

# Python - Listas, Dicionários e Tuplas



Disciplina: Programação para Internet

Prof. Bruno Gomes



@profbrunogomes



# Aula de Hoje

2

- » **Listas**
- » **Dicionários**
- » **Tuplas**



# 1.

## Listas

# Lista em Python

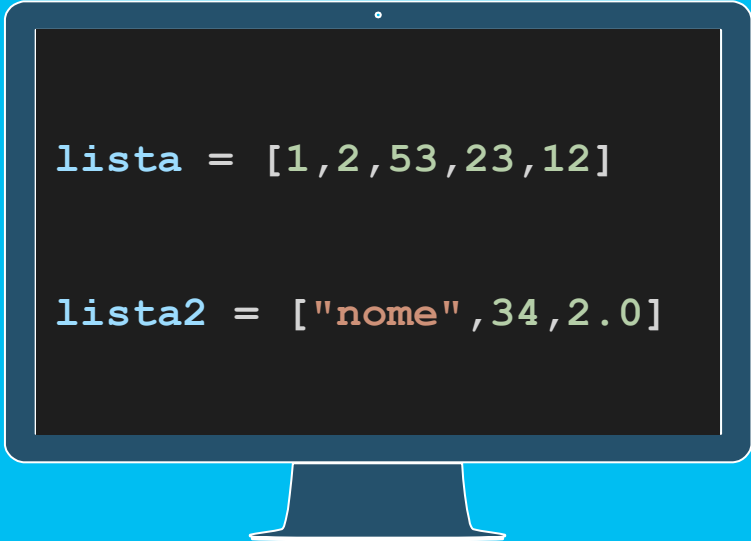
4

## Sintaxe



```
[v1, v2, v3]
```

## Exemplos



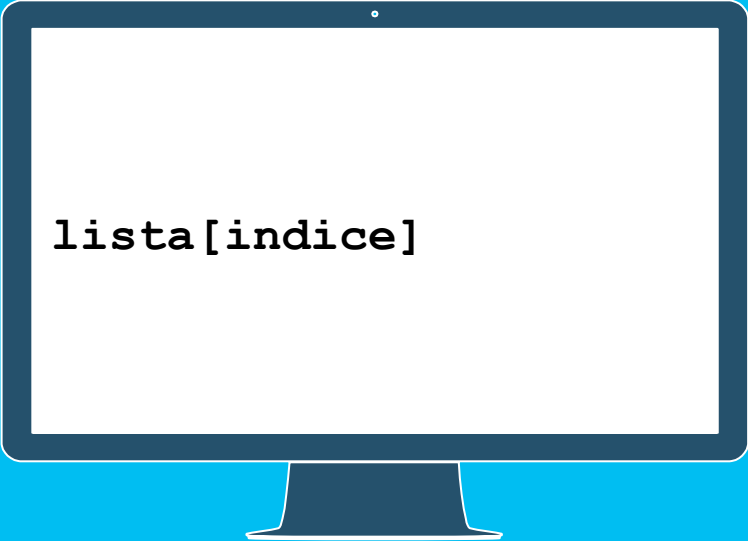
```
lista = [1,2,53,23,12]
```

```
lista2 = ["nome",34,2.0]
```

# Acessando valor de uma lista

5

## Sintaxe



```
lista[indice]
```

## Exemplos



```
lista = [1,2,53,23,12]
```

```
valor = lista[1]
```

```
print(valor)
```

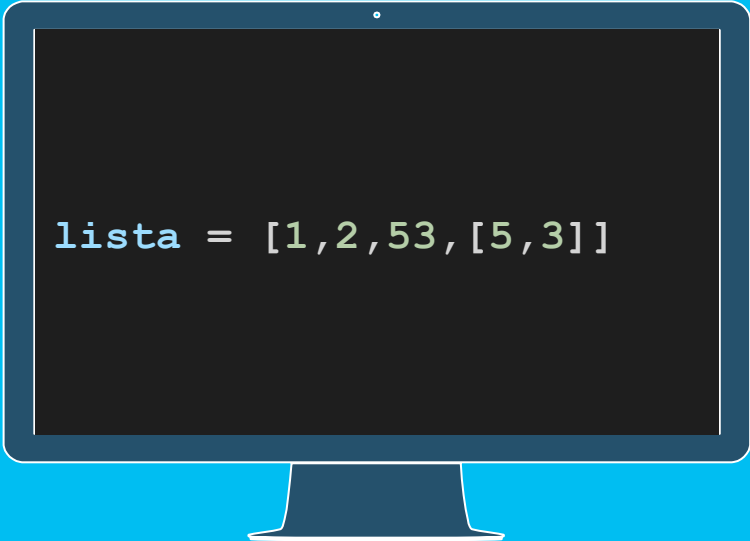
```
lista2 = ["nome",34,2.0]
```

```
print(lista2[2])
```

# Lista dentro de Lista

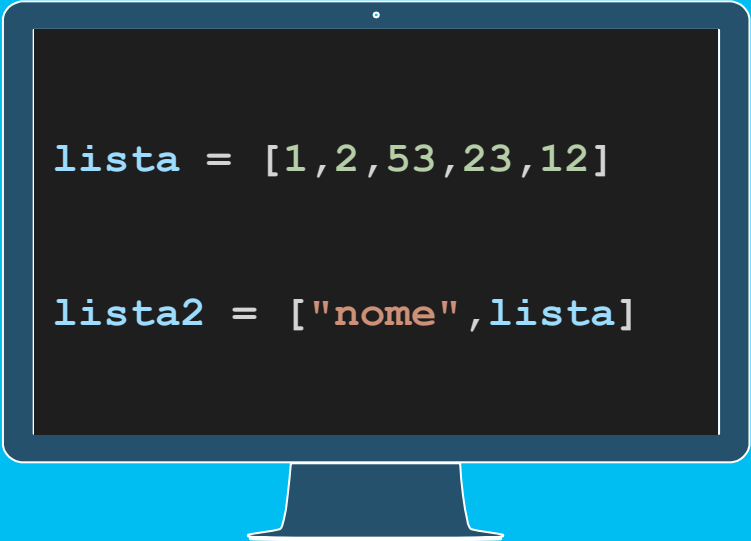
6

## Exemplo



```
lista = [1,2,53,[5,3]]
```

## Exemplo




```
lista = [1,2,53,23,12]
```

```
lista2 = ["nome",lista]
```

# Operações

7

## Concatenação: $\pm$



```
lista = [1,2,53]
lista2 = [5,3,2]

lista3 = lista + lista2
#resultado: [1,2,53,5,3,2]

lista4 = lista + [7,3,5.0]
#resultado: [1,2,53,7,3,5.0]
```



# Operador In

8

- » Verifica se algum elemento está na lista:
  - » retorna true ou false

```
lista = [4,5,10]  
print(5 in lista)
```



# Função len

9

- » Retorna o tamanho da lista:

```
lista = [4,5,10]  
print(len(lista))
```

# Função max

10

- » Retorna o maior valor da lista:

```
lista = [4,10,5]  
print(max(lista))
```

# Função min

11

- » Retorna o menor valor da lista:

```
lista = [4,10,5]  
print(min(lista))
```

# Função append

12

- » Adiciona um elemento à lista:

```
lista = [4,10,5]  
lista.append(7)  
print(lista)
```

# Função count

13

- » Conta quantas vezes um elemento aparece na lista:

```
lista = [4,10,5,4]  
print(lista.count(4))
```

# Função extend

14

- » Distribui elementos de outra lista na lista atual:

```
lista = [4,10,5]  
lista.extend([14,2])  
print(lista)
```

# Função index

15

- » Retorna o índice de um determinado elemento:

```
lista = [4,10,5]  
print(lista.index(10))
```



# Função insert

16

- » Adiciona um elemento em uma posição específica:

```
lista = [4,10,5]  
lista.insert(1, 100)  
print(lista)
```

# Função remove

17

- » Adiciona um elemento em uma posição específica:

```
lista = [4,10,5]  
lista.remove(10)  
print(lista)
```

# Função reverse

18

» Inverte a lista:

```
lista = [4,10,5]  
lista.reverse()  
print(lista)
```

# Função sort

19

» Ordena a lista:

```
lista = [4,10,5]  
lista.sort()  
print(lista)
```

## Listas dentro de Lista (matriz)

20

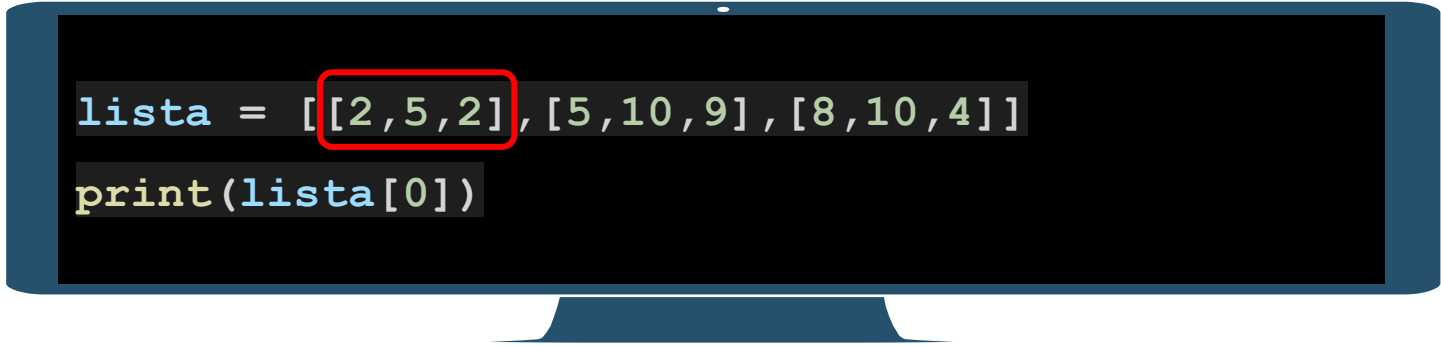
- » É possível salvar uma outra lista em qualquer posição de uma lista

```
lista = [[2,5,2],[5,10,9],[8,10,4]]
```

# Listas dentro de Lista (matriz)

21

» Retornando uma lista:



```
lista = [[2,5,2],[5,10,9],[8,10,4]]  
print(lista[0])
```

# Listas dentro de Lista (matriz)

22

» Retornando um elemento:

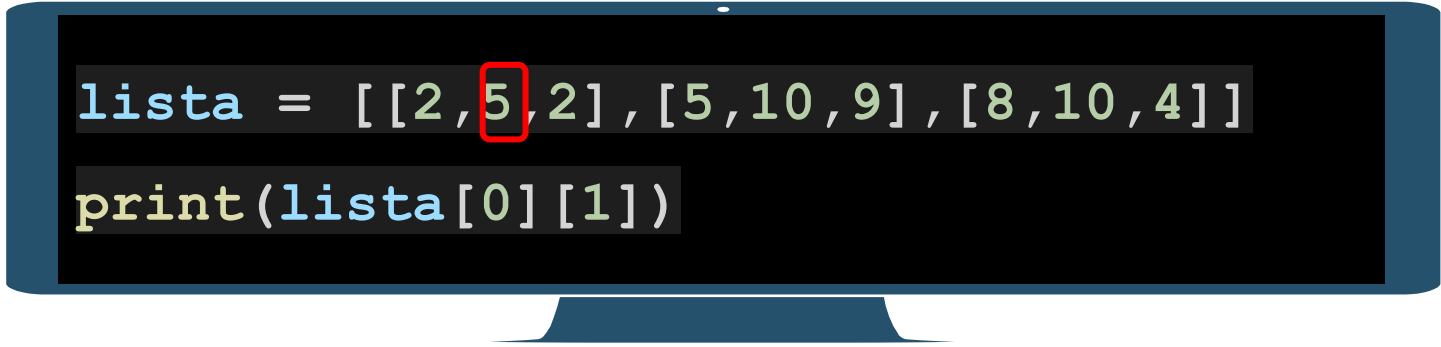
```
lista = [[2, 5, 2], [5, 10, 9], [8, 10, 4]]  
resultado = lista[0]  
print(resultado[1])
```



## Listas dentro de Lista (matriz)

23

» Retornando um elemento (simplificando):



```
lista = [[2, 5, 2], [5, 10, 9], [8, 10, 4]]  
print(lista[0][1])
```

# Dúvidas?



# Vamos Praticar!!

Lista de Exercícios



# 2.

## Dicionário

# Dicionário

27

- » Conhecidas como tabelas de hash (*hash tables*);
- » Cria um mapeamento: coleção de objetos que são armazenados por uma chave;
  - » Chave -> valor associado (qualquer objeto python);
- » Exemplo:
  - » {chave1: valor1, chave2: valor2}

## Observação

28

- » A lista considera a posição relativa de um elemento, já o dicionário leva em conta a chave.

# Dicionário

29

» Criada com { }:

```
dicionario = {"aluno1":80, "aluno2": 85, "aluno3":50}
```



# Dicionário

30

- » Impressão através da chave (utiliza colchetes):

```
dicionario = {"aluno1":80, "aluno2": 85, "aluno3":50}  
print(dicionario["aluno1"])
```

# Dicionário

31

» Adicionando uma nova chave e valor:

```
dicionario = {"aluno1":80, "aluno2": 85, "aluno3":50}  
dicionario["aluno4"] = 100  
print(dicionario)
```

# Dicionário

32

» Substituindo um valor:

```
dicionario = {"aluno1":80, "aluno2": 85, "aluno3":50}  
dicionario["aluno2"] = 90  
print(dicionario["aluno2"])
```

# Dicionário - Função clear

33

» Zera o dicionário:

```
dicionario = {"aluno1":80, "aluno2": 85, "aluno3":50}  
dicionario.clear()  
print(dicionario)
```

# Dicionário - Função len

34

- » Retorna o tamanho do dicionário:

```
dicionario = {"aluno1":80, "aluno2": 85, "aluno3":50}  
print(len(dicionario))
```

# Dicionário - Função keys

35

- » Retorna apenas as chaves do dicionário:

```
dicionario = {"aluno1":80, "aluno2": 85, "aluno3":50}  
print(dicionario.keys())
```

# Dicionário - Função values

36

- » Retorna apenas os valores do dicionário:

```
dicionario = {"aluno1":80, "aluno2": 85, "aluno3":50}  
print(dicionario.values())
```



# Dicionário - Função items

37

- » Retorna todas as combinações de chaves e valores:

```
dicionario = {"aluno1":80, "aluno2": 85, "aluno3":50}  
print(dicionario.items())
```

# Dicionário - Função update

38

- » Atualiza (concatena) um dicionário com combinações de chaves e valores de outro dicionário:

```
dicionario = {"aluno1":80, "aluno2": 85, "aluno3":50}  
dicionario2 = {"aluno4":55, "aluno5": 100}  
dicionario.update(dicionario2)  
print(dicionario)
```

## Dicionário - Dicionário vazio

39

- » É possível criar um dicionário vazio e depois adicionar a combinação de chave e valor individualmente:

```
diccionario = {}  
diccionario["aluno1"] = 80  
diccionario["aluno2"] = 85  
print(diccionario)
```

## Dicionário - Dicionário com chaves diferentes 40

- » É possível criar um dicionário com chaves de tipos diferentes:

```
dicionario = {"aluno1":80, 50: "nota"}  
print(dicionario)
```

## Dicionário - Lista como valores

41

- » É possível adicionar como valor uma lista:

```
dicionario = {"aluno1": [100, 60], "aluno2": [85, 84]}  
print(dicionario["aluno1"])
```

## Dicionário - Lista como valores

42

- » Retornando apenas um elemento da lista de uma chave:

```
dicionario = {"aluno1": [100, 60], "aluno2": [85, 84]}  
print(dicionario["aluno1"][0])
```

## » Organização do Dicionário:

```
diccionario = {  
    "aluno1": [100, 60],  
    "aluno2": [85, 84]  
}  
print(diccionario["aluno1"][0])
```



# Dicionário Aninhado

44

- » Um dicionário dentro de outro dicionário:

```
dicionario = {  
    "aluno1": {"nota1": 100, "nota2": 90},  
    "aluno2": {"nota1": 85, "nota2": 70}  
}  
  
print(dicionario["aluno1"]["nota2"])
```

# Dicionário - operação del

45

- » Deleta o dicionário:

```
dicionario = {"aluno1":80, "aluno2": 85, "aluno3":50}  
del dicionario  
print(dicionario)
```

# Dúvidas?



# Vamos Praticar!!

Lista de Exercícios



# 3.

## Tuplas

# Tuplas

49

- » Semelhante a lista, mas é imutável (não pode ser alterada)
- » Exemplo de uso: dias da semana; datas...
- » Uso de parênteses e vírgulas para separar os elementos:

```
tupla = (item1, item2, item3)
```

# Tupla

50

- » Criando uma tupla:

```
tupla = ("Segunda", "Terça", "Quarta")  
print(tupla)
```



# Tupla

51

- » Criando uma tupla:

```
tupla = ("Segunda", "Terça", "Quarta")  
print(tupla)
```

# Tupla

52

- » Retornando um elemento:

```
tupla = ("Segunda", "Terça", "Quarta")  
print(tupla[0])
```

# Tupla - Funções Suportadas

53

- » len
- » index

# Tupla

54

» Convertendo para uma lista:

```
tupla = ("Segunda", "Terça", "Quarta")  
lista = list(tupla)  
print(lista)
```

# Dúvidas?



# Vamos Praticar!!

Lista de Exercícios

