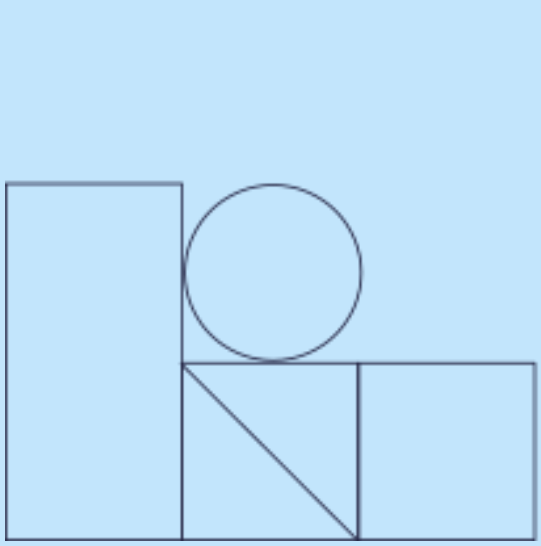
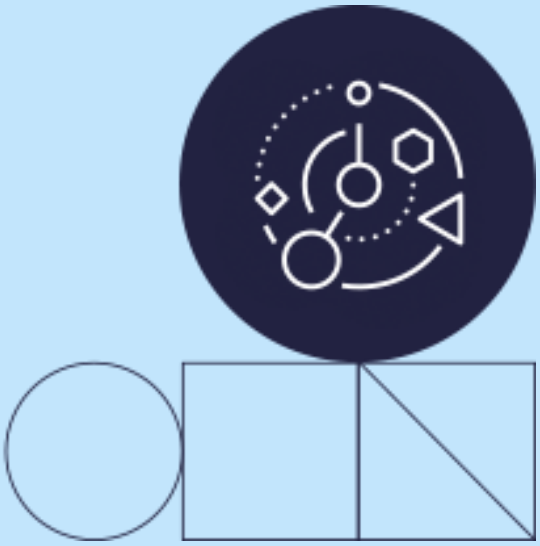
IBM SkillsBuild

IBM

Pruebas con Python

Excepciones asertivas





Índice

Introducción

Las excepciones asertivas (assert) son constructos sintácticas de Python que se configuran como booleanos puros, es decir, que sólo devuelven True o False como resultado de la evaluación de una condición implícita.

Saber cómo escribir afirmaciones en Python le permite escribir fácilmente minipruebas para su código.

Además, los marcos de prueba como PyTest pueden funcionar directamente con afirmaciones para formar UnitTests en pleno funcionamiento.

En primer lugar, revisemos todos los diferentes tipos de afirmaciones que podemos hacer para PyTest.

Lista de declaraciones de afirmación de PyTest Python

```
# Module Imports
from types import *
import pandas as pd
import numpy as np
from collections.abc import Iterable
```

Nota: Siempre que vea # Ejemplo de éxito, esto significa que la prueba de aserción tendrá éxito. Sin embargo, cuando vea # Ejemplo de falla, esto significa que la prueba de aserción fallará.

1. Igual o no igual a [valor]

```
assert 5 == 5 # Success Example

assert 5 == 3 # Fail Example

---------------------------------------
---------------------------------

AssertionError                          Tr
aceback (most recent call last)

<ipython-input-106-db6ee5a4bb16> in
<module>
----> 1 assert 5 == 3 # Fail Example

AssertionError:
assert 5 != 3 # Success Example

assert 5 != 5 # Fail Example

---------------------------------------
---------------------------------
```

```
AssertionError                          Tr
aceback (most recent call last)

<ipython-input-108-de24f583bfdf> in
<module>
----> 1 assert 5 != 5 # Fail Example

AssertionError:
```

2. type() is [valor]

```
assert type(5) is int # Success Example

assert type(5) is not int # Fail Example

---------------------------------------
---------------------------------

AssertionError                          Tr
aceback (most recent call last)

<ipython-input-110-e4cc0467bcd9> in
<module>
----> 1 assert type(5) is not int # Fail
Example

AssertionError:
```

3. isinstance

```
assert type(5) is int # Success Example

assert type(5) is not int # Fail Example

---------------------------------------
---------------------------------

AssertionError                          Tr
aceback (most recent call last)

<ipython-input-110-e4cc0467bcd9> in
<module>
----> 1 assert type(5) is not int # Fail
Example

AssertionError:
```

4. is [Tipo booleano]

```
true = 5==5
assert true is True # Success Example

assert true is False # Fail Example

---------------------------------------
---------------------------------

AssertionError                          Tr
aceback (most recent call last)

<ipython-input-118-45032f0eff77> in
<module>
----> 1 assert true is False # Fail
Example

AssertionError:
```

5. in y not in [iterable]

```
list_one=[1,3,5,6]

assert 5 in list_one # Success Example
assert 5 not in list_one # Fail Example

---------------------------------------
---------------------------------

AssertionError                          Tr
aceback (most recent call last)

<ipython-input-121-90e8a0f2ef02> in
<module>
----> 1 assert 5 not in list_one # Fail
Example

AssertionError:
```

6. Mayor o menor que [valor]

```
assert 5 > 4 # Success Example

assert 5 > 7 # Fail Example

---------------------------------------
---------------------------------

AssertionError                          Tr
aceback (most recent call last)

<ipython-input-123-3068a8105e75> in
<module>
----> 1 assert 5 > 7 # Fail Example

AssertionError:


assert 2 < 4 # Success Example

assert 4 < 2 # Fail Example

---------------------------------------
---------------------------------
```

```
AssertionError                            T
raceback (most recent call last)

<ipython-input-125-089b0fa99ac6> in
<module>
----> 1 assert 4 < 2 # Fail Example

AssertionError:
_____________________________________
```

Debemos tener en cuenta que la lista de ejemplo es **True** porque al menos uno de los números no es un **0**, todos los números por encima de **0** son **True**.

```
assert any(example) == True # Success
Example
assert any(booleans) == True # Success
Example
_____________________________________
```

7. El módulo % es igual a [valor]

```
assert 2 % 2 == 0 # Success Example

assert 2 % 2 == 1 # Fail Example
---------------------------------------
---------------------------------
AssertionError                            T
raceback (most recent call last)

<ipython-input-127-2c429e622b13> in
<module>
----> 1 assert 2 % 2 == 1 # Fail Example

AssertionError:
_____________________________________
```

9. declaracion de afirmación all()

```
>>> all(example)
True

>>> all(booleans)
False
assert all(example) # Success Example

assert all(booleans) # Fail Example

---------------------------------------
---------------------------------
AssertionError                            T
raceback (most recent call last)

<ipython-input-135-c422689f500e> in
<module>
----> 1 assert all(booleans) # Fail
Example

AssertionError:
```

8. declaracion de afirmación any()

```
example = [5,3,1,6,6]
booleans = [False, False,True, False]
>>> any(example)
True

>>> any(booleans)
True
```

10. Objetos personalizados

Es posible identificar si una clase es un tipo específico de objeto. Podemos hacer esto usando:

```
type(object).__name__
df = pd.DataFrame()

>>> type(df).__name__

'DataFrame'

type(df).__name__ == 'DataFrame' # True
Boolean
type(df).__name__ is 'DataFrame' # True
Boolean
type(df).__name__ == type([]).__name__ #
False Boolean
type(df).__name__ is type([]).__name__ #
False Boolean
assert(type(df).__name__ == 'DataFrame')
# Success Example
assert(type(df).__name__ ==
type([]).__name__) # Fail Example

---------------------------------------
---------------------------------

AssertionError                          T
raceback (most recent call last)

<ipython-input-147-2332f54f50a3> in
<module>
----> 1 assert(type(df).__name__ ==
type([]).__name__) # Fail Example

AssertionError:
_____________________________________
```

11. Iterables

También es posible determinar si una variable es iterable con:

```
from collections.abc import Iterable
iterable_item = [3,6,4,2,1]

>>> isinstance(iterable_item, Iterable)
True

>>> isinstance(5, Iterable)
False
assert isinstance(iterable_item,
Iterable) # Success Example

assert isinstance(3, Iterable) # Fail
Example

---------------------------------------
---------------------------------

AssertionError                          T
raceback (most recent call last)

<ipython-input-153-e96805891245> in
<module>
----> 1 assert isinstance(3, Iterable) #
Fail Example

AssertionError:
_____________________________________
```

Combinación de varias declaraciones and/or con declaraciones de afirmación

También es posible combinar múltiples condiciones con **OR** o **AND** y probar los comandos encadenados con la declaración de afirmación:

```
true_statement = 5 == 5 and 10 == 10
false_statement = 5 == 3 and 10 == 2

>>> print(true_statement,
false_statement)
True False
assert true_statement # Success Example

assert false_statement # Fail Example

---------------------------------------
---------------------------------

AssertionError                            T
raceback (most recent call last)

<ipython-input-157-452ef20f327f> in
<module>
----> 1 assert false_statement # Fail
Example

AssertionError:
_____________________________________
true_or_statement = 5 == 5 or 3 == 3
false_or_statement = 7 == 3 or 10 == 1

>>> print(true_or_statement,
false_or_statement)
True False
assert true_or_statement # Success
Example

assert false_or_statement # Fail Example

---------------------------------------
---------------------------------

AssertionError                            T
raceback (most recent call last)

<ipython-input-161-38343a099bdc> in
<module>
----> 1 assert false_or_statement # Fail
Example

AssertionError:
_____________________________________
```

Prueba de varios comandos

También podemos probar más de una cosa a la vez al tener múltiples afirmaciones dentro del mismo método de Python:

```
class Test(object):
    def __init__(self, first_name,
last_name ):
        self.first_name = first_name
        self.last_name = last_name

    def test_all_class_arguments(self):
        print('Testing both of the class
variables to see whether they are both
strings!')

        for _ in [self.first_name,
self.last_name]:
            assert(type(_) is str)
        print('------')
        print('Passed all of the tests')
yay = Test('James' , 'Phoenix') # Success
Example
yay.test_all_class_arguments()

Testing both of the class variables to
see whether they are both strings!
------
```

```
Passed all of the tests
yay = Test(5 , 'Phoenix') # Fail Example
yay.test_all_class_arguments()

Testing both of the class variables to
see whether they are both strings!

--------------------------------------
---------------------------------

AssertionError
 Traceback (most recent call last)

<ipython-input-164-64cb2bee07e3> in
<module>
      1 yay = Test(5 , 'Phoenix') # Fail
Example
----> 2 yay.test_all_class_arguments()

<ipython-input-162-3ae9548ef4b7> in
test_all_class_arguments(self)
      8
      9         for _ in
[self.first_name, self.last_name]:
---> 10             assert(type(_) is
str)
     11         print('------')
     12         print('Passed all of the
tests')

AssertionError:
```

Métodos de afirmación de Python 3.x UnitTest

A continuación, detallamos la lista de todos los métodos de afirmación de UnitTest:

Método	Implementación
assertEqual	a == b
assertNotEqual	a != b
assertTrue	bool(x) is True
assertFalse	bool(x) is False
assertIs	a is b
assertIsNot	a is not b
assertIsNone	x is None
assertIsNotNone	x is not None
assertIn	a in b
assertNotIn	a not in b
assertIsInstance	is instance(a,b)
assertNotIsInstance	not is instance(a,b)
assertRaises	fun(*args,**kwds) raises exc
assertRaisesRegexp	fun(*args,**kwds) raises exc(regex)
assertAlmostEqual	round(a-b,7) == 0
assertNotAlmostEqual	round(a-b,7) != 0
assertGreater	a > b
assertGreaterEqual	a >= b
assertLess	a < b
assertLessEqual	a <= b
assertRegexpMatches	r.search(s)
assertNotRegexpMatches	not r.search(s)
assertItemsEqual	sorted(a) == sorted(b)
assertDictContainsSubset	all the key/value pairs in a exist in b
assertMultiLineEqual	strings
assertSequenceEqual	sequences
assertListEqual	lists
assertTupleEqual	tuples
assertSetEqual	sets or frozensets
assertDictEqual	dicts

Escribir declaraciones de afirmación

Además de usar declaraciones de afirmación simples, al importar el módulo de tipos de Python podemos hacer declaraciones de afirmación más abstractas en Tipos específicos:

```
class Example():
    def __init__(self, id_, name):
        self._id = id_
        self.name = name

    def subtract(self):
        answer = 5 + 5
        return answer

    def test_lambda_function(self):
        assert(lambda x: x is LambdaType)

    def test_subtract_function(self):
        assert(self.subtract is
LambdaType)
example_class = Example("123", 'James
Phoenix')

>>> print(example_class._id,
example_class.name)
123 James Phoenix
_____________________________________
example_class.test_lambda_function() #
Success Example
example_class.test_subtract_function() #
Fail Example

---------------------------------------
---------------------------------

AssertionError
 Traceback (most recent call last)

<ipython-input-169-e96c76763824> in
<module>
----> 1
example_class.test_subtract_function() #
Successs

<ipython-input-165-28a62a6c7adf> in
test_subtract_function(self)
     12
     13     def
test_subtract_function(self):
---> 14         assert(self.subtract is
LambdaType)

AssertionError:
```

Hemos probado dos métodos de instancia de clase para ver si alguno de ellos es una función de estilo lambda.