Zen of Python

Beautiful is better than ugly.

Explicit is better than implicit.

Simple is better than complex.

Complex is better than complicated.

Flat is better than nested.

Sparse is better than dense.

Readability counts.

Special cases aren't special enough to break the rules.

Although practicality beats purity.

Errors should never pass silently.

Unless explicitly silenced.

In the face of ambiguity, refuse the temptation to guess.

There should be one-- and preferably only one --obvious way to do it.

Although that way may not be obvious at first unless you're Dutch.

Now is better than never.

Although never is often better than \*right\* now.

If the implementation is hard to explain, it's a bad idea.

If the implementation is easy to explain, it may be a good idea.

Namespaces are one honking great idea -- let's do more of those!

**Explanation of Key Principals**   
  
**Beautiful is better than ugly.**

The beauty of writing code that is aesthetically pleasing and understandable, not just functional.

**Explicit is better than implicit.**

I believe this to be that code should be straightforward and clear. Rather than relying on assumptions, which can make code difficult to maintain in the long run, especially if the code is being done in a collaborative setting.

**Simple is better than complex.**

It is very easy to overthink code, creating a complex situation with many lines of code, that could be otherwise completed with a single line. Additionally, being simply makes it easier to maintain and read.

**Readability Counts.**

With Python being a very easy code to write and understand, it is important that it is readable. This key principle is like Python’s design as a language that can be read by not just machine, but also a human even with next to no coding experience.

**Special cases aren't special enough to break the rules.**

Avoid making exceptions that complicate the code unless necessary. For example, for a valid email login, it is a special case to ensure there a password has lower/upper case, numbers, symbols, more than X characters etc, which is necessary for security.

**Although practicality beats purity.**

It can be okay to sacrifice tidy/clean in favor of practicality when solving real world issues.

**Errors should never pass silently, Unless explicitly silenced.**

If there is an error, even if it’s a warning and doesn’t crash the code, it should be handled, as by leaving it alone you may end up with unexpected behaviour either now, or as you continue to add to the code.

**In the face of ambiguity, refuse the temptation to guess.**

Don’t make assumptions, ask questions, test, and make sure instead of guessing that you double check to ensure the answer is what you think it is.

**There should be one- and preferably only one- obvious way to do it.**

Unlike other languages such as C#, where there are often multiple ways to perform a task, Python aims to be a one-way straightforward solution. An upside to this is that there is very minor variability between different pieces of code done in Python. Though, a downside to this is when learning it is almost impossible to make anything unique or make it seem like you’ve done anything different to the person next to you, made worse in a world where AI exists, in which the most popular AI tools today all run off Python, though that is unlikely to last forever due to the high latency of runtime performance versus other coding languages such as Rust, making it very likely in the future that Python may become obsolete.

**Although that way may not be obvious at first unless you're Dutch.**

This is a joke referencing the creator of Python, whom is Dutch.

**Now is better than never. Although never is often better than right now.**

This speaks of a balance in your programming. It can be better sometimes to getting something done, instead of being held off indefinitely. However, there can also be risk of rushing into something without designing or considering it properly, which can lead to poor outcomes.

**If the implementation is hard to explain, it's a bad idea.**

KISS – Keep it simple stupid.

**If the implementation is easy to explain, it may be a good idea.**

KISS – Keep it simple stupid.

**Namespaces are one honking great idea -- let's do more of those!**

This encourages the use of spaces/indentations to organize code.