




# Estrutura de decisão (condicional)

python

 Copiar

```
if condição:
    # Código executado se a condição for verdadeira
elif outra_condição:
    # Código executado se a primeira condição for falsa e outra_condição for verdadeira
else:
    # Código executado se todas as condições anteriores forem falsas
```

python

```
# Definindo uma variável
idade = 20

# Estrutura de decisão básica
if idade >= 18:
    print("Você é maior de idade.")
elif 13 <= idade < 18:
    print("Você é adolescente.")
else:
    print("Você é menor de idade.")
```

python

```
# Exemplo com múltiplas condições e operadores lógicos
numero = 10

if numero > 0 and numero % 2 == 0:
    print("O número é positivo e par.")
elif numero > 0 and numero % 2 != 0:
    print("O número é positivo e ímpar.")
else:
    print("O número é zero ou negativo.")
```



# Estrutura de repetição (for loop)

## Sintaxe do for:

python

 Copiar

```
for variável in sequência:  
    # Bloco de código a ser repetido
```

## Exemplo com for:

python

 Copiar

```
# Iterando sobre uma lista  
frutas = ["maçã", "banana", "cereja"]  
for fruta in frutas:  
    print(fruta)
```



# Estrutura de repetição (for loop)

range(5)

gera uma sequência de números de 0 a 4.

O laço for percorre essa sequência, atribuindo cada número à variável i e imprimindo a iteração atual.

## Exemplo com range():

python

 Copiar

```
# Usando range para repetir um bloco de código 5 vezes
for i in range(5):
    print("Iteração:", i)
```








# Estrutura de repetição (for loop)

## O enumerate()

é uma função útil em Python que permite iterar sobre uma sequência, como uma lista ou uma string, **ao mesmo tempo em que mantém um contador automático.**

```
1 frutas = ["maçã", "banana", "pera", "uva", "abacaxi"]
2
3 for i, fruta in enumerate(frutas, start=0):
4     # esse start=0 aqui é opcional.
5     print("Índice:", i, " - Fruta:", fruta)
```

Ln: 5, Col: 44

 Run  Share  Command Line Arguments

```
Índice: 0 - Fruta: maçã
Índice: 1 - Fruta: banana
Índice: 2 - Fruta: pera
Índice: 3 - Fruta: uva
Índice: 4 - Fruta: abacaxi
```

Essa função retorna pares:  
o índice (ou contador) + o valor correspondente da sequência.

# Estrutura de repetição (while loop)

O laço while repete um bloco de código enquanto uma condição especificada é verdadeira.

É útil quando você não sabe com antecedência quantas vezes o laço precisará ser executado.

## Sintaxe do while:

python

 Copiar

```
while condição:  
    # Bloco de código a ser repetido
```

## Exemplo com while:

python

 Copiar

```
# Exemplo simples de while  
contador = 0  
while contador < 5:  
    print("Contagem:", contador)  
    contador += 1
```




# Estrutura de repetição (break / continue)

## break:

Interrompe a execução do laço imediatamente.

python

 Copiar

```
for i in range(10):  
    if i == 5:  
        break  
    print(i)
```

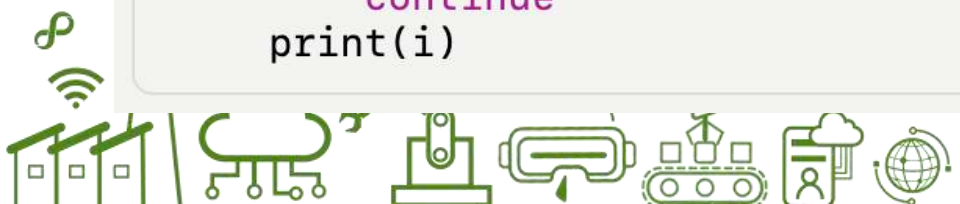
## continue:

Pula para a próxima iteração do laço, ignorando o código restante na iteração atual.

python

 Copiar

```
for i in range(5):  
    if i == 2:  
        continue  
    print(i)
```



## Exemplo Completo com while, break, e continue:

python

```
# Exemplo combinando while com break e continue
contador = 0
while contador < 10:
    contador += 1
    if contador == 5:
        continue # Pula a impressão do número 5
    if contador == 8:
        break # Interrompe o laço ao alcançar 8
    print(contador)
```





# LISTAS, TUPLAS E DICIONÁRIOS

Python possui várias estruturas de dados integradas que permitem armazenar e manipular coleções de dados.

As mais comuns são listas, tuplas e dicionários.

Cada uma dessas estruturas tem características específicas e métodos associados que são úteis para diferentes situações.



# LISTAS

Coleções de elementos mutáveis (podem ser alteradas).

Uma lista pode conter elementos de tipos diferentes, incluindo outras listas.

## Criando uma Lista

python

```
# Criando uma lista de números  
numeros = [1, 2, 3, 4, 5]
```

```
# Lista mista com diferentes tipos de dados  
mista = [1, "Olá", 3.14, True]
```



# 1. Declarando Listas com list()

python

```
# Criando uma lista vazia
lista_vazia = list()
print(lista_vazia)  # Output: []

# Criando uma lista a partir de uma sequência
lista_numeros = list([1, 2, 3, 4, 5])
print(lista_numeros)  # Output: [1, 2, 3, 4, 5]

# Criando uma lista a partir de uma string (a string será dividida em caracteres)
lista_caracteres = list("Python")
print(lista_caracteres)  # Output: ['P', 'y', 't', 'h', 'o', 'n']

# Criando uma lista a partir de um range
lista_range = list(range(5))
print(lista_range)  # Output: [0, 1, 2, 3, 4]
```

## Acessando Elementos

python

 Copiar

```
print(numeros[0])  # Acessa o primeiro elemento: 1
print(numeros[-1]) # Acessa o último elemento: 5
```



# Alguns métodos de lista

Com listas podemos fazer alguns métodos.

- **append()**: Adiciona um elemento ao final da lista.

python

 Copiar

```
numeros.append(6)
print(numeros) # Output: [1, 2, 3, 4, 5, 6]
```

- **remove()**: Remove a primeira ocorrência de um valor específico.

python

 Copiar

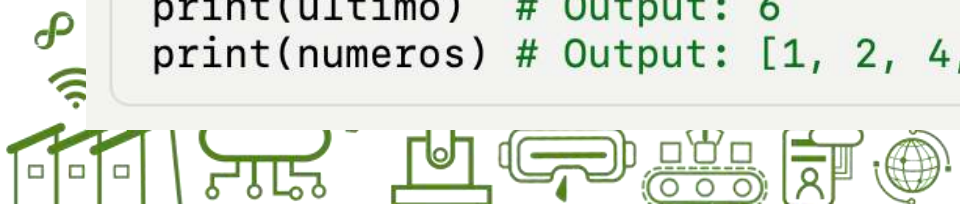
```
numeros.remove(3)
print(numeros) # Output: [1, 2, 4, 5, 6]
```

- **pop()**: Remove e retorna o elemento em uma posição específica (ou o último, se a posição não for especificada).

python

 Copiar

```
ultimo = numeros.pop()
print(ultimo) # Output: 6
print(numeros) # Output: [1, 2, 4, 5]
```



# Alguns métodos de lista

- **index()**: Retorna o índice da primeira ocorrência de um valor.

python

```
posicao = numeros.index(4)
print(posicao)  # Output: 2
```

## Exemplo com len() para uma Lista

python

```
# Definindo uma lista de frutas
frutas = ["maçã", "banana", "cereja", "laranja"]

# Obtendo o tamanho da lista com len()
tamanho = len(frutas)

# Exibindo o tamanho da lista
print("O tamanho da lista de frutas é:", tamanho)
```





# Tupla (tuple)

Tuplas são coleções ordenadas de elementos que são imutáveis (ou seja, uma vez criadas, não podem ser alteradas).

As tuplas são úteis quando você deseja que os dados sejam protegidos contra modificações acidentais.

## Criando uma Tupla

python

```
# Tupla de números
```

```
tupla_numeros = (1, 2, 3, 4, 5)
```

```
# Tupla com tipos mistos
```

```
tupla_mista = (1, "Olá", 3.14, True)
```

## Funções Importantes para Tuplas

- **count ( )**: Retorna o número de ocorrências de um valor específico.

python

```
ocorrencias = tupla_numeros.count(3)
print(ocorrencias)  # Output: 1
```

- **index ( )**: Retorna o índice da primeira ocorrência de um valor.

python

```
posicao = tupla_numeros.index(4)
print(posicao)  # Output: 3
```

Dicionários são coleções desordenadas de pares chave-valor, onde cada chave é única.

Eles são muito úteis para armazenar dados que precisam ser acessados por uma chave identificadora.

## Criando um Dicionário

python

 Copiar

```
# Dicionário com informações de uma pessoa
pessoa = {"nome": "Alice", "idade": 30, "cidade": "São Paulo"}
```

## Acessando Valores

python

 Copiar

```
print(pessoa["nome"]) # Acessa o valor associado à chave "nome": Alice
```



## Funções Importantes para Dicionários

- **keys()**: Retorna uma lista (ou view) das chaves no dicionário.

python

 Copiar

```
chaves = pessoa.keys()
print(chaves)  # Output: dict_keys(['nome', 'idade', 'cidade'])
```

- **values()**: Retorna uma lista (ou view) dos valores no dicionário.

python

 Copiar

```
valores = pessoa.values()
print(valores)  # Output: dict_values(['Alice', 30, 'São Paulo'])
```

- **items()**: Retorna uma lista (ou view) de pares chave-valor (tuplas).

python

 Copiar

```
itens = pessoa.items()
print(itens)  # Output: dict_items([('nome', 'Alice'), ('idade', 30), ('cidade', 'São Paulo')])
```

# Iteração (loop) em um dicionário

```
main.py +
1 variaveis = {
2     "numero": 10,
3     "palavra": "oi",
4     "dinheiro": 31.14,
5     "booleano": True,
6     "frutas": ["maçã", "banana", "uva", "pera", "abacate"]
7 }
8
9 for nome_chave, nome_item in variaveis.items():
10     print("o tipo da variável", nome_chave, "=", type(nome_item))
11
12 |
13 print(type(variaveis))
```

Ln: 12, Col: 5



Run



Share



\$

Command Line Arguments



```
o tipo da variável numero = <class 'int'>
o tipo da variável palavra = <class 'str'>
o tipo da variável dinheiro = <class 'float'>
o tipo da variável booleano = <class 'bool'>
o tipo da variável frutas = <class 'list'>
<class 'dict'>
```

*Esse loop passa por todos os elementos do dicionário (`_dict_`) de `tem`]:*

`_dict_.keys()` = pega cada chave somente

`_dict_.values()` = pega cada valor somente

`_dict_.items()` = pega a chave e o valor



@fpftech.educacional



# Obrigado!



    @fpftech.educacional