



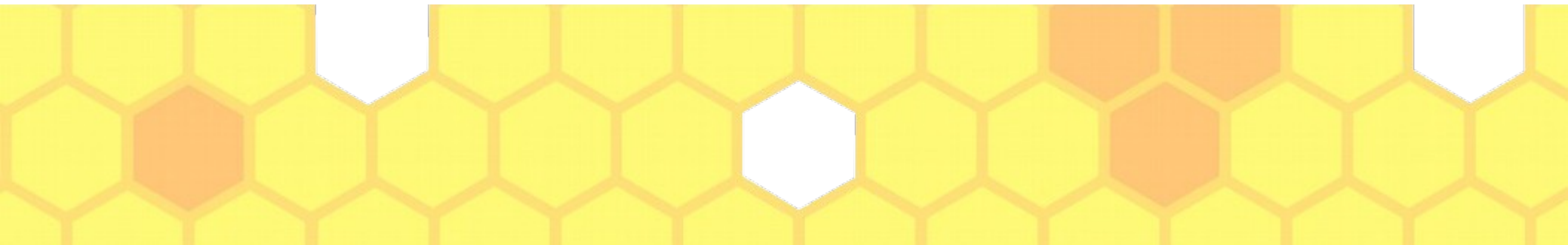
For o retorno...





# Cronograma

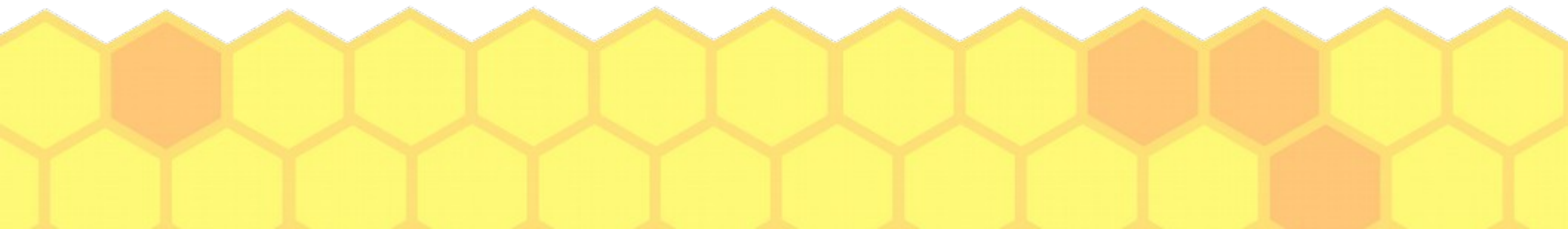
- Até final de setembro



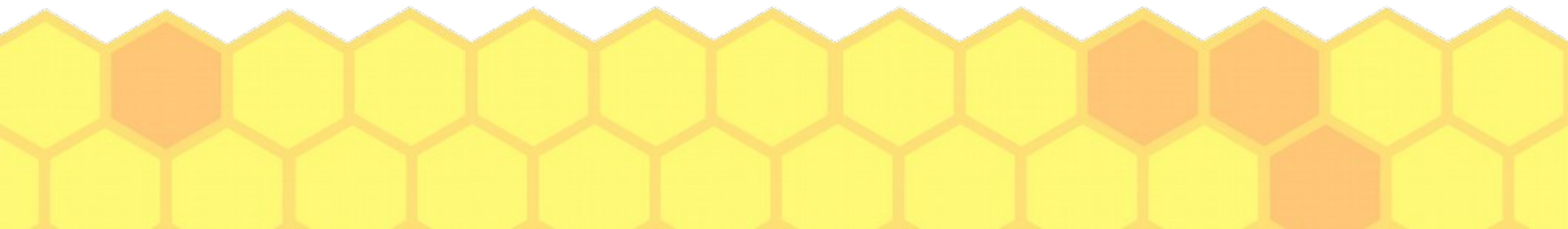
- Hoje 01/09:
  - Resolução de 3 exercícios
  - Revisão sobre listas
  - Interação de listas com o for e indexação
  - Exercícios



- 03/09 – While e controle de fluxo
- **08/09 – Introdução ao banco de dados – Conceitos e características**
- **09/09 – Desenvolvendo Bases de dados, Tabelas e tipos de dados.**
- **10/09 – Segurança dos dados e banco de dados**



- **14/09 – Aulas Presencias (Inglês)**
- 15/09 – Reapresentação, Ambientação
- 16/09 – Exercícios de revisão
- 17/09 – Exercícios de revisão



- 22/09 - Funções Apresentação
- 23/09 - Funções Modulação
- 24/09 - Formatação de string
- 25/09 - Escrita e leitura de arquivos.
- 29/09 - Escrita e leitura de arquivos.
- 30/09 - Exceções (try except)



## 1. Lógica de Programação

- Sistemas computacionais.
- Introdução à lógica.
- Noções de algoritmos de programação.
- Constantes, variáveis e tipos de dados.
- Processamento sequencial e condicional.
- Métodos de repetição.
- Manipulação de vetores.
- Manipulação de matrizes.
- Programação estruturada.

## 2. Banco de dados

- Conceitos
- Características
- Segurança
- Desenvolvendo bases de dados
- Desenvolvendo tabelas
- Tipos de dados

### 2.1. Manipulação de dados

- *INSERT*
- *SELECT*
- *UPDATE*
- *DELETE*

### 2.2. Trabalhando com WHERE

- *AND* e *OR*
- *IN*
- *BETWEEN*
- *NOT BETWEEN*
- *LIKE*
- *NOT LIKE*
- *ORDER BY ASC*
- *ORDER BY DESC*
- *GROUP BY*
- Ordenando dados
- Totalizando dados
- Trabalhando com chaves primárias e estrangeiras
- Associação de múltiplas tabelas

## 3. Desenvolvimento Python

### 3.1. Introdução

- Descrever os principais recursos da tecnologia Python.
- Tipagem dinâmica e forte.
- Escrever, compilar e executar um aplicativo em Python simples.
- Uso IDE - Visual Studio Code.
- PEP8
- Entrada e saída de usuário.
- Expressões e Controle de Fluxo.
- Usar estruturas de decisão - if, else, elif.
- Trabalhando com listas e Tuplas.
- Trabalhando com estruturas de repetição - while, for.
- Formatação de strings.
- Definir exceções.
- Usar as instruções try, except e finally.
- Conversão de tipos.
- Escrita e leitura de arquivos.

### 3.2. Programação Orientada a Objetos

- Definir os conceitos de modelagem: abstração e encapsulamento.
- Classe, Construtor, atributos, propriedades e métodos.
- Decoradores de funções.
- Uso de getters e setters.
- Definir herança, polimorfismo.
- Invocar um método em um objeto específico.
- Descrever a operação completa de construção e de inicialização de objeto
- Métodos - Parâmetros nomeados e opcionais

## 4. Desenvolvimento de Python para Web

### 4.1. Introdução

- Estrutura do protocolo HTTP.
- Requisições, Respostas, Parâmetros, Serviços Web.
- Conhecendo Html.
- Trabalhando com Css, uso, conceitos básicos e Bootstrap.
- Conhecendo o Javascript, conceitos básicos e JQuery.
- Uma breve apresentação sobre frameworks Front-End - Angular e React
- Estrutura e uso de Servidores web e hospedagem.

### 4.2. Framework Flask

- Construindo aplicações web com Flask.
- Servidor web de desenvolvimento
- Ciclo de Requisição e resposta
- Rotas
- Integração com Css e Javascript
- Renderização de templates - Funções de view, página de layout.
- Utilizando banco de dados Mysql no Flask
- Sessão



# Relembrando...

- Listas





# Relembrando...

- Listas
  - Guardam vários valores



# Relembrando...

- Listas
  - Guardam vários valores
  - Podemos recuperar os valores por índices



# Relembrando...

- Listas
  - Guardam vários valores
  - Podemos recuperar os valores por índices
  - Podemos iterar com o for



# Guardar vários valores

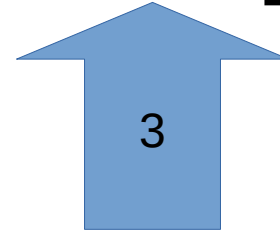
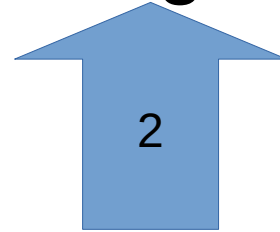
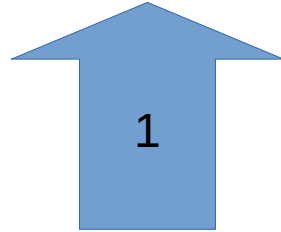
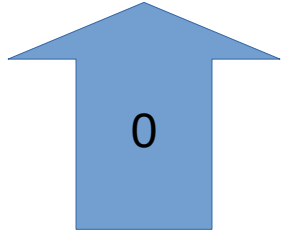
- lista =  
[ 1, 5.99, 'Olá mundo', ['outra', 'lista'], True ]
- Podemos guardar:
  - Variáveis primitivas: Inteiro, float, string, booleano
  - Variáveis tipo coleções: listas, tuplas, dicionários
  - Funções, classes, objetos.



# Recuperando os valores por índices

- Lista =

["Abacaxi", "Banana", "Detergente", "Pão"]



- Lista[ 0 ] → "Abacaxi"

Lista[ 0 : 2 ] → [ "Abacaxi", "Banana" ]



# Inteirando lista com o for

```
>>> Lista = [ 1, 2, "a" ]
```



# Inteirando lista com o for

```
>>> Lista = [ 1, 2, "a" ]
```

```
>>> for i in Lista:  
    print(i)
```





# Inteirando lista com o for

```
>>> Lista = [ 1, 2, "a" ]
```

```
>>> for i in Lista:  
    print(i)
```

Tela:



# Inteirando lista com o for

```
>>> Lista = [ 1, 2, "a" ]
```

```
>>> for i in Lista:  
    print(i)
```

Tela:

1



# Inteirando lista com o for

```
>>> Lista = [ 1, 2, "a" ]
```

```
>>> for i in Lista:  
    print(i)
```

Tela:

1

2



# Inteirando lista com o for

```
>>> Lista = [ 1, 2, "a" ]
```

```
>>> for i in Lista:  
    print(i)
```

Tela:

1

2

3



# Inteirando lista com o for

```
>>> Lista = [ 1, 2, "a" ]
```

```
>>> for i in Lista:  
    print(i)
```

Tela:

1

2

3

'a'

A decorative border at the bottom of the slide consisting of a grid of yellow hexagons with orange outlines. Some hexagons are filled with a solid orange color, creating a patterned effect.

# E se precisar sincronizar mais listas?

- Faça um programa de cadastro de pessoas que receba 5 nomes, idades e e-mails e salve cada um em uma lista. Depois mostre os dados dos clientes.



# Captação de dados

# Apresentação dos resultados





# Captação de dados

# **Criar Listas vazias**



# Captação de dados

**lista\_nomes = [ ]**

**lista\_idades = [ ]**

**lista\_email = [ ]**



# Captação de dados

lista\_nomes = [ ]

lista\_idades = [ ]

lista\_email = [ ]

**# fazer o for para usarmos o loop a nosso favor!**



# Captação de dados

lista\_nomes = [ ]

lista\_idades = [ ]

lista\_email = [ ]

**for i in range(5):**



```
# Captação de dados
```

```
lista_nomes = [ ]
```

```
lista_idades = [ ]
```

```
lista_email = [ ]
```

```
for i in range(5):
```

**# Entrada dos dados via teclado**



```
# Captação de dados
```

```
lista_nomes = [ ]
```

```
lista_idades = [ ]
```

```
lista_email = [ ]
```

```
for i in range(5):
```

```
    nome = input(f 'Digite o nome {1+i}' )
```

```
    idade = input(f 'Digite o idade {1+i}' )
```

```
    email = input(f 'Digite o email {1+i}' )
```



```
# Captação de dados
```

```
lista_nomes = [ ]
```

```
lista_idades = [ ]
```

```
lista_email = [ ]
```

```
for i in range(5):
```

```
    nome = input(f 'Digite o nome {1+i}' )
```

```
    idade = input(f 'Digite o idade {1+i}' )
```

```
    email = input(f 'Digite o email {1+i}' )
```

```
# Adicionar os dados nas listas
```

A decorative border at the bottom of the slide consisting of a grid of yellow hexagons with orange outlines. Some hexagons are filled with a solid orange color, creating a pattern.



# Captação de dados

lista\_nomes = [ ]

lista\_idades = [ ]

lista\_email = [ ]

for i in range(5):

    nome = input(f 'Digite o nome {1+i}' )

    idade = input(f 'Digite o idade {1+i}' )

    email = input(f 'Digite o email {1+i}' )

**lista\_nomes.append( nome )**

**lista\_idades.append( idade )**

**lista\_email.append( email )**

# Captação de dados

lista\_nomes = [ ]

lista\_idades = [ ]

lista\_email = [ ]

for i in range(5):

    nome = input(f 'Digite o nome {1+i}' )

    idade = input(f 'Digite o idade {1+i}' )

    email = input(f 'Digite o email {1+i}' )

**lista\_nomes.append( nome )**

**lista\_idades.append( idade )**

**lista\_email.append( email )**

# Apresentação dos resultados

**# Criar um for para inteirar as listas**



# Apresentação dos resultados

**for i in lista\_nomes:**



# Apresentação dos resultados

for i in lista\_nomes:

**Só mostra os nomes dos clientes**



# Apresentação dos resultados

for i in lista\_nomes:

# Só mostra os nomes dos clientes

**Para mostrar os dados dos clientes (nome, idade, e-mail) vou precisar de outra estratégia!**



# Apresentação dos resultados

for i in lista\_nomes:

# Só mostra os nomes dos clientes

**Qual a estratégia que eu poderia usar para mostrar o nome, idade, e-mail do cliente ao mesmo tempo?**





# Apresentação dos resultados

for i in lista\_nomes:

# Só mostra os nomes dos clientes

**Qual a estratégia que eu poderia usar para mostrar o nome, idade, e-mail do cliente ao mesmo tempo?**

**Sabendo que o for consegue inteirar um lista por vez!**



# Inteirar pelo índice

- As listas devem estar ordenadas e padronizadas.



# Captação de dados

lista\_nomes = [ ]

lista\_idades = [ ]

lista\_email = [ ]

for i in range(5):

    nome = input(f 'Digite o nome {1+i}' )

    idade = input(f 'Digite o idade {1+i}' )

    email = input(f 'Digite o email {1+i}' )

**lista\_nomes.append( nome )**

**lista\_idades.append( idade )**

**lista\_email.append( email )**

# # Captação de dados

```
lista_nomes = [ ]
```

```
lista_idades = [ ]
```

```
lista_email = [ ]
```

```
for i in range(5):
```

```
    nome = input(f 'Digite o nome {1+i}' )
```

```
    idade = input(f 'Digite o idade {1+i}' )
```

```
    email = input(f 'Digite o email {1+i}' )
```

```
    lista_nomes.append( nome )
```

```
    lista_idades.append( idade )
```

```
    lista_email.append( email )
```

Note que todos dados  
são capturados juntos

## # Captação de dados

```
lista_nomes = [ ]
```

```
lista_idades = [ ]
```

```
lista_email = [ ]
```

```
for i in range(5):
```

```
    nome = input(f 'Digite o nome {1+i}' )
```

```
    idade = input(f 'Digite o idade {1+i}' )
```

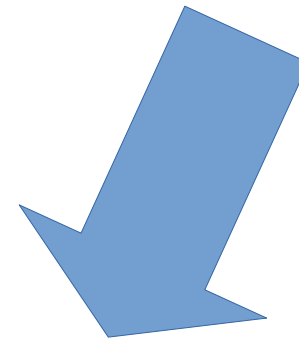
```
    email = input(f 'Digite o email {1+i}' )
```

```
    lista_nomes.append( nome )
```

```
    lista_idades.append( idade )
```

```
    lista_email.append( email )
```

Note que todos dados  
são capturados juntos



## # Captação de dados

```
lista_nomes = [ ]
```

```
lista_idades = [ ]
```

```
lista_email = [ ]
```

```
for i in range(5):
```

```
    nome = input(f 'Digite o nome {1+i}' )
```

```
    idade = input(f 'Digite o idade {1+i}' )
```

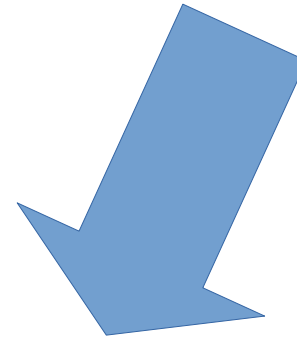
```
    email = input(f 'Digite o email {1+i}' )
```

```
    lista_nomes.append( nome )
```

```
    lista_idades.append( idade )
```

```
    lista_email.append( email )
```

Note que todos dados  
são capturados juntos



## # Captação de dados

```
lista_nomes = [ ]
```

```
lista_idades = [ ]
```

```
lista_email = [ ]
```

```
for i in range(5):
```

```
    nome = input(f 'Digite o nome {1+i}' )
```

```
    idade = input(f 'Digite o idade {1+i}' )
```

```
    email = input(f 'Digite o email {1+i}' )
```

```
    lista_nomes.append( nome )
```

```
    lista_idades.append( idade )
```

```
    lista_email.append( email )
```

Depois é guardado  
nas listas na mesma  
posição

## # Captação de dados

```
lista_nomes = [ ]
```

```
lista_idades = [ ]
```

```
lista_email = [ ]
```

```
for i in range(5):
```

```
    nome = input(f 'Digite o nome {1+i}' )
```

```
    idade = input(f 'Digite a idade {1+i}' )
```


```
    email = input(f 'Digite o email {1+i}' )
```

```
    lista_nomes.append( nome )
```

```
    lista_idades.append( idade )
```

```
    lista_email.append( email )
```

Depois é guardado  
nas listas na mesma  
posição





## # Captação de dados

```
lista_nomes = [ ]
```

```
lista_idades = [ ]
```

```
lista_email = [ ]
```

```
for i in range(5):
```

```
    nome = input(f 'Digite o nome {1+i}' )
```

```
    idade = input(f 'Digite a idade {1+i}' )
```


```
    email = input(f 'Digite o email {1+i}' )
```

```
    lista_nomes.append( nome )
```

```
    lista_idades.append( idade )
```

```
    lista_email.append( email )
```

Depois é guardado  
nas listas na mesma  
posição





# Inteirar pelo índice

- As listas devem estar ordenadas e padronizadas.



# Inteirar pelo índice

- As listas devem estar ordenadas e padronizadas.

[ 'Abioluz', 'Paulo', 'João' ]



# Inteirar pelo índice

- As listas devem estar ordenadas e padronizadas.

[ 'Abioluz', 'Paulo', 'João' ]

[ 35 , 45 , 15 ]



# Inteirar pelo índice

- As listas devem estar ordenadas e padronizadas.

[ 'Abioluz', 'Paulo', 'João' ]

[ 35 , 45 , 15 ]

[ 'abio@a', 'paulo@b', 'joão@c' ]



# Inteirar pelo índice

- As listas devem estar ordenadas e padronizadas.

[ 'Abioluz', 'Paulo', 'João' ]

[ 35 , 45 , 15 ]

[ 'abio@a', 'paulo@b', 'joão@c' ]

**Dados do Abioluz**

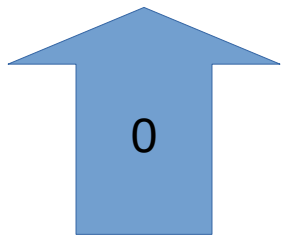
# Inteirar pelo índice

- As listas devem estar ordenadas e padronizadas.

[ 'Abioluz', 'Paulo', 'João' ]

[ 35 , 45 , 15 ]

[ 'abio@a', 'paulo@b', 'joão@c' ]

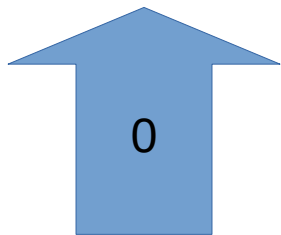


**Dados do Abioluz**

# Inteirar pelo índice

- As listas devem estar ordenadas e padronizadas.

[ 'Abioluz',	'Paulo',	'João' ]
[ 35,	45,	15 ]
[ 'abio@a',	'paulo@b',	'joão@c' ]



**Dados do Abioluz**



# Inteirar pelo índice

- As listas devem estar ordenadas e padronizadas.

[ 'Abioluz', 'Paulo', 'João' ]

[ 35 , 45 , 15 ]

[ 'abio@a', 'paulo@b', 'joão@c' ]

**Dados do Paulo**

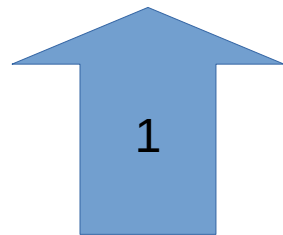
# Inteirar pelo índice

- As listas devem estar ordenadas e padronizadas.

[ 'Abioluz', 'Paulo', 'João' ]

[ 35 , 45 , 15 ]

[ 'abio@a', 'paulo@b', 'joão@c' ]



**Dados do Paulo**

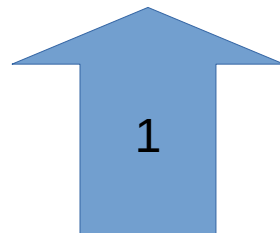
# Inteirar pelo índice

- As listas devem estar ordenadas e padronizadas.

[ 'Abioluz', 'Paulo', 'João' ]

[ 35 , 45 , 15 ]

[ 'abio@a', 'paulo@b', 'joão@c' ]



1

**Dados do Paulo**



# Apresentação dos resultados

for i in lista\_nomes:

**# Primeiro passo: Colocar o range() no for**



# Apresentação dos resultados  
for i in **range( )**:



# Apresentação dos resultados

for i in **range( )**:

**# Segundo passo: descobrir o tamanho da lista.**



# Apresentação dos resultados

for i in **range( )**:

**# Segundo passo: descobrir o tamanho da lista.**

**# Lembrando que as listas possuem o mesmo tamanho.**



```
# Apresentação dos resultados  
tamanho = len( lista_nomes )  
for i in range( ):
```





# Apresentação dos resultados

tamanho = len( lista\_nomes )

for i in range( **tamanho** ):



# Apresentação dos resultados

tamanho = len( lista\_nomes )

for i in range( **tamanho** ):

**# Terceiro passo: Preparar o print(), colocar  
as listas mas sem indexar.**



# Apresentação dos resultados

tamanho = len( lista\_nomes )

for i in range( tamanho ):

**print( f"**

**nome: { lista\_nomes[ ] }**

**Idade: { lista\_idades[ ] }**

**E-mail: { lista\_email[ ] }**

**"" )**



# Apresentação dos resultados

```
tamanho = len( lista_nomes )
```

```
for i in range( tamanho ):
```

```
    print( f"
```

```
nome: { lista_nomes[ ] }
```

```
Idade: { lista_idades[ ] }
```

```
E-mail: { lista_email[ ] }
```

```
"" )
```

**# Quarto e ultimo passo: Adicionar o índice  
para sincronizar as listas**



# Apresentação dos resultados

tamanho = len( lista\_nomes )

for i in range( tamanho ):

print( f"

nome: { lista\_nomes[ ] }

Idade: { lista\_idades[ ] }

E-mail: { lista\_email[ ] }

"" )



# Apresentação dos resultados

tamanho = len( lista\_nomes )

for i in range( tamanho ):

print( f"

nome: { lista\_nomes[ ] }

Idade: { lista\_idades[ ] }

E-mail: { lista\_email[ ] }

"" )



# Apresentação dos resultados

tamanho = len( lista\_nomes )

for **i** in range( tamanho ):

print( f"

nome: { lista\_nomes[ ] }

Idade: { lista\_idades[ ] }

E-mail: { lista\_email[ ] }

"" )



# Apresentação dos resultados

tamanho = len( lista\_nomes )

for i in range( tamanho ):

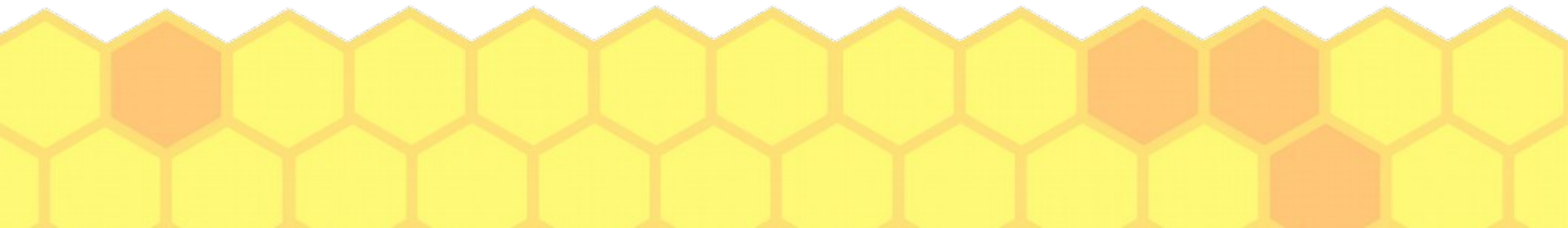
print( f"

nome: { lista\_nomes[ i ] }

Idade: { lista\_idades[ i ] }

E-mail: { lista\_email[ i ] }

"" )





# Apresentação dos resultados

tamanho = len( lista\_nomes )

for i in range( tamanho ):

print( f"

nome: { lista\_nomes[ i ] } ←

Idade: { lista\_idades[ i ] } ←

E-mail: { lista\_email[ i ] } ←

"" )



- Na tela:

nome: Abioluz

Idade: 35

E-mail: abio@a

nome: Paulo

Idade: 45

E-mail: paulo@b

nome: João

Idade: 15

E-mail: joão@c

