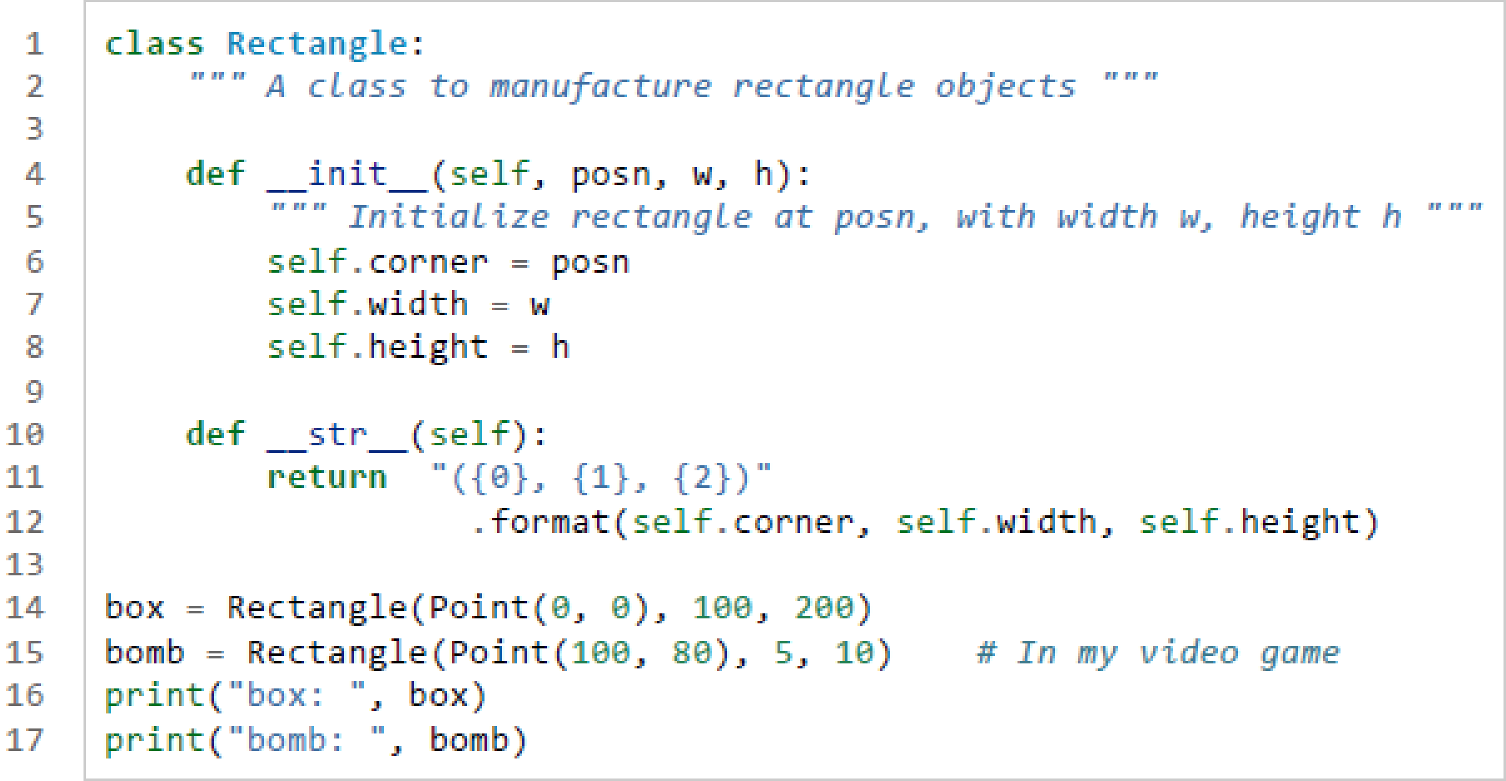
CES 22- 2017 Aula 4

Cópia, Dicionário...

Objetivos

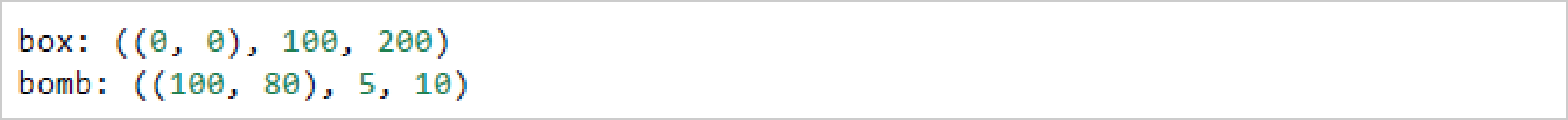
- Comparações e cópias de objetos
- Dicionários

Criando uma classe retângulo

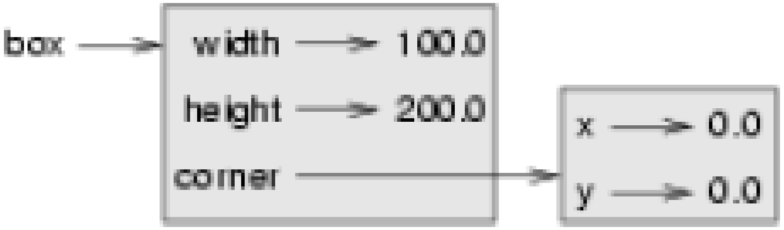


```python
class Rectangle:
    """ A class to manufacture rectangle objects """

    def __init__(self, posn, w, h):
        """ Initialize rectangle at posn, with width w, height h """
        self.corner = posn
        self.width = w
        self.height = h

    def __str__(self):
        return  "({0}, {1}, {2})"
                  .format(self.corner, self.width, self.height)

box = Rectangle(Point(0, 0), 100, 200)
bomb = Rectangle(Point(100, 80), 5, 10)    # In my video game
print("box: ", box)
print("bomb: ", bomb)
```

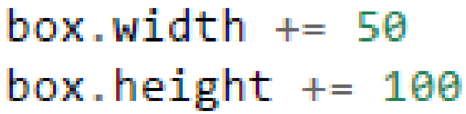
Resultado



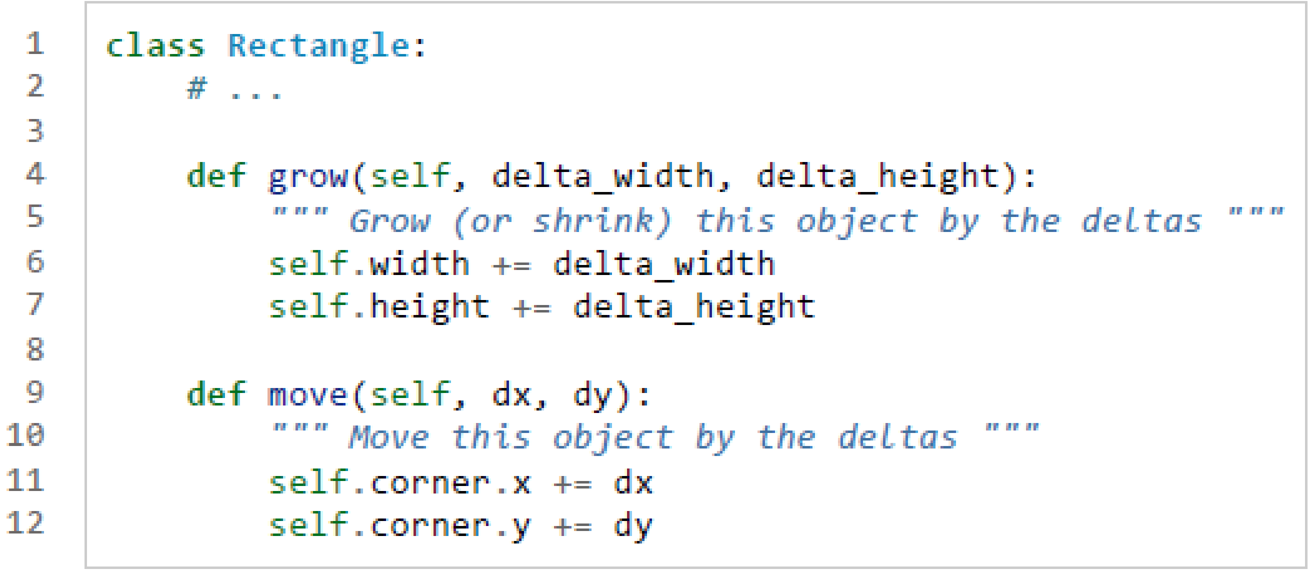
```
box: ((0, 0), 100, 200)
bomb: ((100, 80), 5, 10)
```



Objetos são mutáveis

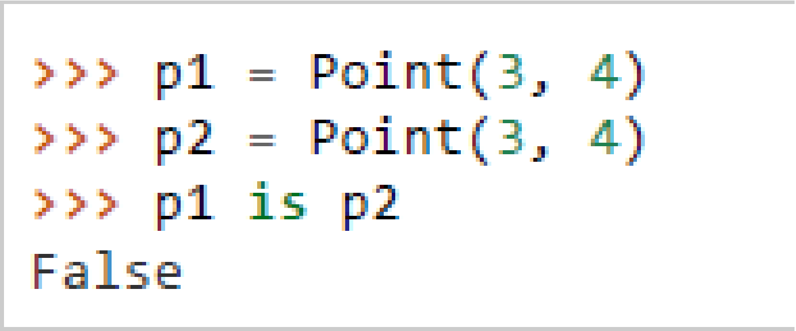


```
box.width += 50
box.height += 100
```

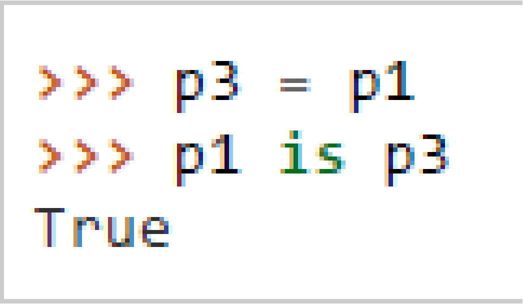


```
class Rectangle:
    # ...

    def grow(self, delta_width, delta_height):
        """ Grow (or shrink) this object by the deltas """
        self.width += delta_width
        self.height += delta_height

    def move(self, dx, dy):
        """ Move this object by the deltas """
        self.corner.x += dx
        self.corner.y += dy
```

Comparações de objetos



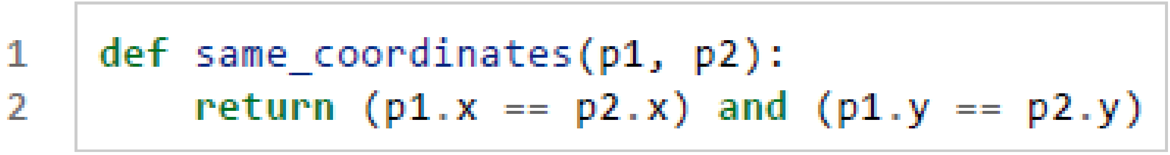
```
>>> p1 = Point(3, 4)
>>> p2 = Point(3, 4)
>>> p1 is p2
False
```



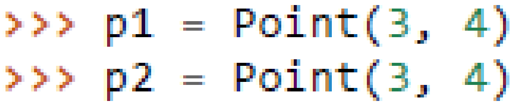
```
>>> p3 = p1
>>> p1 is p3
True
```

Shallow Equality

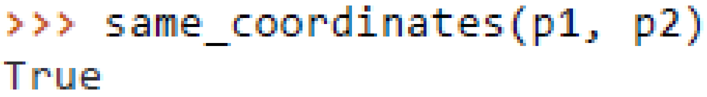
Deep equality



```python
def same_coordinates(p1, p2):
    return (p1.x == p2.x) and (p1.y == p2.y)
```

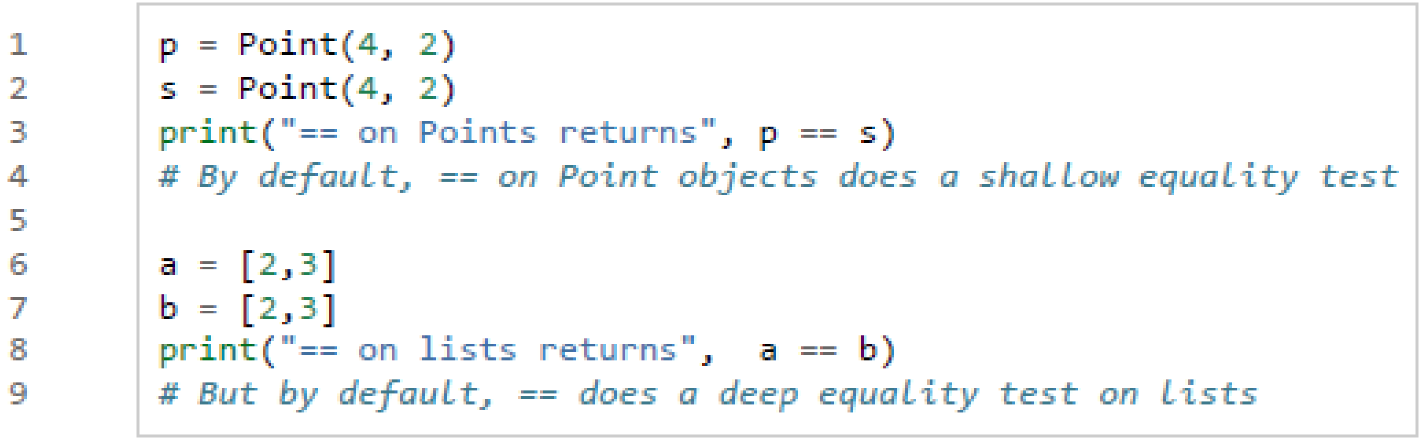


```
>>> p1 = Point(3, 4)
>>> p2 = Point(3, 4)
```

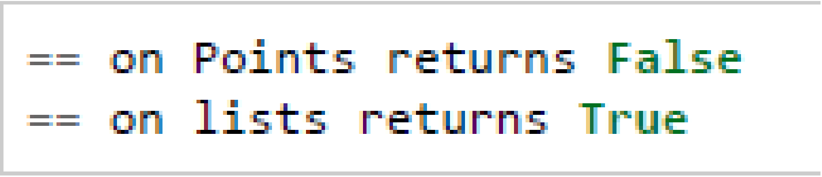


```
>>> same_coordinates(p1, p2)
True
```

Cuidado com ==

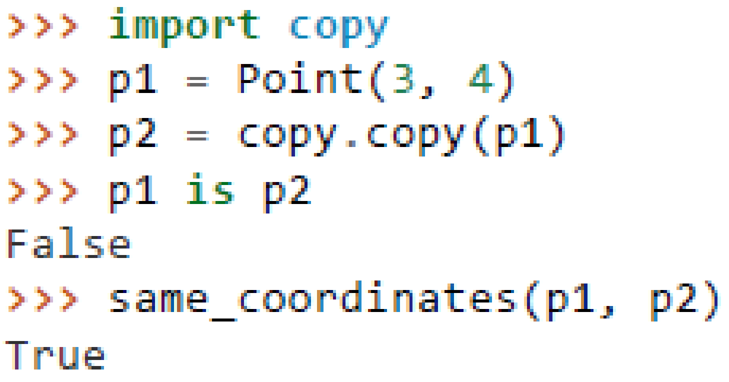


```
p = Point(4, 2)
s = Point(4, 2)
print("== on Points returns", p == s)
# By default, == on Point objects does a shallow equality test

a = [2,3]
b = [2,3]
print("== on lists returns",  a == b)
# But by default, == does a deep equality test on lists
```



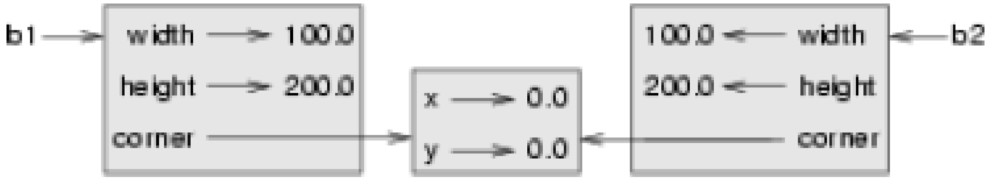
```
== on Points returns False
== on lists returns True
```

Copiar objetos

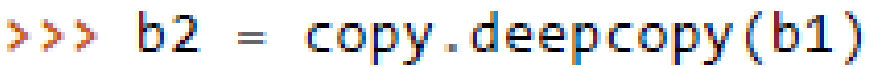


```
>>> import copy
>>> p1 = Point(3, 4)
>>> p2 = copy.copy(p1)
>>> p1 is p2
False
>>> same_coordinates(p1, p2)
True
```

Copiar retângulo usando copy



Usar deepcopy



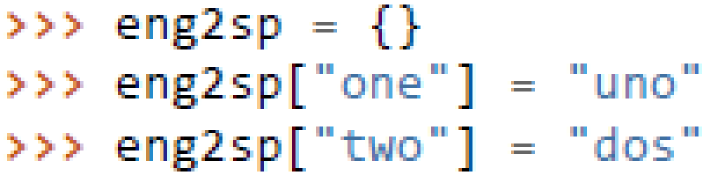
```
>>> b2 = copy.deepcopy(b1)
```

Exercício

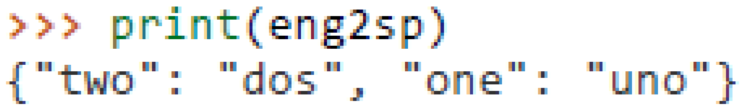
- Exercício 16.6.5 (Colisão de Sprites)

Dicionários

- Dicionário mapeiam chaves de qualquer tipo para valores.



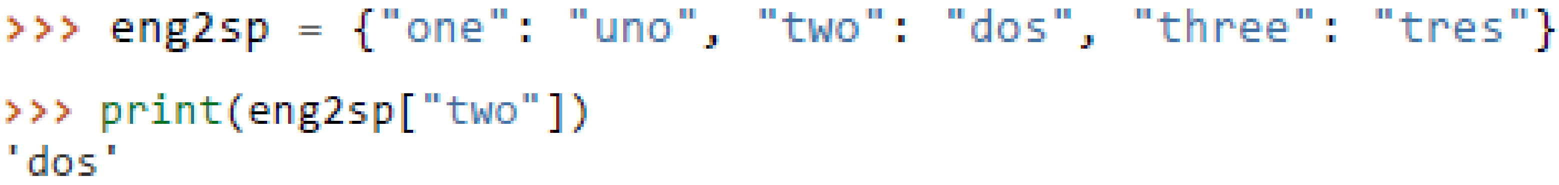
```
>>> eng2sp = {}
>>> eng2sp["one"] = "uno"
>>> eng2sp["two"] = "dos"
```



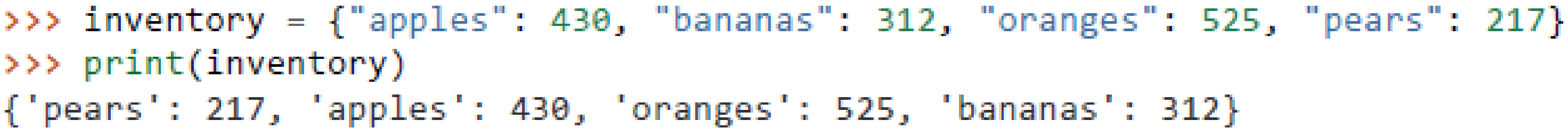
```
>>> print(eng2sp)
{"two": "dos", "one": "uno"}
```

Dicionários são implementados com Hashing.
Dicionários com Hash são muito mais rápidos que buscas em listas ou tuplas.

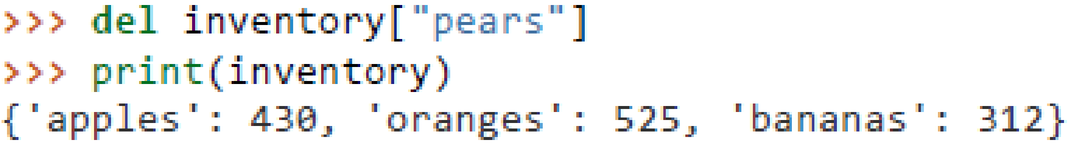
Cont.



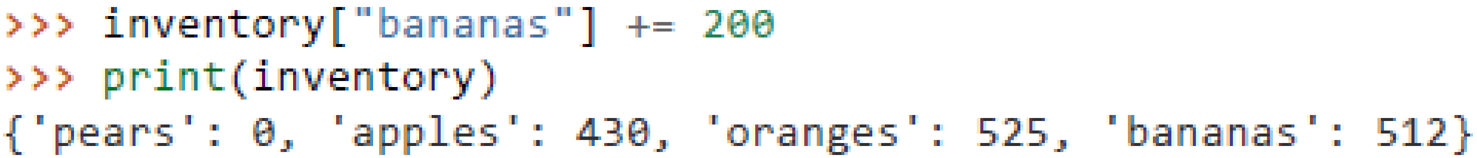
```
>>> eng2sp = {"one": "uno", "two": "dos", "three": "tres"}
>>> print(eng2sp["two"])
'dos'
```



```
>>> inventory = {"apples": 430, "bananas": 312, "oranges": 525, "pears": 217}
>>> print(inventory)
{'pears': 217, 'apples': 430, 'oranges': 525, 'bananas': 312}
```

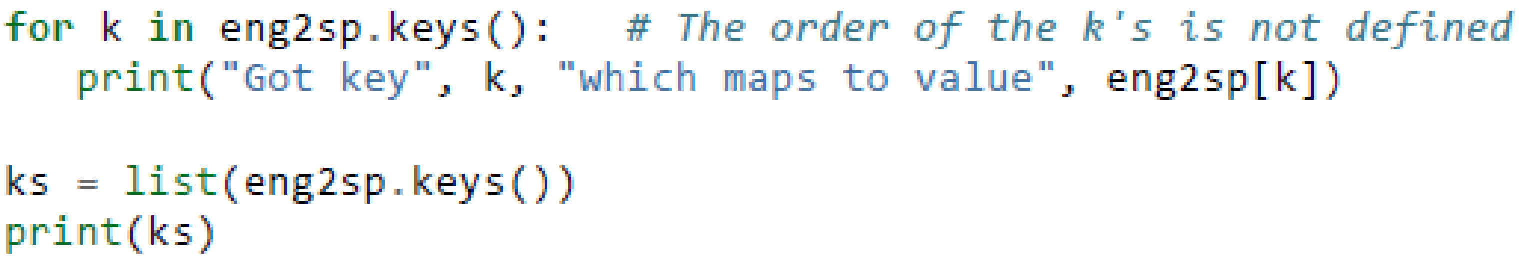


```
>>> del inventory["pears"]
>>> print(inventory)
{'apples': 430, 'oranges': 525, 'bananas': 312}
```

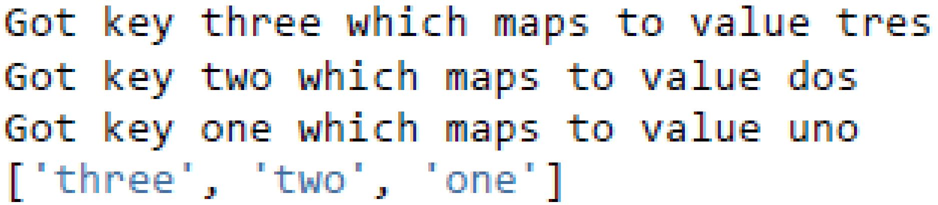


```
>>> inventory["bananas"] += 200
>>> print(inventory)
{'pears': 0, 'apples': 430, 'oranges': 525, 'bananas': 512}
```

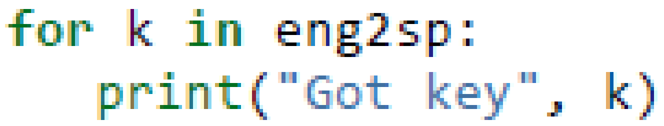
Métodos de dicionário



```
for k in eng2sp.keys():   # The order of the k's is not defined
   print("Got key", k, "which maps to value", eng2sp[k])

ks = list(eng2sp.keys())
print(ks)
```

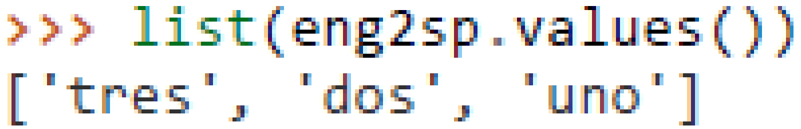


```
Got key three which maps to value tres
Got key two which maps to value dos
Got key one which maps to value uno
['three', 'two', 'one']
```

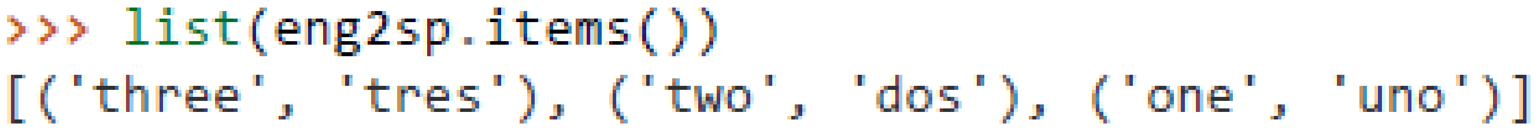


```
for k in eng2sp:
   print("Got key", k)
```

Cont.



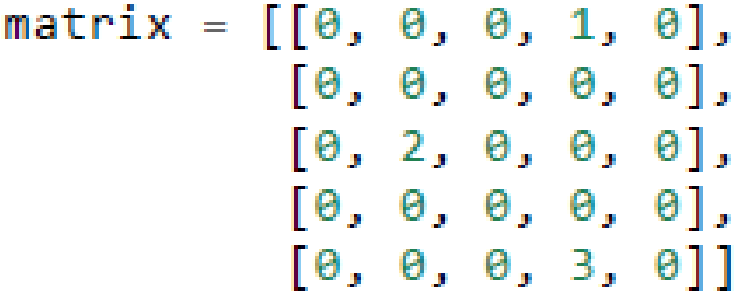
```
>>> list(eng2sp.values())
['tres', 'dos', 'uno']
```



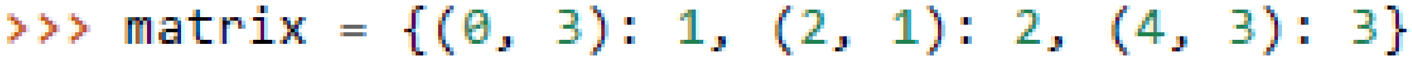
```
>>> list(eng2sp.items())
[('three', 'tres'), ('two', 'dos'), ('one', 'uno')]
```

Matriz esparsa

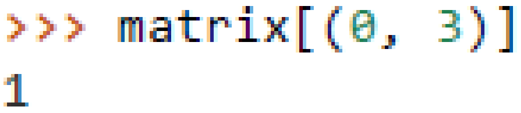
$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \end{bmatrix}$$



```
matrix = [[0, 0, 0, 1, 0],
          [0, 0, 0, 0, 0],
          [0, 2, 0, 0, 0],
          [0, 0, 0, 0, 0],
          [0, 0, 0, 3, 0]]
```

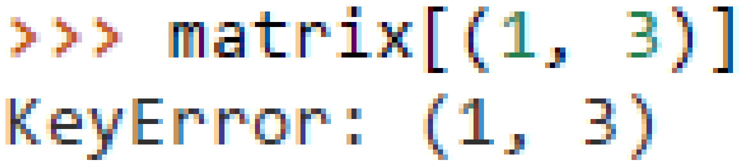


```
>>> matrix = {(0, 3): 1, (2, 1): 2, (4, 3): 3}
```

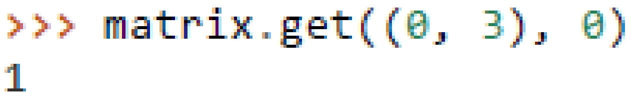


```
>>> matrix[(0, 3)]
1
```

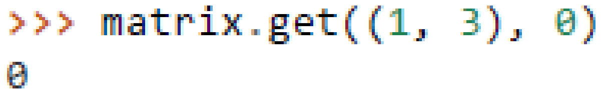
Método get



```
>>> matrix[(1, 3)]
KeyError: (1, 3)
```



```
>>> matrix.get((0, 3), 0)
1
```



```
>>> matrix.get((1, 3), 0)
0
```

Exercício

- Exercício 20.8.3