Funções

cubos
// academy //

Bloco de código que executa uma tarefa específica

Função

cubos academy

Motivação

Imagine que tenho as seguintes **atividades** para serem feitas **todos os meses para cada aluno**:

  - Calcular a média final;
  - Calcular a presença nas aulas;
- Fazer isso de forma manual **para cada aluno**, mesmo que usando loops, seria uma **atividade muito custosa**.
- Se a gente fizer um **bloco de código** que receba o **contexto** de cada aluno e nos retorne a informação definida, economizaria muito tempo, **como uma fórmula** que a gente só encaixa os **parâmetros**.

Funções

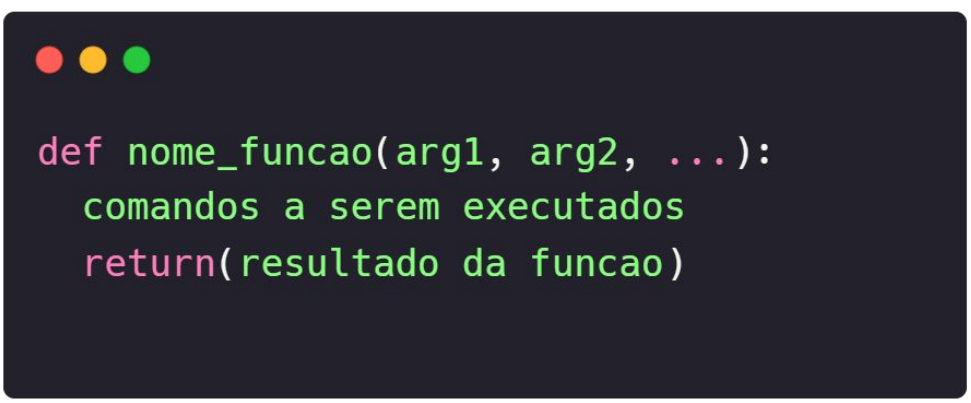
Argumentos

Retorno

Funções

Pedaço de código que realiza alguma tarefa específica:

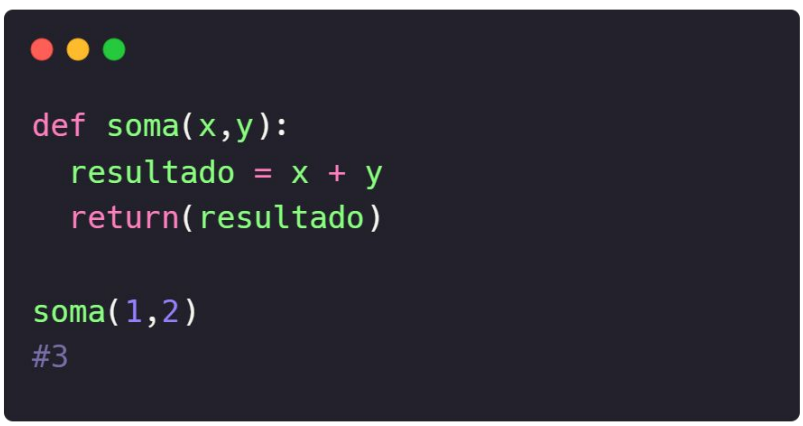
- Podem ser chamadas em qualquer parte do programa
  - Quantas vezes desejarmos
- Deixa nosso programa mais limpo e organizado
  - Evita escrever código repetitivo



```
def nome_funcao(arg1, arg2, ...):
  comandos a serem executados
  return(resultado da funcao)
```

Exemplo

Função que recebe dois números **x** e **y** e retorna o valor da sua soma



```python
def soma(x,y):
  resultado = x + y
  return(resultado)

soma(1,2)
#3
```

Exemplo

Função que recebe o nome, idade, altura e profissão de alguém e retorna o texto:

"Olá! Meu nome é **<nome>**, tenho **<idade>** anos, **<altura>**m de altura e sou **<profissão>**."

```python
def introducao(nome, idade, altura, profissao):
  print(f"Olá! Meu nome é {nome}, tenho {idade} anos, {altura}m de altura e sou {profissao}.")

introducao("helena", 26, 1.67, "estatistica")

#Olá! Meu nome é helena, tenho 26 anos, 1.67m de altura e sou estatistica.
```

Exemplo

Função que recebe uma idade e retorna a faixa etária da pessoa:

- Até 18 anos: “Jovem”
- De 19 a 60 anos: “Adulto”
- Acima de 60 anos: “Idoso”

```python
def faixa_etaria(idade):
  if idade <= 18:
    fx_etaria = "Jovem"
  elif idade <= 60:
    fx_etaria = "Adulto"
  else:
    fx_etaria = "Idoso"

  return(fx_etaria)

faixa_etaria(50)

#Adulto
```

Exemplo

Função que recebe uma lista de números e retorna a soma dos seus valores

```python
def soma(lista):
  total = 0
  for item in lista:
    total += item
  return(total)

minha_lista = [1,2,3,4]
soma(minha_lista)

#10
```

Exercícios

cubos academy

Escopo

cubos academy

Escopo

Dentro de um programa python teremos:

- **Variáveis Globais:** podem ser acessadas em qualquer momento no código, a partir do momento em que foi declarada
- **Variáveis Locais:** só podem ser acessadas dentro da função em que ela foi declarada

```python
idade_glob = 23

def calc_idade(ano_nasc, ano_atual):
  idade_loc = ano_atual - ano_nasc
  return(idade_loc)
```

Vamos ver como
funcionam os escopos

Na prática, no Colab!

cubos
// academy //

Utilizando Funções

cubos
academy

Utilizando Funções

- Não há limitações para a utilização de funções!
- Podemos usá-las:
  - Como argumento de uma função
  - Dentro de uma função

```python
#função que efetua soma
def soma(lista):
  total = 0
  for item in lista:
    total += item
  return(total)

#função que efetua produto
def produto(lista):
  prod = 1
  for item in lista:
    prod = prod * item
  return(prod)
```

Função Dentro de Outra Função

```python
#função que retorna a soma ou o produto
def calcula(lista, nome_operacao):
  if nome_operacao == 'soma':
    resultado = soma(lista)
  elif nome_operacao == 'produto':
    resultado = produto(lista)

  print(f"O resultado é {resultado}")

minha_lista = [1,2,3,4]

#chama as funções para a lista criada
calcula(minha_lista, 'produto')
calcula(minha_lista, 'soma')

#O resultado é 24
#O resultado é 10
```

Função como Argumento

```python
#função que retorna a soma ou o produto
def calcula(lista, operacao):
  resultado = operacao(lista)
  print(f"O resultado é {resultado}")

minha_lista = [1,2,3,4]

#chama as funções para a lista criada
calcula(minha_lista, produto)
calcula(minha_lista, soma)

#O resultado é 24
#O resultado é 10
```

Helena Guimarães
Professora Cubos Academy

cubos
// academy //

www.cubos.academy