

**Week 3.1**

**Student Worksheet**

**Part 1**

**Activities (Time: 40 min)**

**I. Before you read**

**Part 1. What’s scanning? Mark the sentences T (True) or False (F).**

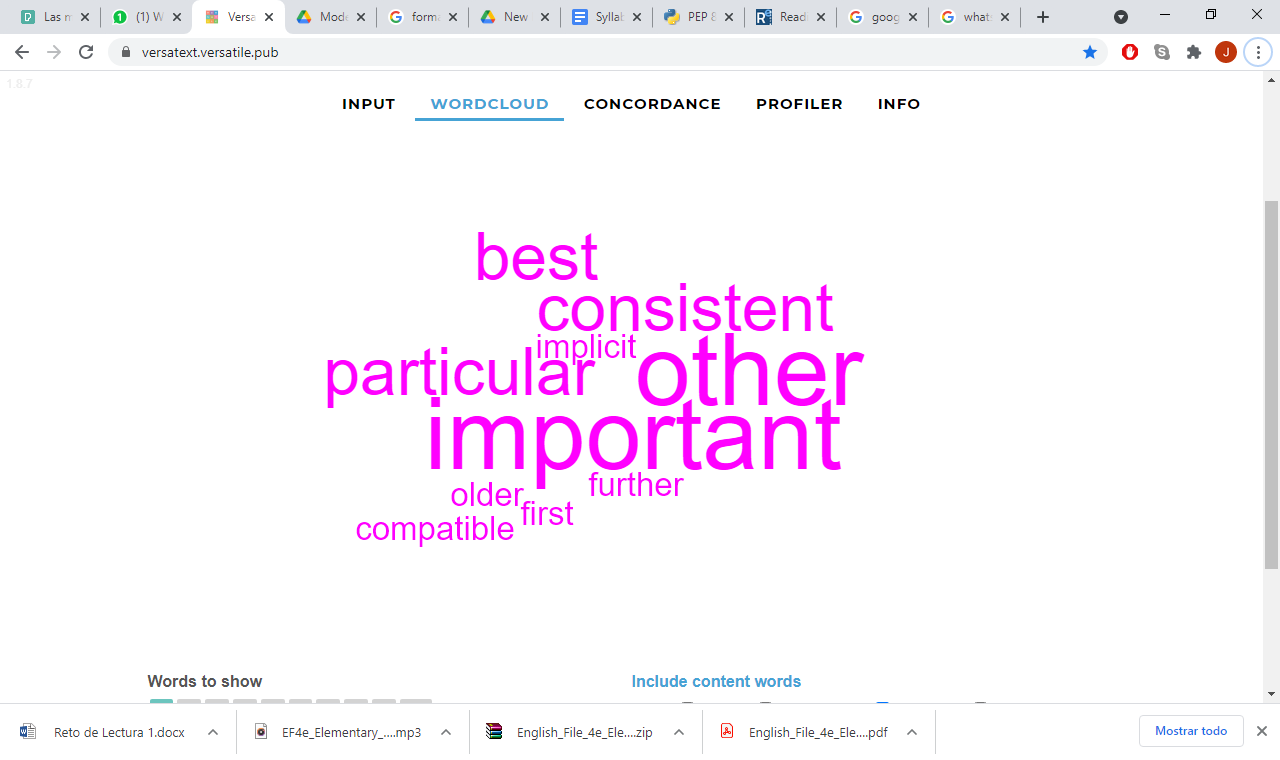
1. It is a strategy to understand the general idea of the text. \_\_\_\_
2. It is a machine to copy files. \_\_\_\_
3. It is a reading strategy to identify specific information. \_\_\_\_
4. It is a technique to study English. \_\_\_\_

**Part 2. Circle the correct option.**

The steps to scan a text are:

1. Read the complete text, identify cognates and find the general idea
2. Keep in mind the main question(s), read the text quickly identifying key information, pay attention to titles, subtitles and abstract to find the answer(s)
3. Read the beginning and the ending of each paragraph to find general ideas
4. Start checking the context and the cognates, the read titles, subtitles and abstract and connect main ideas

**Part 3. Check the following words carefully and answer the questions:**



1. Are these words adjectives/nouns/verbs?
2. Which words are cognates in this list? Mention them.
3. Which words don’t you know their meaning? Circle them and look them up in a dictionary.

**Part 4. Look at the chart below. Match the word with its correct meaning.**

|  |  |  |
| --- | --- | --- |
| **Word** | **#** | **Meaning** |
| 1. Comprising |  | **a.** adj. silly, stupid, a lack of good sense |
| 2. Consistency |  | **b.** verb. to include or contain |
| 3. Indentation |  | **c.** noun. a space of the lay-out in a page |
| 4. Insights |  | **d.** noun. harmony or uniformity |
| 5. Foolish |  | **e.** noun. intuitive understanding or idea |
| 6. Readability |  | **f.** noun. easy to read |
| 7. Standard |  | **g.** noun. design or structure |
| 8. Lay-out |  | **h.** noun. common, usual |

**II. While you read**

**Part 5. Scan. Find these answers to the questions in the reading as quickly as you can. You will check your answers later.**

1. What´s a style guide about?
2. What are 2 reasons to avoid applying these guidelines?
3. What’s the limit of characters in each line?
4. What are the differences between single and double quoted strings?

**PEP\* 8 -- Style Guide for Python Code**

**Introduction**

This document gives coding conventions for the Python code comprising the standard library in the main Python distribution. Please see the companion informational PEP describing style guidelines for the C code in the C implementation of Python [1].

This document and PEP 257 (Docstring Conventions) were adapted from Guido's original Python Style Guide essay, with some additions from Barry's style guide [2].

This style guide evolves over time as additional conventions are identified and past conventions are rendered obsolete by changes in the language itself.

Many projects have their own coding style guidelines. In the event of any conflicts, such project-specific guides take precedence for that project.

**A Foolish Consistency is the Hobgoblin of Little Minds**

One of Guido's key insights is that code is read much more often than it is written. The guidelines provided here are intended to improve the readability of code and make it consistent across the wide spectrum of Python code. As PEP 20 says, "Readability counts".

A style guide is about consistency. Consistency with this style guide is important. Consistency within a project is more important. Consistency within one module or function is the most important.

However, know when to be inconsistent -- sometimes style guide recommendations just aren't applicable. When in doubt, use your best judgment. Look at other examples and decide what looks best. And don't hesitate to ask!

In particular: do not break backwards compatibility just to comply with this PEP!

Some other good reasons to ignore a particular guideline:

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.

**Code Lay-out**

**Indentation**

Use 4 spaces per indentation level.

Continuation lines should align wrapped elements either vertically using Python's implicit line joining inside parentheses, brackets and braces, or using a hanging indent [7]. When using a hanging indent the following should be considered; there should be no arguments on the first line and further indentation should be used to clearly distinguish itself as a continuation line:

# Correct:

# Aligned with opening delimiter.

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

var\_three, var\_four)

# Add 4 spaces (an extra level of indentation) to distinguish arguments 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)

# Wrong:

# 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)

**Tabs or Spaces?**

Spaces are the preferred indentation method.

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

Python 3 disallows mixing the use of tabs and spaces for indentation.

Python 2 code indented with a mixture of tabs and spaces should be converted to using spaces exclusively.

When invoking the Python 2 command line interpreter with the -t option, it issues warnings about code that illegally mixes tabs and spaces. When using -tt these warnings become errors. These options are highly recommended!

**Maximum Line Length**

Limit all lines 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.

Limiting the required editor window width makes it possible to have several files open side by side, and works well when using code review tools that present the two versions in adjacent columns.

The default wrapping in most tools disrupts the visual structure of the code, making it more difficult to understand. The limits are chosen to avoid wrapping in editors with the window width set to 80, even if the tool places a marker glyph in the final column when wrapping lines. Some web based tools may not offer dynamic line wrapping at all.

Some teams strongly prefer a longer line length. For code maintained exclusively or primarily by a team that can reach agreement on this issue, it is okay to increase the line length limit up to 99 characters, provided that comments and docstrings are still wrapped at 72 characters.

The Python standard library is conservative and requires limiting lines to 79 characters (and docstrings/comments to 72).

The preferred way of wrapping long lines is by using Python's implied line continuation inside parentheses, brackets and braces. Long lines can be broken over multiple lines by wrapping expressions in parentheses. These should be used in preference to using a backslash for line continuation.

Backslashes may still be appropriate at times. For example, long, multiple with-statements cannot use implicit continuation, so backslashes are acceptable:

with open('/path/to/some/file/you/want/to/read') as file\_1, \

open('/path/to/some/file/being/written', 'w') as file\_2:

file\_2.write(file\_1.read())

**String Quotes**

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.

For triple-quoted strings, always use double quote characters to be consistent with the docstring convention in PEP 257.

'blah': one (); two(); three()

**Glossary**

1. PEP: stands for Python Enhancement Proposal. A PEP is a design document providing information to the Python community, or describing a new feature for Python or its processes or environment.

**III. After you read**

**Part 6. Read the following statements and tick yes or no in each case.**

|  |  |  |
| --- | --- | --- |
| **Statements** | **Yes** | **No** |
| 1. I should scan a text to find specific information |  |  |
| 2. I read this text in detail |  |  |
| 3. I identify general ideas of the text while I am scanning |  |  |
| 4. I can predict the meaning of the words that are similar to Spanish |  |  |

Click on the link below to read the full text: <https://www.python.org/dev/peps/pep-0008/>

**Answer Key**

**Part 1** Answers: 1. F 2. F 3. T 4. F

**Part 2** Answer: 2

**Part 3** Answers: 1. Adjectives 2. Consistent/ particular / important / implicit/ compatible 3. Answers may vary.

**Part 4** Answers: 1. d 2. d 3. c 4. e 5.a 6. F 7.h 8. g

**Part 5.** Answers:

1. style guide is about consistency

2. 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.

3. Limit all lines to a maximum of 79 characters.

4. In Python, single-quoted strings and double-quoted strings are the same

**Part 6. Read the following statements and tick yes or no in each case.**

|  |  |  |
| --- | --- | --- |
| **Statements** | **Yes** | **No** |
| 1. I should scan a text to find specific information | X |  |
| 2. I read this text in detail |  | X |
| 3. I identify general ideas of the text while I am scanning |  | X |
| 4. I can predict the meaning of the words that are similar to Spanish | X |  |