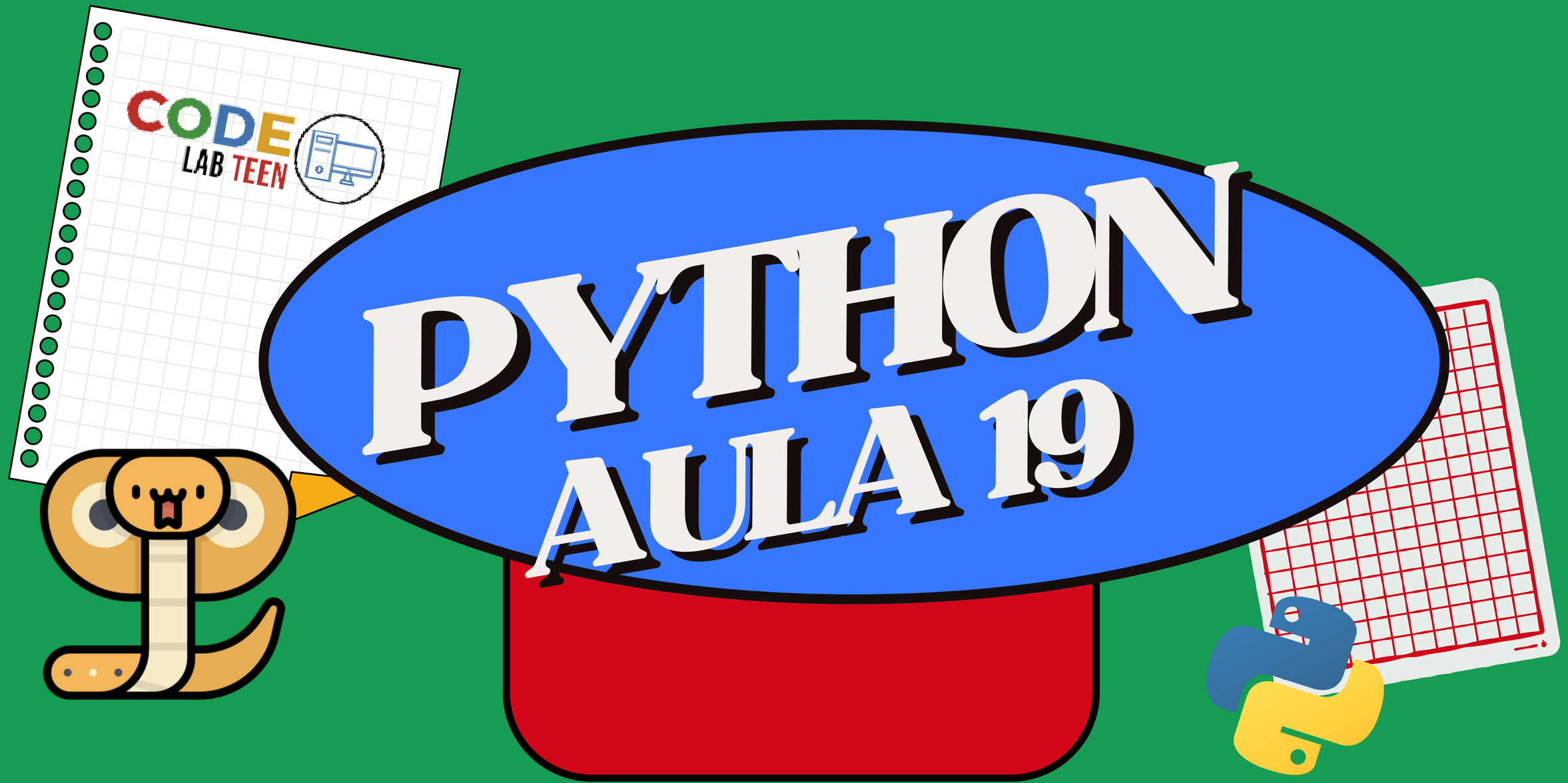

CODE
LAB TEEN
PYTHON
AULA 19

BEM - VINDOS!

AGENDA

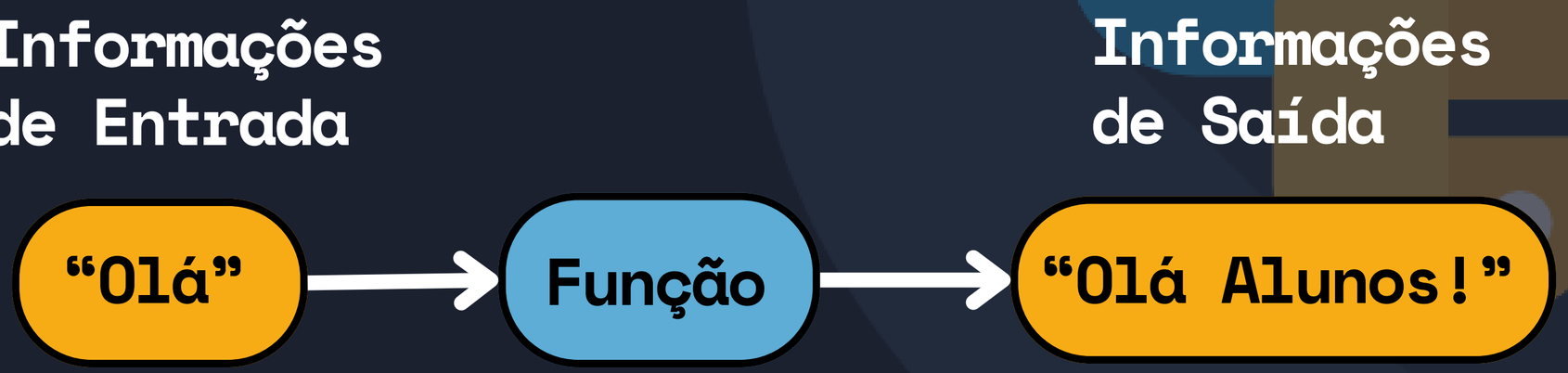
Hoje vamos revisar as estruturas de funções e alguns de seus exemplos.

FUNÇÕES - REVISÃO

Uma função é um tipo de recurso que pega informações e devolve para nós uma outra informação.

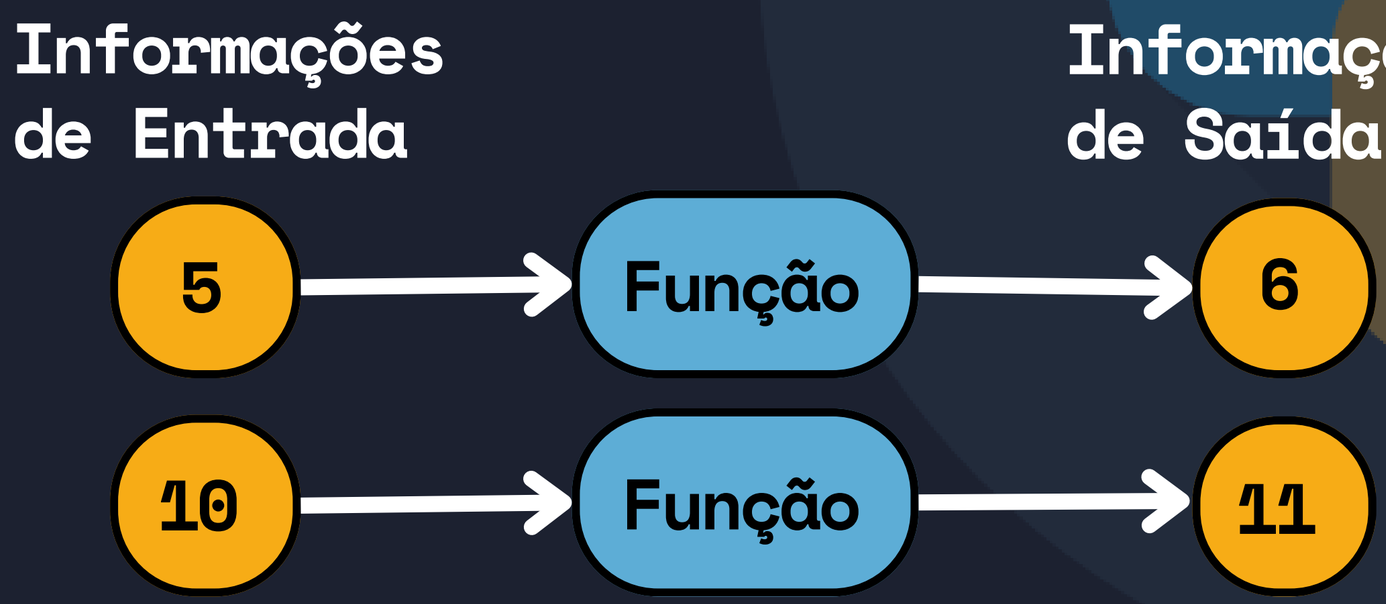
FUNÇÕES - REVISÃO

Note que, a função apresentada pega uma palavra QUALQUER e adiciona “ Alunos!” ao final. Logo, se colocarmos uma outra entrada...



FUNÇÕES - REVISÃO

Uma função também pode fazer contas. Dessa forma, podemos fazer uma função que soma o número um a qualquer outro número dado.

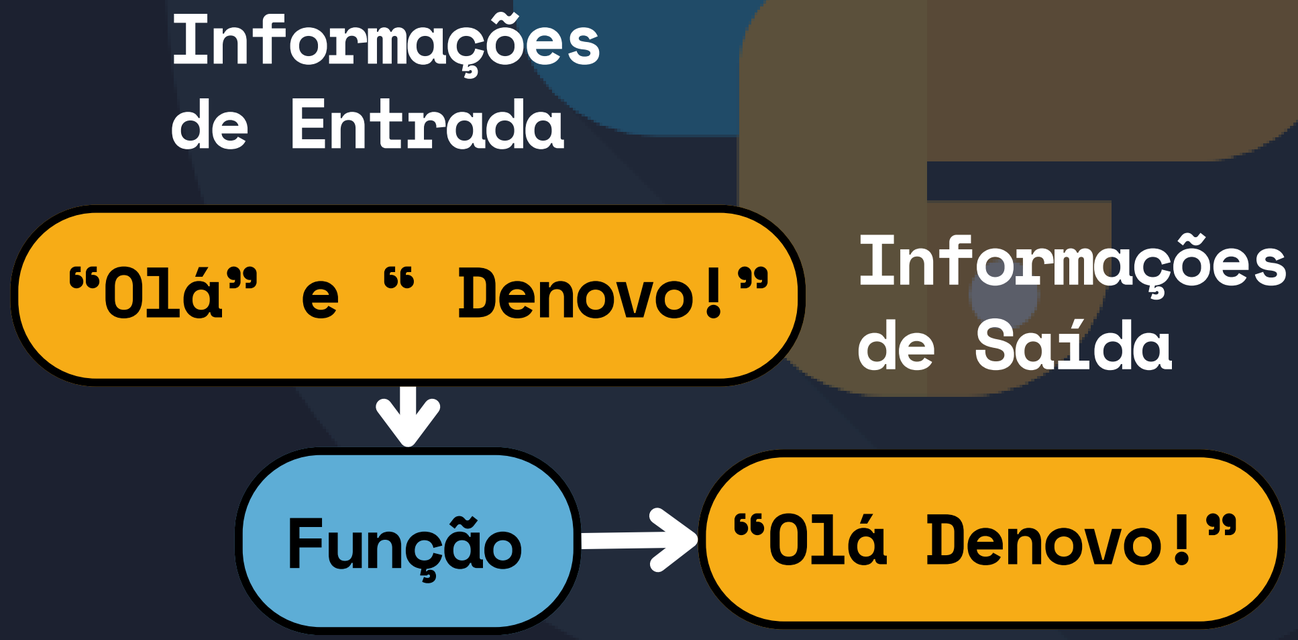


CODE LAB TEEN

FUNÇÕES - REVISÃO

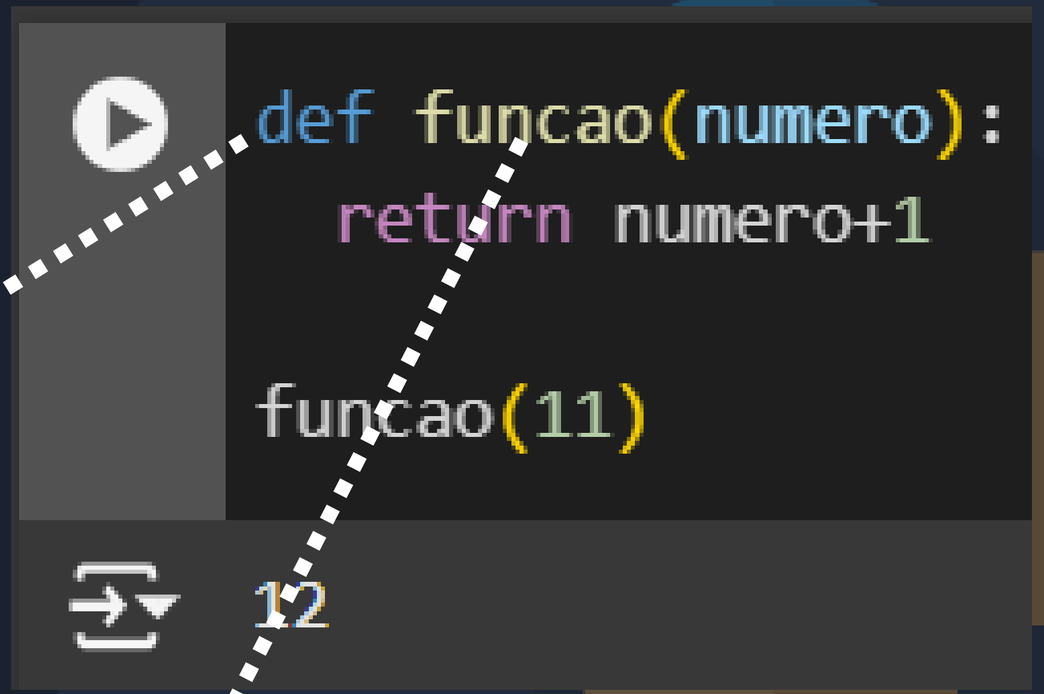
Em programação, uma função sempre recebe parâmetros (informações de entrada) e devolve para nós algum tipo de resultado (informações de saída).

OBS: Podemos passar mais de um parâmetro (informações de Entrada) para a função.



FUNÇÕES - REVISÃO

Na linguagem de programação Python, as funções tem uma estrutura especial.



def: diz para o computador que estamos iniciando uma função.

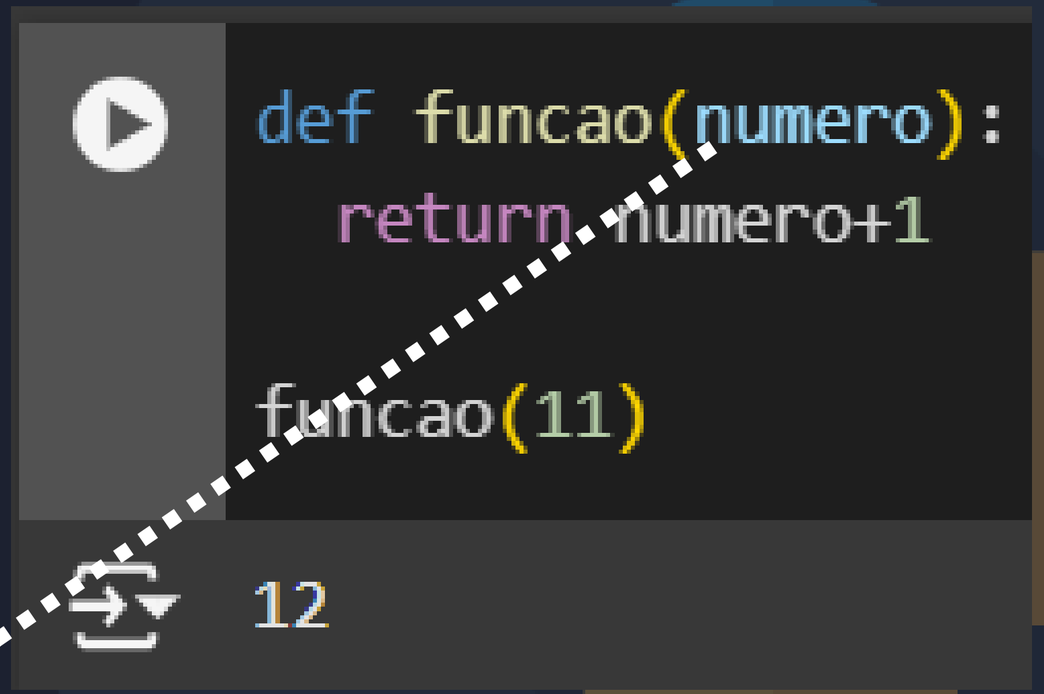
nome da função: pode ser qualquer nome, assim como a variavel.

CODE LAB TEEN

FUNÇÕES - REVISÃO

Na linguagem de programação Python, as funções tem uma estrutura especial.

parâmetros: são as informações de entrada da função que serão guardados em variáveis, neste caso chamada numero.



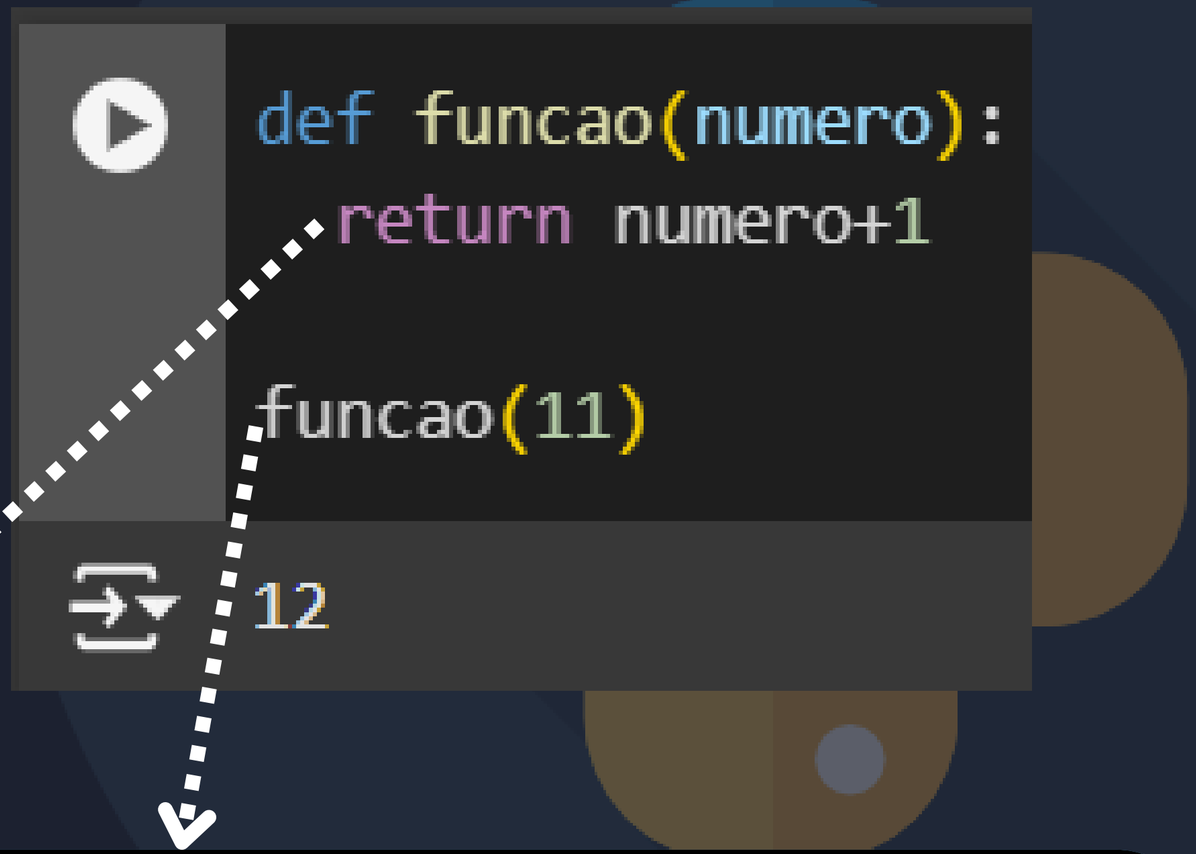
IMPORTANTE: Note a identação que forma o “bloco” da função.

CODE LAB TEEN

FUNÇÕES - REVISÃO

Na linguagem de programação Python, as funções tem uma estrutura especial.

return: diz para o código que ele vai devolver uma informação de saída.



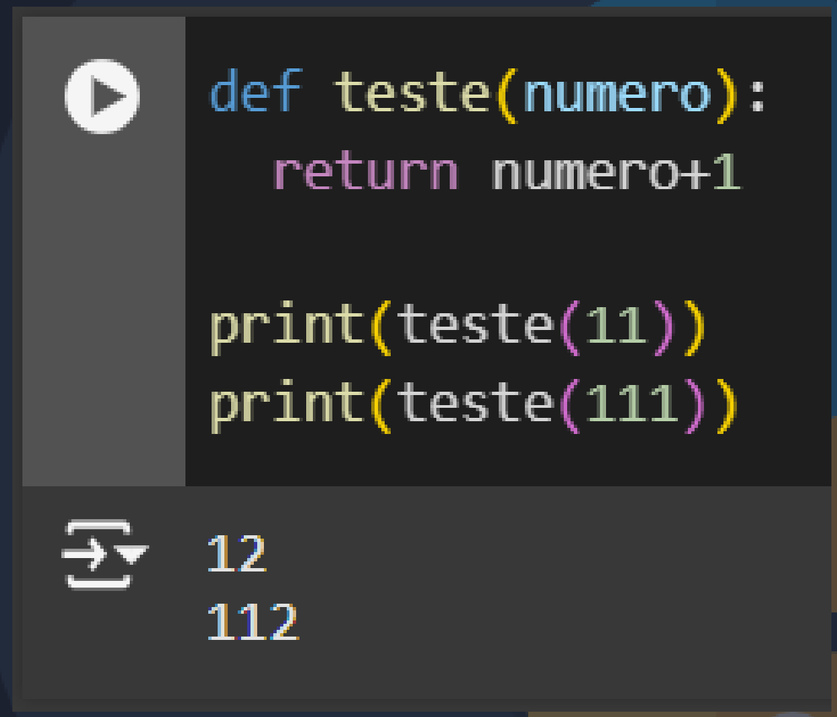
OBS: Note que chamamos a função e pedidos para que ela some 11 com 1. Logo, a função retorna 12.

CODE LAB TEEN

FUNÇÕES - REVISÃO

Vamos alterar o código e ver o que acontece, para fixar a sintaxe.

Note a mudança no nome da função e o fato de que podemos reutiliza-la no código para somar 1 a QUALQUER valor que quisermos.



CODE LAB TEEN

FUNÇÕES - REVISÃO

Vamos alterar o código e ver o que acontece, para fixar a sintaxe.

```
def teste(numero_um, numero_dois):
    return numero_um + numero_dois

print(teste(11, 45))
```

```
56
```

Modificamos a função teste para que ela receba mais de um parâmetro. Veja que, os parametros são separados por virgulas.

Quando chamamos a função com mais de um parâmetro também colocamos uma virgula. A função está somando o numero_um (11) e o numero_dois (45).

CODE LAB TEEN

FUNÇÕES - REVISÃO

Vamos alterar o código e ver o que acontece, para fixar a sintaxe.

Agora, a modificação mostra que tudo que está dentro da função é um pedaço de código. Logo, podemos declarar variáveis e até fazer laços.

```
def teste(numero_um, numero_dois):
  soma = numero_um + numero_dois
  return soma

print(teste(11, 45))
```

```
56
```

Neste caso, armazenamos o resultado na variável soma e depois devolvemos ela usando o return.

CODE LAB TEEN

FUNÇÕES - REVISÃO

Podemos ter funções também sem parâmetros, apenas que façam uma ação quando chamamos

```
def bomdia ():
  return "Bom dia, alunos!"


bomdia()
```

```
'Bom dia, alunos!'
```

As funções em Python tem o poder de fazer tudo aquilo que quisermos!!!

Neste caso temos uma função que apenas nos dá “Bom dia, alunos!” toda vez que chamamos ela

FUNÇÕES - REVISÃO

Podemos ter tudo dentro de uma função e apenas chamá-las com parâmetros para deixar o código mais organizado

Como neste exemplo temos uma função de tabuada pronta! Basta chamarmos e pedir o número que queremos

```python
def tabuada(numero):
    for i in range(11):
        print(f'{numero} x {i} = {numero * i}')

tabuada(5)
```

```
5 x 0 = 0
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

CODE LAB TEEN

FUNÇÕES - REVISÃO

Mas, qual a vantagem de usarmos as funções?

A vantagem das funções é a capacidade que temos de reutilizar várias vezes o mesmo algoritmo sem ter retrabalhos

Além de otimizações de código, organização e praticidade

CODE LAB TEEN

FUNÇÕES - REVISÃO

Comparações:

```python
def tabuada(numero):
    for i in range(11):
        print(f'{numero} x {i} = {numero * i}')

tabuada(5)
tabuada(4)
```

```
5 x 0 = 0
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
4 x 0 = 0
4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
4 x 10 = 40
```

Os resultados são iguais!!

```python
for i in range(11):
    print(f'{5} x {i} = {5 * i}')
for i in range(11):
        print(f'{4} x {i} = {4 * i}')
for i in range(11):
        print(f'{3} x {i} = {3 * i}')
```

```
5 x 0 = 0
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
4 x 0 = 0
4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
4 x 10 = 40
```

CODE LAB TEEN

FUNÇÕES - REVISÃO

Comparações:

```python
def tabuada(numero):
    for i in range(11):
        print(f'{numero} x {i} = {numero * i}')

tabuada(5)
tabuada(4)
```

```
5 x 0 = 0
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
4 x 10 = 40
```

Com funções utilizamos duas linhas apenas para fazer duas tabuadas

```python
for i in range(11):
    print(f'{5} x {i} = {5 * i}')
for i in range(11):
        print(f'{4} x {i} = {4 * i}')
for i in range(11):
        print(f'{3} x {i} = {3 * i}')
```

```
5 x 0 = 0
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
4 x 10 = 40
```

Se fizermos sem função usaríamos o DOBRO de linhas de código

CODE LAB TEEN

OBRIGADO!

Contem para gente o que você achou da aula de hoje:

https://forms.gle/e5ZtC8fhagHarBjDA

CODELAB TEEN