.
* When the code needs to remain compatible with older versions of Python that don't support the feature recommended by the style guide.

1. Summarize proper indentation style. Explain some examples of good and bad indentation style.

Use 4 spaces per indentation level.

**Yes:**

# Aligned with opening delimiter.

foo = long\_function\_name(var\_one, var\_two,

var\_three, var\_four)

# More indentation included to distinguish this from the rest.

def long\_function\_name(

var\_one, var\_two, var\_three,

var\_four):

print(var\_one)

# Hanging indents should add a level.

foo = long\_function\_name(

var\_one, var\_two,

var\_three, var\_four)

**No:**

# Arguments on first line forbidden when not using vertical alignment.

foo = long\_function\_name(var\_one, var\_two,

var\_three, var\_four)

# Further indentation required as indentation is not distinguishable.

def long\_function\_name(

var\_one, var\_two, var\_three,

var\_four):

print(var\_one)

1. What should be used for whitespace, tabs or spaces?

Tabs should be used solely to remain consistent with code that is already indented with tabs.

1. Why does PEP 08 recommend limiting line length to a maximum of 79 characters?

* For flowing long blocks of text with fewer structural restrictions (docstrings or comments), the line length should be limited to 72 characters.

1. What does PEP 08 have to say about ‘single string’ quotes and “double string” quotes? How would you quote a string with an apostrophe like don’t?

* In Python, single-quoted strings and double-quoted strings are the same. This PEP does not make a recommendation for this. Pick a rule and stick to it. When a string contains single or double quote characters, however, use the other one to avoid backslashes in the string. It improves readability.

1. Summarize proper use of “whitespace”. Explain some examples of good and bad whitespace style.

* Avoid trailing whitespace anywhere. Because it's usually invisible, it can be confusing: e.g. a backslash followed by a space and a newline does not count as a line continuation marker. Some editors don't preserve it and many projects (like CPython itself) have pre-commit hooks that reject it.

Always surround these binary operators with a single space on either side: assignment (=), augmented assignment (+=, -= etc.), comparisons (==, <, >, !=, <>, <=, >=, in, not in, is, is not), Booleans (and, or, not).

If operators with different priorities are used, consider adding whitespace around the operators with the lowest priority(ies). Use your own judgment; however, never use more than one space, and always have the same amount of whitespace on both sides of a binary operator.

**Yes:**

ham[1:9], ham[1:9:3], ham[:9:3], ham[1::3], ham[1:9:]

ham[lower:upper], ham[lower:upper:], ham[lower::step]

ham[lower+offset : upper+offset]

ham[: upper\_fn(x) : step\_fn(x)], ham[:: step\_fn(x)]

ham[lower + offset : upper + offset]

**No:**

ham[lower + offset:upper + offset]

ham[1: 9], ham[1 :9], ham[1:9 :3]

ham[lower : : upper]

ham[ : upper]

1. What is the recommended use of a “block comment”. Provide an example of a block comment and explain how a block comment is different from a “inline comment”

* Block comments generally apply to some (or all) code that follows them, and are indented to the same level as that code. Each line of a block comment starts with a # and a single space (unless it is indented text inside the comment).

Paragraphs inside a block comment are separated by a line containing a single #.

1. Summarize proper use of “inline comments”. Explain some examples of good and bad inline comment style.

* An inline comment is a comment on the same line as a statement. Inline comments should be separated by at least two spaces from the statement. They should start with a # and a single space.

1. Which naming styles have you seen used on example code provided in class so far?

b (single lowercase letter)

B (single uppercase letter)

lowercase

lower\_case\_with\_underscores

UPPERCASE

UPPER\_CASE\_WITH\_UNDERSCORES

CapitalizedWords (or CapWords, or CamelCase -- so named because of the bumpy look of its letters [4]). This is also sometimes known as StudlyCaps.

mixedCase (differs from CapitalizedWords by initial lowercase character!)

Capitalized\_Words\_With\_Underscores (ugly!)

1. What is the recommended naming style for types and variables? Provide an example of a variable definition using this style.

* Names of type variables introduced in PEP 484 should normally use CapWords preferring short names: T, AnyStr, Num. It is recommended to add suffixes \_co or \_contra to the variables used to declare covariant or contravariant behavior correspondingly. Examples: from typing import TypeVar

VT\_co = TypeVar('VT\_co', covariant=True)

KT\_contra = TypeVar('KT\_contra', contravariant=True)

1. What is the recommended naming style for function names? Provide an example of a function definition using this style.

* Function names should be lowercase, with words separated by underscores as necessary to improve readability.
* Mixed Case is allowed only in contexts where that's already the prevailing style (e.g. threading.py), to retain backwards compatibility.

1. What is the recommended naming style for constants? Provide an example of a constant definition using this style.

* Constants are usually defined on a module level and written in all capital letters with underscores separating words. Examples include MAX\_OVERFLOW and TOTAL.