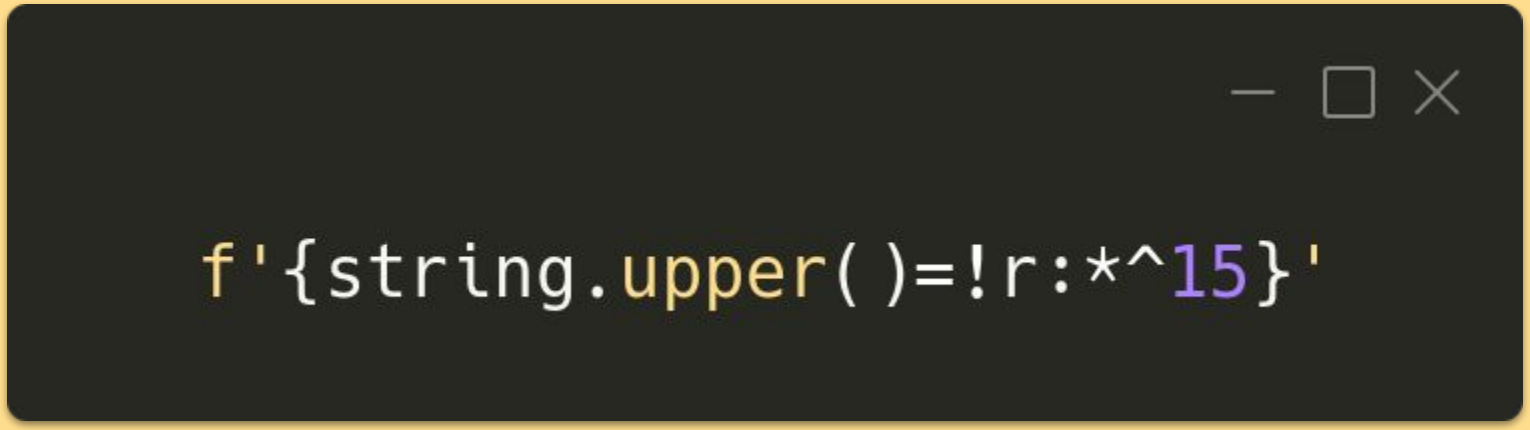


Live de Python #181

Roteiro

1. Tipos de formatação
O básico necessário

2. Entendendo o format
O lado esquedo

3. Uma mini linguagem
O lado direito do format

4. Alguns exemplos legais
Tentando refazer um cardápio

picpay.me/dunossauro

apoia.se/livedepython

PIX

Ajude o projeto

picpay.me/dunossauro

apoia.se/livedepython

PIX

Ajude o projeto

Acássio Anjos, Ademar Peixoto, Alex Lima, Alexandre Harano, Alexandre Santos, Alexandre Tsuno, Alexandre Villares, Alynne Ferreira, Alysson Oliveira, Amaziles Carvalho, Andre Azevedo, André Rocha, Antonio Lins, Arnaldo Turque, Artur Zalewska, Bruno Barcellos, Bruno Batista, Bruno Freitas, Bruno Guizi, Bruno Oliveira, Caio Nascimento, Carlos Chiarelli, Cleber Santos, César Almeida, Davi Ramos, David Kwast, Diego Guimarães, Diego Ubirajara, Dilenon Delfino, Dino Aguilar, Donivaldo Sarzi, Elias Soares, Emerson Rafael, Eric Niens, Eugenio Mazzini, Euripedes Borges, Everton Alves, Fabiano Gomes, Fabio Barros, Fabio Castro, Flavkaze Flavkaze, Flávio Meira, Francisco Alencar, Franklin Silva, Fábio Barros, Gabriel Simonetto, Gabriel Soares, Gabriela Santiago, Geandreson Costa, Guilherme Castro, Guilherme Felitti, Guilherme Gall, Guilherme Ostrock, Gustavo Suto, Henrique Junqueira, Henrique Machado, Ismael Ventura, Israel Fabiano, Israel Gomes, Italo Silva, Jair Andrade, Jairo Rocha, Johnny Tardin, Jonatas Leon, Jonatas Oliveira, Jorge Plautz, Jose Mazolini, José Gomes, José Prado, João Lugão, Juan Gutierrez, Jônatas Silva, Kaio Peixoto, Kaneson Alves, Leonardo Cruz, Leonardo Mello, Leonardo Nascimento, Lidiane Monteiro, Lorena Ribeiro, Lucas Barros, Lucas Mello, Lucas Mendes, Lucas Oliveira, Lucas Polo, Lucas Teixeira, Lucas Valino, Luciano Ratamero, Luciano Silva, Maiquel Leonel, Marcela Campos, Marcelino Pinheiro, Marcos Ferreira, Maria Clara, Marina Passos, Matheus Vian, Murilo Cunha, Natan Cervinski, Nicolas Teodosio, Osvaldo Neto, Patric Lacouth, Patricia Minamizawa, Patrick Brito, Patrick Gomes, Paulo Tadei, Pedro Pereira, Peterson Santos, Rafael Lino, Reinaldo Silva, Renan Moura, Revton Silva, Richard Nixon, Riverfount Riverfount, Robson Maciel, Rodrigo Ferreira, Rodrigo Mende, Rodrigo Vaccari, Rodrigo Vieira, Ronaldo Silva, Rui Jr, Samanta Cicilia, Sandro Mio, Sara Selis, Silvio Xm, Thiago Araujo, Thiago Borges, Thiago Bueno, Thiago Moraes, Tony Dias, Tony Santos, Tyrone Damasceno, Vinícius Bastos, Vlademir Souza, Vítor Gomes, Wellington Abreu, Wendel Rios, Wesley Mendes, Willian Lopes, Willian Rosa, Wilson Duarte, Yuri Fialho, Yury Barros, Érico Andrei

Obrigado você

Dá pra formatar de quais maneiras?

Tipos

Tipos de formatação

```python
live, de, python = 'Live', 'de', 'Python'

'%s %s %s' % (live, de, python)      # Live de Python
'{} {} {}'.format(live, de, python)  # Live de Python
f'{live} {de} {python}'              # Live de Python
```

O incrível lado esquerdo

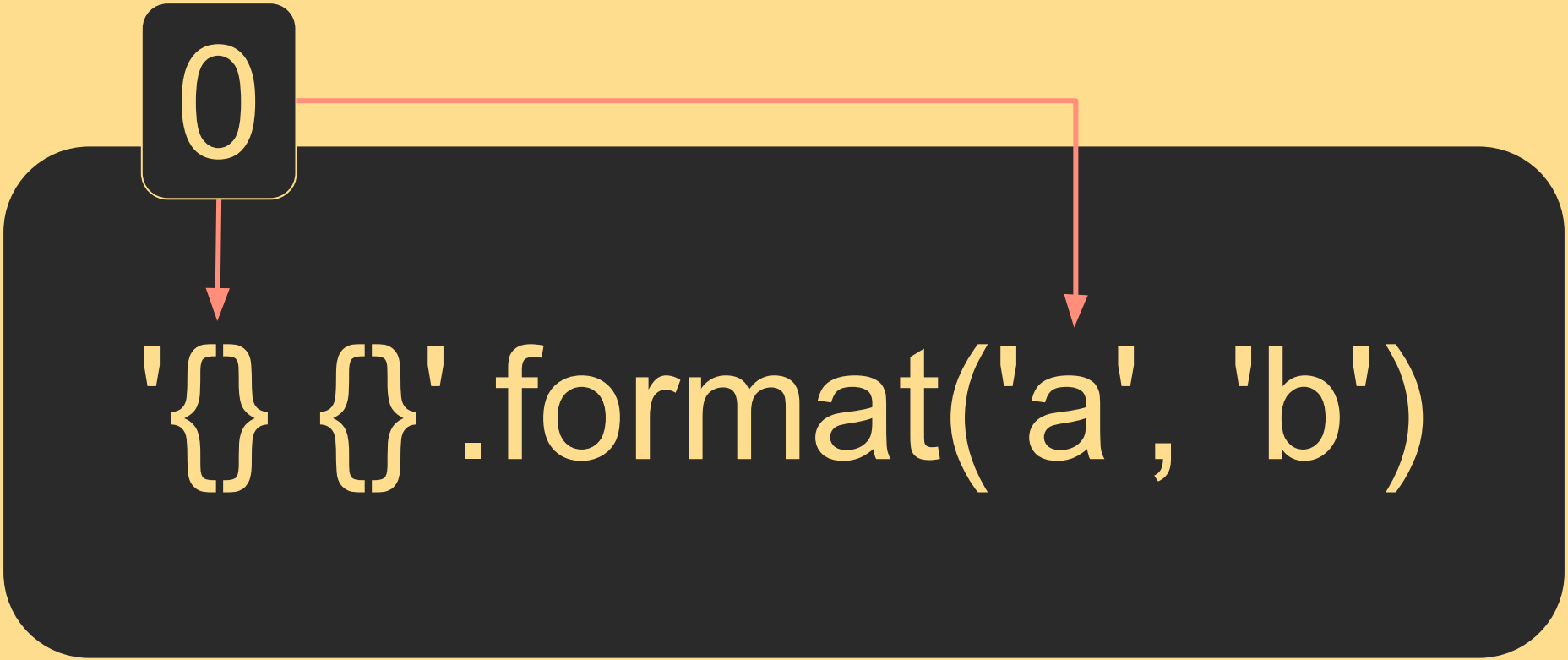
Format

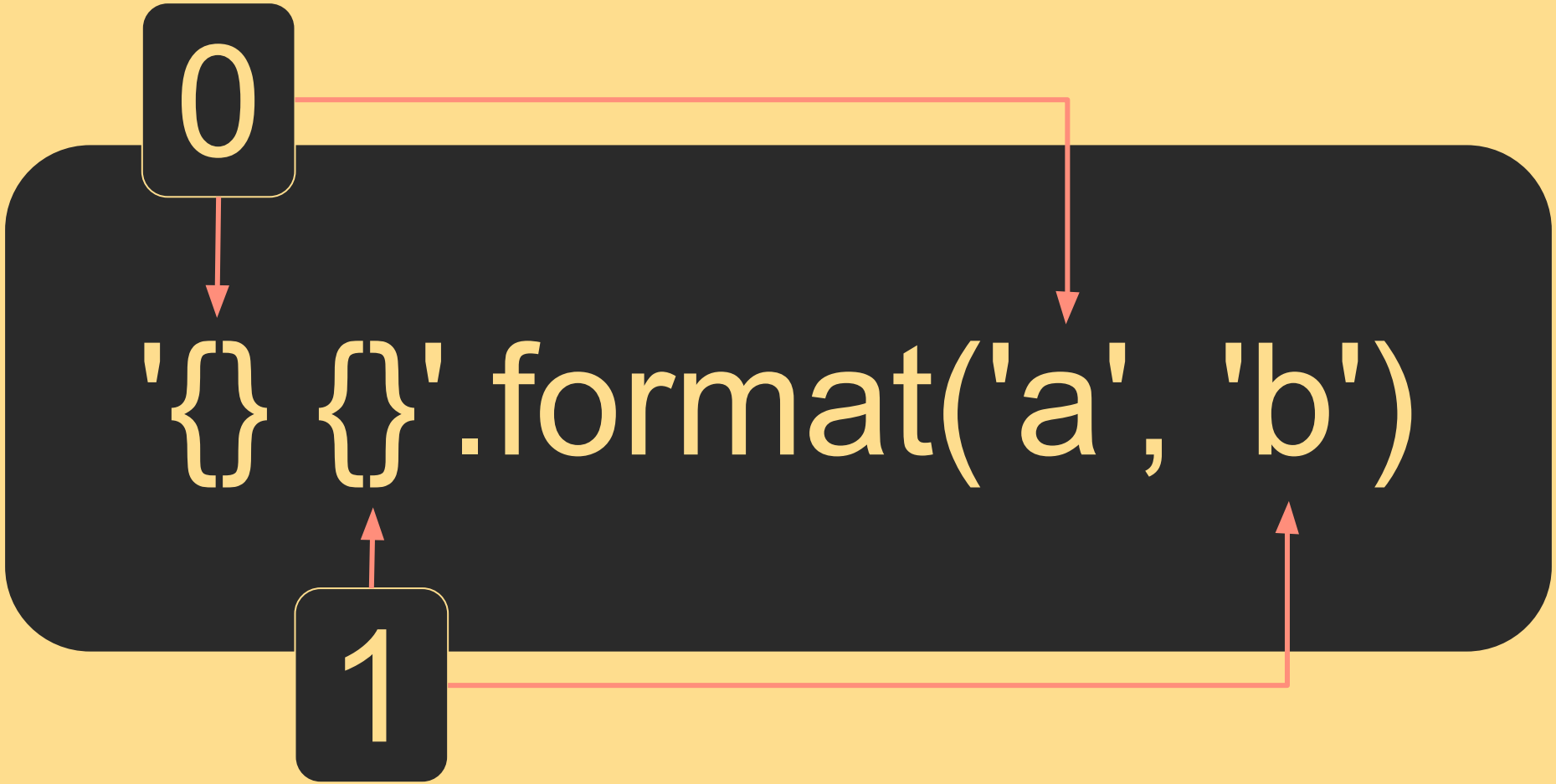
```
'{}'.format('a')
```

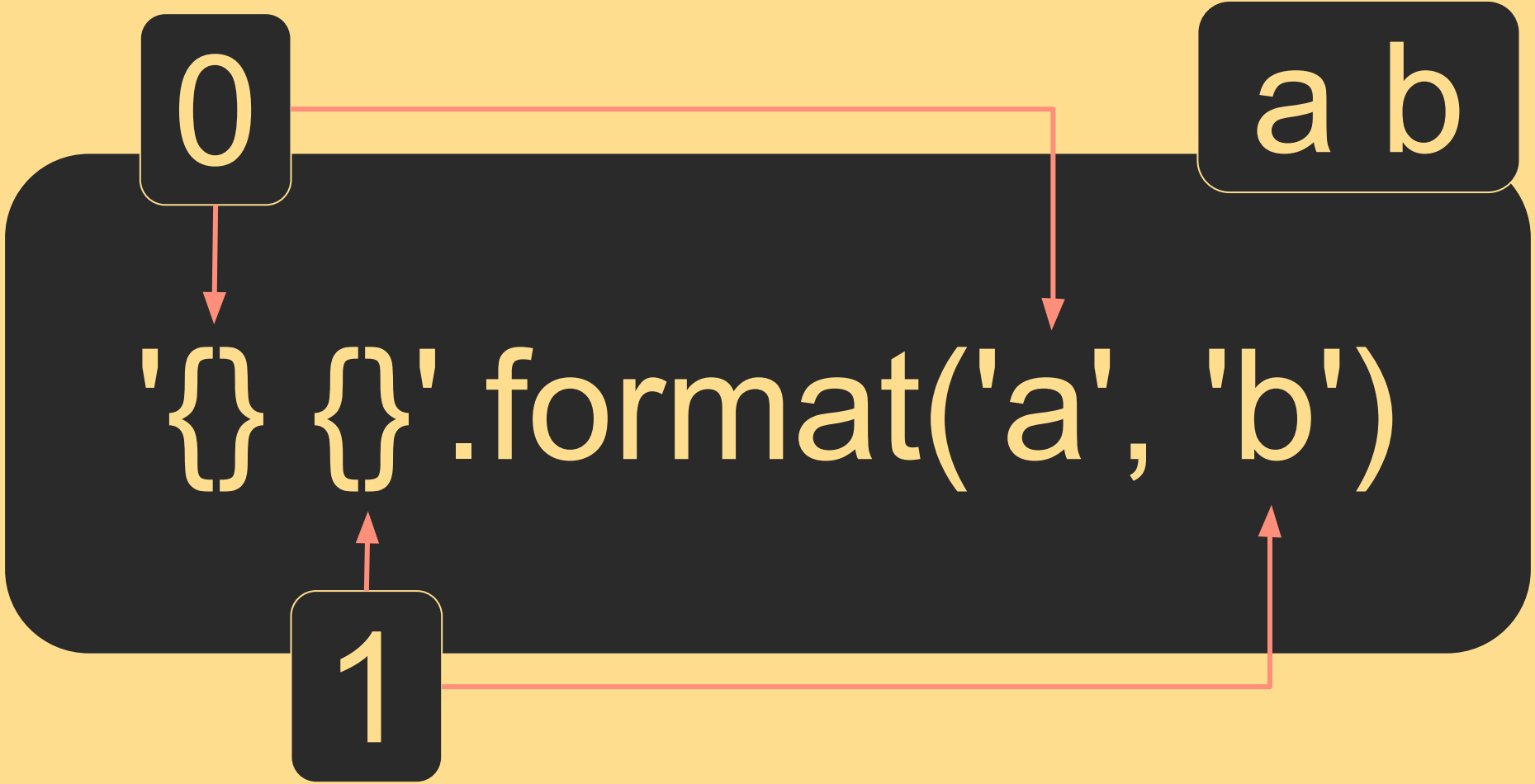

'{}'.format('a')
a

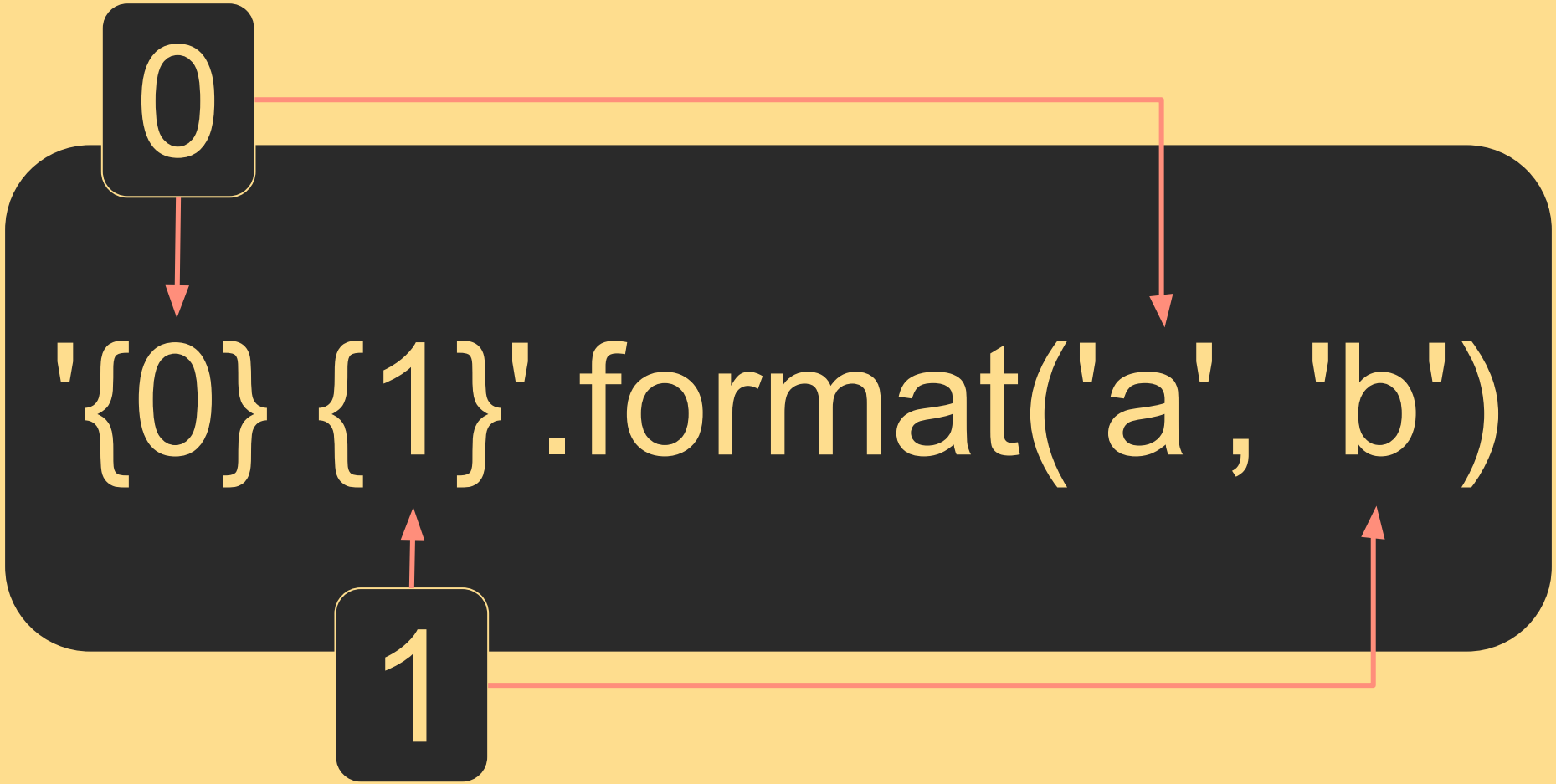
```
'{} {}'.format('a', 'b')
```

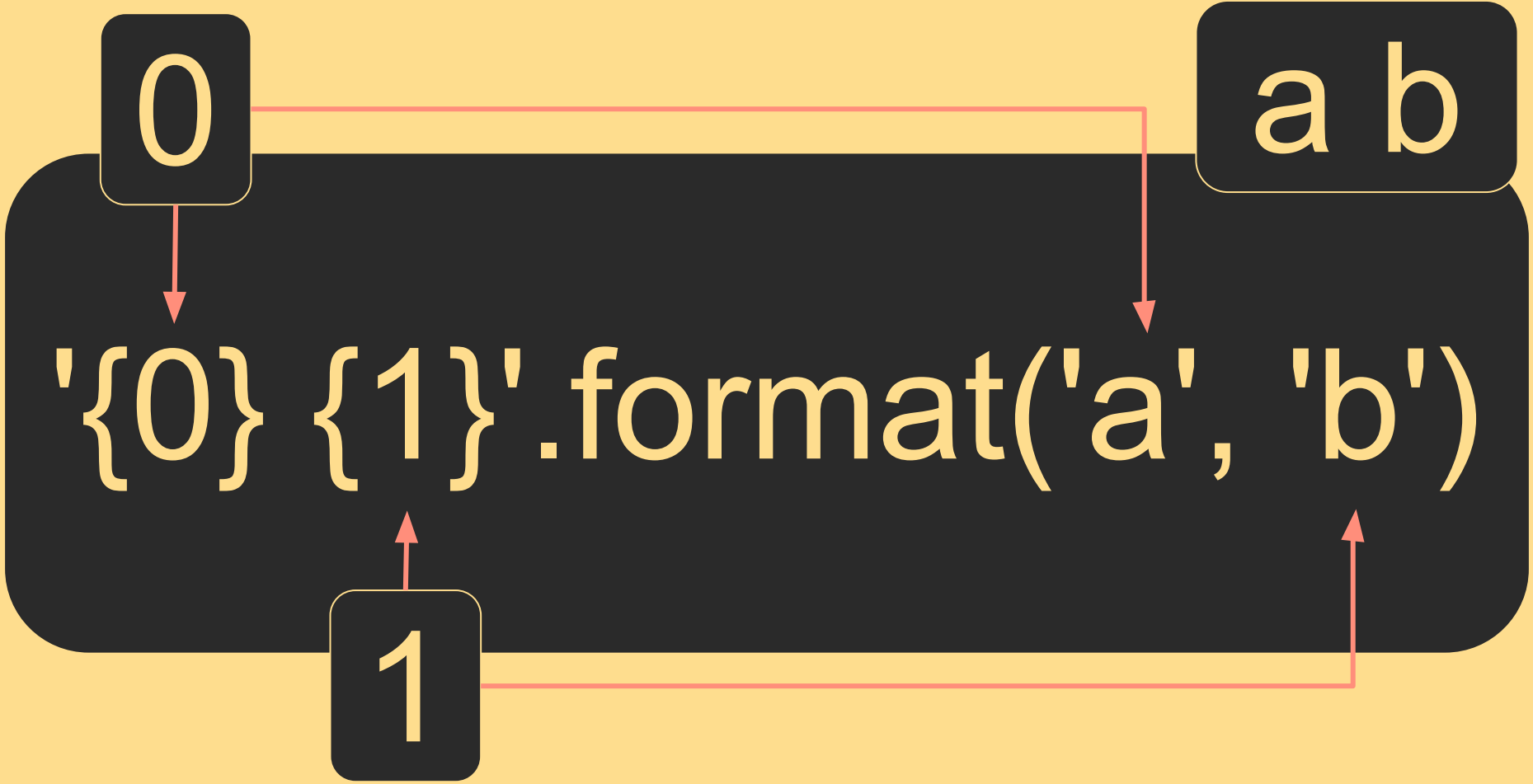

'{} {}'.format('a', 'b')
a b

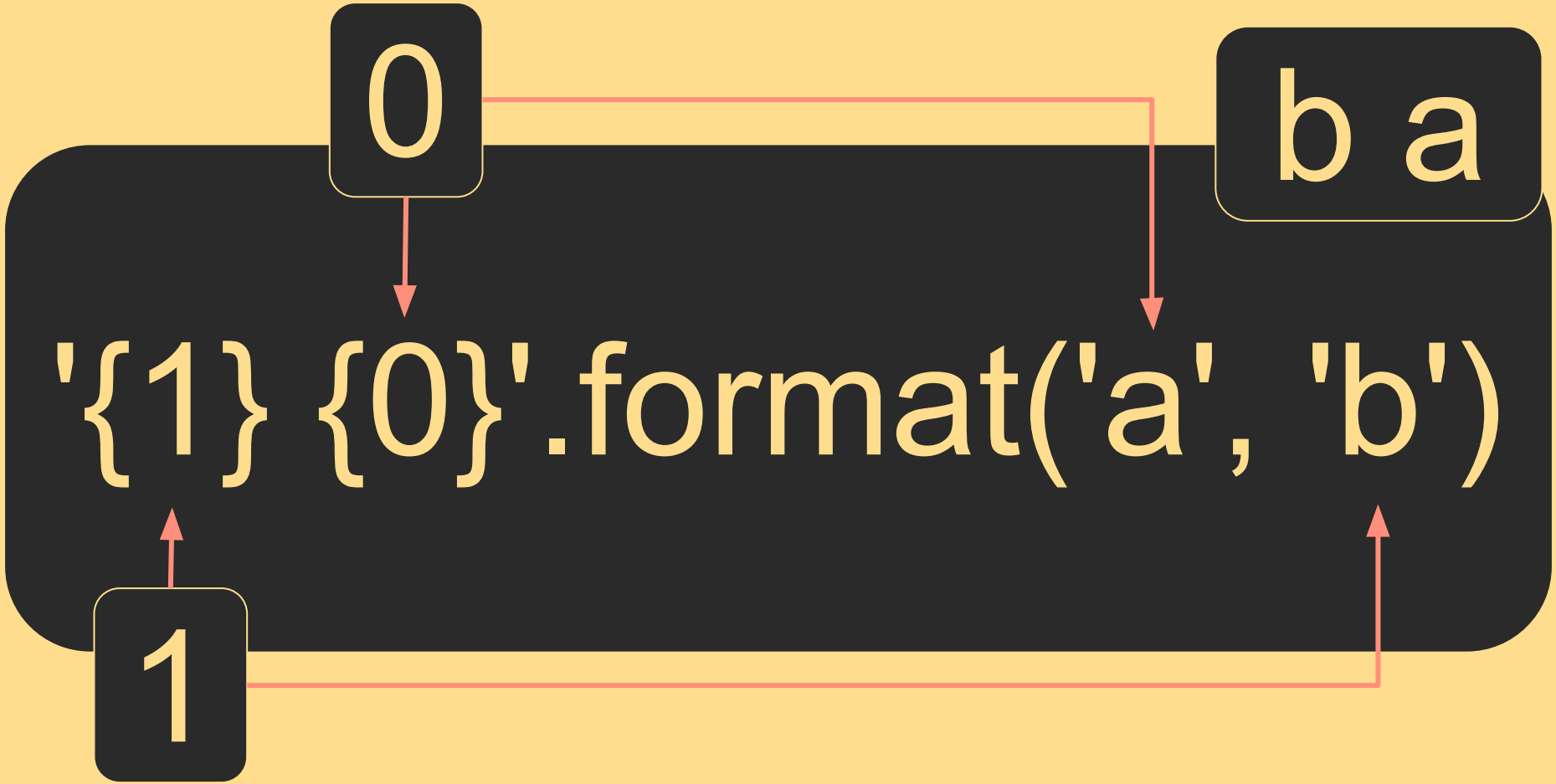

0
'{} {}'.format('a', 'b')


0
'{} {}'.format('a', 'b')
1


0
a b
'{} {}'.format('a', 'b')
1


0
'{0} {1}'.format('a', 'b')
1


0
a b
'{0} {1}'.format('a', 'b')
1


0
b a
'{1} {0}'.format('a', 'b')
1

Parâmetros nomeados

Agora fica mais interessante

```
'{nome} {sobrenome}'.format(
    nome='Eduardo',
    sobrenome='Mendes'
)
```

```
'{nome} {sobrenome}'.format(
    nome='Eduardo',
    sobrenome='Mendes'
)
```

Eduardo Mendes

```
'{sobrenome} {nome}'.format(
    nome='Eduardo',
    sobrenome='Mendes'
)
```

Mendes Eduardo

```
nome = 'Eduardo'
sobrenome = 'Mendes'

f'{sobrenome} {nome}'
```

Mendes Eduardo

Chamadas de métodos

Sim, dá pra fazer também

```
nome = 'Eduardo'

f'{nome.upper()}'
```

```
EDUARDO
```

```
nome = 'Eduardo'

f'{nome.lower()}'
```

```
eduardo
```

```
lista = [1, 2, 3, 4]

f'{lista[0]} {lista[1:]}'
```

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

Conversões

Convertendo para outros tipos de texto

O esquema das conversões

O valores podem ser convertidos para outros tipos de texto, o que acrescenta coisas no nosso vocabulário de strings

{string[!<conversão>]}

Tipos disponíveis

texto = 'Ração'

{texto!s}	Ração
{texto!r}	"Ração"
{texto!a}	Ra\\xe7\\xe3o

Como essa conversão acontece?

O python, dependendo do tipo. Vai chamar uma função diferente para a formatação da string

`{texto!s}` ➡ `str(texto)`

Como essa conversão acontece?

O python, dependendo do tipo. Vai chamar uma função diferente para a formatação da string

{texto!r} ➡ repr(texto)

Como essa conversão acontece?

O python, dependendo do tipo. Vai chamar uma função diferente para a formatação da string

{texto!a} ➡ ascii(texto)

Olhando mais a fundo

{Exemplo()!s}

str(texto)

```
1  class Exemplo:
2      texto = 'Ração'
3
4      def __str__(self):
5          return f'str -> {self.texto}'
6
7      def __repr__(self):
8          return f'repr -> {self.texto}'
```

Olhando mais a fundo

{Exemplo()!r}

repr(texto)

```
1  class Exemplo:
2      texto = 'Ração'
3
4      def __str__(self):
5          return f'str -> {self.texto}'
6
7      def __repr__(self):
8          return f'repr -> {self.texto}'
```

Olhando mais a fundo

{Exemplo()!a}

ascii(texto)

repr(texto)

```python
class Exemplo:
    texto = 'Ração'

    def __str__(self):
        return f'str -> {self.texto}'

    def __repr__(self):
        return f'repr -> {self.texto}'
```

https://docs.python.org/3/library/functions.html#ascii

O sinal de Igual

Debugar nunca foi tão gostoso

O sinal de igual (3.8+)

O sinal de igual pode auxiliar em logs e quando estamos tentando debugar algo. Ele nos mostra o nome e o valor da variável

```
nome = 'Eduardo'

f'{nome=}'
```

O sinal de igual

O sinal de igual pode auxiliar em logs e quando estamos tentando debugar algo. Ele nos mostra o nome e o valor da variável

```
nome = 'Eduardo'

f'{nome=}'
```

```
"nome='Eduardo'"
```

Uma conclusão

É possível ver o nome e valor da variável usando `=`. É possível converter a string em outros tipos de texto e também fazer chamada de métodos.

```
{[nome][=][!converção]}
```

Uma breve revisão

```python
string = 'Melão'

f'{string}'              # Melão
f'{string.upper()}'      # MELÃO
f'{string[-1]}'          # o
f'{string=}'             # string='Melão'
f'{string!a}'            # 'Mel\\xe3o'
f'{string.upper()=!a}'   # string.upper()='MEL\\xc3O'
```

Mini linguagem

O temido lado direito

Standard Format Specifiers

If an object does not define its own format specifiers, a standard set of format specifiers is used. These are similar in concept to the format specifiers used by the existing '%' operator, however there are also a number of differences.

The general form of a standard format specifier is:

```
[[fill]align][sign][#][0][minimumwidth][.precision][type]
```

https://www.python.org/dev/peps/pep-3101/

The general form of a *standard format specifier* is:

```
format_spec     ::=  [[fill]align][sign][#][0][width][grouping_option][.precision][type]
fill            ::=  <any character>
align           ::=  "<" | ">" | "=" | "^"
sign            ::=  "+" | "-" | " "
width           ::=  digit+
grouping_option ::=  "_" | ","
precision       ::=  digit+
type            ::=  "b" | "c" | "d" | "e" | "E" | "f" | "F" | "g" | "G" | "n" | "o" | "s" | "x" | "X" | "%"
```

https://docs.python.org/3/library/string.html#format-specification-mini-language

Vamos começar do básico

Tudo que faz parte da mini-linguagem é dividido por dois pontos **:**

Vamos começar do básico

Tudo que falamos antes, vai ficar do lado esquerdo dos dois pontos

```
{string=!s:}
```

Mas o que podemos fazer?

Mais links

- https://www.python.org/dev/peps/pep-3101/

- https://docs.python.org/3/library/string.html#format-specification-mini-language

- https://docs.python.org/3/whatsnew/3.8.html

- https://realpython.com/python-formatted-output/