Principios del Zen

INGENIERIA DE SOFTWARE I

Estudiante:
Mamani Mamani Ilma Magda

Ingeneria Estadistica e Informatica

April 18, 2024

INTRODUCCIÓN

Translation to English: The Zen of Python is a collection of 20 software principles that influence the design of the Python Programming Language. Of these, 19 were written by Tim Peters in June of 1999. These principles are intended to provide guidelines for writing code in Python.

PRINCIPIOS

"de Zen"

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.

Errors should never pass silently. Unless explicitly silenced.

Ejemplo:

"Los casos especiales no son lo suficientemente especiales como para romper las reglas"

```python
def divide(numerador, denominador):
    try:
        resultado = numerador / denominador
    except ZeroDivisionError:
        print("Error: División por cero.")
        resultado = None
    except TypeError:
        print("Error: Ambos argumentos deben ser números.")
        resultado = None
    except Exception as e:
        print(f"Error: {str(e)}")
        resultado = None
    finally:
        return resultado

# Caso normal
print(divide(55, 2))  # Devuelve 5.0

# Casos especiales
print(divide(99, 0))  # Devuelve None y muestra "Error: División por cero."
print(divide(77, 'a'))  # Devuelve None y muestra "Error: Ambos argumentos deben ser números."

```

```
27.5
Error: División por cero.
None
Error: Ambos argumentos deben ser números.
None
```

University of Idaho

Ejemplo:

"Los casos especiales no son lo suficientemente especiales como para romper las reglas"

This code defines a function called "divide" that takes two arguments, **numerator** and **denominator**, and performs a division. The code is designed to handle various special cases of errors that could occur during division.

The Zen principle applies in the sense that, regardless of the errors that may occur, the function follows a consistent rule of error handling.

In all these special cases, the "rule" followed is to handle the error, print an error message, and return None. This rule is consistently applied, no matter how "special" the error case is. This ensures that the divide function is robust and handles errors in a predictable manner.

¡Thank you for your attention!