The lazy programmer's guide to bulletproof code.

Juan Pablo Yamamoto

Outline

Elogio de la Pereza

José Galaviz Casas

“Nuestra pereza para hacer estas labores ha exacerbado nuestro ingenio; en el mejor de los casos las tareas aburridas pueden ser hechas por algo, ya no por alguien, así hemos inventado artefactos que trabajan en nuestro lugar ...

El demonio de la pereza es también el de la invención y el descubrimiento.” [1]

Ejemplo

Dada una lista y un índice dentro de esta, regresa la misma lista pero sin el elemento en el índice dado.

Ejemplo

```
def elimina(lista, indice):
    parte_inicio = lista[0:indice]
    parte_final = lista[indice+1:len(lista)]

    return parte_inicio + parte_final
```

Ejemplo

```
# Elimina el primer índice
assert elimina([0,1,2,3,4], 0) == [1,2,3,4]

# Elimina el último índice
assert elimina([0,1,2,3,4], 4) == [0,1,2,3]

# Elimina un elemento del centro
assert elimina([0,1,2,3,4], 2) == [0,1,3,4]
```

Ejemplo

```
$ pytest test.py

========================= test session starts ==========================
platform linux -- Python 3.8.5, pytest-6.2.1, py-1.10.0, pluggy-0.13.1
rootdir: /home/jpyamamoto/property-testing-python/code
collected 3 items

test.py ...                                                      [100%]

========================== 3 passed in 0.01s ===========================
```

Ejemplo

```
>>> elimina("abcdefghij", 0)
"bcdefghij"
```

Ejemplo

```
>>> elimina([0,1,2,3,4,5], 1.5 )

Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "test.py", line 6, in elimina
    return lista[0:indice] + lista[indice+1:len(lista)]
TypeError: slice indices must be integers or None or have an
    __index__ method
```

mypy [2]

http://www.mypy-lang.org/

Type Specification

mypy

Biblioteca para el análisis estático de tipos.

```python
def fibonacci(n: int) -> int:
  # código

def hola_mundo(nombre: str) -> str:
  return "Hola " + nombre

from typing import List
def suma(lista: List[int]) -> int:
  # código

# >= Python 3.9
def suma(lista: list[int]) -> int:
  # código
```

mypy

Biblioteca para el análisis estático de tipos.

```
from typing import List

def elimina(lista: List[object], indice: int) -> List[object]:
    parte_inicio = lista[0:indice]
    parte_final = lista[indice+1:len(lista)]

    return parte_inicio + parte_final
```

mypy

Biblioteca para el análisis estático de tipos.

```
from typing import List

def elimina(lista: List[object], indice: int) -> List[object]:
    parte_inicio = lista[0:indice]
    parte_final = lista[indice+1:len(lista)]

    return parte_inicio + parte_final
```

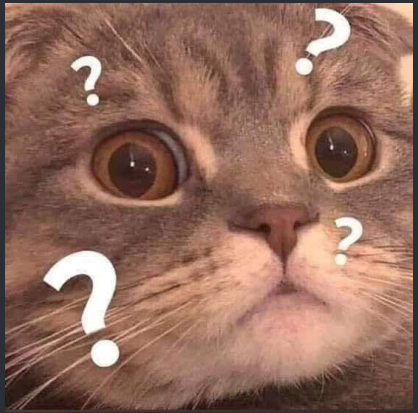
mypy

```
def ejemplo() -> None:
    elimina("abc", 0)
```

```
$ mypy test.py

test.py:9: error: Argument 1 to "elimina" has incompatible type "str";
     expected "List[object]"
Found 1 error in 1 file (checked 1 source file)
```

mypy

```
def ejemplo() -> None:
    elimina("abc", 0)

$ mypy test.py

test.py:9: error: Argument 1 to "elimina" has incompatible type "str";
    expected "List[object]"
Found 1 error in 1 file (checked 1 source file)
```



hypothesis [3]

https://hypothesis.works/

Property-Based Testing

Hypothesis

Biblioteca para *property-based testing* (Pruebas basadas en propiedades), mediante la ejecución de pruebas unitarias automatizadas y aleatorias.

- Pruebas con grandes conjuntos de entradas.
- Entradas arbitrarias.
- Prueba casos extremos.
- Implementación rápida.

Hypothesis

```
from typing import List
from hypothesis import given, strategies as st

@given(st.lists(st.integers()), st.integers())
def test_longitud(lista, indice):
    nueva_lista = elimina(lista, indice)
    assert len(lista) > len(nueva_lista)
```

Hypothesis

```
from typing import List
from hypothesis import given, strategies as st

@given(st.lists(st.integers()), st.integers())
def test_longitud(lista, indice):
    nueva_lista = elimina(lista, indice)
    assert len(lista) > len(nueva_lista)
```

Hypothesis

```
$ pytest test.py

============================= FAILURES ==============================
_____________________________ test_longitud _____________________________

    @given(st.lists(st.integers()), st.integers())
>   def test_longitud(lista, indice):

test.py:9:
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

lista = [], indice = 0

    @given(st.lists(st.integers()), st.integers())
    def test_longitud(lista, indice):
        nueva_lista = elimina(lista, indice)
>       assert len(lista) > len(nueva_lista)
E       assert 0 > 0
E        +  where 0 = len([])
E        +  and 0 = len([])

test.py:11: AssertionError
---------------------------- Hypothesis ----------------------------
Falsifying example: test_longitud(lista=[], indice=0)
```

Hypothesis

```
from typing import List
from hypothesis import given, assume, strategies as st

@given(st.lists(st.integers( min_value=0 )), st.integers())
def test_longitud(lista, indice):
    assume (indice < len(lista))

    nueva_lista = elimina(lista, indice)
    assert len(lista) > len(nueva_lista)
```

Hypothesis

```
$ pytest test.py

======================== test session starts ========================
platform linux -- Python 3.8.5, pytest-6.2.3, py-1.10.0, pluggy-0.13.1
plugins: hypothesis-6.8.9
collected 1 item

test.py .                                                      [100%]
========================= 1 passed in 1.56s =========================
```

¿Hemos terminado?

El código funciona correctamente cuando:

- El índice no es negativo.
- El índice es menor que la longitud de la lista.

Veamos qué sucede cuando ejecutamos nuestra función con un índice que no cumple los requisitos de funcionamiento correcto.

¿Hemos terminado?

```
# Índice negativo.
>>> elimina([0,1,2,3,4,5], -1)
[0, 1, 2, 3, 0, 1, 2, 3, 4]

# Índice fuera del arreglo.
>>> elimina([0,1,2,3,4], 10)
[0, 1, 2, 3, 4]
```

What?

deal [4]

https://deal.readthedocs.io/

Design by Contract

Deal

Biblioteca para la implementación del paradigma *design by contract* (diseño por contrato).

- Pre-condiciones.
- Invariantes.
- Post-condiciones.

Deal

```
from typing import List
import deal

@deal.pre(lambda lista, indice: 0 <= indice < len(lista))
def elimina(lista: List[object], indice: int) -> List[object]:
    parte_inicio = lista[0:indice]
    parte_final = lista[indice+1:len(lista)]

    return parte_inicio + parte_final
```

Deal

```
# Índice negativo.
>>> elimina([0,1,2], -1)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
deal.PreContractError: expected 0 <= indice < len(lista)
    (where lista=[0,1,2], indice=-1)

# Índice fuera del arreglo.
>>> elimina([0,1,2], 10)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
deal.PreContractError: expected 0 <= indice < len(lista)
    (where lista=[0,1,2], indice=10)
```


MI CÓDIGO
YO

Conclusiones

- Sé perezoso: desarrolla **sólo una vez**.
- Implementa pruebas unitarias.
- Utiliza especificaciones de tipos.
- Utiliza pruebas basadas en propiedades.
- Utiliza diseño por contrato.
- Toma awa y sé feliz.

Sobre mí

Juan Pablo Yamamoto Zazueta.

```
> Computer Scientist
> Facultad de Ciencias @ UNAM
> Microsoft Learn Student Ambassador
```

Redes Sociales:

- Instagram (@no.compila)
- LinkedIn (/in/jpyamamoto)
- GitHub (JPYamamoto)
- Website (jpyamamoto.com)

Enlaces

[1] Galaviz, José. *El elogio de la Pereza*. Las Prensas de Ciencias, 2003.

[2] Mypy. Website: `http://www.mypy-lang.org/`.

[3] Hypothesis. Website: `https://hypothesis.works/`.

[4] Deal. Website: `https://deal.readthedocs.io/`.