



IN

INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

AULA 01 – LISTAS E TUPLAS

O QUE IREMOS APRENDER

- 01** LISTAS E SUAS CARACTERÍSTICAS
- 02** DEFININDO E EXIBINDO VALORES DE UMA LISTA
- 03** MÉTODOS PARA MANIPULAÇÃO DE LISTAS
- 04** TUPLAS E SUAS CARACTERÍSTICAS
- 05** DEFININDO E EXIBINDO VALORES DE UMA TUPLA
- 06** MÉTODOS PARA MANIPULAÇÃO DE TUPLAS
- 07** ATIVIDADES PRÁTICAS
- 08** PEDINDO AJUDA AO CHATGPT

For

A sintaxe básica do for como estrutura de repetição é a seguinte:

```
for variavel in range(inicio, fim, passo):
```

variavel: Variável que assume o valor de cada iteração.

range(inicio, fim, passo):

Gera uma sequência numérica de inicio até fim-1 com o incremento definido por passo.

Função range()

range() - Criando uma Sequência Numérica

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

A função range() é uma ferramenta útil para gerar sequências numéricas em um for.

Break no for

```
numero_procurado = 7

for i in range(1, 11):
    if i == numero_procurado:
        print(f"Número {numero_procurado} encontrado!")
        break
    print(i)
```

Neste exemplo, o loop for é interrompido assim que o número procurado é encontrado. Se o número 7 estiver presente na sequência de 1 a 10, a mensagem será exibida, e o loop será interrompido.

Continue no for

Exemplo: Pular Números Ímpares

```
for i in range(1, 11):
    if i % 2 != 0:
        continue
    print(i)
```

Neste exemplo, o continue é usado para pular os números ímpares. Se a condição `i % 2 != 0` for verdadeira, o continue é acionado, e o código abaixo não é executado nessa iteração.

Break e continue

Exemplo: Parar o Loop quando Encontrar um Número Primo

```
for num in range(2, 20):
    for i in range(2, num):
        if num % i == 0:
            # Não é um número primo, pular para o próximo
            break
    else:
        # Chegou aqui apenas se não houve break no loop interno
        print(f"{num} é um número primo!")
```

Neste exemplo, o loop externo percorre números de 2 a 19. O loop interno verifica se o número é primo. Se encontrar um divisor, o break é acionado, e o loop interno é encerrado. Se nenhum divisor é encontrado, a mensagem é impressa.

Listas e suas características

Em Python, além das variáveis que podem armazenar apenas um tipo de dados, temos um “tipo de variável” que pode armazenar muitas coisas de uma vez: as **listas**.

Uma lista em Python <**class 'list'**> é uma estrutura de dados que pode guardar números inteiros e floats, textos (strings), valores lógicos (booleanos) e muitos outros.

Assim, com as listas em Python, é possível armazenar vários **diferentes tipos de dados** de uma só vez, reduzindo o número de variáveis que precisariam ser definidas em nosso código.



Listas e suas características

Veja a seguir alguns exemplos com listas em Python:

```
# Definindo uma lista de números
lista_de_numeros = [1,2,3,4,5]

# Definindo uma lista de letras
lista_de_letras = ['a', 'i', 'u', 'e', 'o']

# Definindo uma lista de valores lógicos
lista_de_logicos = [True, False, False, True]
# Lista com diferentes tipos de dados
lista_mista = ['Gabriel', 12, True]
```

ATIVIDADE PRÁTICA 1

Faça a definição de uma lista contendo os números de 1 até 5. Finalmente, utilize o print() para exibir os valores da lista.

ATIVIDADE PRÁTICA 2

Faça a definição de uma lista contendo as vogais.
Finalmente, utilize o print() para exibir os valores da lista.

Definindo e exibindo valores de uma lista

Durante a realização das atividades, o que você percebeu quando utilizou o `print()` para exibir os valores da lista? Como esses valores foram exibidos na tela (terminal)?

CÓDIGO PYTHON

```
1 # Lista com os cinco primeiros números pares  
2 numero_pares = [0,2,4,6,8]  
3 print(numero_pares)
```

SAÍDA NO TERMINAL

```
PS C:\Users\tiinf\Desktop\Estudo Dirigido - MongoDB> python ed01.py  
[0, 2, 4, 6, 8]  
PS C:\Users\tiinf\Desktop\Estudo Dirigido - MongoDB>
```

Note que os valores da lista sempre serão exibidos no formato em que ela foi definida, ou seja, entre os colchetes e separados por vírgula, assim como definido no código.

Definindo e exibindo valores de uma lista

Imagine, no entanto, a seguinte situação: uma lista 'notas' que representa as notas finais dos três semestres letivos de um aluno:

notas = [10.0, 9.8, 8.7].

Como você faria para mostrar ao usuário apenas a nota 9.8?



Acessando valores de uma lista

Se quisermos exibir apenas um dos valores guardados na lista, basta especificarmos a posição (**índice**) da lista onde o **valor** desejado se encontra.

NOTAS <CLASS 'LIST'>			
10.0	9.8	8.7	valores
0	1	2	índices

Assim sendo, a nota **10.0** encontrar-se no **índice 0**, enquanto a nota 9.8 no **índice 1**, e, finalmente, a 8.7 no **índice 2**. Essa lógica se aplica a **qualquer lista** que você criar em Python!

ATIVIDADE PRÁTICA 3

Defina uma lista com 5 itens que tenha, pelo menos, 3 classes diferentes. Utilize o print() para exibir o terceiro elemento dessa lista.

Acessando valores de uma lista

Em Python, podemos utilizar número negativos para acessar os elementos de uma lista. Assim, o índice -1 representa o último elemento, enquanto o -2 o penúltimo, e o -3 o antepenúltimo, e assim por diante, seja qual for o tamanho da lista:



```
1 # Lista com as cinco primeiras letras do alfabeto
2
3 letras = ['a', 'b', 'c', 'd', 'e']
4
5 print(letras[-1]) # letra 'e'
6 print(letras[-2]) # letra 'd'
7 print(letras[-3]) # letra 'c'
8 print(letras[-4]) # letra 'b'
9 print(letras[-5]) # letra 'a'
```

Métodos para manipulação de listas

Em python existem diversos métodos de manipulação de listas, mas os principais são:

Finalidade	Sintaxe	Função
Adição de itens	.append()	Adicionar um elemento ao final da lista
Adição de itens	.insert()	Adicionar um elemento na posição informada
Remoção de itens	.remove()	Remover um elemento de uma lista através de seu valor
Remoção de itens	.pop()	Remover um elemento de uma lista através de seu índice

Métodos para manipulação de listas

Veja alguns exemplos práticos da adição de itens:



```
1 # Adicionando itens ao final da lista com método .append()
2 numeros = [1, 2, 3, 4, 5]
3
4 numeros.append(6)
5
6 print(numeros) # [1, 2, 3, 4, 5, 6]
7
8 # Adicionando itens em dada posição da lista com método .insert()
9 numeros = [10, 30, 40, 50]
10 letras = ['a', 'e', 'o', 'u']
11 pesos = [1.2, 3.4, 5.3, 6.7]
12
13 # Inserindo valores nas respectivas posições das listas
14 numeros.insert(1, 20) # Inserindo 20 na posição 1
15 letras.insert(2, 'i') # Inserindo 'i' na posição 2
16 pesos.insert(2, 4.0) # Inserindo 4.0 na posição 3
17
18 # Exibindo as inserções dos valores
19 print(numeros) # [10, 20, 30, 40, 50]
20 print(letras) # ['a', 'e', 'i', 'o', 'u']
21 print(pesos) # [1.2, 3.4, 4.0, 5.3, 6.7]
```



Métodos para manipulação de listas

Agora veja alguns exemplos práticos da remoção de itens:



```
1  notas = [
2      9.0, # indice 0
3      8.7, # indice 1
4      9.9, # indice 2
5      8.7, # indice 3
6      7.9 # indice 4
7  ]
8
9  notas.pop(0) # removendo a nota 9.0
10 notas.pop(1) # removendo a nota 9.9
11 notas.pop(2) # removendo a nota 7.9
12
13 print(notas) # [8.7, 8.7]
```

```
# Lista com nomes dos alunos finalistas
finalistas_obmep = [
    "Maria da Silva Santos",
    "Pedro Henrique Oliveira",
    "Ana Carolina Pereira Costa",
    "João Fernandes Rodrigues",
    "Laura Nunes Almeida"
]

# Utilizando o método .remove() para remover um finalista
finalistas_obmep.remove("Ana Carolina Pereira Costa")

# Exibindo alista após a remoção do finalista
print(finalistas_obmep)
```

Iterando listas em Python com FOR

Anteriormente, você aprendeu alguns métodos para manipulação de listas em Python, sendo: `.append()`, `.insert()`, `.pop()` e `.remove()`. Agora aprenderemos um pouco sobre iterações de listas em Python com o laço FOR. Para isso, é importante lembrar que ele percorre coleções e, por isso, podemos iterar listas, pois elas também são coleções.



Iterando listas em Python com FOR

No exemplo ao lado, definimos uma lista de compras e adicionamos valores a ela. Após o `print()`, utilizamos o laço **FOR** para percorrer cada item da lista de compras:

SAÍDA
DE
DADOS

```
LISTA DE COMPRAS
[ ] 2 pcts. de Arroz
[ ] 6 pcts. de Feijão
[ ] 2 pcts. de Farinha de Mandioca
[ ] 4kg de Linguiça Calabresa
[ ] 4kg de Charque
[ ] 2kg de Bacon de Barriga
[ ] 2kg de Pé e Orelha de porco
[ ] 1 pct de folha de louro
[ ] 6 mói de couve
[ ] 5kg de laranja
```

```
# Definição da Lista de Compras
lista_compras = [
    "2 pcts. de Arroz",
    "6 pcts. de Feijão",
    "2 pcts. de Farinha de Mandioca",
    "4kg de Linguiça Calabresa",
    "4kg de Charque",
    "2kg de Bacon de Barriga",
    "2kg de Pé e Orelha de porco",
    "1 pct de folha de louro",
    "6 mói de couve",
    "5kg de laranja",
]

print("LISTA DE COMPRAS", end='\n\n')

# Percorrendo a Lista de Compras
for item in lista_compras:
    print("[ ]", item)
```

IN

Tuplas e suas características:

Tuplas são estruturas semelhantes a **listas**, mas com uma distinção importante: são **imutáveis**. Ou seja, uma vez que você cria uma tupla, não pode modificar seus elementos. Isso pode ser útil em situações em que você deseja garantir a integridade dos dados.

Vamos explorar como declarar e acessar elementos em tuplas, bem como entender as situações em que a imutabilidade das tuplas é uma **vantagem**.



Definindo e exibindo valores de uma tupla:

As **listas** e **tuplas** são estruturas de dados em Python que armazenam coleções de elementos. A principal diferença entre elas reside na mutabilidade. Listas são **mutáveis**, o que significa que seus elementos **podem ser modificados** após a criação. Em contrapartida, tuplas são **imutáveis**, tornando seus elementos **fixos** após a declaração.

Além disso, a sintaxe das tuplas é **diferente** das listas, observe o exemplo abaixo:

```
# Lista mutável
minha_lista = [1, 2, 3, 4, 5]
minha_lista[0] = 10 # Modificando um elemento
print(minha_lista) # Saída: [10, 2, 3, 4, 5]

# Tupla imutável
minha_tupla = (1, 2, 3, 4, 5)
# minha_tupla[0] = 10 # Isso resultará em um erro, pois as tuplas são imutáveis
print(minha_tupla) # Saída: (1, 2, 3, 4, 5)
```

Métodos para a Manipulação de Tuplas

Embora as tuplas sejam imutáveis, existem alguns métodos que podem ser utilizados para realizar operações e manipulações específicas. Aqui estão alguns:



```
1 # Criando uma tupla
2 frutas = ("maçã", "banana", "laranja", "abacaxi")
3
4 # Método index(): Retorna o índice do primeiro elemento especificado
5 indice_laranja = frutas.index("laranja")
6 print("Índice da laranja:", indice_laranja) # Saída: 2
7
8 # Método count(): Retorna o número de ocorrências de um elemento
9 quantidade_bananas = frutas.count("banana")
10 print("Qunatidade de bananas:", quantidade_bananas)
11
12 # Desempacotando uma tupla
13 maca, banana, laranja, abacaxi = frutas
14 print("Fruta 1:", maca)
15 print("Fruta 2:", banana)
16 print("Fruta 3:", laranja)
17 print("Fruta 4:", abacaxi)
```



PEDINDO AJUDA AO CHATGPT



You

pode me ensinar a fazer listas em python de forma resumida?

< 3 / 3 >



ChatGPT

Claro! Em Python, você pode criar listas de diversas maneiras. Aqui estão algumas formas resumidas de criar listas:

1. Usando colchetes `[]`:

Você pode criar uma lista simplesmente colocando os elementos entre colchetes, separados por vírgulas. Por exemplo:

```
python
```

Copy code

```
minha_lista = [1, 2, 3, 4, 5]
```

2. Usando a função `list()`:

Você pode converter outros tipos de dados iteráveis (como strings, tuplas, conjuntos, etc.) em listas usando a função `list()`. Por exemplo:

```
python
```

Copy code

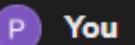
```
minha_string = "hello"
lista_da_string = list(minha_string)
```



3. Usando compreensão de listas (list comprehension):

IN

PEDINDO AJUDA AO CHATGPT



You

me indique 5 exercícios sobre listas



ChatGPT

Claro! Aqui estão cinco exercícios envolvendo listas em Python que podem ajudar a praticar e aprimorar suas habilidades:

1. Média de uma lista:

Escreva uma função em Python que receba uma lista de números como entrada e retorne a média dos números na lista.

2. Remover duplicatas:

Escreva uma função que receba uma lista como entrada e retorne uma nova lista com todas as duplicatas removidas, mantendo a ordem original dos elementos.

3. Inversão de lista:

Escreva uma função que receba uma lista como entrada e retorne uma nova lista com os elementos na ordem inversa.

4. Ordenação parcial:

Escreva uma função que receba uma lista de números como entrada e retorne uma nova lista com os números pares ordenados de forma crescente e os ímpares mantidos na mesma ordem original.

5. Soma de elementos consecutivos:

Escreva uma função que receba uma lista de números como entrada e retorne uma lista contendo a soma de cada par de elementos consecutivos da lista original.

IN

ATIVIDADE PRÁTICA 4

Crie uma tupla para representar as informações de três palestrantes, cada uma contendo o nome, o tema da palestra e a instituição à qual estão vinculados.

Exiba na tela as informações do terceiro palestrante, incluindo nome, tema da palestra e instituição.

DESAFIO PRÁTICO

Suponha que você está gerenciando uma competição esportiva e tem uma lista de tuplas representando os resultados das equipes em diferentes modalidades. Cada tupla contém o nome da equipe, seguido por uma lista de pontuações obtidas em cada rodada da competição.

- 1.Calcule a média das pontuações de cada equipe e armazene esses valores em uma nova lista chamada medias.
- 2.Ordene a lista medias em ordem decrescente.
- 3.Crie uma nova lista chamada classificacao que contém tuplas, onde cada tupla contém o nome da equipe e sua média de pontuações.
- 4.Exiba na tela a classificação final das equipes, mostrando o nome da equipe e sua média, da equipe com a pontuação mais alta para a mais baixa.

**SE LIGA NO
CONTEÚDO DA
PRÓXIMA AULA!**

AULA 02 DE PYTHON:
DICIONÁRIOS E SETS



INFINITY SCHOOL
VISUAL ART CREATIVE CENTER

Aula 02 - DICIONÁRIO E SETS

SETS E SUAS CARACTERÍSTICAS

OPERAÇÕES MATEMÁTICAS COM SETS

DICIONÁRIOS E SUAS CARACTERÍSTICAS

PERCORRENDO DICIONÁRIOS

OPERAÇÕES SUPORTADAS NOS DICIONÁRIOS



IN

INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

ATÉ A PRÓXIMA...