



## Módulo 3.1 - Loops

💰 5000



Olá, desenvolvedores! O nosso conteúdo do dia englobará algumas das estruturas fundamentais do Python (e qualquer outra linguagem), os nossos loops. Esperamos que vocês possam tirar o máximo possível de proveito do conteúdo que estamos disponibilizando, não esqueçam de praticar bastante para fixar bem.

### for...in

O Python não remove e tem loops *for* no sentido tradicional. Ele tem apenas loops *for...in*, o qual chamaremos de loop *for*. O loop *for* do Python pega um iterável e retorna cada item dele, um por vez, até que todos os itens tenham sido iterados ou o loop tenha terminado. Ele funciona no estilo *iteração* e *iterável*, que veremos mais detalhadamente numa aula futura. No exemplo a seguir, `carrinho_compras` é uma lista de itens de compra. Dentro do loop *for*, você tem acesso à iteração atual do item a cada passagem pelo loop.

```
carrinho_compras = ['Leite', 'Ovos', 'Presunto', 'Queijo', 'Pão', 'Café']
for item in carrinho_compras:
    print(item)

>>> Leite
>>> Ovos
>>> Presunto
>>> ...
>>> Café
```

Aqui está um exemplo da impressão de cada letra de uma string, uma de cada vez.

```
for letra in "Olá Mundo":
    print(letra)
```



## Módulo 3.1 - Loops

💰 5000



Olá, desenvolvedores! O nosso conteúdo do dia englobará algumas das estruturas fundamentais do Python (e qualquer outra linguagem), os nossos loops. Esperamos que vocês possam tirar o máximo possível de proveito do conteúdo que estamos disponibilizando, não esqueçam de praticar bastante para fixar bem.

### for...in

O Python não remove e tem loops *for* no sentido tradicional. Ele tem apenas loops *for...in*, o qual chamaremos de loop *for*. O loop *for* do Python pega um iterável e retorna cada item dele, um por vez, até que todos os itens tenham sido iterados ou o loop tenha terminado. Ele funciona no estilo *iteração* e *iterável*, que veremos mais detalhadamente numa aula futura. No exemplo a seguir, `carrinho_compras` é uma lista de itens de compra. Dentro do loop *for*, você tem acesso à iteração atual do item a cada passagem pelo loop.

```
carrito_compras = ['Leite', 'Ovos', 'Presunto', 'Queijo', 'Pão', 'Café']
for item in carrito_compras:
    print(item)

>>> Leite
>>> Ovos
>>> Presunto
>>> ...
>>> Café
```

Aqui está um exemplo da impressão de cada letra de uma string, uma de cada vez.

```
for letra in "Olá Mundo":
    print(letra)
```



## Módulo 3.1 - Loops

💰 5000



Olá, desenvolvedores! O nosso conteúdo do dia englobará algumas das estruturas fundamentais do Python (e qualquer outra linguagem), os nossos loops. Esperamos que vocês possam tirar o máximo possível de proveito do conteúdo que estamos disponibilizando, não esqueçam de praticar bastante para fixar bem.

### for...in

O Python não remove e tem loops *for* no sentido tradicional. Ele tem apenas loops *for...in*, o qual chamaremos de loop *for*. O loop *for* do Python pega um iterável e retorna cada item dele, um por vez, até que todos os itens tenham sido iterados ou o loop tenha terminado. Ele funciona no estilo *iteração* e *iterável*, que veremos mais detalhadamente numa aula futura. No exemplo a seguir, `carrinho_compras` é uma lista de itens de compra. Dentro do loop *for*, você tem acesso à iteração atual do item a cada passagem pelo loop.

```
carrinho_compras = ['Leite', 'Ovos', 'Presunto', 'Queijo', 'Pão', 'Café']
for item in carrinho_compras:
    print(item)

>>> Leite
>>> Ovos
>>> Presunto
>>> ...
>>> Café
```

Aqui está um exemplo da impressão de cada letra de uma string, uma de cada vez.

```
for letra in "Olá Mundo":
    print(letra)
```

Módulo 2.2 -  
Listas



## Módulo 3.1 - Loops

💰 5000



Olá, desenvolvedores! O nosso conteúdo do dia englobará algumas das estruturas fundamentais do Python (e qualquer outra linguagem), os nossos loops. Esperamos que vocês possam tirar o máximo possível de proveito do conteúdo que estamos disponibilizando, não esqueçam de praticar bastante para fixar bem.

### for...in

O Python não remove e tem loops *for* no sentido tradicional. Ele tem apenas loops *for...in*, o qual chamaremos de loop *for*. O loop *for* do Python pega um iterável e retorna cada item dele, um por vez, até que todos os itens tenham sido iterados ou o loop tenha terminado. Ele funciona no estilo *iteração* e *iterável*, que veremos mais detalhadamente numa aula futura. No exemplo a seguir, `carrinho_compras` é uma lista de itens de compra. Dentro do loop *for*, você tem acesso à iteração atual do item a cada passagem pelo loop.

```
carrinho_compras = ['Leite', 'Ovos', 'Presunto', 'Queijo', 'Pão', 'Café']
for item in carrinho_compras:
    print(item)

>>> Leite
>>> Ovos
>>> Presunto
>>> ...
>>> Café
```

Aqui está um exemplo da impressão de cada letra de uma string, uma de cada vez.

```
for letra in "Olá Mundo":
    print(letra)
```



## Módulo 3.1 - Loops

💰 5000



Olá, desenvolvedores! O nosso conteúdo do dia englobará algumas das estruturas fundamentais do Python (e qualquer outra linguagem), os nossos loops. Esperamos que vocês possam tirar o máximo possível de proveito do conteúdo que estamos disponibilizando, não esqueçam de praticar bastante para fixar bem.

### for...in

O Python não remove e tem loops *for* no sentido tradicional. Ele tem apenas loops *for...in*, o qual chamaremos de loop *for*. O loop *for* do Python pega um iterável e retorna cada item dele, um por vez, até que todos os itens tenham sido iterados ou o loop tenha terminado. Ele funciona no estilo *iteração* e *iterável*, que veremos mais detalhadamente numa aula futura. No exemplo a seguir, `carrinho_compras` é uma lista de itens de compra. Dentro do loop *for*, você tem acesso à iteração atual do item a cada passagem pelo loop.

```
carrinho_compras = ['Leite', 'Ovos', 'Presunto', 'Queijo', 'Pão', 'Café']
for item in carrinho_compras:
    print(item)

>>> Leite
>>> Ovos
>>> Presunto
>>> ...
>>> Café
```

Aqui está um exemplo da impressão de cada letra de uma string, uma de cada vez.

```
for letra in "Olá Mundo":
    print(letra)
```



## Módulo 3.1 - Loops

💰 5000



Olá, desenvolvedores! O nosso conteúdo do dia englobará algumas das estruturas fundamentais do Python (e qualquer outra linguagem), os nossos loops. Esperamos que vocês possam tirar o máximo possível de proveito do conteúdo que estamos disponibilizando, não esqueçam de praticar bastante para fixar bem.

### for...in

O Python não remove e tem loops *for* no sentido tradicional. Ele tem apenas loops *for...in*, o qual chamaremos de loop *for*. O loop *for* do Python pega um iterável e retorna cada item dele, um por vez, até que todos os itens tenham sido iterados ou o loop tenha terminado. Ele funciona no estilo *iteração* e *iterável*, que veremos mais detalhadamente numa aula futura. No exemplo a seguir, `carrinho_compras` é uma lista de itens de compra. Dentro do loop *for*, você tem acesso à iteração atual do item a cada passagem pelo loop.

```
carrinho_compras = ['Leite', 'Ovos', 'Presunto', 'Queijo', 'Pão', 'Café']
for item in carrinho_compras:
    print(item)

>>> Leite
>>> Ovos
>>> Presunto
>>> ...
>>> Café
```

Aqui está um exemplo da impressão de cada letra de uma string, uma de cada vez.

```
for letra in "Olá Mundo":
    print(letra)
```



## Módulo 3.1 - Loops

💰 5000



Olá, desenvolvedores! O nosso conteúdo do dia englobará algumas das estruturas fundamentais do Python (e qualquer outra linguagem), os nossos loops. Esperamos que vocês possam tirar o máximo possível de proveito do conteúdo que estamos disponibilizando, não esqueçam de praticar bastante para fixar bem.

### for...in

O Python não remove e tem loops *for* no sentido tradicional. Ele tem apenas loops *for...in*, o qual chamaremos de loop *for*. O loop *for* do Python pega um iterável e retorna cada item dele, um por vez, até que todos os itens tenham sido iterados ou o loop tenha terminado. Ele funciona no estilo *iteração* e *iterável*, que veremos mais detalhadamente numa aula futura. No exemplo a seguir, `carrinho_compras` é uma lista de itens de compra. Dentro do loop *for*, você tem acesso à iteração atual do item a cada passagem pelo loop.

```
carrito_compras = ['Leite', 'Ovos', 'Presunto', 'Queijo', 'Pão', 'Café']
for item in carrito_compras:
    print(item)

>>> Leite
>>> Ovos
>>> Presunto
>>> ...
>>> Café
```

Aqui está um exemplo da impressão de cada letra de uma string, uma de cada vez.

```
for letra in "Olá Mundo":
    print(letra)
```



## Módulo 3.1 - Loops

💰 5000



Olá, desenvolvedores! O nosso conteúdo do dia englobará algumas das estruturas fundamentais do Python (e qualquer outra linguagem), os nossos loops. Esperamos que vocês possam tirar o máximo possível de proveito do conteúdo que estamos disponibilizando, não esqueçam de praticar bastante para fixar bem.

### for...in

O Python não remove e tem loops *for* no sentido tradicional. Ele tem apenas loops *for...in*, o qual chamaremos de loop *for*. O loop *for* do Python pega um iterável e retorna cada item dele, um por vez, até que todos os itens tenham sido iterados ou o loop tenha terminado. Ele funciona no estilo *iteração* e *iterável*, que veremos mais detalhadamente numa aula futura. No exemplo a seguir, `carrinho_compras` é uma lista de itens de compra. Dentro do loop *for*, você tem acesso à iteração atual do item a cada passagem pelo loop.

```
carrinho_compras = ['Leite', 'Ovos', 'Presunto', 'Queijo', 'Pão', 'Café']
for item in carrinho_compras:
    print(item)

>>> Leite
>>> Ovos
>>> Presunto
>>> ...
>>> Café
```

Aqui está um exemplo da impressão de cada letra de uma string, uma de cada vez.

```
for letra in "Olá Mundo":
    print(letra)
```