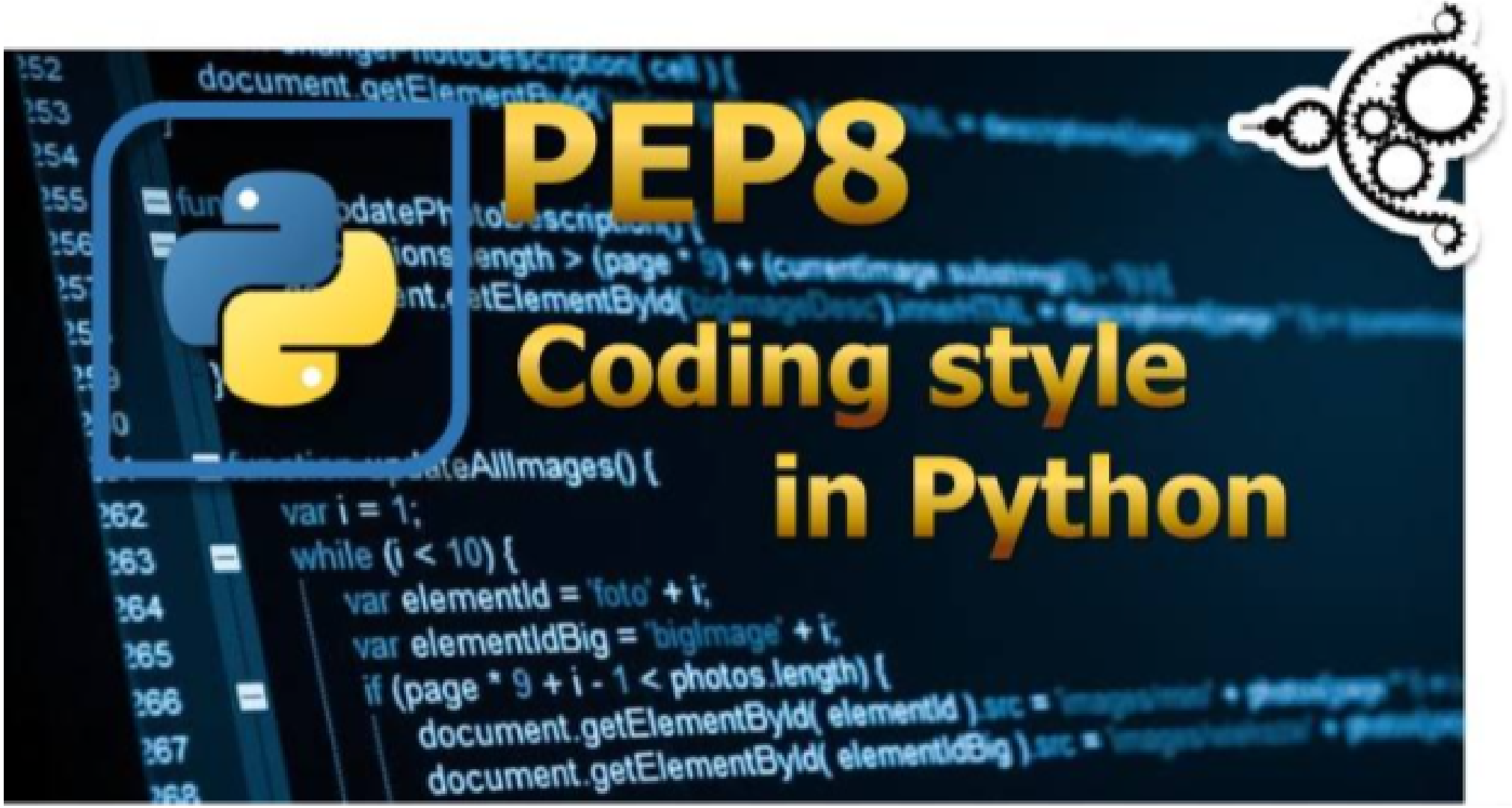
Python


PEP8
Coding style
in Python
var i = 1;
while (i < 10) {
var elementId = 'foto' + i;
var elementIdBig = 'bigImage' + i;
if (page * 9 + i - 1 < photos.length) {
document.getElementById(elementId)
document.getElementById(elementIdBig)

What is PEP for Python?

In this article, we will explain to you the **PEP** i.e Python Enhancement Proposal in Python.

PEP is an abbreviation for **Python Enhancement Proposal**. A PEP is a design document that informs the Python community or describes a new feature for Python, its processes, or its environment. The PEP should provide a brief technical description of the feature as well as its reasoning. PEPs are intended to be the primary mechanisms for proposing important new features, gathering community input on a problem, and documenting Python design decisions.

The PEP author is responsible for building consensus within the community and documenting opposing (dissenting) opinions.

Because the PEPs are stored as text files in a versioned repository, their revision history serves as the feature proposal's historical record. This historical record can be accessed using standard git commands to get previous versions, and it can also be browsed on GitHub.

Types of PEP

PEP is classified into three types −

- A **Standards Track** PEP defines a new Python feature or implementation. It may also specify an interoperability standard that will be supported outside of the standard library for current Python versions until a future PEP adds standard library support.
- An **Informational** PEP addresses a Python design issue or offers general guidance or information to the Python community without proposing a new feature. Users and implementers are allowed to ignore or follow Informational PEPs because they do not necessarily represent a Python community consensus or recommendation.
- A **Process PEP** describes a Python-related process or proposes a change to (or event in) a process. Process PEPs are similar to Standards Track PEPs in that they apply to areas other than the Python language. They may suggest an implementation, but not to the Python codebase; they frequently require community consensus; thus, unlike Informational PEPs, they are more than recommendations that users are not free to disregard/ignore them. Procedures, guidelines, changes to the decision-making process, and modifications to the tools or environment used in Python development are all examples. Any meta-PEP is a Process PEP as well.

Why PEP 8 is Important?

PEP 8 enhances the readability of the Python code, but why is readability so important? Let's understand this concept.

Creator of Python, Guido van Rossum said, **"Code is much more often than it is written."** The code can be written in a few minutes, a few hours, or a whole day but once we have written the code, we will never rewrite it again. But sometimes, we need to read the code again and again.

At this point, we must have an idea of why we wrote the particular line in the code. The code should reflect the meaning of each line. That's why readability is so much important.