UNIVERSIDAD NACIONAL DEL ALTIPLANO

FACULTAD DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA

ESCUELA PROFESIONAL DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA

Zen Python

Presentado por:
Chañi Puaccar Beckham Vieri

Docente:
Ing. Fred Torres Cruz

Semestre:
7mo

Curso:
INGENIERIA DE SOFTWARE I

Puno - Perú
2024

Índice

Zen Python

If you ever open the Python console and type (import this) you will see the lines with the famous Python Zen:

```
>>> import this
The Zen of Python, by Tim Peters

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!
>>>
```

Python Zen consists of 19 rules that must be correctly applied to any code to turn it into Python code; by Tim Peters.

0.1. Errors should never pass silently.

When you work under pressure and find mistakes, you tend to hide or take too many actions or drastic actions such as the following:

```
#Los errores nunca deberian ucurrir silenciosamente
##########################################
def resolucion(operacion):
        if operacion == "suma":
                return a + b
        elif operacion == "resta":
                return a - b
        elif operacion == "multiplicacion":
                return a * b
        else:
                try:
                        return a / b
                except ZeroDivisionError:
                        print("No se puede dividir entre 0")
                        return "Operacion erronea"


while True:
        try:
                a = int(input("dijite a: "))
                b = int(input("dijite b: "))
                break
        except ValueError:
                print("Se necesita un dato numerico para hacer la operacion indicada")

operacion = input("que operacion desea realizar: ")
print(resolucion(operacion))
```