

1 ZEN DE PYTHON

Es un conjunto de 19 principios de diseño para escribir código Python de alta calidad. Estos principios fueron escritos por Tim Peters, activo colaborador de Python, escribió hace años, los principios que bien podrían ser la filosofía o Zen de Python. Estos principios se incorporaron dentro del propio Python y se pueden ver escribiendo `import this` en el prompt interactivo de Python.

```
>>> 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!
```

19 PRINCIPIOS DE ZEN DE PYTHON

- 1.-Bello es mejor que feo
- 2.-Explícito es mejor que implícito
- 3.-SIMPLE ES MEJOR QUE COMPLEJO
- 4.-Complejo es mejor que complicado
- 5.-Plano es mejor que anidado
- 6.-Espaciado es mejor que denso
- 7.-La legibilidad cuenta
- 8.-Los casos especiales no son tan especiales
- 9.-Aunque la practicidad le gana a la pureza
- 10.-Los errores nunca deben pasar silenciosamente
- 11.-A menos que se silencien explícitamente
- 12.-Frente a la ambigüedad, evita la tentación de adivinar
- 13.-Debería haber una -y preferiblemente sólo una- forma obvia de hacerlo
- 14.-Aunque esa forma puede no ser obvia al principio a menos que seas holandés
- 15.-Ahora es mejor que nunca
- 16.-Aunque nunca es a menudo mejor que ahora mismo
- 17.-Si la implementación es difícil de explicar, es una mala idea
- 18.-Si la implementación es fácil de explicar, puede que sea una buena idea
- 19.-Los espacios de nombres son una gran idea, ¡hagamos más de esos!

2 SIMPLE ES MEJOR QUE COMPLEJO

Cuanto más simple sea el código más fácil será trabajar con él o realizar futuras modificaciones. Además del código se tiene en cuenta el resultado, cuanto más simple sea una interfaz más fácil lo tendrá el usuario final para trabajar con ella. Comprueba este código y mira cuál te resulta más simple.

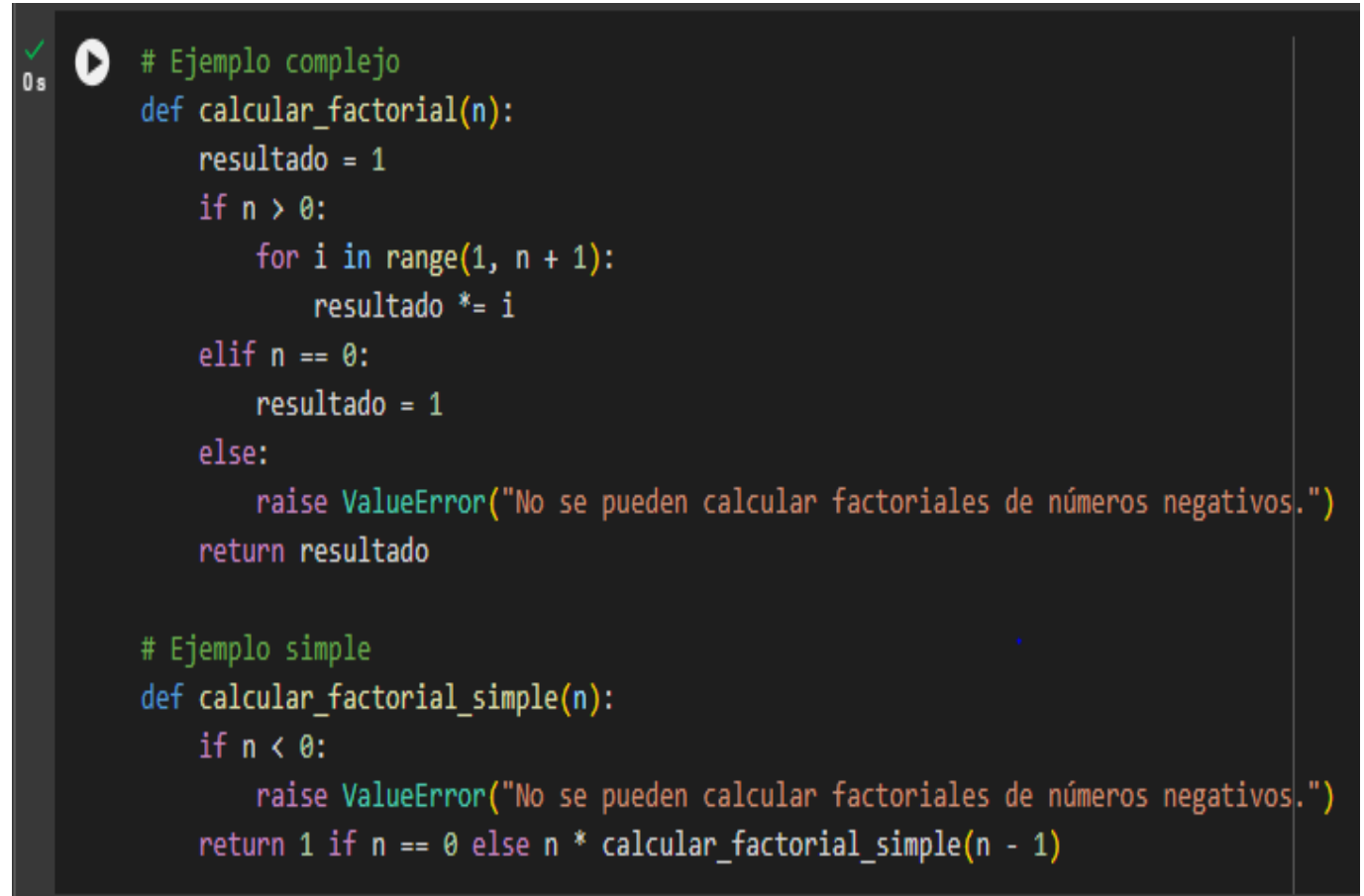
Ejemplo 1

```
a = [1,2,3,4,5,6,7,8,9] for x in range(len(a)): print(a[x])
```

Ejemplo 2

```
numeros = [1,2,3,4,5,6,7,8,9] for numero in numeros: print(numero)
```

EJEMPLO EN CÓDIGO



```
# Ejemplo complejo
def calcular_factorial(n):
    resultado = 1
    if n > 0:
        for i in range(1, n + 1):
            resultado *= i
    elif n == 0:
        resultado = 1
    else:
        raise ValueError("No se pueden calcular factoriales de números negativos.")
    return resultado

# Ejemplo simple
def calcular_factorial_simple(n):
    if n < 0:
        raise ValueError("No se pueden calcular factoriales de números negativos.")
    return 1 if n == 0 else n * calcular_factorial_simple(n - 1)
```