Python – błędy I sztuczki nie tylko dla nowicjuszy

Filip Hoffmann
Python Developer w Daftcode
filip.hoffmann@daftcode.pl

Domyślne wartości argumentów funkcji

In [1]:
```
def foo(bar=2):
    print(bar)

foo()
foo(bar=3)
foo()
```

Domyślne wartości argumentów funkcji

In [1]:
```
def foo(bar=2):
    print(bar)

foo()
foo(bar=3)
foo()
```

```
2
3
2
```

Domyślne wartości argumentów funkcji

```
def foo(bar=[]):
    bar.append('yolo')
    print(bar)

foo()
foo(bar=[])
foo()
```

Domyślne wartości argumentów funkcji

```
def foo(bar=[]):
    bar.append('yolo')
    print(bar)

foo()
foo(bar=[])
foo()
```

```
['yolo']
['yolo']
['yolo', 'yolo']
```

Domyślne wartości argumentów funkcji

Wartości domyślne argumentów funkcji są ewaluowane TYLKO RAZ

Domyślne wartości argumentów funkcji

```
def foo(bar=None):
    if bar is None:
        bar = []
    bar.append('yolo')
    print(bar)

foo()
foo(bar=[])
foo()
```

```
['yolo']
['yolo']
['yolo']
```

Nieintuicyjne dziedziczenie

```
class A:
    x = 1


class B(A):
    pass


class C(A):
    pass


print(A.x)
print(B.x)
print(C.x)
```

Nieintuicyjne dziedziczenie

```
class A:
    x = 1


class B(A):
    pass


class C(A):
    pass


print(A.x)
print(B.x)
print(C.x)
```

```
1
1
1
```

Nieintuicyjne dziedziczenie

```
B.x = 2
print(A.x)
print(B.x)
print(C.x)
```

Nieintuicyjne dziedziczenie

```
B.x = 2
print(A.x)
print(B.x)
print(C.x)
```

```
1
2
1
```

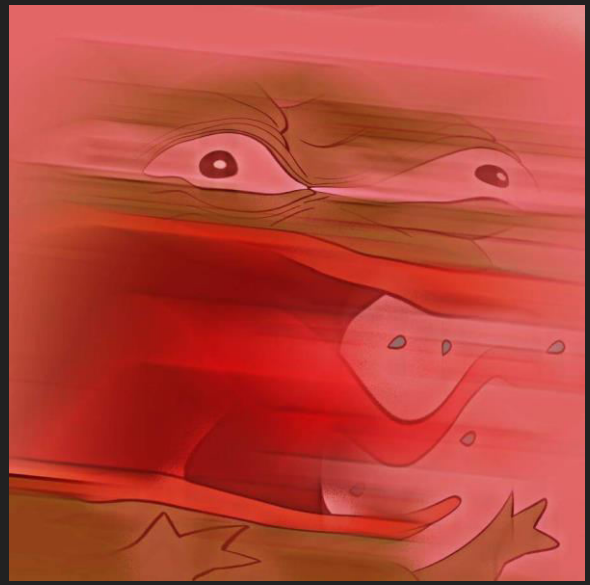
Nieintuicyjne dziedziczenie

```
A.x = 3

print(A.x)
print(B.x)
print(C.x)
```

Nieintuicyjne dziedziczenie

```
A.x = 3

print(A.x)
print(B.x)
print(C.x)
```

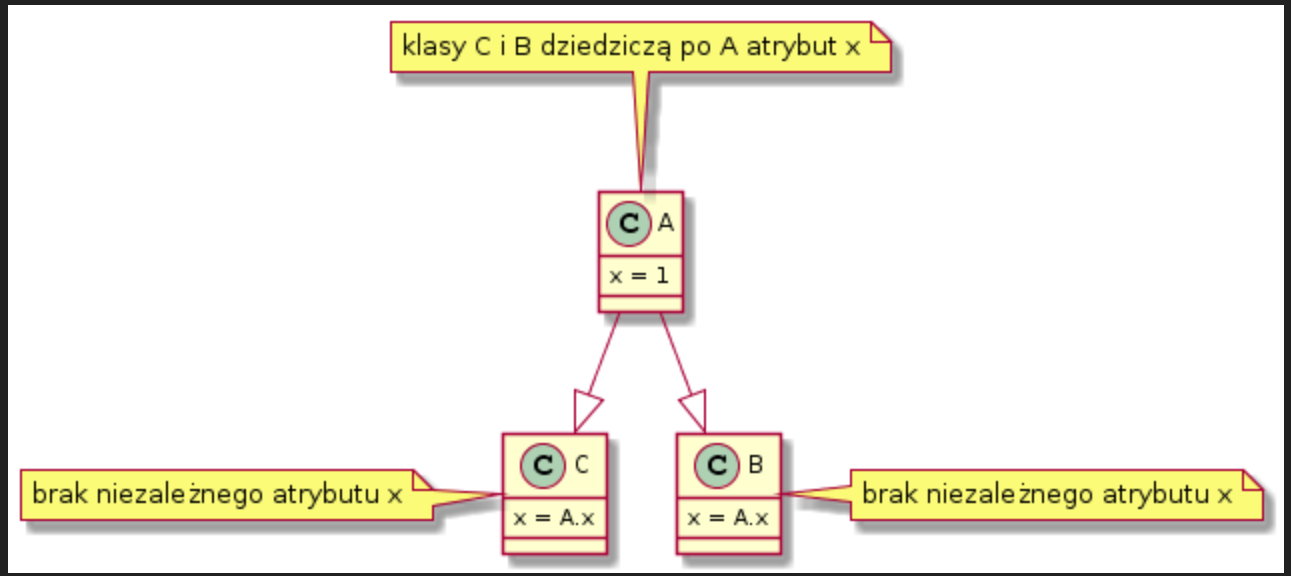
```
3
2
3
```



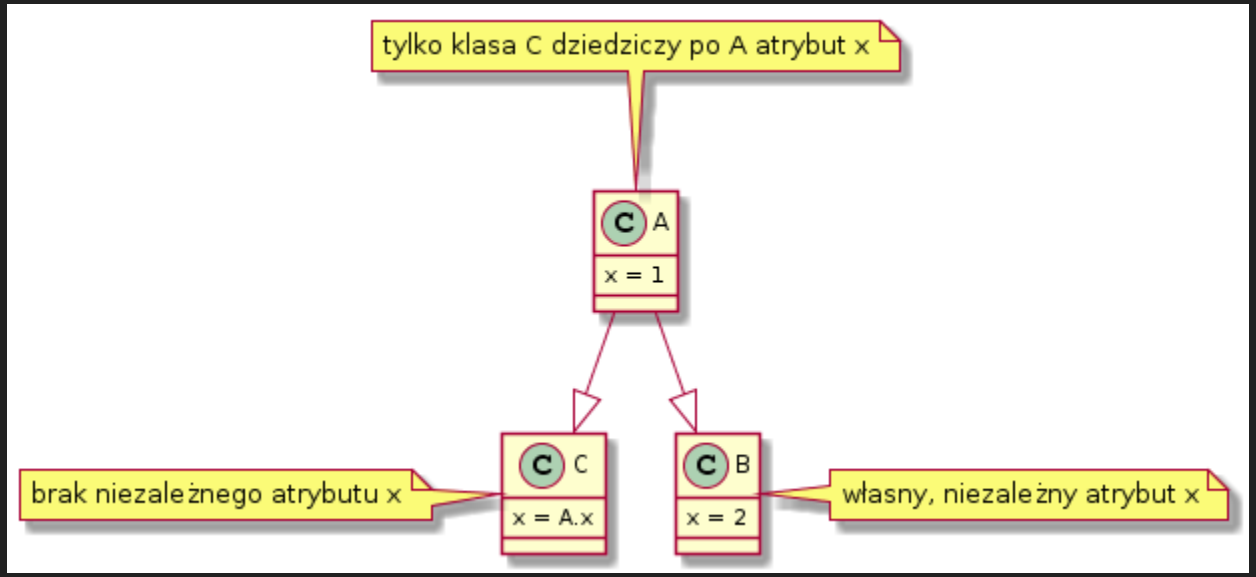
Nieintuicyjne dziedziczenie

Dlaczego C.x też się zmieniło? Zmieniliśmy tylko A.x 0_0

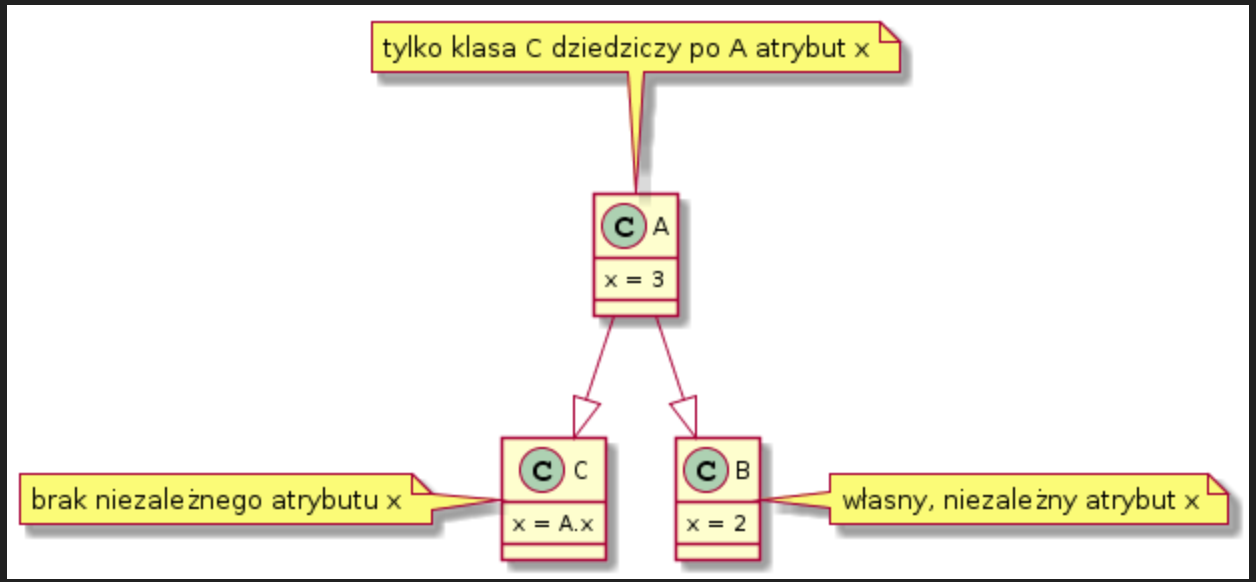
Nieintuicyjne dziedziczenie



Nieintuicyjne dziedziczenie



Nieintuicyjne dziedziczenie



Lokalnie czy globalnie?

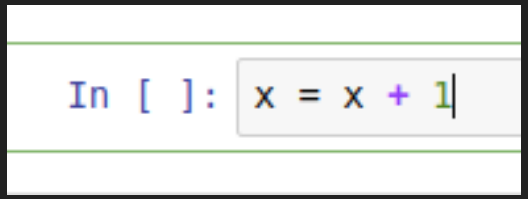
```
x = 10


def foo():
    x += 1
    print(x)

foo()
```

Lokalnie czy globalnie?

```
x = 10


def foo():
    x += 1
    print(x)

foo()
```

```
---------------------------------------------------------------------------
UnboundLocalError                         Traceback (most recent call last)
<ipython-input-7-485e8cd9c901> in <module>()
      6     print(x)
      7
----> 8 foo()

<ipython-input-7-485e8cd9c901> in foo()
      3
      4 def foo():
----> 5     x += 1
      6     print(x)
      7

UnboundLocalError: local variable 'x' referenced before assignment
```

Lokalnie czy globalnie?



Lokalnie czy globalnie?

```
x = 10


def foo():
    x += 1 # x = x + 1
    print(x)

foo()
```

Lokalnie czy globalnie?

```
x = 10


def foo(x):
    x += 1
    return x


x = foo(x)
```

Lokalnie czy globalnie?

```
x = []


def foo():
    x += [1]
    print(x)


foo()
```

```
---------------------------------------------------------------------------
UnboundLocalError                         Traceback (most recent call last)
<ipython-input-10-ff633868f597> in <module>()
      7
      8
----> 9 foo()

<ipython-input-10-ff633868f597> in foo()
      3
      4 def foo():
----> 5     x += [1]
      6     print(x)
      7

UnboundLocalError: local variable 'x' referenced before assignment
```

Lokalnie czy globalnie?

```
x = []


def foo():
    x.append(1)
    print(x)


foo()
```

```
[1]
```

Magia tupli

```
my_tuple = (1, 2, 3)
my_tuple += (4, 5, 6)
print(my_tuple)
```

```
(1, 2, 3, 4, 5, 6)
```

Magia tupli

```
my_tuple = ([1, 2, 3], 1, 2, 3)
my_tuple[0].append(4)
print(my_tuple)
```

```
([1, 2, 3, 4], 1, 2, 3)
```

Magia tupli

```
my_tuple[0] += [5, 6]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-26-6352e08731d5> in <module>()
----> 1 my_tuple[0] += [5, 6]

TypeError: 'tuple' object does not support item assignment
```

Magia tupli

```
my_tuple[0] += [5, 6]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-26-6352e08731d5> in <module>()
----> 1 my_tuple[0] += [5, 6]

TypeError: 'tuple' object does not support item assignment
```

```
print(my_tuple)
```

```
([1, 2, 3, 4, 5, 6], 1, 2, 3)
```

Magia tupli

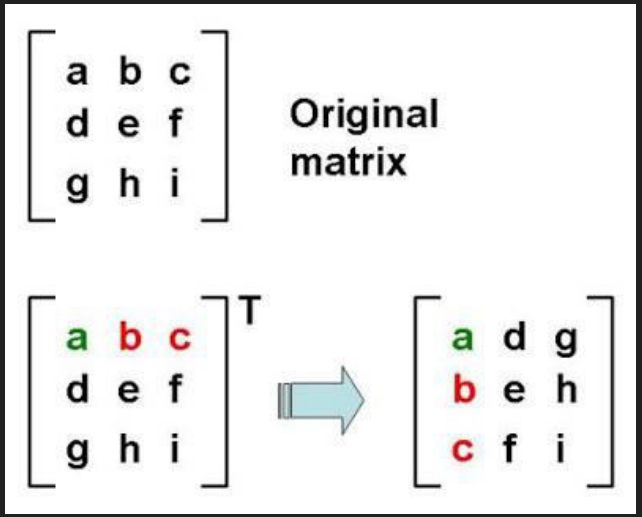
```
my_tuple[0].extend([5, 6])
my_tuple[0] = my_tuple[0]
```

Magia tupli

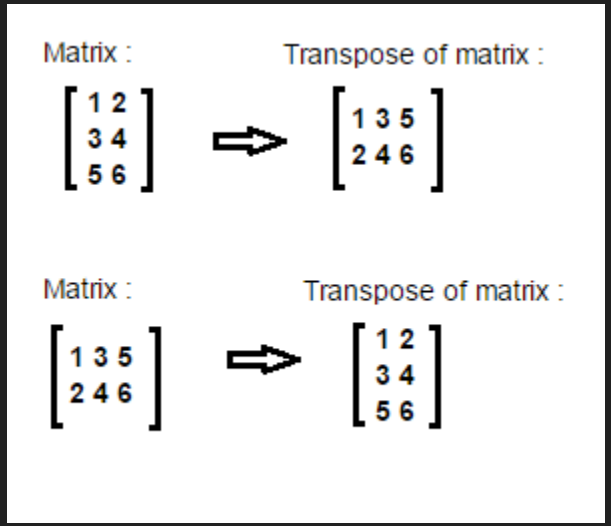
```
my_dict = {}
my_dict[my_tuple] = 'yolo'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-28-13b0ea0e58e3> in <module>()
      1 my_dict = {}
----> 2 my_dict[my_tuple] = 'yolo'

TypeError: unhashable type: 'list'
```

Kreatywny zip



Kreatywny zip



Kreatywny zip

```
matrix = [[1, 2, 3], [1, 2, 3]]
matrix = list(zip(*matrix))
print(matrix)
```

```
[(1, 1), (2, 2), (3, 3)]
```

Kreatywny zip

```
matrix = [1, 2, 3]
matrix2 = [1, 2, 3]
list(zip(matrix, matrix2))
```

```
[(1, 1), (2, 2), (3, 3)]
```

Kreatywny zip

```
matrix = [[1, 2, 3], [1, 2, 3]]
print(*matrix)
```

```
[1, 2, 3] [1, 2, 3]
```

Kreatywny zip

```
[list(row) for row in zip(*matrix)]
```

```
[[1, 1], [2, 2], [3, 3]]
```

Przycinanie sekwencji like a boss

```
seq = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print(seq[7::])
```

```
[8, 9, 10]
```

Przycinanie sekwencji like a boss

```
LASTTHREE = slice(-3, None)
print(seq[LASTTHREE])
```

```
[8, 9, 10]
```

Przycinanie sekwencji like a boss

```
seq[slice(-3, None)] == seq[7::]
```

```
True
```

Spłaszczanie listy

```
a = [[1, 2], [1, 2, 3], [3, 4, 5]]
a = sum(a, [])
print(a)
```

```
[1, 2, 1, 2, 3, 3, 4, 5]
```

Spłaszczanie listy

```
import itertools
```

```
a = [[1, 2], [1, 2, 3], [3, 4, 5]]
list(itertools.chain.from_iterable(a))
```

```
[1, 2, 1, 2, 3, 3, 4, 5]
```

Thank you!

Careers
https://daftcode.pl/careers