Para alterar a largura de um elemento, usamos a propriedade `width` .

index.html style.css Browser

One theory of the Earth

CONTINUAR

Para alterar a altura de um elemento, usamos a propriedade `height` .

index.html style.css Browser

```
<!doctype html>
<html>
 <head>
  <link rel="stylesheet"
href="style.css">
 </head>
 <body>
  <h3>One theory of the Earth</h3>
  <img src="https://mimo.app/r
/earth.png">
 </body>
</html>
```

</> CONTINUAR

Para alterar a altura de um elemento, usamos a propriedade `height` .

index.html style.css Browser

```
img {
  height: 100px;
}
```

CONTINUAR

Medimos a altura e a largura de um elemento em **pixels** , como `50px` . Pixels são pequenos pontos que compõem o que você vê na tela.

index.html style.css Browser

Two different theories

</>

CONTINUAR

O que acontece quando definimos `height` como `100px` nesta regra?

style.css index.html Browser

```
h1 {
 height: 100px;
 background-color: teal;
}
```

A propriedade `font-size` está definida como `100px`

Todos os elementos `h1` têm sua propriedade `height` definida como `100px`

</> CONTINUAR

O que acontece quando definimos `height` como `100px` nesta regra?

style.css index.html Browser

A propriedade `font-size` está definida como `100px`

Todos os elementos `h1` têm sua propriedade `height` definida como `100px`

</> CONTINUAR

Faça esta regra dobrar a largura da outra digitando `width` .

index.html style.css Browser

Easy Coffee Guide

7g for a single shot

14g for a double shot

</>

CONTINUAR

Para definir a cor, vamos adicionar `red` no final do valor da borda.

Podemos tornar essa borda vermelha definindo o valor da cor como `red` .

index.html style.css Browser

</> CONTINUAR

Para definir a cor, vamos adicionar `red` no final do valor da borda.

Podemos tornar essa borda vermelha definindo o valor da cor como `red` .

index.html style.css Browser

</> CONTINUAR

`border-radius` é uma propriedade que arredonda os cantos de um elemento. Se definirmos o raio como `10px` , a borda se curvará `10px` antes do canto.

index.html style.css Browser

Movie Depository

Curb Your Enthusiasm is an award winning show created by Larry David, the cocreator of the hit show Seinfeld

</>

CONTINUAR

maneira fácil de fazer com que as imagens tenham uma ótima aparência em uma página da web.

Para arredondar os cantos de todas as imagens usando `border-radius`, digite `100px;`.

index.html style.css Browser

Window Seat Digest

A collection of views from airplanes

</>

CONTINUAR

`border-radius` funciona em todos os elementos, até mesmo em imagens. É uma maneira fácil de fazer com que as imagens tenham uma ótima aparência em uma página da web.

Para arredondar os cantos de todas as imagens usando `border-radius` , digite `100px;` .

index.html style.css Browser

```
img {
 border-radius: 100px ;
 border: solid 5px black;
}
```

`border-radius` funciona em todos os elementos, até mesmo em imagens. É uma maneira fácil de fazer com que as imagens tenham uma ótima aparência em uma página da web.

Para arredondar os cantos de todas as imagens usando `border-radius` , digite `100px;` .

index.html style.css Browser

```
<!doctype html>
<html>
 <head>
  <link rel="stylesheet"
href="style.css">
 </head>
 <body>
  <h1>Window Seat Digest</h1>
  <h3>A collection of views from
airplanes</h3>
   <img src="https://mimo.app/r
/wing1.png">
   <img src="https://mimo.app/r
/wing2.png">
 </body>
</html>
```

</>

CONTINUAR

13 SEGUIDOS!

Faça uma borda visível ao redor de todos os elementos `img` .

index.html style.css Browser

```
<html>
 <head>
  <link rel="stylesheet"
href="style.css">
 </head>
 <body>
  <h1>The Frame Shop</h1>
  <h3>Minimalist frames around
your photos</h3>
  <img src="https://mimo.app/r
/plant.png">
 </body>
</html>
```

CONTINUAR

Arredonde as bordas de todos os elementos `img` .

index.html style.css Browser

BOM TRABALHO!

Usando o seletor `img` , defina a altura e a largura como `100px` . Em seguida, defina uma borda visível `2px` de espessura ao redor deles.

Toque no código para inserir a solução

index.html style.css Browser

```
<!doctype html>
<html>
 <head>
  <link rel="stylesheet"
href="style.css">
 </head>
 <body>
  <h2>My Favorite Things</h2>
  <img
src="https://mimo.app/r/cat.png">
  <img
src="https://mimo.app/r/coffee
.png">
 </body>
</html>
```

BOM TRABALHO!

Usando o seletor `img` , defina a altura e a largura como `100px` . Em seguida, defina uma borda visível `2px` de espessura ao redor deles.

Toque no código para inserir a solução

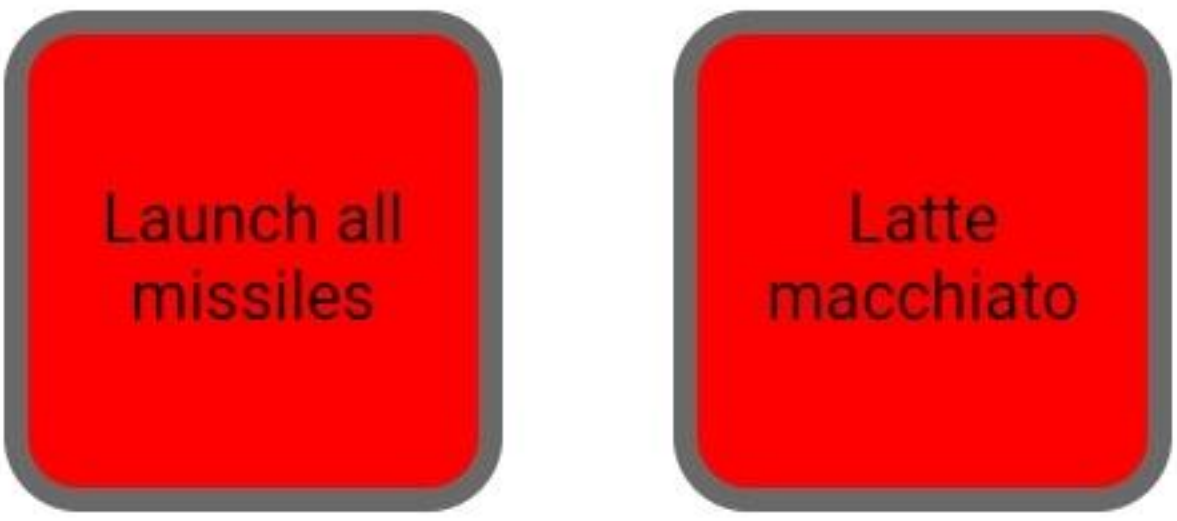
index.html style.css Browser

```
img{
height: 100px;
width: 100px;
border: solid 2px;
}
```

Para definir as margens direita e esquerda para `15px` em uma linha, podemos usar a abreviação de `margin` .

index.html style.css Browser

Command center



TotallyNotABadDesign Incorported

</> CONTINUAR

Para definir as margens direita e esquerda para `15px` em uma linha, podemos usar a abreviação de `margin` .

index.html style.css Browser

```
button {
  margin: 0 15px 0 15px;
  width: 100px;
  height: 100px;
  border-radius: 15px;
  background-color: red;
  border: solid 5px dimGray;
}

body {
  text-align: center;
}

p {
  margin: 30px 0 0 0;
  font-size: 8px;
}
```

</> CONTINUAR

Para definir as margens direita e esquerda para `15px` em uma linha, podemos usar a abreviação de `margin` .

index.html style.css Browser

```
<!doctype html>
<html>
 <head>
  <link rel="stylesheet"
href="style.css">
 </head>
 <body>
  <h3>Command center</h3>
  <button>Launch all missiles</
button>
  <button>Latte macchiato</button>
  <p>TotallyNotABadDesign
Incorported</p>
 </body>
</html>
```

</>

CONTINUAR

Qual é a ordem dos valores da propriedade `border-radius` ?

Superior esquerdo, superior direito, inferior direito, inferior esquerdo

Inferior direito, inferior esquerdo, superior esquerdo, superior direito

CONTINUAR

Defina o canto superior direito e o canto inferior esquerdo como `100px` e os outros cantos como `0` .

index.html style.css Browser

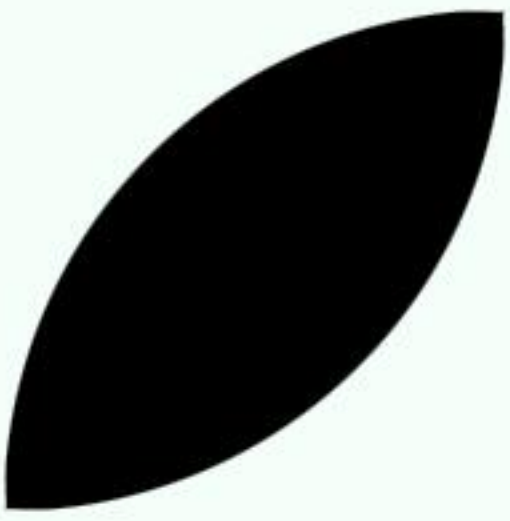
Bojack Horseman Quotes

"Nobody ever wants honeydew, but it's always there!"

64%

Defina o canto superior esquerdo como `100px` e o canto inferior direito como `100px` .

index.html style.css Browser



Palmie

Cosmetics Website

</> CONTINUAR

64%

Defina as margens esquerda e direita para `90px` .

index.html style.css Browser

Military General Ranking

General

Lieutenant General

Major General

</>

CONTINUAR

Defina as margens esquerda e direita para `90px` .

index.html style.css Browser

```
.general {
 margin-left: 90px;
 margin-right: 90px;
}

.lieutenantGeneral {
 margin-left: 30px;
 margin-right: 30px;
}

p {
 text-align: center;
 background-color: lightblue;
 border: 1px solid black;
}
```

</> CONTINUAR

5 SEGUIDOS!

Defina as margens laterais como `50px` , a margem superior como `30px` e a margem inferior como `20px` .

index.html style.css Browser

The London Thames Newspaper

CONTINUAR

Defina o espaçamento superior e inferior para `20px` e os preenchimentos esquerdo e direito para `30px` .

index.html style.css Browser

The Winning Shot

A recap of last minute winning shots throughout history.

</> CONTINUAR

3. Por fim, defina a `width` da sua imagem com um valor adequado para você. Por exemplo, `100px` .

index.html style.css Browser

```
<html>
  <head>
    <link href="style.css"
rel="stylesheet" >
  </head>
  <body>
    <img
src="https://mimo.app/i/emmy.png">
    <h1>Emmy</h1>
    <h2>Aspiring Software
Developer</h2>
    <p>Welcome to my page!</p>
    <a
href="https://twitter.com/getmimo"
target="_blank">Twitter</a>
    <a
href="https://www.tiktok.com/@mimo
_org" target="_blank">TikTok</a>
  </body>
</html>
```

3. Por fim, defina a `width` da sua imagem com um valor adequado para você. Por exemplo, `100px` .

index.html style.css Browser

```
img{
border-radius: 50px;
width: 100px;
}
```

Leaderboard

Liga de Prata

1d 19h 8m

		UhWhat	5.351
	A	Ashley	5.349
		Valiero	3.885
4		Jonas	2.675
5		Talia	2.400
6		Adriaen	2.028
7		Jefferson	1.960
8		Bert	1.922
9		Mude	1.886

Aprender | Leaderboard | Comunidade | Perfil

DESAFIO

Crie um programa de inventário simples para uma loja de camisas. O programa deve aumentar a variável `sales` em 1 e diminuir a variável `inventory` em 1 quando uma camisa for vendida.

1. Entre a inicialização da variável e as instruções de impressão, aumente o valor da variável `sales` em `1` usando um operador.
2. Diminua o valor da variável `inventory` em `1` usando um operador.

script.py

```
sales = 0
inventory = 10
sales += 1
inventory -= 1
print(f'Sales: {sales}')
print(f'Inventory: {inventory}')
```

Adicionamos um valor a um índice específico com `insert()`. Ao codificar `insert(0, "lemon")`, adicionaremos o valor ao início da lista.

script.py

```
shopping = ["kiwis", "peas"]
shopping.insert(0, "lemon")
print(shopping)
```

Adicione `"sugar"` ao final da lista codificando `append("sugar")`.

script.py

```
cookies = ["flour", "butter"]
cookies.append("sugar")
print(cookies)
```

Saída

```
['flour', 'butter', 'sugar']
```

CONTINUAR

Quantos valores por vez podemos adicionar a uma lista usando `.append()` e `.insert()`?

- Um valor de cada vez
- Quantos quisermos

CONTINUAR

Qual instrução usamos para remover o último elemento de uma lista?

`.insert()`

`.pop()`

CONTINUAR

Como podemos salvar o valor removido por `.pop()` ?

Não podemos salvá-lo, `.pop()` não nos devolve um valor

Armazenando-o em uma variável

CONTINUAR

Para onde vai um novo valor que adicionamos com `.append()` ?

No início da lista

No final da lista

ENVIAR

Qual instrução usamos para adicionar um valor a uma lista?

`.pop()`

`.append()`

ENVIAR

O valor removido também pode ser armazenado dentro de uma variável. Podemos ver isso aqui quando codificamos `print(removed)` .

script.py

```
todo = ["call mom", "dishes",
"painting"]
removed = todo.pop(0)
```

jan. 11 - jan. 18, 2024

Parabéns!

Você terminou #9 de 50 no Liga de Prata e avançou para Liga de Ouro.

COMPARTILHAR RESULTADO

CONTINUAR

Leaderboard

Liga de Ouro

6d 23h 35m

		Jefferson	310
		Kelvin Palma	310
		basile	310
4		Arielzito	280
5		Yuriy	250
6		motikuko2	220
7		Бобровник Екатерина	190
8		L	180

ZONA DE PERIGO

Perfil de usuário

PRO

Jefferson

I'm a apprentice , trying to be a better dev. My main thinking is the discipline = sucess.

3	23.284	Ouro
SEQUÊNCIA	XP	LIGA

6 dias — MELHOR SEQUÊNCIA

Certificados

- SQL ✓
- HTML ✓

Troféus da Leaderboard

Arielzito

Não sabemos nada sobre Arielzito

SEGUIR

1	31.013	Ouro
SEQUÊNCIA	XP	LIGA

66 dias MELHOR SEQUÊNCIA

Playgrounds

Não há Playgrounds no perfil de Arielzito

Para iterar através os elementos da lista `final_scores`, escrevemos `for`, uma variável como `score`, a palavra `in` e a lista `final_scores`

script.py

```
final_scores = [17, 22, 34, 13]

for score in final_scores:
  print(score)
```

O que esse código exibe no console?

script.py

```
consoles = ["Playstation", "Xbox"]

for console in consoles:
  print(console)
```

Nada é exibido no console

`"Playstation"` e `"Xbox"`

ENVIAR

O que acontece quando usamos `len()` em uma lista que não possui nenhum valor?

Causa um loop infinito

Ele retorna o valor `0`

CONTINUAR

O que há de errado com esse código?

script.py

```
signups = [100]

result = len()

print(result)
```

`len()` não está sendo usado com um valor de lista

Não há nada de errado com este código

</> CONTINUAR

Exiba o comprimento desta lista no console.

script.py

```
square_meters = [55, 67, 100]

print(len(square_meters))
```

Saída

```
3
```

O que há de errado com esse código?

script.py

```
singers = ["Bowie", "Freddie"]

for singer in singers:
  print(singer)
```

Saída

```
Bowie
Freddie
```

A lista `singers` não foi criada corretamente

A posição de `in` e `for` é confusa

Não há nada de errado com este código

</>

CONTINUAR

Complete a condição verificando se o comprimento dos `sodas` é maior que `3` .

script.py

```
sodas = ["coke", "fanta"]

if  len(sodas)  > 3:
    print("Too much soda!")
```

DESAFIO

para incluir o item atual do menu do jantar!

script.py

```
meals = ["omelet", "salad",
"chicken"]

print(f"Lunch menu: {meals[1]}")

meals.insert(2,"pizza")

print(f"Dinner menu: {meals[2]}")
```

Saída

```
Lunch menu: salad
Dinner menu: pizza
```

Todas verificações aprovadas **+ 20 XP**

Condição 1

Make sure to print 'Lunch menu: salad' to the console

Con

Ma
prin

CONTINUAR

1. Use uma operação de lista para substituir `"Iliana"` por `"Jack"` (`sub_1`).
2. Substitua `"Anders"` por `"Celeste"` (`sub_2`).
3. Substitua `"Gabrielle"` por `"Mary"` (`sub_3`).

Todas verificações aprovadas **+ 20 XP**

Condição 1

Make sure to sub 'Jack' in for 'Iliana'

Con

Ma

CONTINUAR

DESAFIO

```
shopping_list = ["dish soap",
"kleenex", "batteries", "aluminum
foil", "pet food", "toothpaste",
"lightbulbs"]

for item in shopping_list:
 print(f"Don't forget to buy
{item}")
```

Saída

```
Don't forget to buy dish soap
Don't forget to buy kleenex
Don't forget to buy batteries
Don't forget to buy aluminum foil
Don't forget to buy pet food
Don't forget to buy toothpaste
Don't forget to buy lightbulbs
```

Todas verificações aprovadas **+ 20 XP**

Condição 1

Make sure to print 'Don't forget to buy {item}!' for each item

Con

Ma
one
pro

CONTINUAR

Ao usar `sort()` em uma lista de strings como `names`, os elementos são classificados em ordem alfabética.

script.py

```
names = ["Cloe", "Ana", "Bill"]
names.sort()

print(names)
```

Saída

```
['Ana', 'Bill', 'Cloe']
```

</> CONTINUAR

O que `sort()` faz quando aplicado em uma lista de números?

Ele classifica os números em ordem decrescente

Ele classifica os números em ordem crescente

CONTINUAR

O que acontece quando classificamos uma lista contendo valores ponto flutuante e inteiros?

Os elementos da lista são classificados de acordo com seu valor numérico

Metade da lista conterá apenas números inteiros e a outra metade apenas flutuantes

ENVIAR

Como `sort()` classifica uma lista de strings?

Só podemos usar `sort()` em listas de números

Em ordem alfabética

CONTINUAR

Exiba `sold_tickets` , classifique a lista e exiba-a novamente para ver a diferença.

script.py

```
sold_tickets = [7, 20, 12, 30]
print(sold_tickets)
sold_tickets.sort()
print(sold_tickets)
```

Saída

```
[7, 20, 12, 30]
[7, 12, 20, 30]
```

CONTINUAR

Codifique `.sort()` para classificar os pontos de teste de um aluno em ordem crescente.

script.py

```
test_points = [97, 76, 81]
test_points .sort()

print(test_points)
```

Classifique esta lista de números negativos e positivos para pontos e penalidades que um usuário obteve em um jogo com `overview.sort()`.

script.py

```
overview = [3, -1, 4, -2]
overview.sort()

print(overview)
```

Saída

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

</> CONTINUAR

O que há de errado com esse código?

script.py

```
jeans_sizes = [25, 28, 40]
smallest_size = jeans_sizes.min()
```

- Não podemos salvar o menor valor de uma variável
- Deve ser `min(jeans_sizes)`
- Não há nada de errado com o código

ENVIAR

O que há de errado com esse código?

script.py

```
active_users = [103, 200, 140,
288]
sort(active_users)
```

Não há nada de errado com o código

Deve ser `active_users.sort()`

Só podemos usar `sort()` dentro de uma instrução `print()`

ENVIAR

O que há de errado com esse código?

script.py

```
cart = [50, 17, 20]
print(cart.sum())
```

Não há nada de errado com o código

Deve ser `sum(cart)`

Não podemos usar `sum()` , pois `cart` não é uma lista

Ao usar `+` , a segunda lista é anexada no final da primeira lista.

script.py

```
dataset_1 = [1, 2, 3]
dataset_2 = [4, 5]

print(dataset_1 + dataset_2)
```

Também podemos usar `+` para combinar listas contendo diferentes tipos de valores, como os `seats` numéricos e os booleanos `taken`.

script.py

```
seats = [1, 2, 3]
taken = [True, True, False]

print(seats + taken)
```

Saída

```
[1, 2, 3, True, True, False]
```

</> CONTINUAR

Codifique + para juntar as duas listas que armazenam a taxa de crescimento de uma planta.

script.py

```
growth_rate_1 = [0.5, 0.7, 0.8]
growth_rate_2 = [0.8, 0.6, 0.6]

print(growth_rate_1 +
growth_rate_2)
```

Para contar com que frequência um valor aparece em uma lista como `answers`, começamos com o nome da lista, um ponto `.` e depois `count()`.

script.py

```
answers = ["yes", "no",
"sometimes", "yes"]
answers . count()
```

Para contar com que frequência `"yes"` aparece na lista `answers`, codificamos `"yes"` entre os parênteses de `count()`.

script.py

```
answers = ["yes", "no",
"sometimes", "yes", "no"]
print(answers.count( "yes" ))
```

Podemos usar `count()` com qualquer tipo de valor, como verificar quantas vagas livres restam consecutivas pelo número de valores `True` .

script.py

```
free_seats = [False, False, True,
True, False]
print(free_seats.count( True ))
```

Saída

```
2
```

CONTINUAR

7 SEGUIDOS!

Verificar se um elemento como `"sugar"` está em uma lista como `"ingredients"` nos dá um booleano, `True` ou `False`.

script.py

```
ingredients = ["flour", "butter",
"sugar", "eggs"]

print( "sugar"  in  ingredients )
print("chocolate" in ingredients)
```

Saída

```
True
False
```

CONTINUAR

Como podemos reutilizar o resultado dado por `count()` ?

Criando uma variável e armazenando-a dentro

Não podemos reutilizar o resultado, pois ele sempre muda

ENVIAR

Descubra com que frequência a temperatura foi de 32 graus Fahrenheit codificando 32 entre parênteses.

script.py

```
temperatures = [40, 32, 32, 38]
print(temperatures.count( 32 ))
```

Saída

```
2
```

Verifique se o elemento `13` está `in` lista `winning_numbers` .

script.py

```
winning_numbers = [2, 36, 40, 13]

print( winning_numbers  in  13 )
```

Exiba `False` verificando se o elemento `0` faz parte da lista.

script.py

```
winning_numbers = [2, 36, 40, 13]

print( 0  in winning_numbers)
```

Saída

```
False
```

</> CONTINUAR

Codifique a variável `has_13`, que armazena o resultado da verificação se `13` está na lista `winning_numbers`.

script.py

```
winning_numbers = [2, 36, 40, 13]
has_13 = 13 in winning_numbers
print(has_13)
```

BOM TRABALHO!

Codifique um valor para contar quantas vezes o tamanho M foi comprado.

script.py

```
bought = ["L", "M", "S", "M", "M"]
print(bought.count( "M" ))
```

Saída

```
3
```

</>

CONTINUAR

DESAFIO

```
humidity = [77.78, 65.51, 74.42,
80.48, 71.71, 68.39, 65.10, 71.26,
75.95, 81.29, 71.54, 80.85, 84.45,
84.23, 83.18, 68.59, 80.83, 84.72,
78.23, 74.44, 83.20, 74.90, 80.59,
75.09, 82.46, 67.96, 77.65, 69.51,
74.57, 72.98, 66.01, 67.89, 82.52,
65.15, 70.07, 66.84, 74.74, 82.16,
73.73, 82.23, 66.47, 68.70, 71.28,
67.42, 82.87, 66.41, 80.43, 83.42,
74.70, 83.97]

min_humidity = min(humidity)
max_humidity = max(humidity)
```

Saída

```
65.1
84.72
```

Todas verificações aprovadas **+ 20 XP**

st

Condição 3

Make sure to print the correct messages.

CONTINUAR

DESAFIO

Temos um torneio mensal e registramos o nome do campeão na lista `champions`. Recentemente, descobrimos que um participante regular `"Tooti3"` estava trapaceando. Temos que verificar se `"Tooti3"` ganhou algum torneio e removê-lo da lista. Descubra quantas vezes `"Tooti3"` aparece na lista `champions` usando `count()` e imprima o resultado no console.

script.py

```
champions = ["Miracle+", "Tooti3",
"Orustat", "Emkay", "mizuhana",
"CaptainSpark", "NichMercs",
"mizuhana", "dabian", "Cyle",
"Tooti3", "Flaker"]
```

Armazenamos os valores em `new_users` como elementos individuais dentro de `users_list`. Podemos exibi-los codificando `print(users_list)`.

script.py

```
new_users = "Ann Jon Alex"

users_list = new_users.split()

print ( users_list )
```

Saída

```
['Ann', 'Jon', 'Alex']
```

</> CONTINUAR

Podemos especificar exatamente como queremos dividir uma string colocando um separador entre parênteses, como acontece com `" , "` aqui.

ⓘ Toque no código para inserir a solução

script.py

```
user = "Lauren,25,F,Architect"

user_1 = user.split( "," )

print(user_1)
```

Saída

```
['Lauren', '25', 'F', 'Architect']
```

Em vez de criar uma nova variável, podemos reatribuir a variável original à string atualizada codificando `special` .

script.py

```
special = "Today's special is
pasta"

special =
special.replace("pasta","pizza")

print(special)
```

Saída

```
Today's special is pizza
```

O que esse código exibe no console?

script.py

```
answer = "The answer is Belgium"

updated = answer.replace("Belgium"
,"Finland")

print(updated)
```

Saída

```
The answer is Finland
```

A resposta é Bélgica

A resposta é Finlândia

</>

CONTINUAR

Atualize a string chamando `replace()` para corrigir o erro de capitalização

script.py

```
answer = "The answer is INcorrect"

updated = answer . replace ("INcorrect", "incorrect" )

print(updated)
```

Saída

```
The answer is incorrect
```

Reatribua a variável `monthly` ao valor atualizado e exiba-a no console.

script.py

```
monthly = "Monthly reduction is
25%"

monthly = monthly.replace("25%","15%")

print(monthly)
```

Divida a string `signups` em `-` .

script.py

```
signups = "44-22-44-11"

signups_list = signups . split
( "_" )
```

```
print(signups_list)
```

Saída

```
['44', '22', '44', '11']
```

</> CONTINUAR

Exiba a variável `path_list` .

script.py

```
file_path = "documents/code/
example"

path_list = file_path.split("/")

print(path_list)
```

Saída

```
['documents', 'code', 'example']
```

CONTINUAR

Qual é o separador padrão ao usar `split()` ?

- Um periodo
- Um espaço em branco
- Um traço

CONTINUAR

O que esse código exibe no console?

script.py

```
usernames = "aut_43 tommo rodger"

usernames_list = usernames.split()

print(usernames_list)
```

Saída

```
['aut_43', 'tommo', 'rodger']
```

`aut_43 tommo rodger`

Nada

`['aut_43', 'tommo', 'rodger']`

</>

CONTINUAR

O que `split()` faz?

- Ele se junta a uma string
- Ele substitui parte de uma string
- Ele divide uma string em uma lista

CONTINUAR

Reatribua a variável original ao valor atualizado.

script.py

```
yearly = "Discount increased to
2%"

 yearly  =
yearly.replace("2%","4%")

print(yearly)
```

Saída

```
Discount increased to 4%
```

Atualize a variável `release_date` .

script.py

```
release_date = "Release date: 24th July"

updated = release_date.replace("24th July","31st August")

print(updated)
```

Saída

```
Release date: 31st August
```

</> CONTINUAR

BOM TRABALHO!

O que esse código exibe no console?

script.py

```
answer = "Wales + Ireland"

updated =
answer.replace("+","and")

print(updated)
```

Saída

```
Wales and Ireland
```

- `Wales and Ireland`
- `WalesandIreland`
- `Wales + Ireland`

</> CONTINUAR

DESAFIO

Um professor deseja criar uma lista com os nomes de todos os alunos de sua turma. Ele recebeu uma longa string contendo todos os nomes, cada nome separado por vírgula `,`

Ajude-o a criar uma `names_list` que armazene os nomes individuais separados por vírgula. Em seguida, imprima a lista.

Toque no código para inserir a solução

script.py

```
student_names =
"Samantha,Mcgrath,Peyton,Kerim,Nad
ia,Sandra,Sarah,Alex"
```

DESAFIO

Você trabalha como desenvolvedor de software e decidiu usar um conjunto específico de tecnologias de programação para sua próxima aplicação. Em uma solicitação de última hora do cliente, você concordou em usar React em vez de Angular.

1. Atribua novamente `tech_stack` e use uma operação de string para substituir "Angular" por "React".
2. Crie uma variável `tech_stack_list` que armazene os nomes de `tech_stack` como uma lista.
3. Imprimir `tech_stack_list`

script.py

```
tech_stack = "Angular Node Mongo
Express"
```

Quando escrevemos programas maiores, muitas vezes temos que reescrever códigos que executam operações específicas.

Por exemplo, digamos que para um jogo você precise reescrever o código que faz um personagem pular. Geralmente, usamos funções para reutilizar o código.

CONTINUAR

Módulos são arquivos que contêm variáveis, funções, etc. Aqui, criamos `mod.js` como um módulo. Você pode ter quantos módulos quiser.

mod.js script.js

```
const pi = 3.14;
```

Para usar variáveis de um módulo, temos que exportá-las usando a palavra-chave `export` e especificar `default` para exportar apenas um objeto.

mod.js script.js

```
const pi = 3.14;

export default pi;
```

Saída

```
3.14
```

</> CONTINUAR

Para usar uma variável de um módulo em um arquivo diferente, precisamos primeiro importá-la usando `import from` . Aqui, importaremos `pi` de `mod.js` .

mod.js script.js

```
import pi from "./mod.js";
```

6 SEGUIDOS!

Agora que importamos a variável, podemos usá-la. Vamos exibir o valor de `pi` no console.

script.js mod.js

```
const pi = 3.14;
export default pi;
```

Saída

```
3.14
```

</> CONTINUAR

O que há de errado com a instrução `import` fornecida?

script.js

```
import var1;
```

`import` deve vir depois `var1`

Não especifica o nome do arquivo

CONTINUAR

13 SEGUIDOS!

Importe `pi` de `mod.js` .

mod.js script.js

```
import pi from "./mod.js";
```

Saída

```
(node:6) Warning: To load an ES
module, set "type": "module" in
the package.json or use the .mjs
extension.
/home/default/code/file.js:3
export default pi;
^^^^^^

SyntaxError: Unexpected token 'export'
    at wrapSafe (internal/modules/cjs/
loader.js:915:16)
    at Module._compile (internal/
modules/cjs/loader.js:963:27)
    at Object.Module._extensions..js
(internal/modules/cjs/
loader.js:1027:10)
```

</>

CONTINUAR

Os módulos não estão restritos apenas a variáveis. Também podemos importar funções de módulos para usar em um arquivo diferente.

CONTINUAR

Para comparar se dois números são iguais, usamos o **operador de igualdade** , `===` .

script.js

```
5 === 5
```

Ao comparar, existem apenas dois resultados: `true` ou `false`.

CONTINUAR

Para verificar se um número não é igual a outro número, usamos o **operador de desigualdade** , `!==` .

script.js

```
console.log(1 !== 10);
```

6 SEGUIDOS!

O que esse código exibe no console?

script.js

```
let income = 1400;
let savings = 900;
savings = 1000;
income = 1500;

console.log(savings);
```

Saída

1000

- 1000
- 900
- 1500

</>

CONTINUAR

Leaderboard

Liga de Ouro 2d 23h 42m

		Yuriy	7.886
		Jefferson	3.450
	W	WikingProgramowania	3.102
4	A	Aaron Goldstein	2.670
5		Charlie	2.572
6		Virgil	2.520
7		Alex	2.340
8		UhWhat	2.321
9		Adel Ghoualem	2.010

Leaderboard

Liga de Ouro 2d 23h 42m

		Yuriy	7.886
		Jefferson	3.450
	W	WikingProgramowania	3.102
4	A	Aaron Goldstein	2.670
5		Charlie	2.572
6		Virgil	2.520
7		Alex	2.340
8		UhWhat	2.321
9		Adel Ghoualem	2.010

Tipos e comparações

0%

Desafio de programação

Condicionais

Codifique condicionais para construir programas que tomam decisões

Se o número à esquerda não for menor que o número à direita, como em `235 < 1`, o resultado será `false`.

script.js

```
console.log(235 < 1);
```

Saída

```
false
```

O que esse código exibe no console?

script.js

```
console.log(1 > 1);
```

Saída

```
false
```

- `false`
- `true`

</> CONTINUAR

Para verificar se um número é maior ou igual a outro número, usamos o **operador maior ou igual a** , `>=` .

script.js

```
console.log(3099 >= 3099);
```

Saída

```
true
```

</> CONTINUAR

Verifique se `batteryLevel` é menor ou igual a `20` com o operador menor ou igual a, `<=` .

script.js

```
const batteryLevel = 10;
const low = batteryLevel <= 20;
console.log("Low battery: " +
low);
```

Armazene o resultado de `points >= 10` na variável `levelTwo` .

script.js

```
const points = 12;
const levelTwo = points >= 10;
console.log("Level 2: " +
levelTwo);
```

Saída

```
Level 2: true
```

Para verificar se uma string é igual a outra string, podemos usar o **operador de igualdade estrita** , `===` .

script.js

```
console.log("apple" === "apple");
```

CONTINUAR

Armazene uma string no `wallpaper` para exibir `false` no console.

script.js

```
const wallpaper = "bliss.png";
console.log(wallpaper !==
"bliss.png");
```

Saída

```
false
```

</> CONTINUAR

BOM TRABALHO!

Verifique se `bigCity` não é igual a `smallCity` codificando `!==` .

script.js

```
const bigCity = "Tokyo";
const smallCity = "Zurich";
console.log(bigCity !==
smallCity);
```

Saída

```
true
```

</> CONTINUAR

Já vimos e usamos muitos tipos diferentes de valores. Na linguagem de programação, esses diferentes tipos de valores são chamados **de tipos** .

CONTINUAR

6 SEGUIDOS!

Como são chamados valores como booleanos, strings e números?

Variáveis

Tipos

CONTINUAR

Atribua uma **string** à variável `userName`, um **número** a `age` e um **booleano** a `isMale`.

script.js

```
const userName = "Joey";
const age = 28;
const isMale = true;
console.log(userName, age,
isMale);
```

null

BOM TRABALHO!

Atribua o valor `57` à variável `constscore`.

script.js

```
const score = 57;
console.log(score);
```

Saída

```
57
```

</>

CONTINUAR

Combine uma string e um número para exibir a pontuação de um jogador.

script.js

```
const result = "Your score: " +
99;
console.log(result);
```

Saída

```
Your score: 99
```

</> CONTINUAR

Também podemos juntar strings e booleanos.

Combine uma string e um booleano para exibir o resultado de um aluno. Eles passaram?

script.js

```
const result = "You passed: " +
true;
console.log(result);
```

Saída

```
You passed: true
```

Que tipo é o resultado da união de uma string com um número?

número

string

CONTINUAR

Qual é o resultado da união de uma string e um booleano?

Um booleano

Uma string

CONTINUAR

Adicione um valor de string ao valor numérico e verifique o resultado da string .

script.js

```
console.log(10 + "10");
```

Saída

```
1010
```

CONTINUAR

Codifique o valor da string e adicione a variável numérica.

script.js

```
const fee = 8;
console.log( "Entry fee: "  +
fee );
```

Saída

```
Entry fee: 8
```

BOM TRABALHO!

Adicione a variável booleana `isWinner` à string.

script.js

```
const isWinner = true;
console.log("You won: " +
isWinner );
```

Saída

```
You won: true
```

CONTINUAR

Com operadores lógicos, podemos conectar duas ou mais condições para decidir um resultado.

Ligue a lâmpada conectando o interruptor e a bateria.

Light the bulb

Bulb is **off**

Connect

On

Is battery connected?
No.

Is switch on?
No.

CONTINUAR

Com operadores lógicos, podemos conectar duas ou mais condições para decidir um resultado.

Ligue a lâmpada conectando o interruptor e a bateria.

Light the bulb

Bulb is **off**

Disconnect

Is battery connected?
Yes.

On

Is switch on?
No.

CONTINUAR

Com operadores lógicos, podemos conectar duas ou mais condições para decidir um resultado.

Ligue a lâmpada conectando o interruptor e a bateria.

Light the bulb

Bulb is **on**

Disconnect

Is battery connected?
Yes.

Off

Is switch on?
Yes.

CONTINUAR

Com operadores lógicos, podemos conectar duas ou mais condições para decidir um resultado.

Ligue a lâmpada conectando o interruptor e a bateria.

Light the bulb

Bulb is **on**

Disconnect

Is battery connected?
Yes.

Off

Is switch on?
Yes.

CONTINUAR

Um operador lógico conecta as duas condições da bateria e do interruptor para decidir se a lâmpada deve ser ligada ou desligada.

Light the bulb

Bulb is **off**

Disconnect

On

Is battery connected?
Yes.

Is switch on?
No.

CONTINUAR

O operador **AND**&& retorna `true` somente se todas as condições forem `true` .

script.js

```
let isBatteryOn = true;
let isSwitchOn = true;
console.log(isBatteryOn && isSwitchOn);
```

Saída

```
true
```

</> CONTINUAR

Quando operandos e operadores computam um valor booleano juntos, ele forma uma expressão lógica, como `isBatteryOn && isSwitchOn`.

script.js

```
let isBatteryOn = false;
let isSwitchOn = false;
console.log( isBatteryOn  &&
isSwitchOn );
```

Saída

```
false
```

CONTINUAR

Você também pode armazenar o resultado de uma expressão lógica em uma variável.

script.js

```
let isBatteryOn = false;
let isSwitchOn = false;
let result = isBatteryOn &&
isSwitchOn;
console.log(result);
```

Saída

```
false
```

CONTINUAR

5 SEGUIDOS!

O que esse código exibe no console para a variável `result` ? `true` ou `false` ? Percorra o código passo a passo para descobrir.

script.js

```
let cost = 50;
let sellPrice = 60;
const profit = sellPrice - cost;
console.log(profit);
const result = cost < sellPrice &&
profit > 0;
console.log("result: " + result);
```

Saída

```
10
result: true
```

`false`

`true`

</>

CONTINUAR

6 SEGUIDOS!

Verifique se uma carteira de motorista pode ser emitida para uma pessoa. A idade deve ser igual ou superior a 18 anos e devem passar no exame de direção.

script.js

```
let age = 18;
let isPass = true;
const isEligible = age >= 18 &&
isPass;
console.log(isEligible);
```

Saída

```
true
```

CONTINUAR

Verifique se o usuário pode enviar um e-mail. Para isso, o destinatário deverá ser válido e o assunto preenchido.

script.js

```
let isRecipientValid = true;
let isSubjectFilled = false;
const isMailSent =
isRecipientValid &&
isSubjectFilled;
console.log("Mail Sending
Successful?: " + isMailSent);
```

Saída

```
Mail Sending Successful?: false
```

</> CONTINUAR

BOM TRABALHO!

Armazene uma expressão lógica na variável `willSiteLoad` que verifica se o WiFi está conectado e a URL é válida para carregar um site.

script.js

```
let isWifiConnected = true;
let isURLValid = true;
const willSiteLoad =
isWifiConnected && isURLValid;
console.log(willSiteLoad);
```

Saída

```
true
```

CONTINUAR

Vimos que o operador AND exige que todas as condições sejam verdadeiras. Mas e se for suficiente que apenas uma condição seja verdadeira?

Light the bulb

Bulb is **on**

Disconnect

Is battery connected? **Yes.**

On

Is power connected? **No.**

CONTINUAR

Vimos que o operador AND exige que todas as condições sejam verdadeiras. Mas e se for suficiente que apenas uma condição seja verdadeira?

Light the bulb

Bulb is **on**

Connect

Off

Is battery connected? **No.**

Is power connected? **Yes.**

CONTINUAR

Por exemplo, a lâmpada aqui acende se a bateria estiver conectada *ou* a energia estiver conectada.

Light the bulb

Bulb is **on**

Disconnect

Is battery connected? **Yes.**

On

Is power connected? **No.**

CONTINUAR

Por exemplo, a lâmpada aqui acende se a bateria estiver conectada *ou* a energia estiver conectada.

Light the bulb

Bulb is **on**

Connect

Is battery connected? **No.**

Off

Is power connected? **Yes.**

CONTINUAR

Por exemplo, a lâmpada aqui acende se a bateria estiver conectada *ou* a energia estiver conectada.

Light the bulb

Bulb is **on**

Disconnect

Is battery connected? **Yes.**

Off

Is power connected? **Yes.**

CONTINUAR

Podemos usar `!` para negar expressões lógicas também. Para fazer isso, colocamos a expressão lógica entre parênteses.

script.js

```
let isBatteryOn = true;
let isPowerOn = false;

console.log( ! ( isBatteryOn &&
isPowerOn ) );
```

Verifique se uma pessoa pode comprar um carro dependendo se o empréstimo foi aprovado *ou* se o valor em dinheiro é maior que o custo do carro.

script.js

```
let isLoanApproved = true;
let cash = 30000;
let cost = 40000;
const isCarBought = isLoanApproved
   (cash    cost);
console.log(isCarBought);
```

Verifique se uma pessoa pode comprar um carro dependendo se o empréstimo foi aprovado *ou* se o valor em dinheiro é maior que o custo do carro.

script.js

```
let isLoanApproved = true;
let cash = 30000;
let cost = 40000;
const isCarBought = isLoanApproved
&& (cash > cost);
console.log(isCarBought);
```

Verifique se uma pessoa pode comprar um carro ou não.

script.js

```
let loanRejected = false;
let cash = 30000;
let cost = 40000;
const insufficientFunds = cash <
cost;
const canBuyCar = !(loanRejected
&& insufficientFunds);
console.log(canBuyCar);
```

Saída

```
true
```

CONTINUAR

Salve um arquivo se o armazenamento na nuvem for maior que o tamanho do arquivo *ou* se o armazenamento do sistema for maior que o tamanho do arquivo.

script.js

```
let cloudStorage = 4;
let systemStorage = 8;
let fileSize = 5;
const result = (cloudStorage |
fileSize) |  (systemStorage >
fileSize);
console.log("Saved? : " + result);
```

&& | | >

BOM TRABALHO!

Salve um arquivo se o armazenamento na nuvem for maior que o tamanho do arquivo *ou* se o armazenamento do sistema for maior que o tamanho do arquivo.

script.js

```
let cloudStorage = 4;
let systemStorage = 8;
let fileSize = 5;
const result = (cloudStorage > fileSize) || (systemStorage > fileSize);
console.log("Saved? : " + result);
```

Saída

```
Saved? : true
```

CONTINUAR

Que tipo tem `message` ?

script.js

```
const isBooked = true;
const message = "Ticket booked: "
+ isBooked;
console.log(message);
```

Saída

```
Ticket booked: true
```

- Número
- String
- boleano

</> CONTINUAR

Salve o valor `"1001"` na variável `result`.

script.js

```
const result = "100" + 1;
console.log(result);
```

Saída

```
1001
```

O que esse código exibe no console?

script.js

```
console.log("Repeat song: " +
true);
```

Saída

```
Repeat song: true
```

O booleano `true`

A string `"Repeat song: "`

A string `"Repeat song: true"`

</>

CONTINUAR

12 SEGUIDOS!

Os usuários podem fazer login se *não* falharem no teste do bot. Use um operador para verificar se o usuário consegue fazer login.

script.js

```
let failedBotTest = false;
const loggedIn = ! failedBotTest;
console.log("Logged in? : " +
loggedIn);
```

Saída

```
Logged in? : true
```

</> CONTINUAR

Verifique se o estoque está disponível ou não.

script.js

```
let isStockEmpty = false;
const isAvailable = !
isStockEmpty;
console.log(isAvailable);
```

DESAFIO

Crie uma variável `const` chamada `isPasswordCorrect` para verificar se a senha inserida pelo usuário é a mesma que o sistema armazenou anteriormente.

1. Inicialize uma variável `const` `isPasswordCorrect` para verificar se `userPassword` é exatamente igual a `savedPassword`.

Saída:

```
Access granted: true
```

script.js

```
const userPassword = "283746";

const savedPassword = "283746";



console.log("Access granted: " +
 isPasswordCorrect);
```

Crie uma variável `const` chamada `isPasswordCorrect` para verificar se a senha inserida pelo usuário é a mesma que o sistema armazenou anteriormente.

1. Inicialize uma variável `const` `isPasswordCorrect` para verificar se `userPassword` é exatamente igual a `savedPassword` .

Saída:

```
Access granted: true
```

Saída

Access granted: true

Todas verificações aprovadas **+ 20 XP**

Condição 2

Make sure to not change the string values of the variables `savedPassword` and `userPassword`.

CONTINUAR

script.js

```
const userPassword = "283746";

const savedPassword = "283746";

const isPasswordCorrect =
 userPassword === savedPassword;


console.log("Access granted: " +
 isPasswordCorrect);
```

Saída

```
Access granted: true
```

Todas verificações aprovadas **+ 20 XP**

Condição 2

Make sure to not change the string values of the variables `savedPassword` and `userPassword`.

CONTINUAR

`Drinking Age: true` se o cliente tiver idade legal para beber e `Of Legal Drinking Age: false` se o cliente for menor de idade. Use a variável `ageCheck` .

script.js

```
const age = 23;

const ageCheck = age >= 21;

console.log(
 "Of Legal Drinking Age: " +
ageCheck
 );
```

Saída

```
Of Legal Drinking Age: true
```

Todas verificações aprovadas **+ 20 XP**

Condição 6

Make sure that the output in the console is correct.

CONTINUAR

1. Na primeira linha, inicialize uma variável `const age` . Dê a ele o valor numérico de `23` .
2. Em seguida, crie outra variável `ageCheck` que verifica se a variável `age` é maior ou igual à idade legal para beber. A idade legal para beber é `21` .
3. Por último, modifique o `console.log` para que imprima `Of Legal Drinking Age: true` se o cliente tiver idade legal para beber e `Of Legal Drinking Age: false` se o cliente for menor de idade. Use a variável `ageCheck` .

script.js

Saída

```
Of Legal Drinking Age: true
```

Todas verificações aprovadas **+ 20 XP**

Condição 6

Make sure that the output in the console is correct.

CONTINUAR

DESAFIO

Você deseja verificar se tem dinheiro suficiente para comprar novos fones de ouvido.

1. Complete a variável `hasEnoughMoney` para que ela armazene o resultado booleano de se `price` é menor *ou* igual a `budget` .

Saída:

```
I have enough money:
false
```

script.js

```
const budget = 23;

const price = 80;

const hasEnoughMoney = price
budget;

console.log("I have enough
money:");
```

Desafio de programação

Condicionais

Codifique condicionais para construir programas que tomam decisões

IR PARA O PRÓXIMO

145 3

JavaScript

145 3

Python

145 3

Python

145 3

Python

145 3

Python

145 3

Python

145 3

Python

145 3

Python

Programas como o abaixo precisam executar regularmente a mesma tarefa continuamente.

Increase volume

CONTINUAR

Programas como o abaixo precisam executar regularmente a mesma tarefa continuamente.

Increase volume

Volume increased!

CONTINUAR

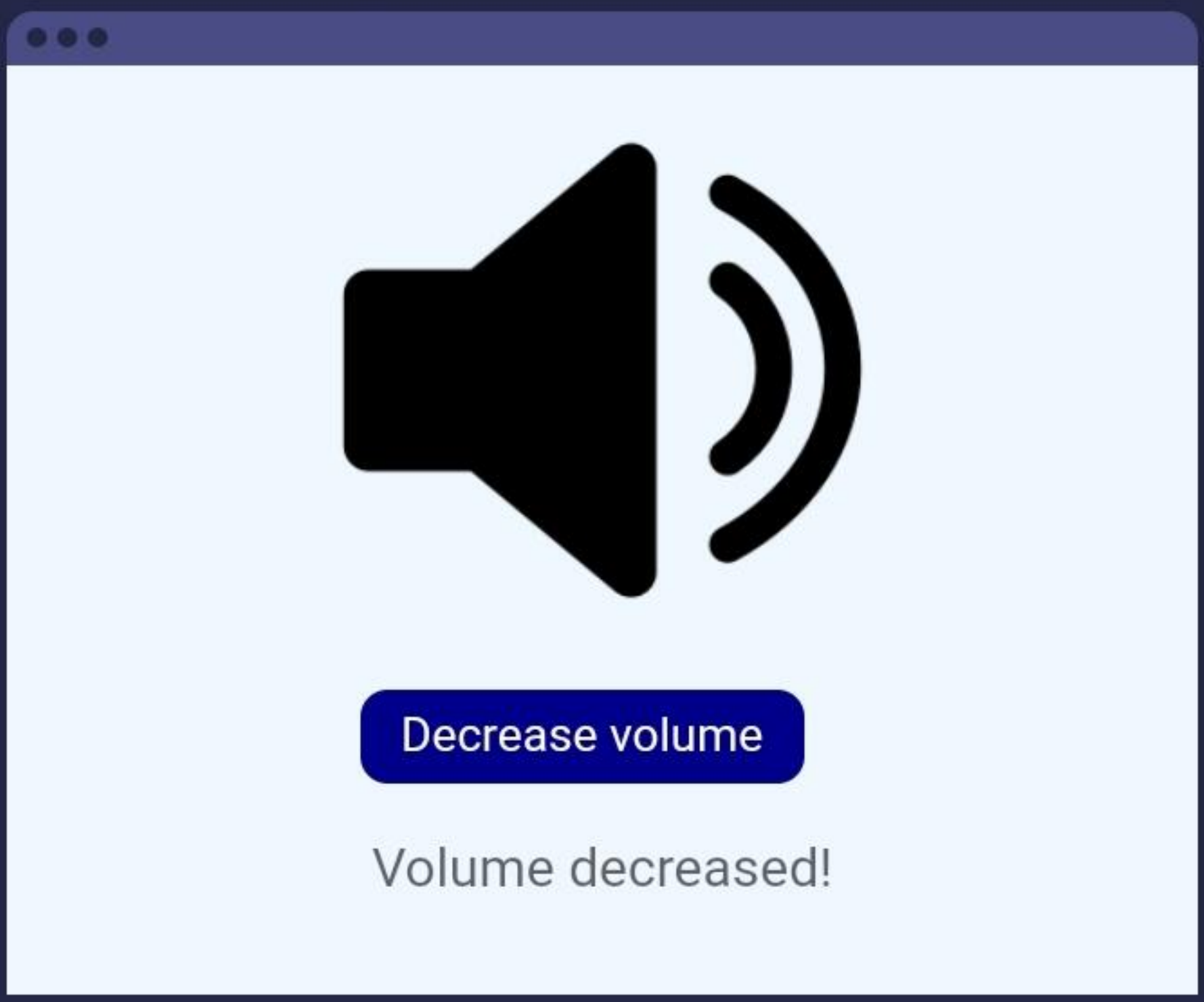
Programas como o abaixo precisam executar regularmente a mesma tarefa continuamente.

Increase volume

Volume increased!

CONTINUAR

Em vez de reescrever o mesmo código, podemos usar **funções** para agrupar códigos relacionados e executar a tarefa em um só lugar.

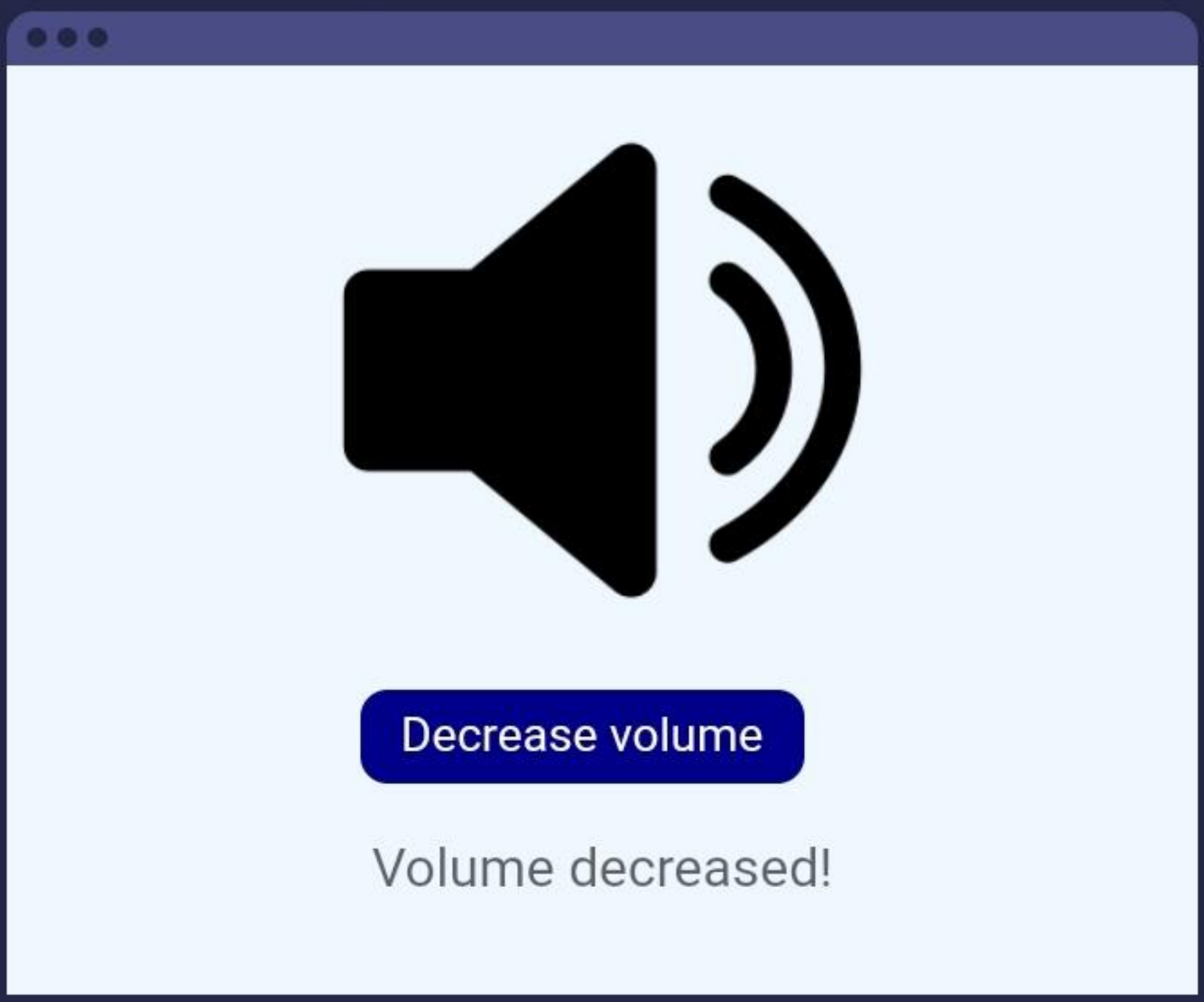


Decrease volume

Volume decreased!

CONTINUAR

Em vez de reescrever o mesmo código, podemos usar **funções** para agrupar códigos relacionados e executar a tarefa em um só lugar.



Decrease volume

Volume decreased!

CONTINUAR

Como nomeamos uma função?

Incluindo espaços no nome da função

Usando o snake case

CONTINUAR

11 SEGUIDOS!

Para onde vai o código que queremos agrupar?

Após a definição da função, recuado por dois espaços

Entre parênteses `(` e `)`

CONTINUAR

Dê um nome a esta função em snake case.

script.py

```
def night_routine():
  print("Lights off")
  print("Alarm set")
night_routine()
```

nightroutine

15 SEGUIDOS!

Adicione dois pontos para marcar o início do bloco de código.

script.py

```
def morning_routine() :
  print("Lights on")
  print("Alarm off")
morning_routine()
```

Saída

```
Lights on
Alarm off
```

CONTINUAR