

# PYTHON

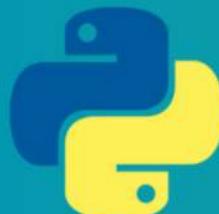
professor



Lázaro  
Santos



The Edge

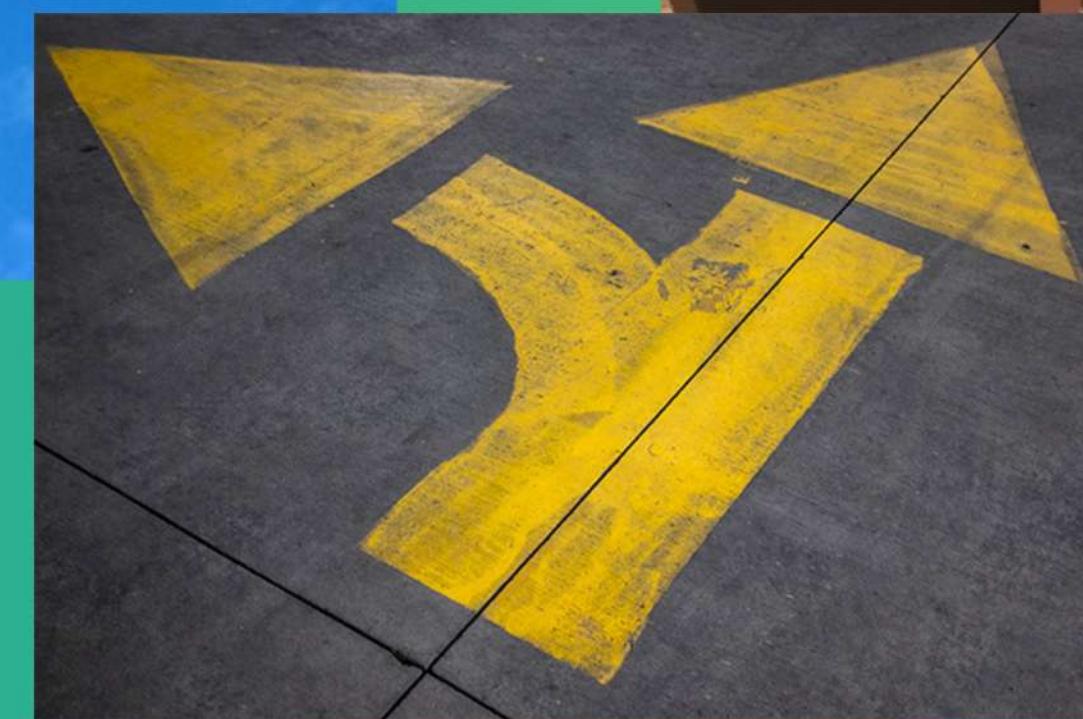


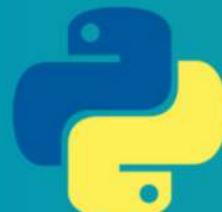
# IF & ELSE

If = SE

Else = SENÃO

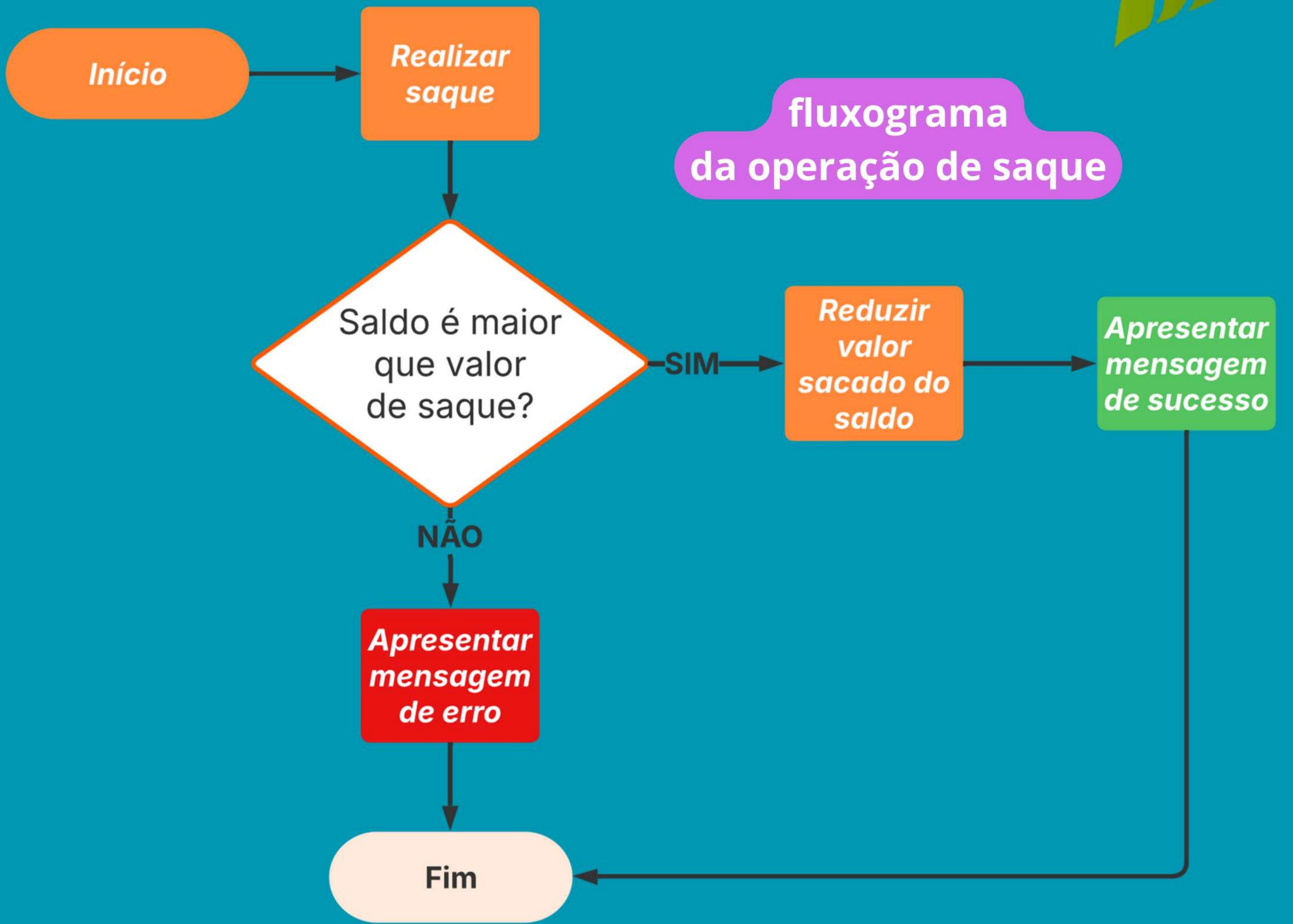
Elif = SENÃO, SE

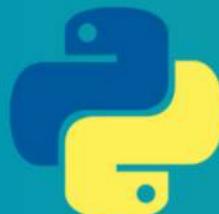




# CONDIÇÃO PARA SAQUE

SE o valor de saque é MENOR OU IGUAL ao valor do saldo bancário da conta, ENTÃO realizar operação de saque.

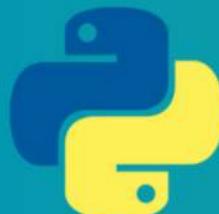




# CONDIÇÃO NO CÓDIGO

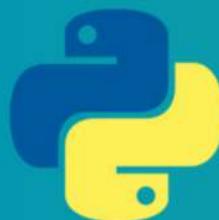
```
saldo = int( input("Digite o saldo bancário") )
saque = int( input("Digite o valor de saque") )

if saldo >= saque:
    saldo = saldo - saque
    print("Você realizou um saque com sucesso.")
else:
    print("Você não possui saldo suficiente para realizar essa operação.")
```



# IF & ELSE COM MÉDIA DE ALUNO

```
nota1 = int( input("Digite a primeira nota") )
nota2 = int( input("Digite a segunda nota") )
media = (nota1+nota2)/2
if media >= 6:
    print("Aprovado")
else:
    print("Reprovado")
```



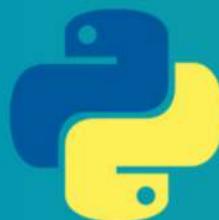
# IF & ELSE LISTA DE COMPRAS

```
valor_total_compra = 0
valor_feijao = 5.5
valor_arroz = 4
valor_macarrao = 2

produto = input("Digite o código do produto")

if produto == "feijão":
    valor_total_compra += valor_feijao
elif produto == "arroz":
    valor_total_compra += valor_arroz
elif produto == "macarrão":
    valor_total_compra += valor_macarrao

print(valor_total_compra)
```



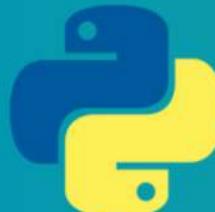
# DESAFIO

Desafio: SE porcentagem <= 0.0, Exibir "Insira um número maior que 0"

```
valor_parte = float(input("Insira o valor parte: "))
porcentagem = float(input("Insira o valor da porcentagem: "))
valor_total = valor_parte * (porcentagem/100)
print(valor_total)
```



*Incentação*



# INDENTAÇÃO

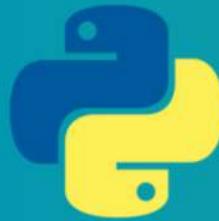
**Definição:** Em informática e tipografia, indentação refere-se ao ato ou efeito de indentar, ou seja, inserir um determinado espaço entre a margem da página e o início do texto de um parágrafo.

“Tudo vale a pena quando a alma não é pequena.”

Fernando Pessoa#

“Tudo vale a pena quando a alma não é pequena.”

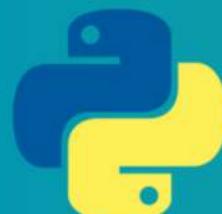
Fernando Pessoa#



# INDENTAÇÃO

Indentação é o uso de espaços em branco (ou tabulações) no início de uma linha de código para afastar o código da margem esquerda. Em muitas linguagens de programação a indentação é usada apenas para deixar o código mais bonito e legível.

**Por que a Indentação é Crucial em Python?**  
Aqui é onde Python é especial: a indentação não é apenas para organização, ela é parte da sintaxe da linguagem.



# BLOCOS DE INDENTAÇÃO

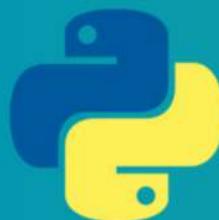
Bloco 1

Bloco 2

Bloco 3

Bloco 2

Bloco 1



# INDENTAÇÃO NO PYTHON

```
valor = int(input("Digite um valor: "))
```

```
if valor > 0:
```

```
    print("Valor positivo")
```

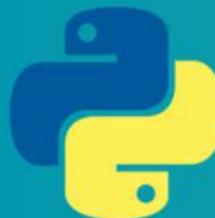
```
    if valor % 2 == 0:
```

```
        print("Valor par")
```

```
    print("Fim do programa")
```

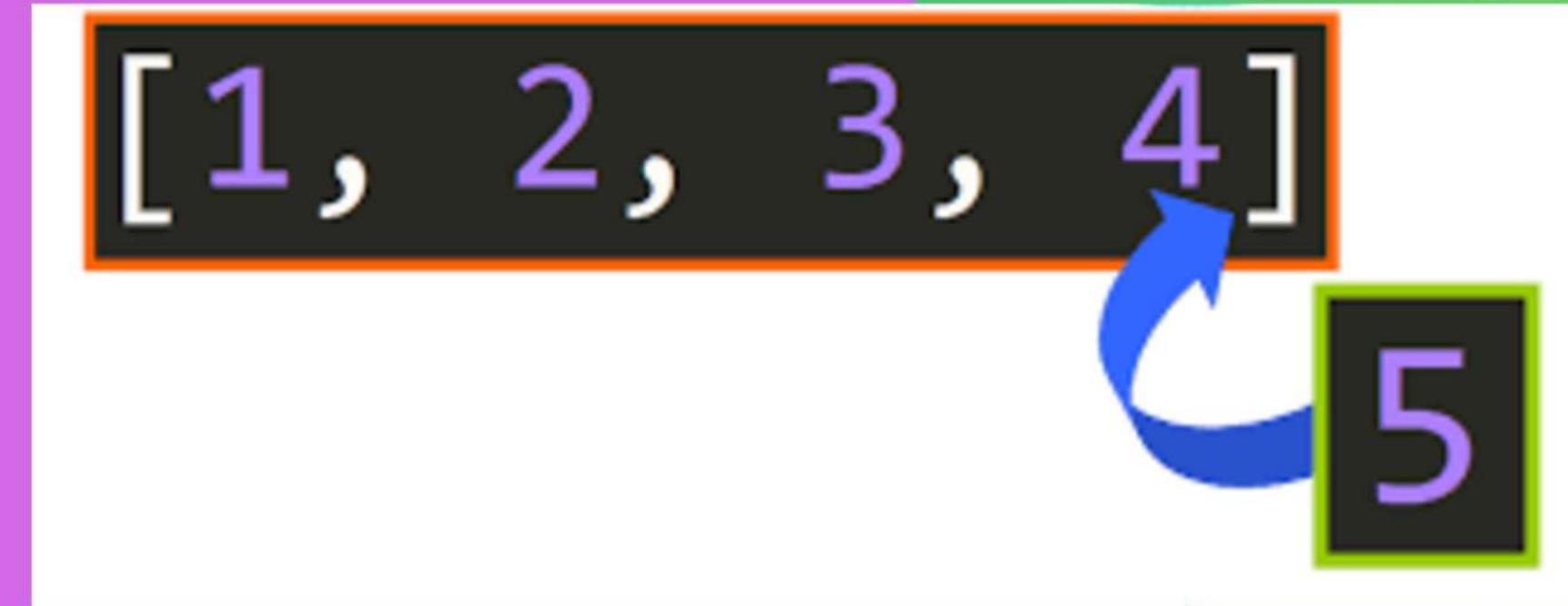


*Listas*



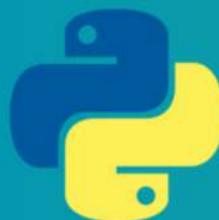
# LISTAS

Listas são extremamente úteis; é o tipo de dado mais comum do python.



É uma simples “coleção” de itens.

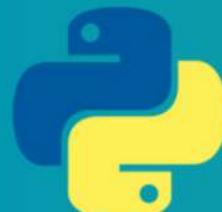
Você pode guardar strings, inteiros, floats, variáveis, funções e até... listas!



# LISTAS

```
números = [ 1, 4, 7, 21, 98, 156]
coordenadas = [ 1.5694, 9.3991]
criaturas_mitológicas = ["Unicórnio", "Vampiro",
    "Minotauro", "Dragão", "Chupacabra"]

print(números[3]) # 21
print(criaturas_mitológicas[1]) # Vampiro
print(coordenadas[-1]) # 9.3991
```



# CONCATENANDO LISTAS

```
print(números + coordenadas)
# [1, 4, 7, 21, 98, 156, 1.5694, 9.3991]
```



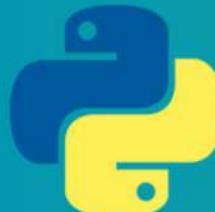
# FUNÇÕES DE LISTAS

```
números.insert(4,5) .insert()  
print(números)  
# [1, 4, 7, 21, 5, 98, 156]
```

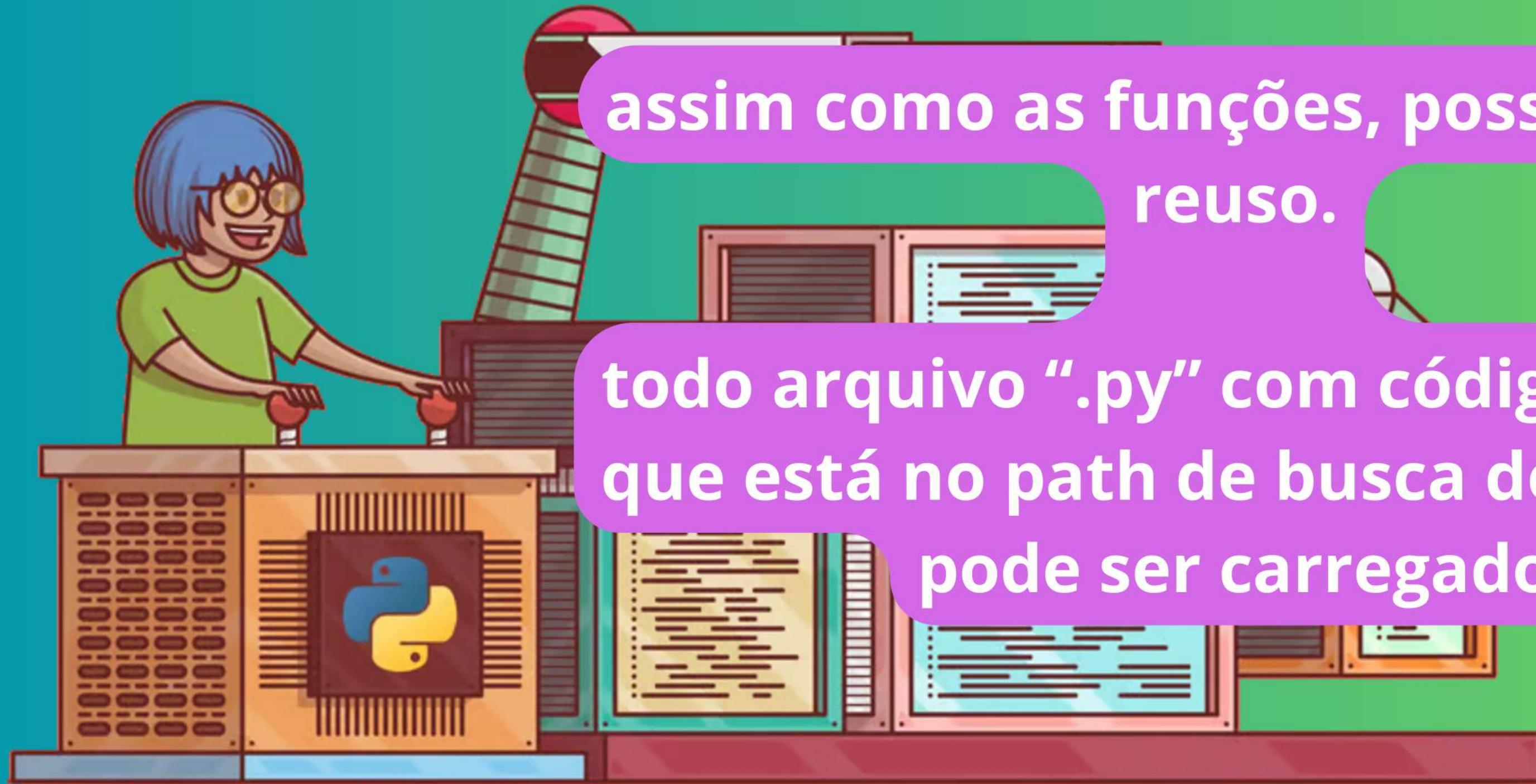
```
números.pop(4) .pop()  
print(números)  
# [1, 4, 7, 21, 98, 156]
```



*Mochlos*

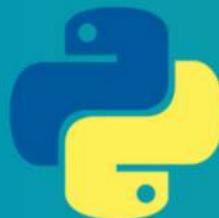


# MÓDULOS



assim como as funções, possibilitam o reuso.

todo arquivo “.py” com código válido e que está no path de busca de módulos pode ser carregado



# PATH

quando importamos um módulo, o interpretador percorre uma série de caminhos - os chamados paths -, para procurar a implementação do módulo importado.

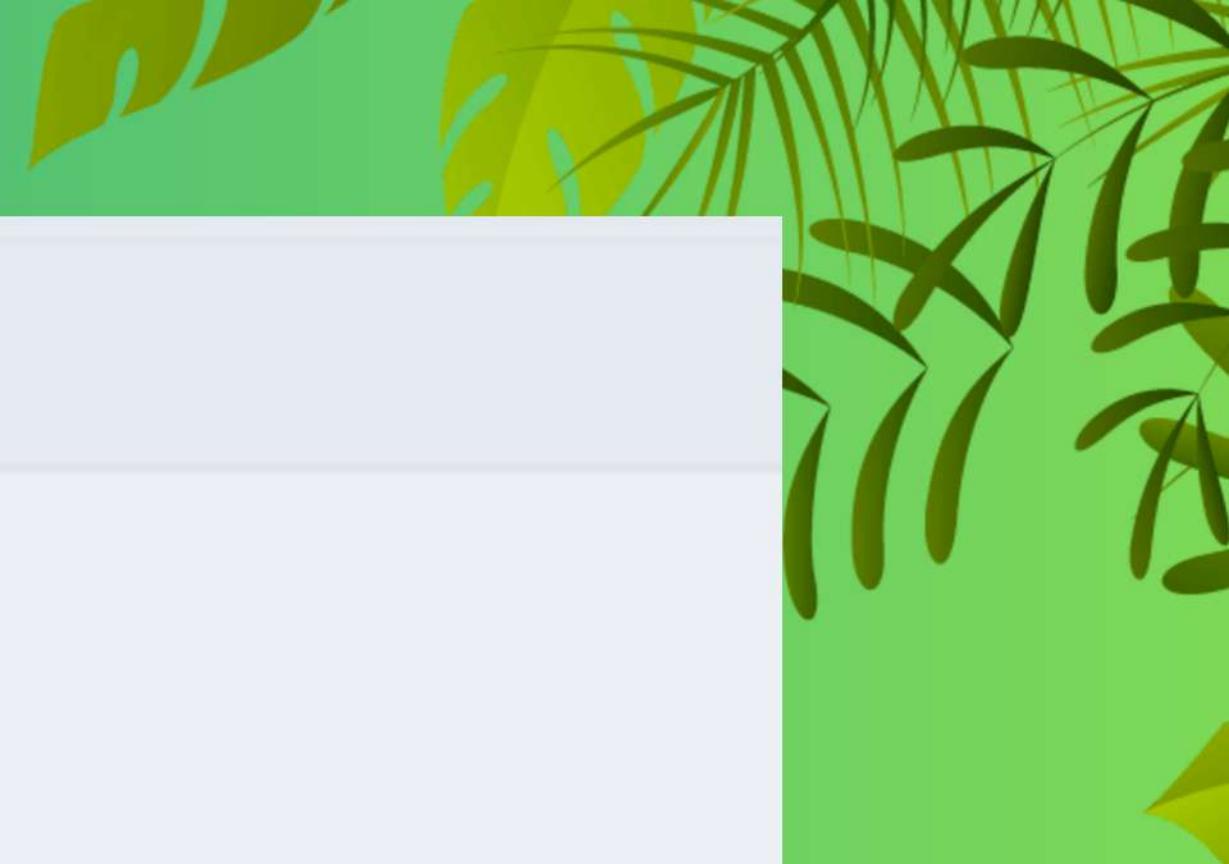
Por padrão, o diretório de trabalho, de onde o interpretador foi chamado, está nessa lista. Por isso, quando criamos um arquivo `meu_modulo.py`, se abrimos o interpretador no mesmo diretório dele, podemos executar com sucesso `import meu_modulo` e utilizar o seu código.

cofre.py

senha.py ×

senha.py > [?] minha\_senha

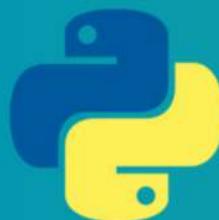
```
1 minha_senha = "123456"
2
3 def verificar_senha(nova_senha):
4     if len(nova_senha) > 6:
5         print("Válida")
6     else:
7         print("Inválida")
```



```
cofre.py  x  senha.py
cofre.py > {} senha
1 import senha
2
3 print(senha.minha_senha)
4
```

para fazer o import, basta digitar import e o nome  
do arquivo

para acessar uma variável de um módulo,  
use o “.”



# MÓDULO RANDOM

```
import random

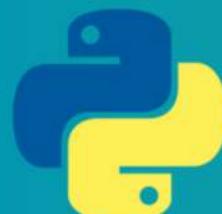
numero_aleatorio = random.random()
print(numero_aleatorio)
```



# MÓDULO RANDOM

```
import random

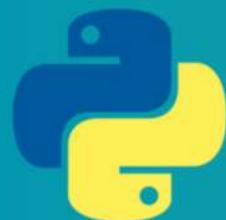
numero_aleatorio = random.randint(0,10)
print(numero_aleatorio)
```



# MÓDULO RANDOM

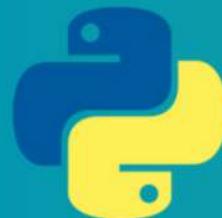
```
import random

cesta = [ 'Maçã', 'Banana', 'Uva', 'Limão']
print(random.choice(cesta))
```



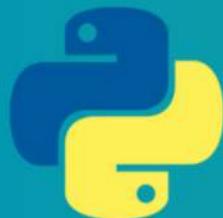
# MÓDULO TIME

```
import time  
  
print(time.asctime())
```



# MÓDULO TURTLE

```
import turtle  
  
turtle.circle(50)  
turtle.getscreen()._root.mainloop()
```



# DESAFIO

traduza o resultado do  
módulo “this”

```
import this
```

# REFERÊNCIA BIBLIOGRÁFICA:



MAYER, Christian. Coffee Break Python Slicing: 24 Workouts to Master Slicing in Python, Once and for All. Amazon: Amazon, 2018.

# Coffee Break Python

# Slicing



**24** Workouts to Master Slicing in Python, Once and for All

CHRISTIAN MAYER