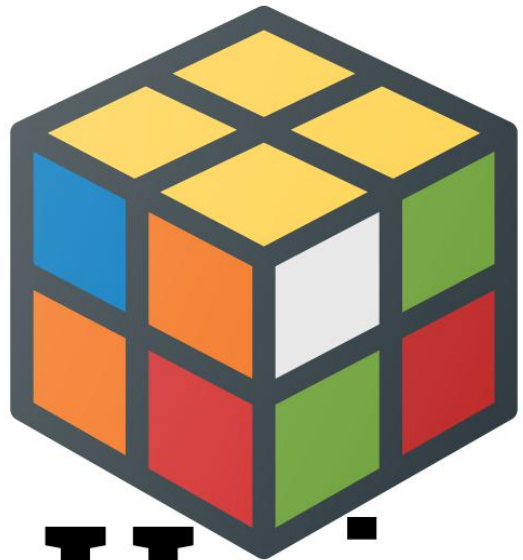


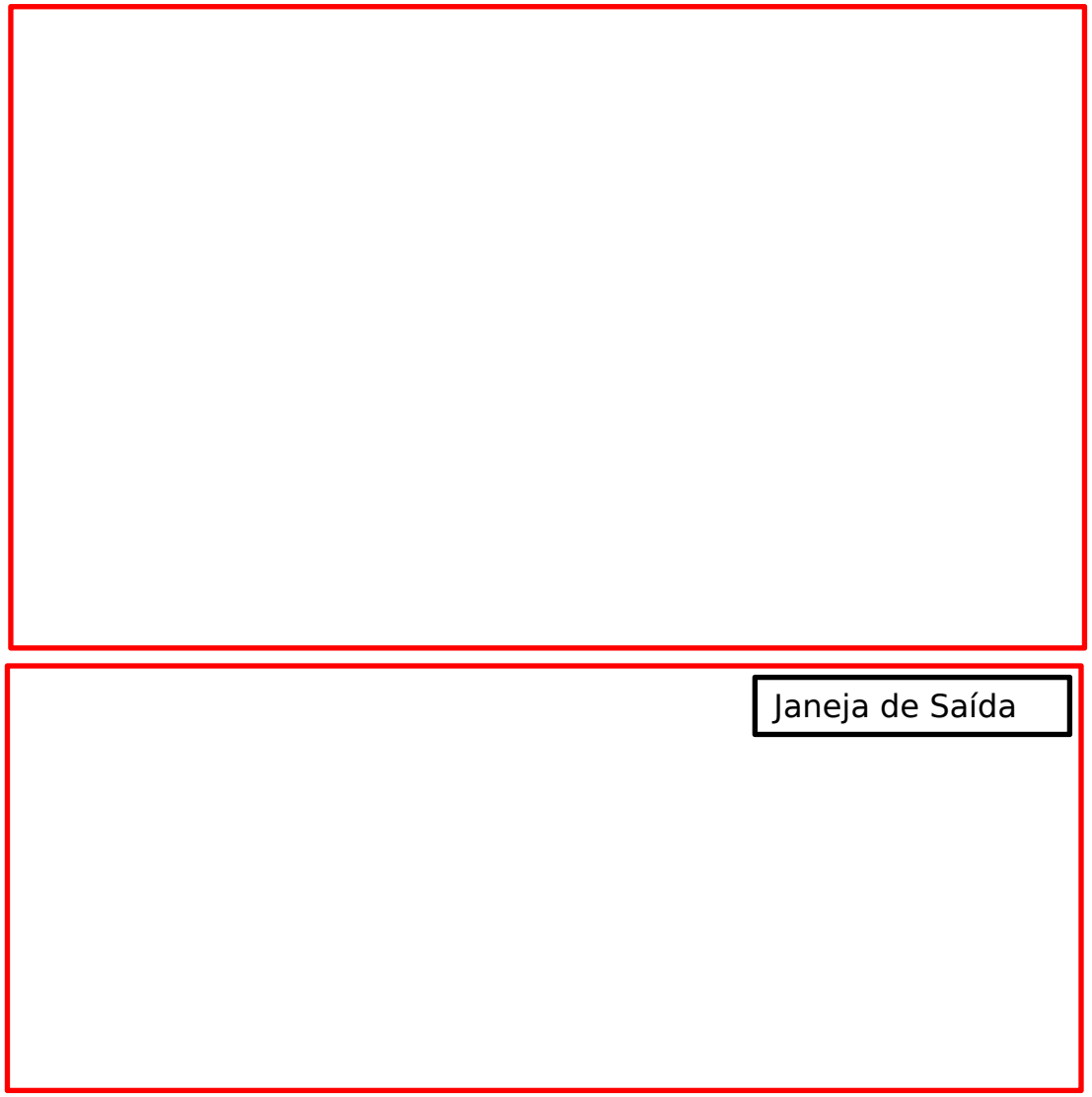
Geek University

Evolua seu lado geek!

www.geekuniversity.com.br

Alocação e Gerência Memória em Python

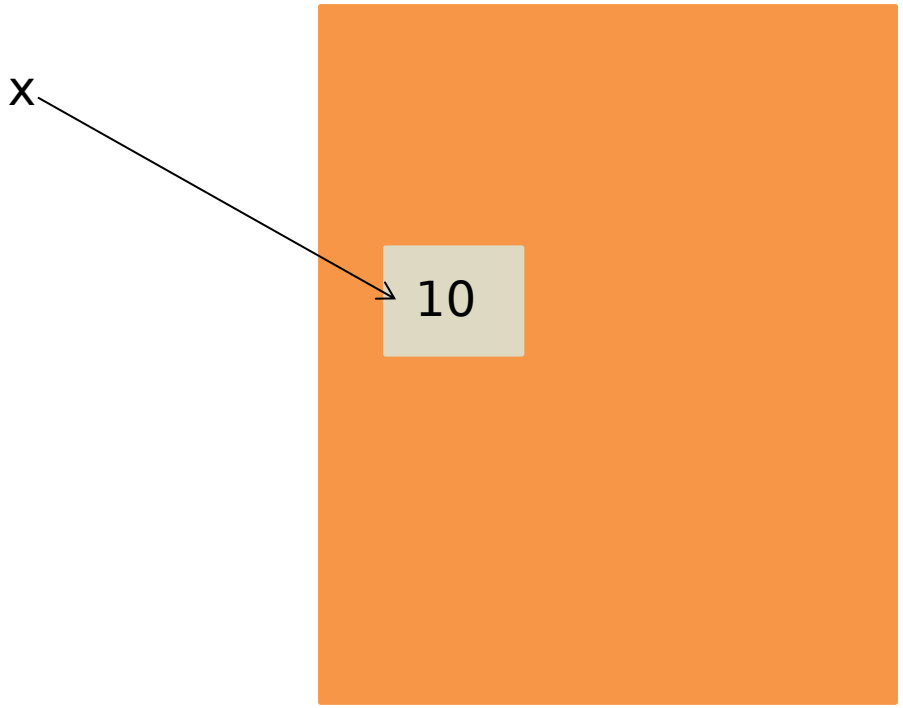
Alocação e Gerência Memória em Python



Alocação e Gerência Memória em Python

x = 10

Janeja de Saída

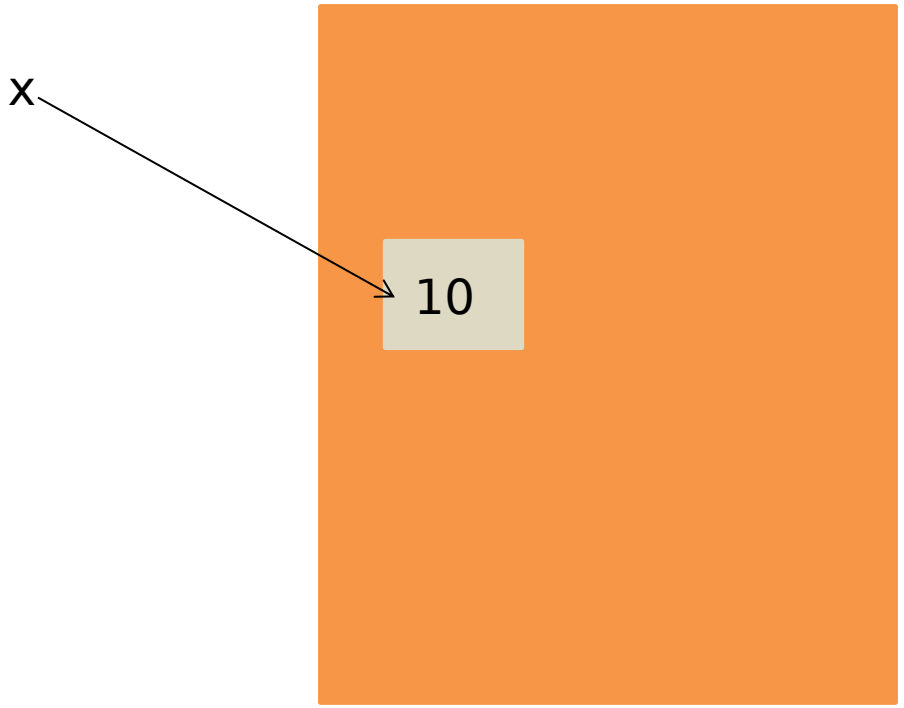


Alocação e Gerência Memória em Python

```
x = 10
print(type(x))
```

```
<class 'int'>
```

Janeja de Saída

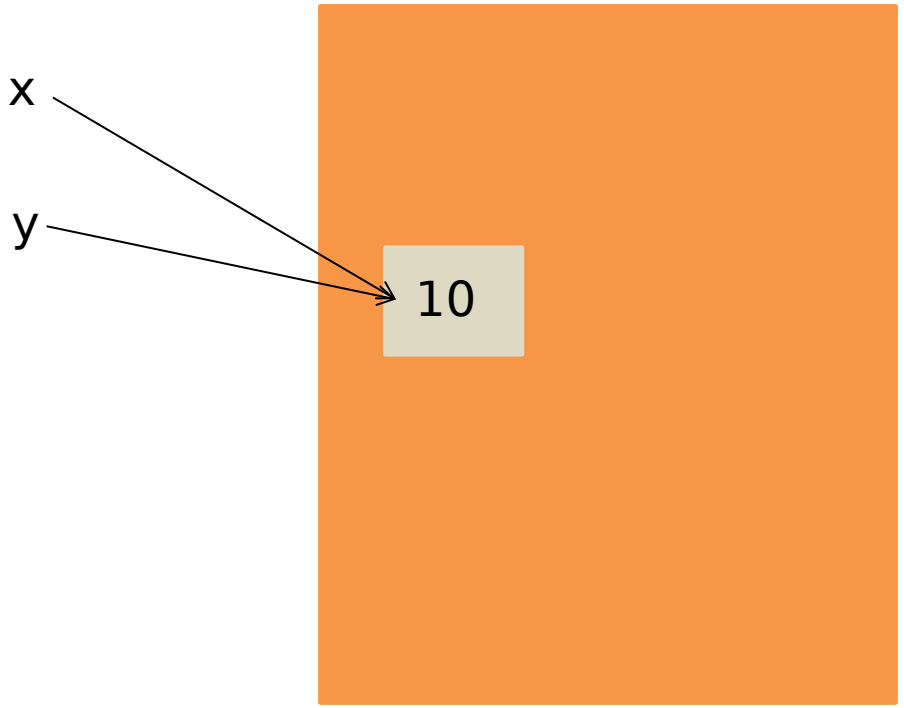


Alocação e Gerência Memória em Python

```
x = 10
print(type(x))

y = x
```

Janeja de Saída

```
<class 'int'>
```

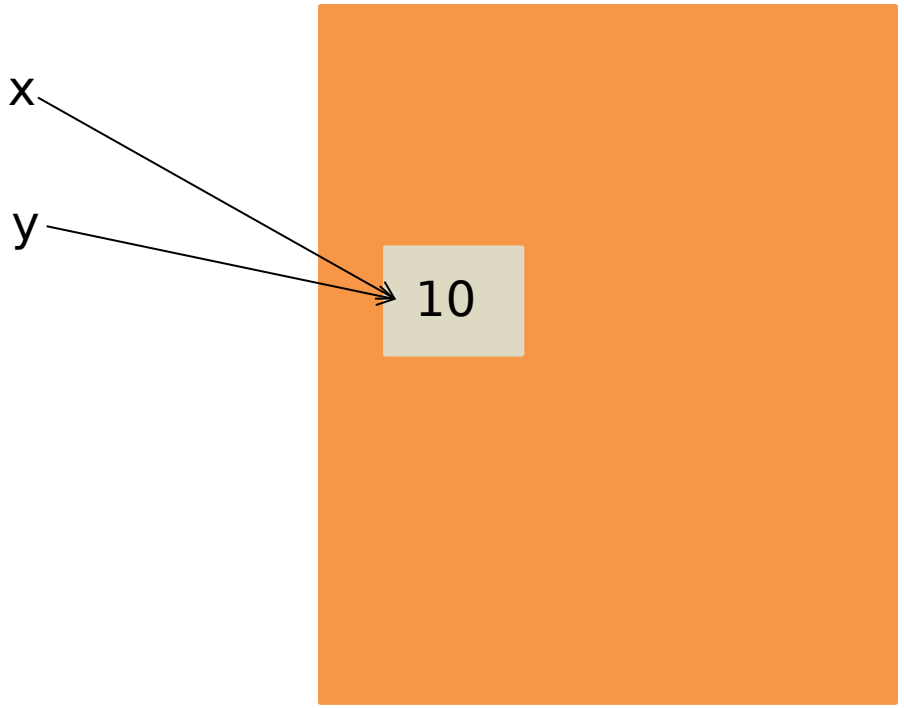


Alocação e Gerência Memória em Python

```
x = 10
print(type(x))

y = x

if(id(x) == id(y)):
    print('x e y referenciam ao mesmo objeto')
```

Janeja de Saída

```
<class 'int'>
```

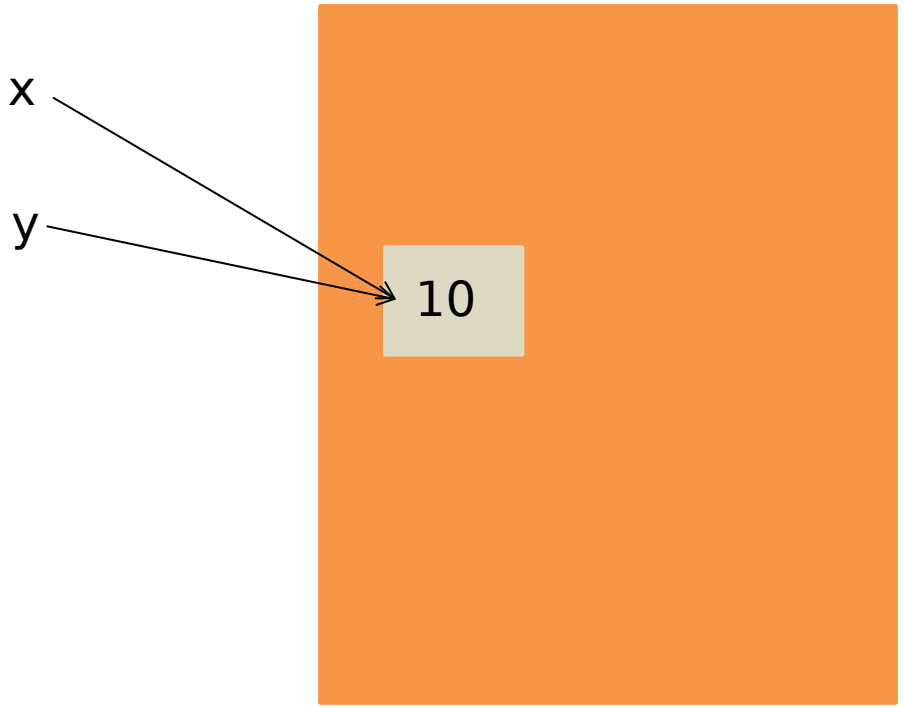


Alocação e Gerência Memória em Python

```
x = 10
print(type(x))

y = x

if(id(x) == id(y)):
    print('x e y referenciam ao mesmo objeto')
```

Janeja de Saída

```
<class 'int'>

x e y referenciam ao mesmo objeto
```

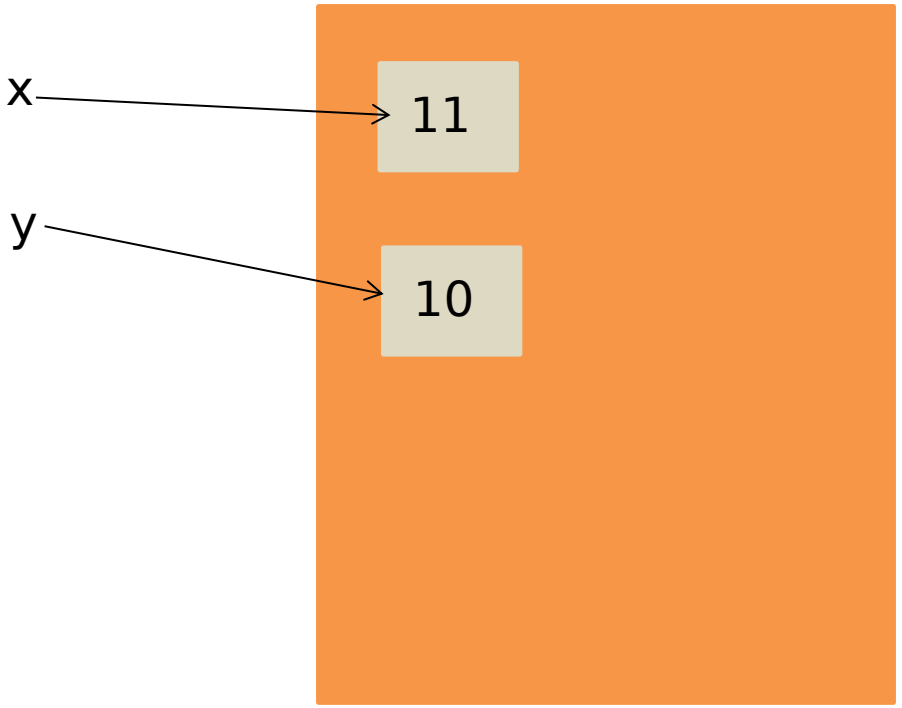


Alocação e Gerência Memória em Python

```
x = 10
print(type(x))

y = x

x = x + 1
if(id(x) == id(y)):
    print('x e y referenciam objetos diferentes')
```

Janeja de Saída

```
<class 'int'>

x e y referenciam objetos diferentes
```

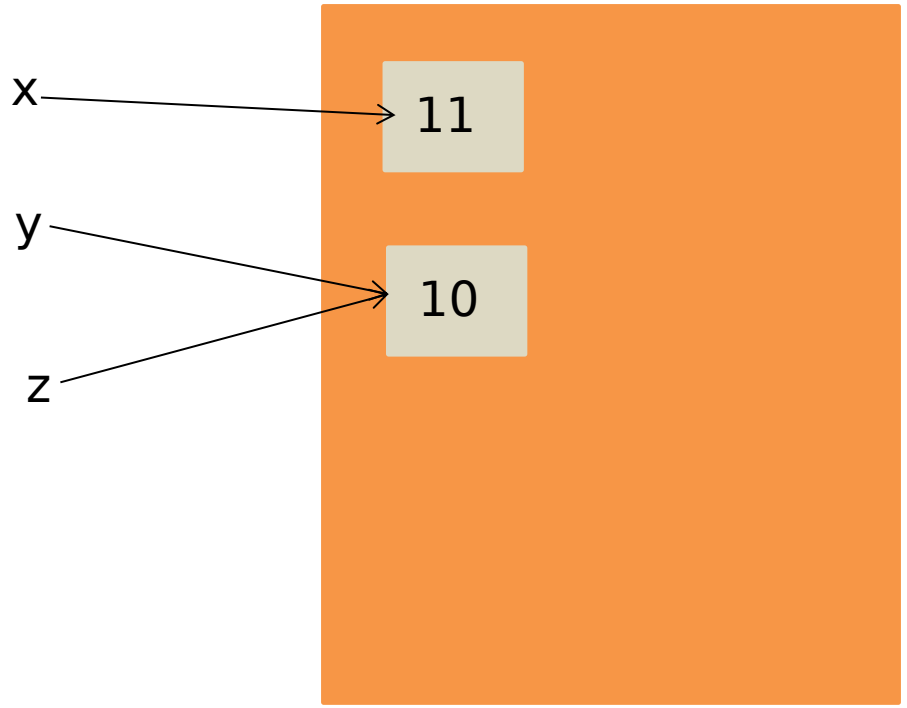


Alocação e Gerência Memória em Python

```
x = 10
print(type(x))

y = x

x = x + 1

z = 10
if(id(y) == id(z)):
    print('y e z apontam para a mesma memória')
else:
    print('y e z apontam para objetos diferentes')
```

Janeja de Saída

```
<class 'int'>

y e z apontam para a mesma memória
```

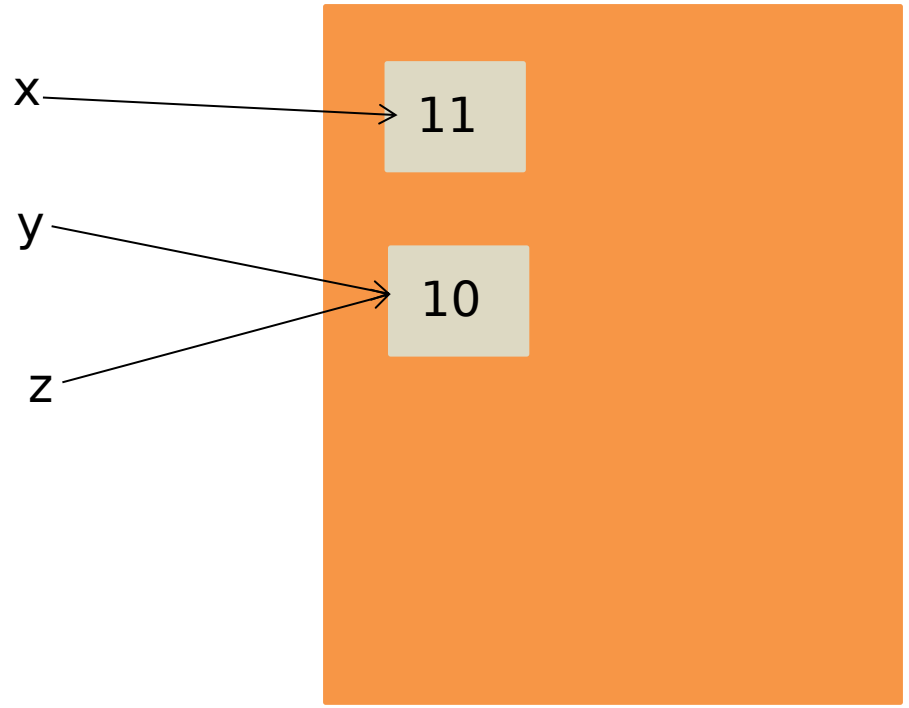


Alocação e Gerência Memória em Python

```
x = 10
print(type(x))

y = x

x = x + 1

z = 10
if(id(y) == id(z)):
    print('y e z apontam para a mesma memória')
else:
    print('y e z apontam para objetos diferentes')
```

Janeja de Saída

```
<class 'int'>

y e z apontam para a mesma memória
```



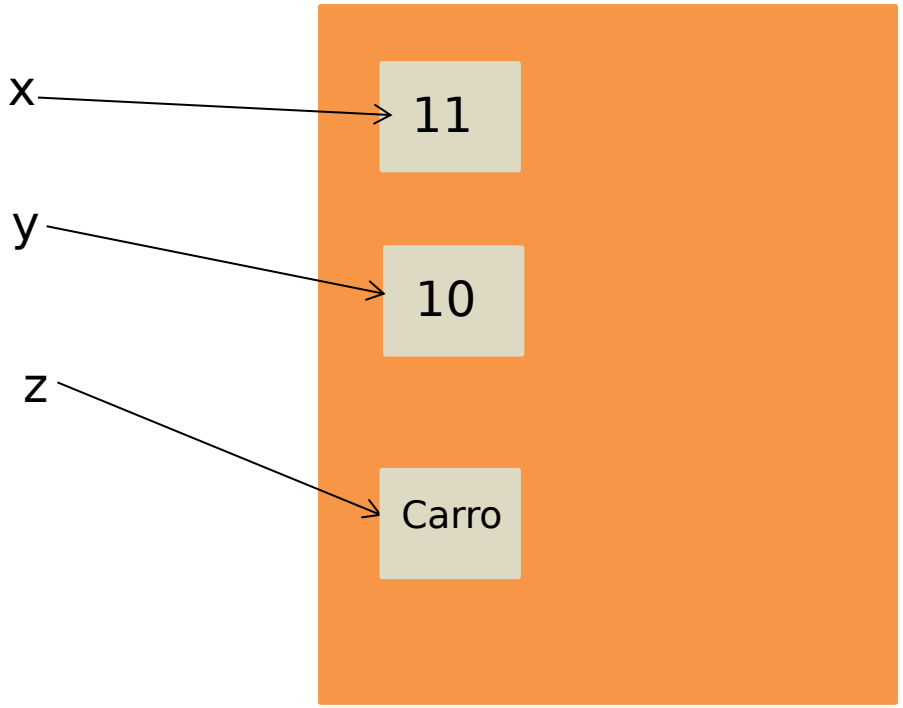
Ou seja, Python inteligentemente reutiliza valores já alocados para novos valores, economizando espaço em memória!

Alocação e Gerência Memória em Python

```
x = 10
print(type(x))

y = x

x = x + 1

z = 10

class Carro:
    pass

z = Carro()
```

Janeja de Saída

```
<class 'int'>
```

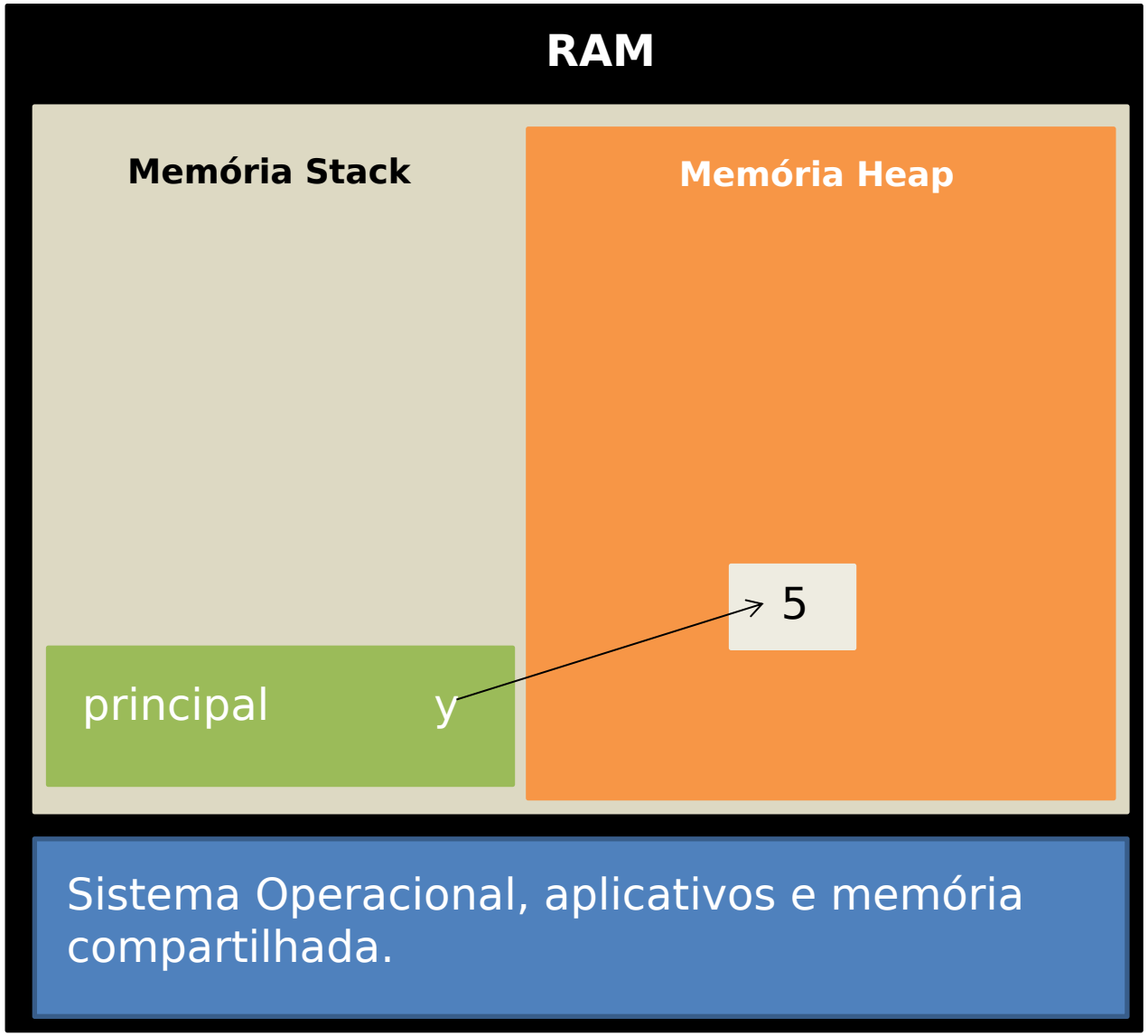


Alocação e Gerência Memória em Python

Como isso funciona por baixo do capô?

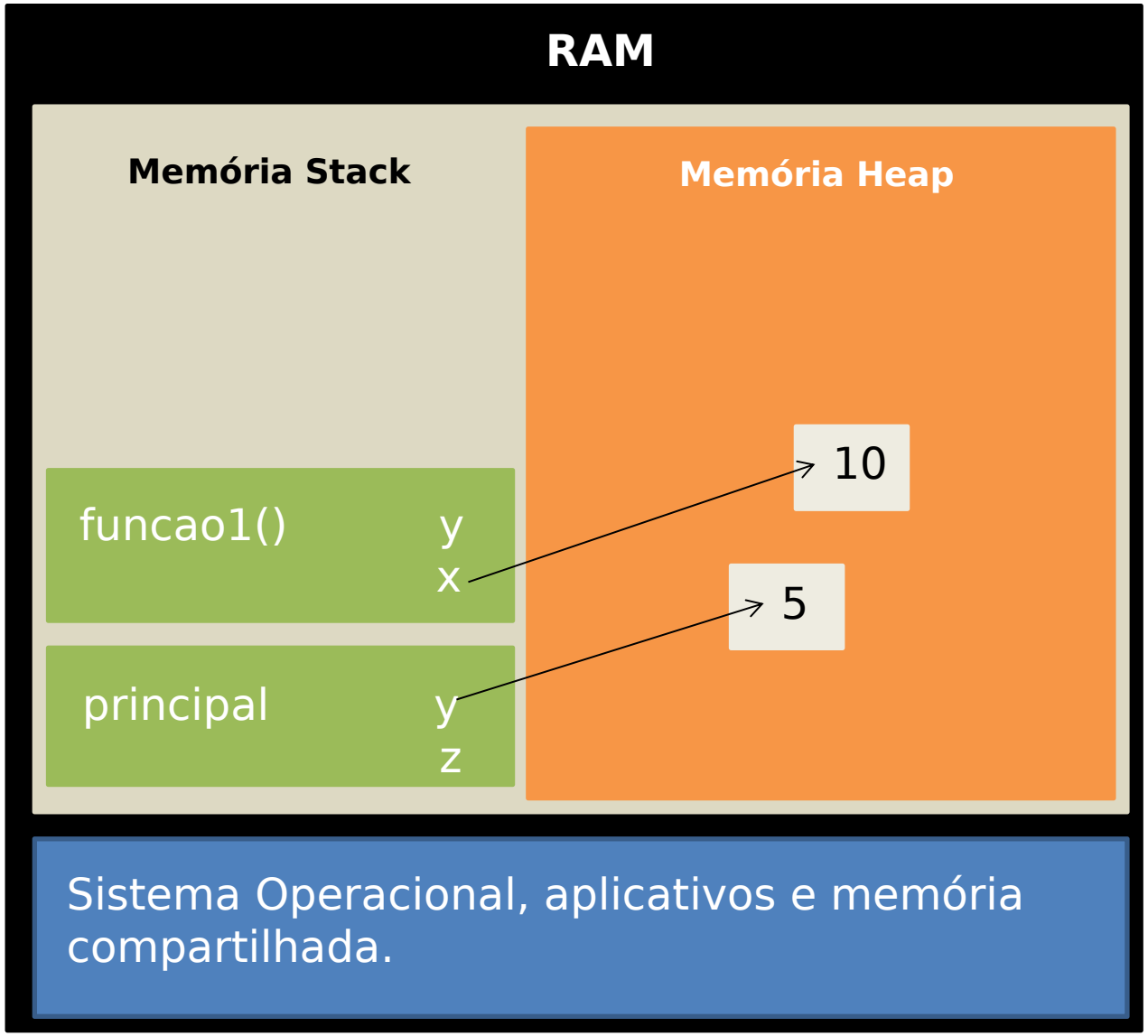
Alocação e Gerência Memória em Python

#principal
y = 5



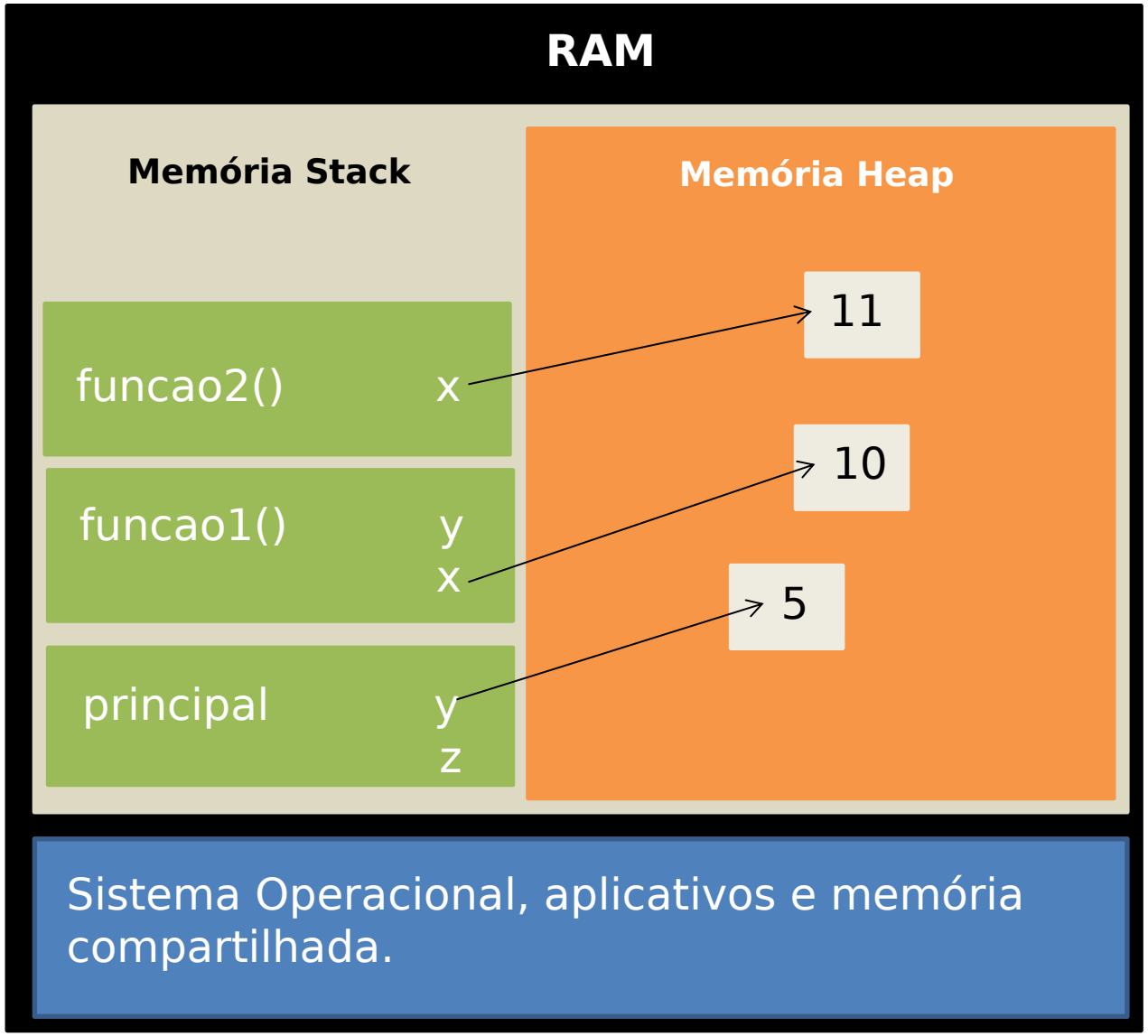
Alocação e Gerência Memória em Python

```
def funcao1(x):
    x = x * 2
    y = funcao2(x)
    return y

#principal
y = 5
z = funcao1(y)
```



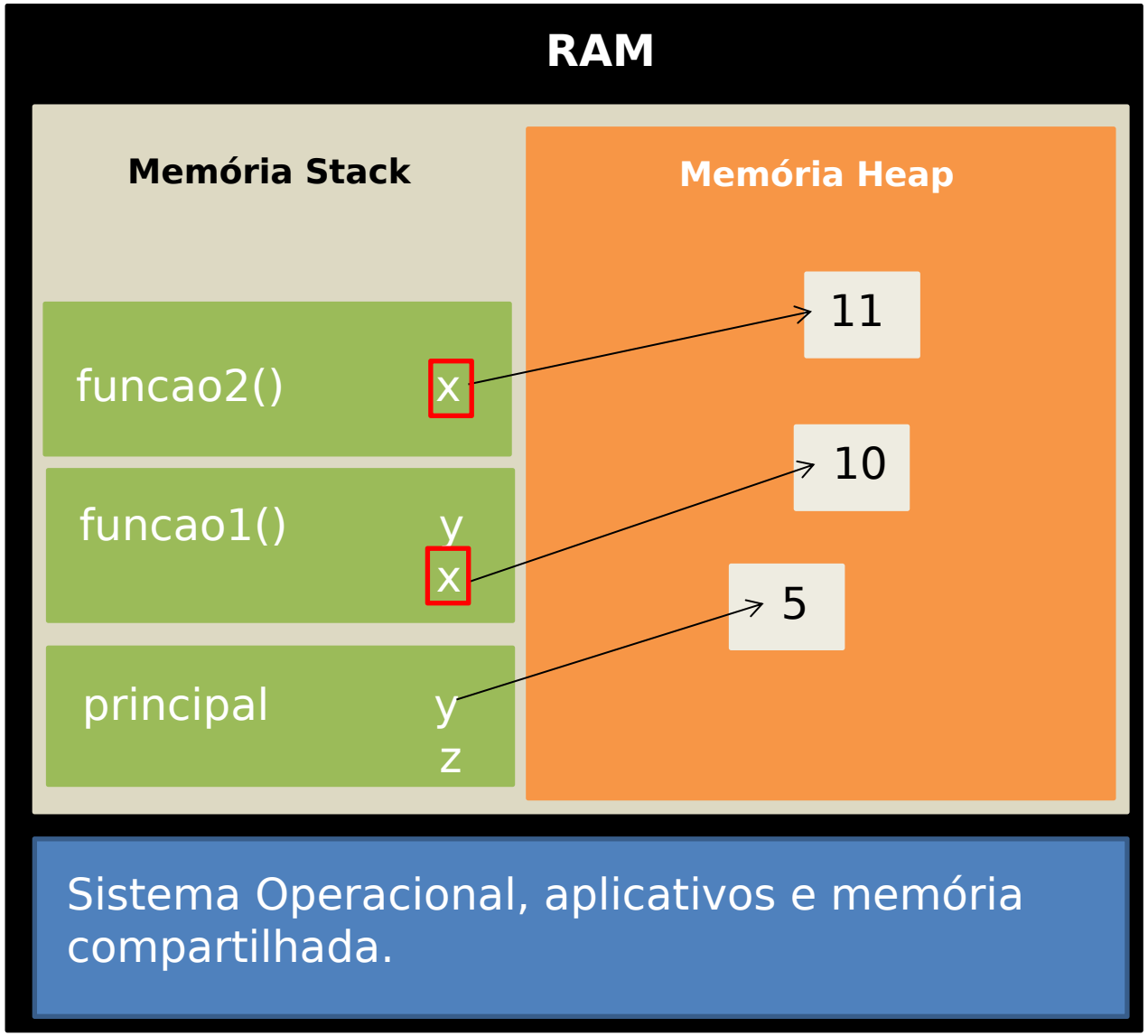
Alocação e Gerência Memória em Python

```
def funcao2(x):
    x = x + 1
    return x

def funcao1(x):
    x = x * 2
    y = funcao2(x)
    return y

#principal
y = 5
z = funcao1(y)
```



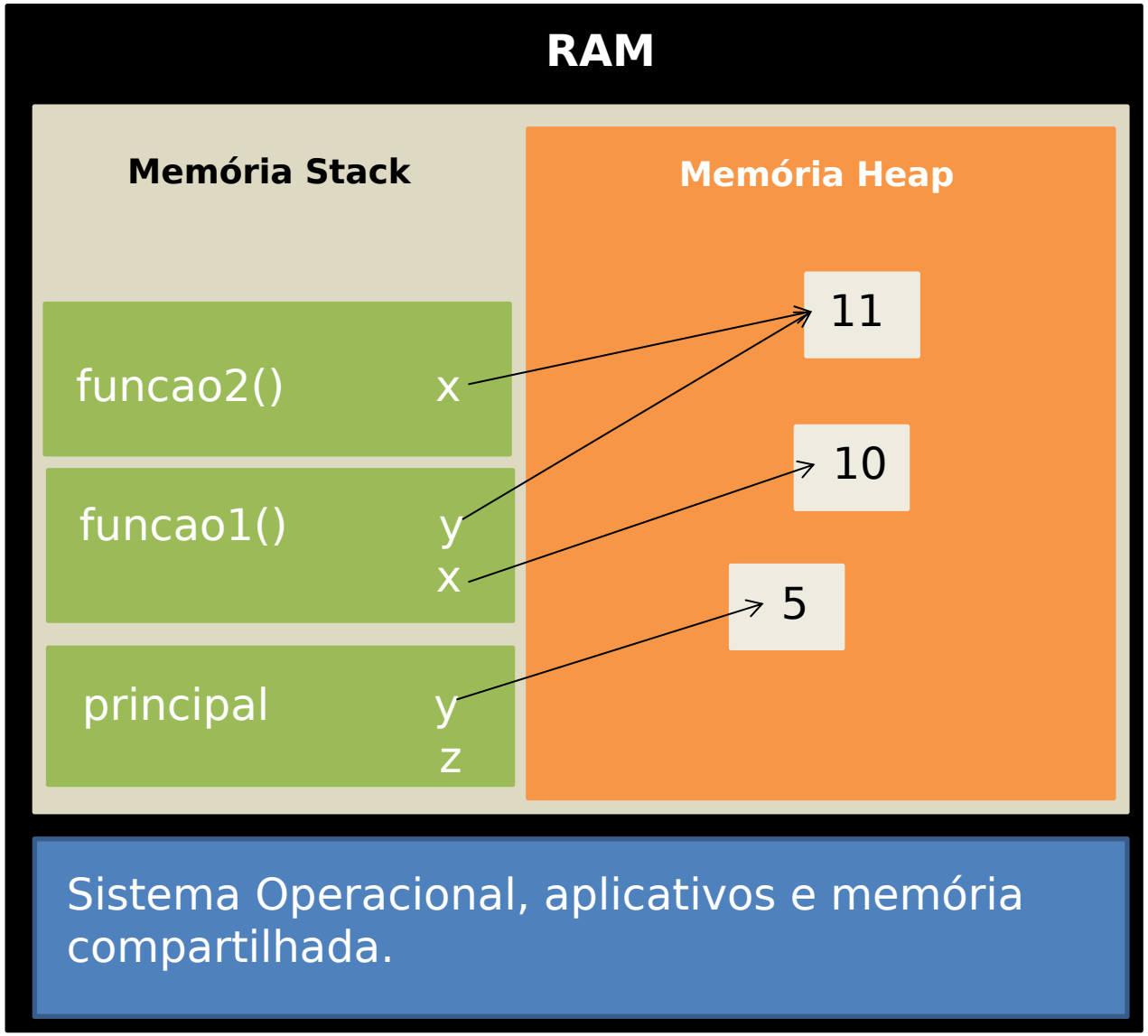
Alocação e Gerência Memória em Python

```
def funcao2(x):
    x = x + 1
    return x

def funcao1(x):
    x = x * 2
    y = funcao2(x)
    return y

#principal
y = 5
z = funcao1(y)
```



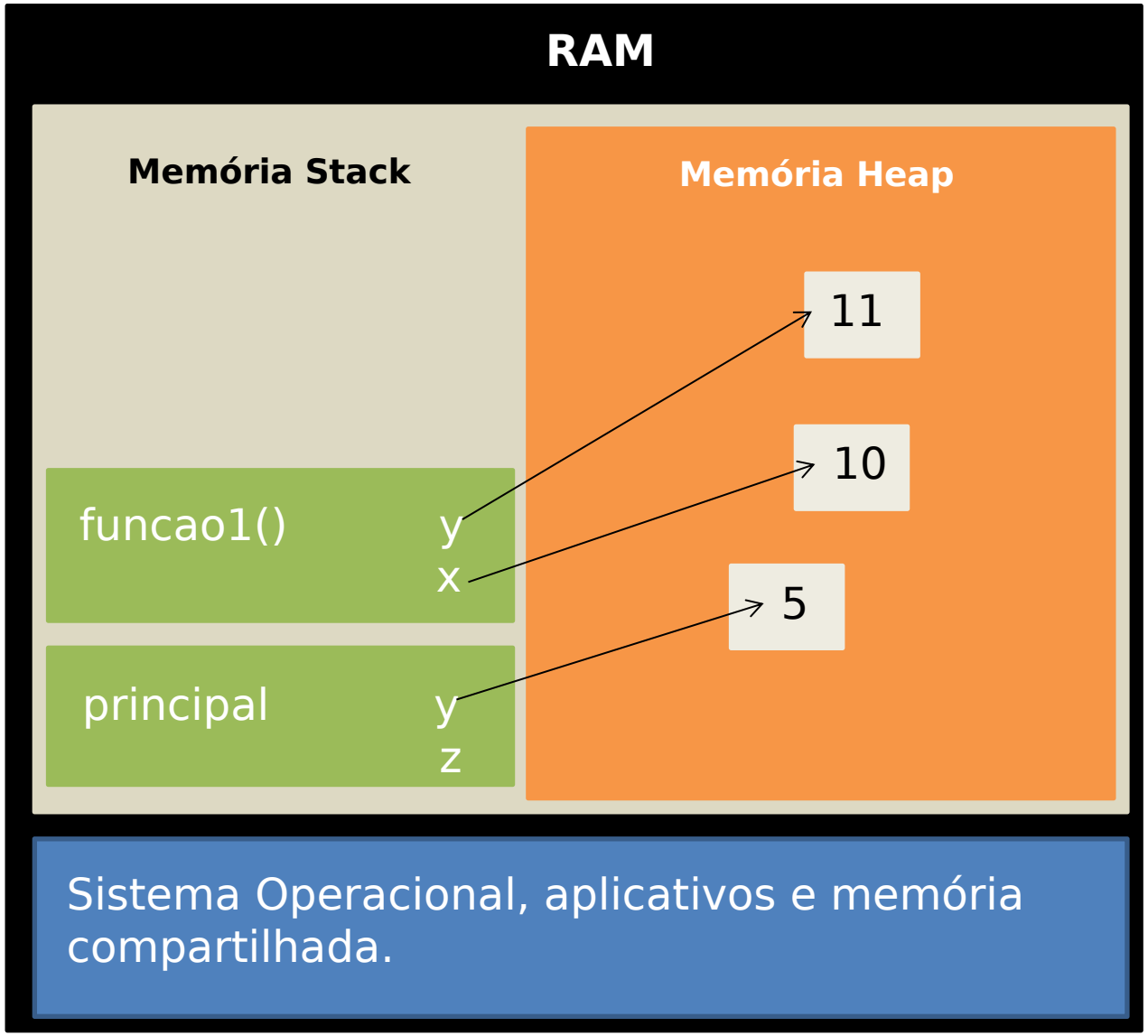
Alocação e Gerência Memória em Python

```
def funcao2(x):
    x = x + 1
    return x

def funcao1(x):
    x = x * 2
    y = funcao2(x)
    return y

#principal
y = 5
z = funcao1(y)
```



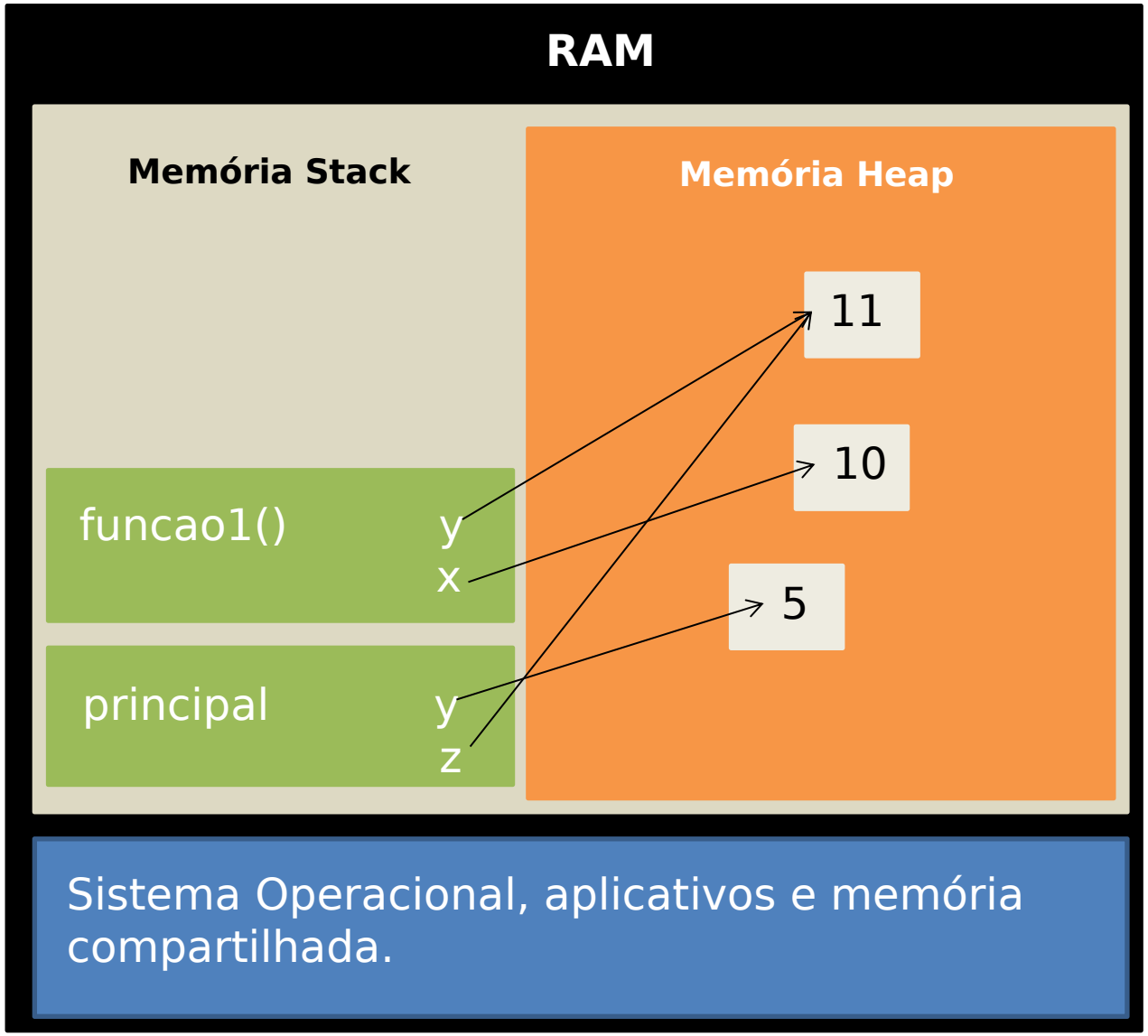
Alocação e Gerência Memória em Python

```
def funcao2(x):
    x = x + 1
    return x

def funcao1(x):
    x = x * 2
    y = funcao2(x)
    return y

#principal
y = 5
z = funcao1(y)
```



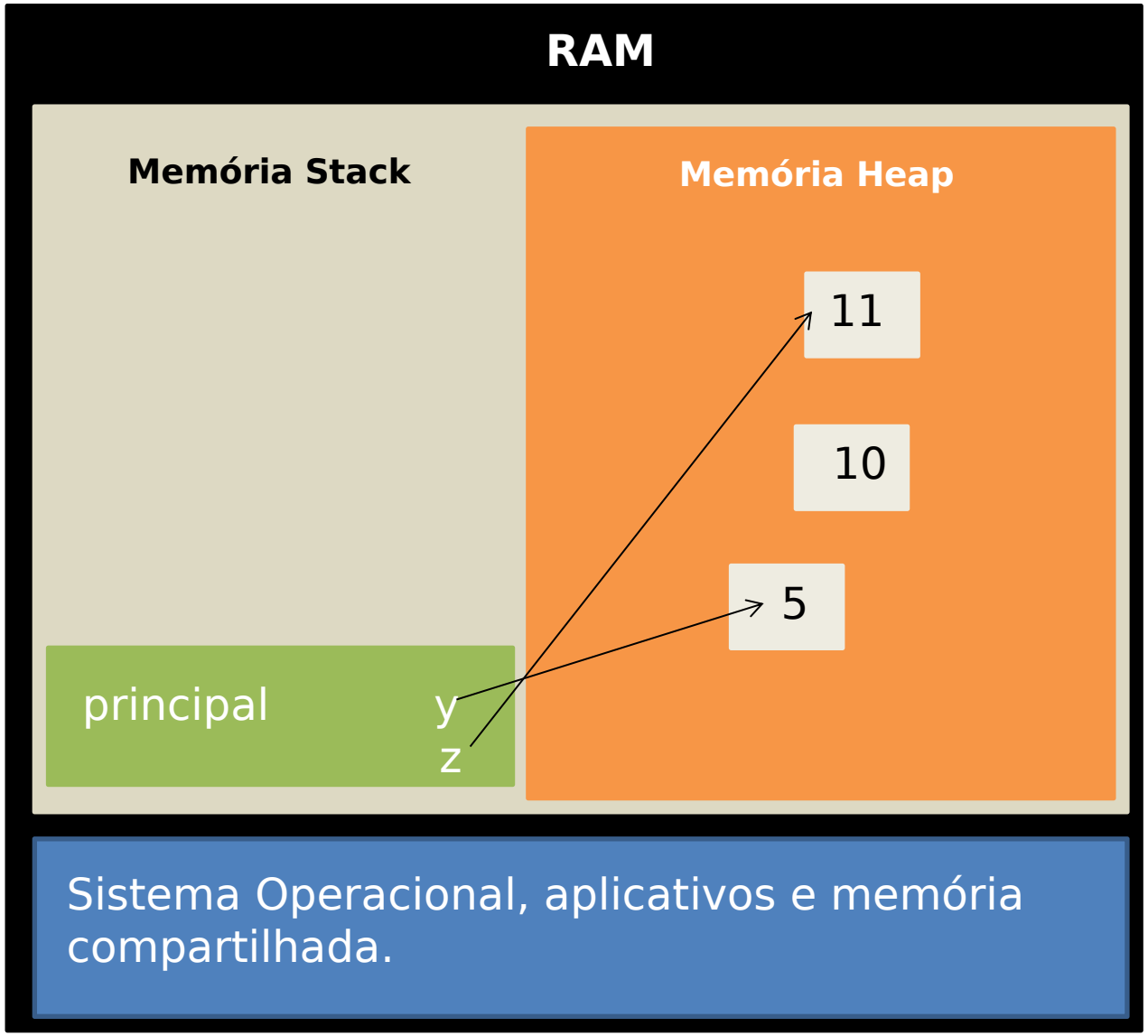
Alocação e Gerência Memória em Python

```
def funcao2(x):
    x = x + 1
    return x

def funcao1(x):
    x = x * 2
    y = funcao2(x)
    return y

#principal
y = 5
z = funcao1(y)
```



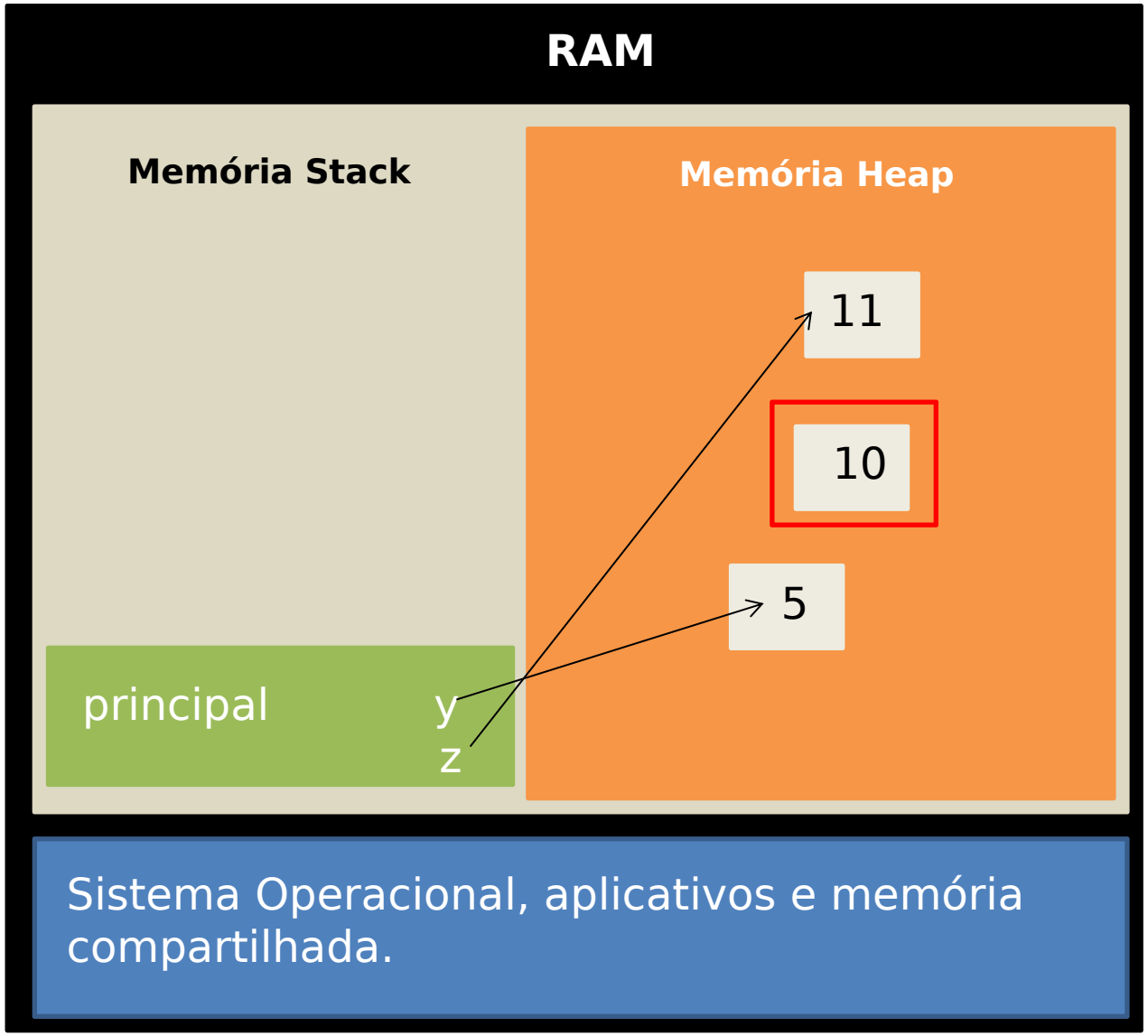
Alocação e Gerência Memória em Python

```
def funcao2(x):
    x = x + 1
    return x

def funcao1(x):
    x = x * 2
    y = funcao2(x)
    return y

#principal
y = 5
z = funcao1(y)
```



Alocação e Gerência Memória em Python

```
def funcao2(x):
    x = x + 1
    return x

def funcao1(x):
    x = x * 2
    y = funcao2(x)
    return y

#principal
y = 5
z = funcao1(y)
```

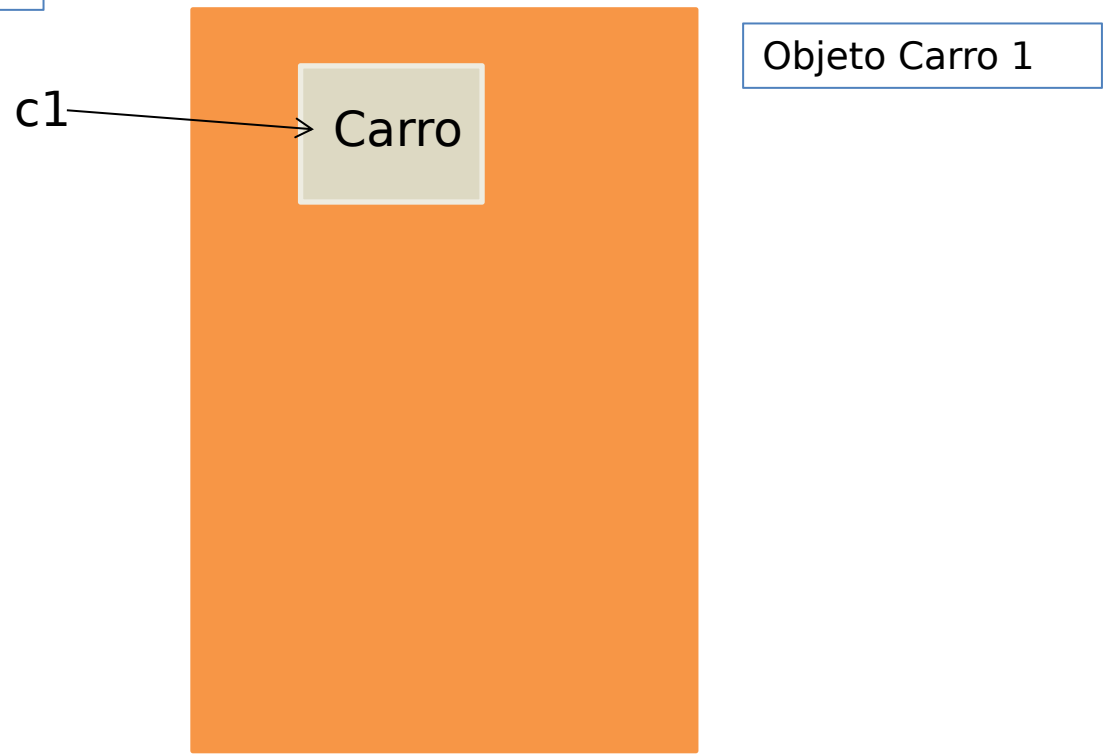


O que acontece com os dados em memória quando não são mais referenciados por nenhum objeto?

Alocação e Gerência Memória em Python

```
class Carro:
    def __init__(self, pneus):
        self.pneus = pneus

    def get_pneus(self):
        return self.pneus
```

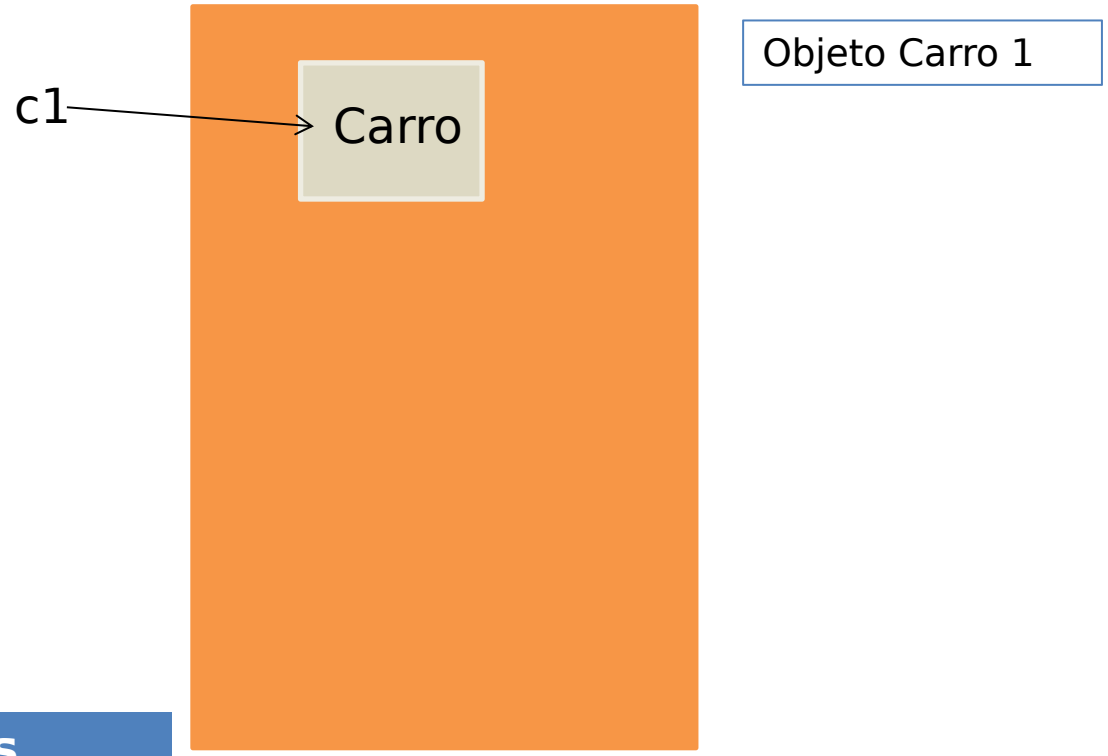
```
c1 = Carro(4)
```



Alocação e Gerência Memória em Python

```
class Carro:
    def __init__(self, pneus):
        self.pneus = pneus

    def get_pneus(self):
        return self.pneus
```

```
c1 = Carro(4)
```

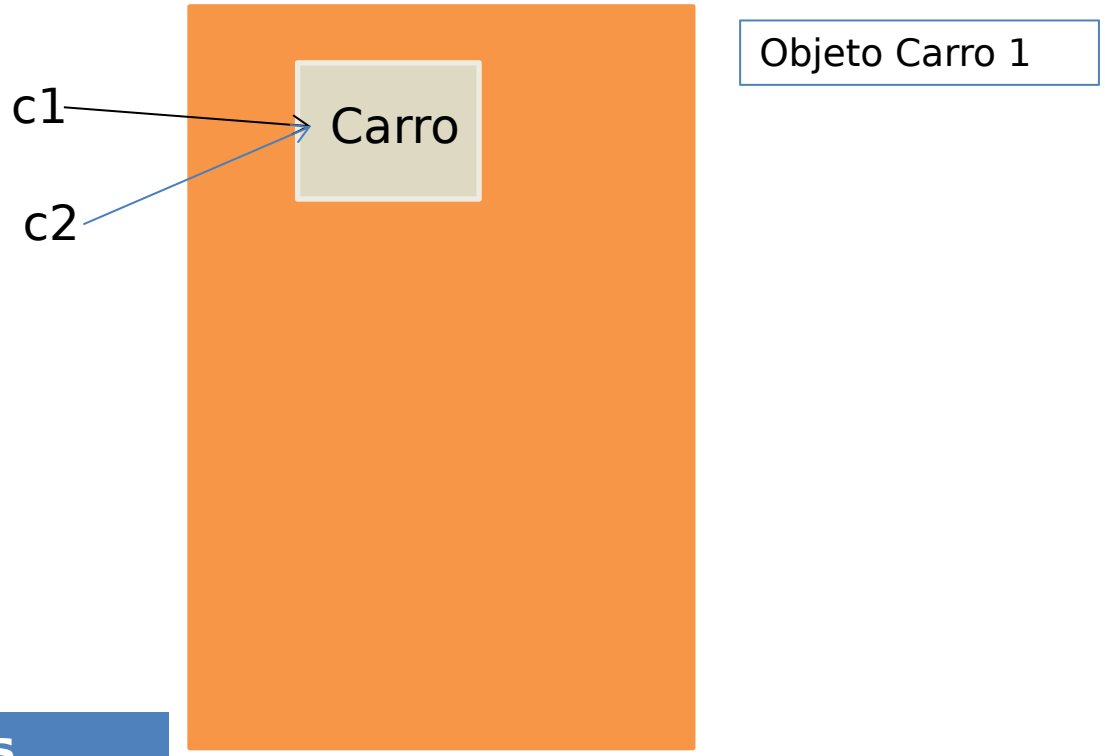


Objeto	Número de referências
Objeto Carro 1	1

Alocação e Gerência Memória em Python

```
class Carro:
    def __init__(self, pneus):
        self.pneus = pneus

    def get_pneus(self):
        return self.pneus
```

```
c1 = Carro(4)

c2 = c1
```

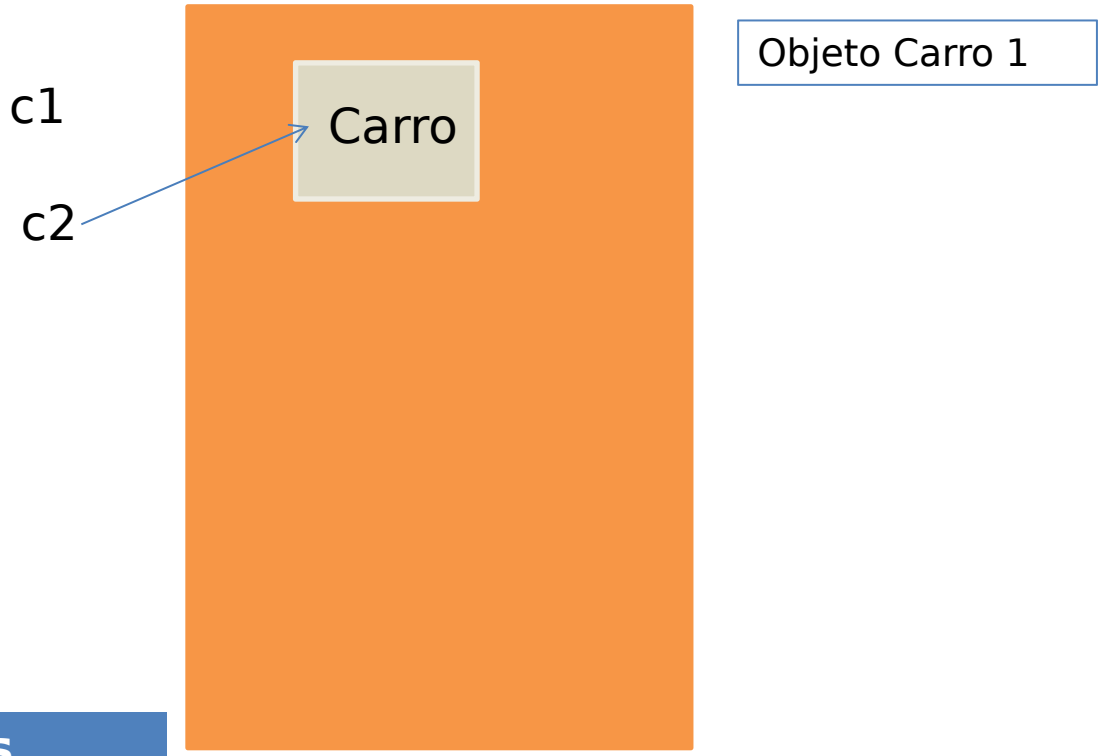


Objeto	Número de referências
Objeto Carro 1	2

Alocação e Gerência Memória em Python

```
class Carro:
    def __init__(self, pneus):
        self.pneus = pneus

    def get_pneus(self):
        return self.pneus
```

```
c1 = Carro(4)

c2 = c1

c1 = None
```

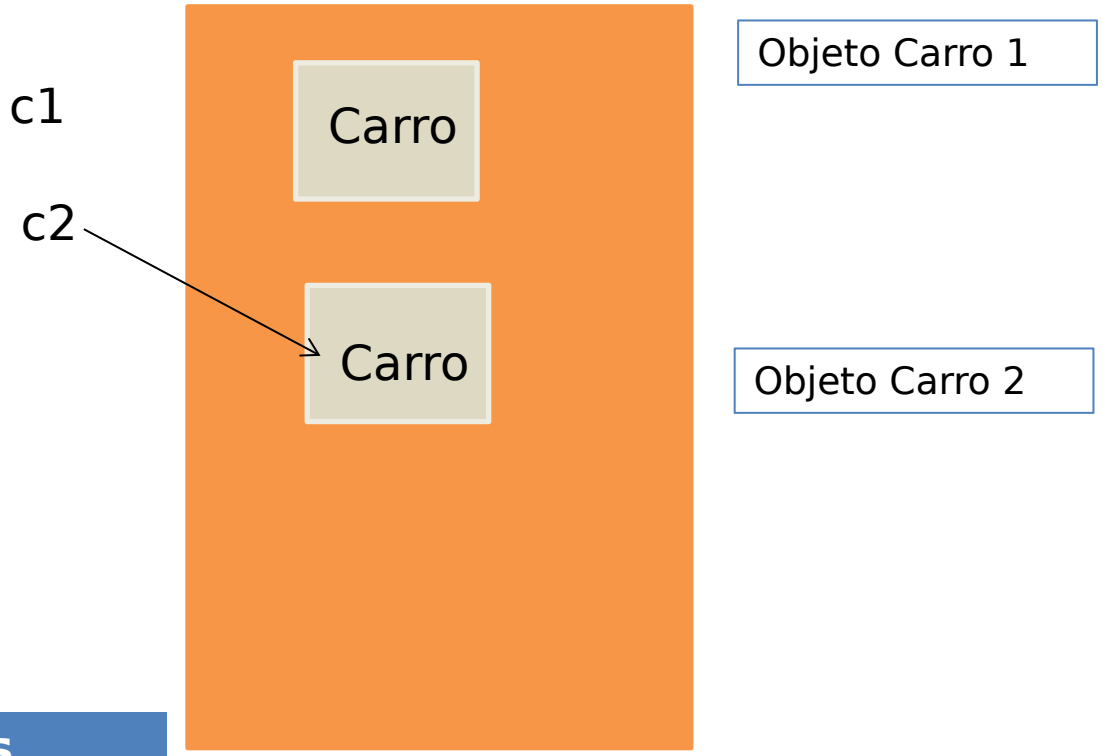


Objeto	Número de referências
Objeto Carro 1	1

Alocação e Gerência Memória em Python

```
class Carro:
    def __init__(self, pneus):
        self.pneus = pneus

    def get_pneus(self):
        return self.pneus
```

```
c1 = Carro(4)

c2 = c1

c1 = None

c2 = Carro(4)
```

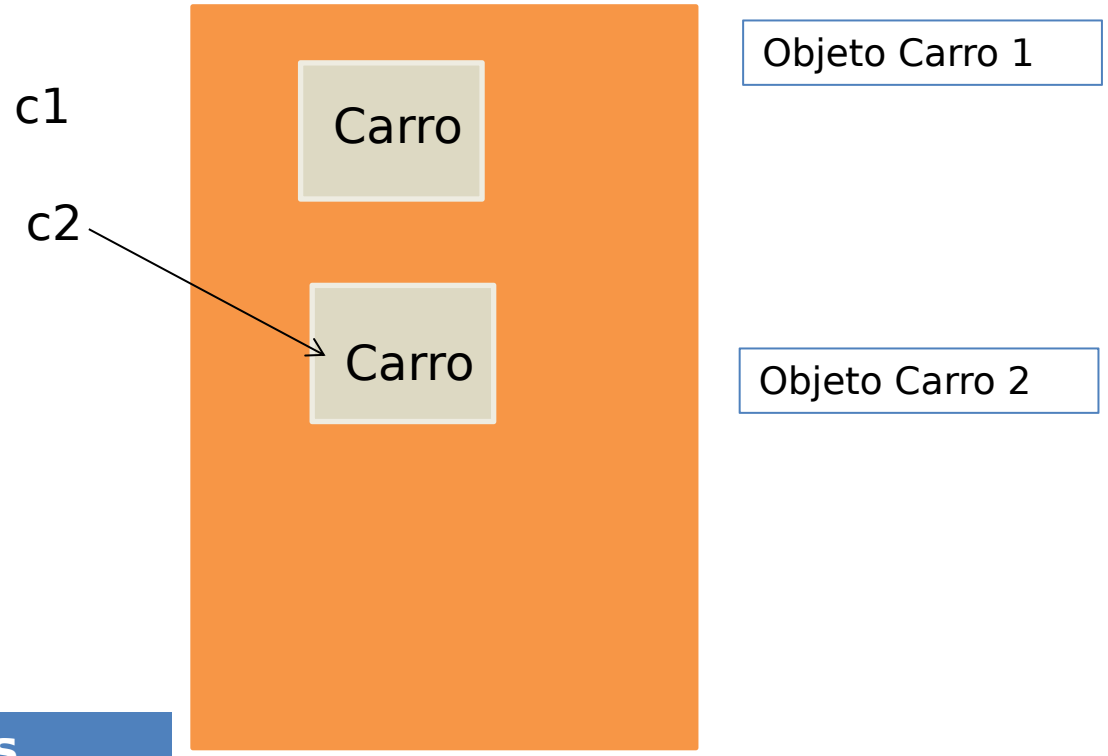


Objeto	Número de referências
Objeto Carro 1	0 Dead Object
Objeto Carro 2	1

Alocação e Gerência Memória em Python

```
class Carro:
    def __init__(self, pneus):
        self.pneus = pneus

    def get_pneus(self):
        return self.pneus
```

```
c1 = Carro(4)

c2 = c1

c1 = None

c2 = Carro(4)
```



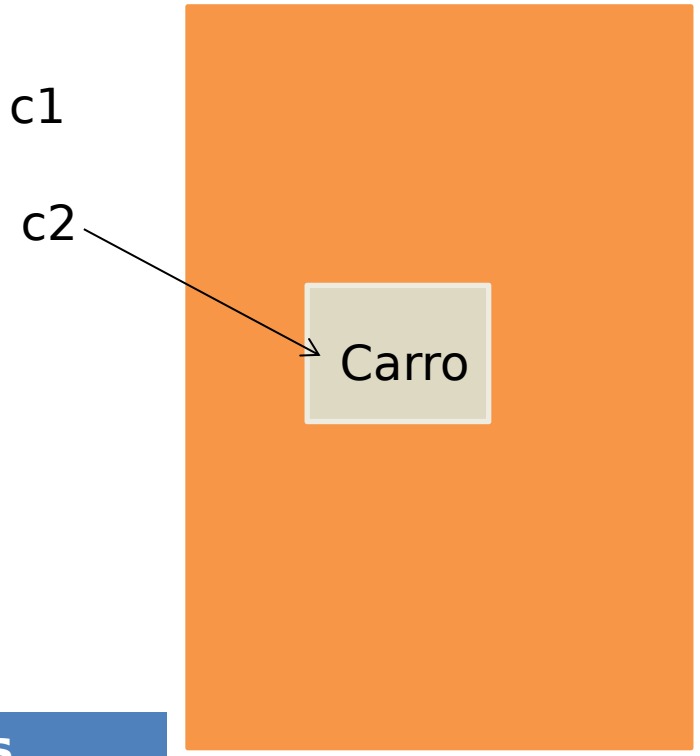
Objeto	Número de referências
Objeto Carro 1	0 Dead Object
Objeto Carro 2	1

Garbage Collector

Alocação e Gerência Memória em Python

```
class Carro:
    def __init__(self, pneus):
        self.pneus = pneus

    def get_pneus(self):
        return self.pneus
```

```
c1 = Carro(4)

c2 = c1

c1 = None

c2 = Carro(4)
```



Objeto	Número de referências
~~Objeto Carro 1~~	~~0~~
Objeto Carro 2	1

O algoritmo utilizado pelo Garbage Collector do Python é chamado de Reference Counting.

Alocação e Gerência Memória em Python

Diferentes linguagens de programação utilizam diferentes algoritmos para o Garbage Collector.

Ate mesmo diferentes implementações da linguagem Python utilizam diferentes implementações para o Garbage Collector: CPython, PyPy, IronPython, Stackless, Jython...

Alocação e Gerência Memória em Python

Comparando Python e outras linguagens como Java e C

Alocação e Gerência Memória em Python

Comparando Python e outras linguagens como Java e C

	Python	Java / C
Declaração	x = 10	int x = 10;

Alocação e Gerência Memória em Python

Comparando Python e outras linguagens como Java e C

	Python	Java / C
Declaração	x = 10	int x = 10;
Declaração de tipo	Não necessário. Dinamicamente tipado.	Obrigatório. Estaticamente tipado.

Alocação e Gerência Memória em Python

Comparando Python e outras linguagens como Java e C

	Python	Java / C
Declaração	x = 10	int x = 10;
Declaração de tipo	Não necessário. Dinamicamente tipado.	Obrigatório. Estaticamente tipado.
O que é 10?	Um objeto criado na memória heap.	Um dado primitivo armazenado em 4 bytes em Java ou 2 bytes em C.

Alocação e Gerência Memória em Python

Comparando Python e outras linguagens como Java e C

	Python	Java / C
Declaração	x = 10	int x = 10;
Declaração de tipo	Não necessário. Dinamicamente tipado.	Obrigatório. Estaticamente tipado.
O que é 10?	Um objeto criado na memória heap.	Um dado primitivo armazenado em 4 bytes em Java ou 2 bytes em C.
O que x contém?	Referência para o Objeto 10.	Local de memória onde 10 está armazenado.

Alocação e Gerência Memória em Python

Comparando Python e outras linguagens como Java e C

	Python	Java / C
Declaração	x = 10	int x = 10;
Declaração de tipo	Não necessário. Dinamicamente tipado.	Obrigatório. Estaticamente tipado.
O que é 10?	Um objeto criado na memória heap.	Um dado primitivo armazenado em 4 bytes em Java ou 2 bytes em C.
O que x contém?	Referência para o Objeto 10.	Local de memória onde 10 está armazenado.
x = x + 1	x é referenciado a um novo objeto com valor 11	x continua apontando para o mesmo local de memória onde o valor foi alterado para 11

Alocação e Gerência Memória em Python

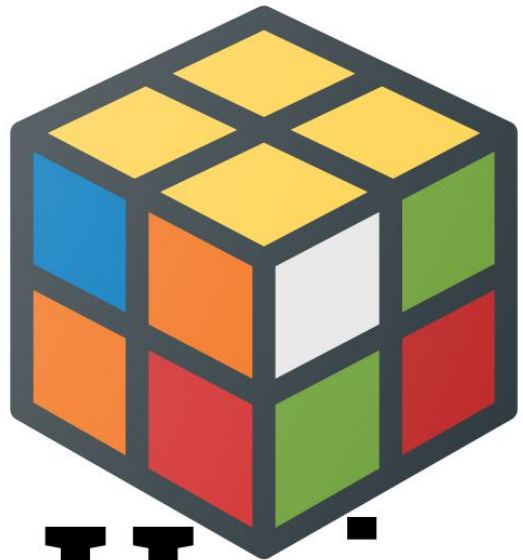
Comparando Python e outras linguagens como Java e C

	Python	Java / C
Declaração	x = 10	int x = 10;
Declaração de tipo	Não necessário. Dinamicamente tipado.	Obrigatório. Estaticamente tipado.
O que é 10?	Um objeto criado na memória heap.	Um dado primitivo armazenado em 4 bytes em Java ou 2 bytes em C.
O que x contém?	Referência para o Objeto 10.	Local de memória onde 10 está armazenado.
x = x + 1	x e referenciado a um novo objeto com valor 11	x continua apontando para o mesmo local de memoria onde o valor foi alterado para 11
x = 10 y = 10	Ambas, x e y são referenciadas para o mesmo objeto.	x e y são duas variáveis apontando para locais diferentes da memória.

Alocação e Gerência Memória em Python

Relembrando tudo...

- Métodos e variáveis são criadas na memória stack;
- Os objetos e instancias são criadas na memória heap;
- Um novo stack é criado durante a invocação de uma função ou método;
- Stacks são destuidas sempre que uma função ou método retorna valor;
- Garbage Collector é um mecanismo para limpar dead objects;



Geek University

Evolua seu lado geek!

www.geekuniversity.com.br