

Estrutura de Dados

Listas e Tuplas

Prof. Dr. Danilo Barbosa

Conteúdo

- Conceito de Listas
- Conceito de tuplas
- Concatenação de tuplas
- Empacotamento e despacotamento de tuplas
- Atividade
- Referências



Conceito de Listas

A lista é uma estrutura de dados que agrupa objetos sequencialmente.

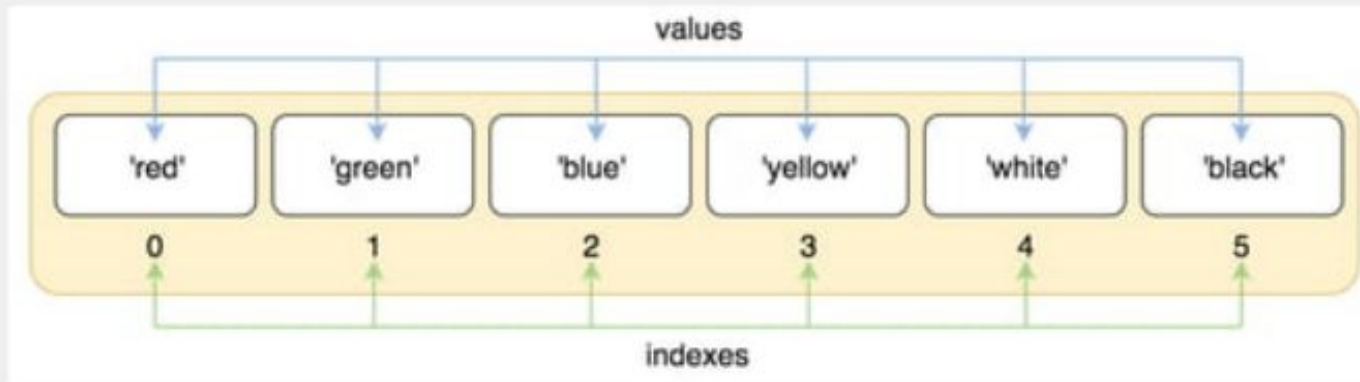


Figura: Ilustração de uma estrutura de lista com 6 objetos do tipo **str** (texto)

- Cada item da lista é identificado por um **índice**.
- O primeiro item da lista tem índice 0



Criação de Listas

- Uma lista é definida em Python como uma sequência de valores ou objetos entre colchetes.
- A lista pode ser atribuída a uma variável, e então todos os objetos da lista serão acessíveis por meio dessa única variável.
- É possível mostrar o conteúdo da lista com **print**.

Exemplo de código

```
lista = [10, 20, 30, 40] #lista com 4 objetos int  
print(lista) #mostra a lista no terminal
```



Criação de Listas

Exemplo de código – lista de **int**

```
lista = [10, 20, 30, 40] #lista com 4 objetos int  
print(lista) #mostra a lista no terminal
```

Exemplo de código – lista de **str**

```
#lista com 3 objetos str  
lista = ['Filipe', 'Joao', 'Ana']  
print(lista) #mostra a lista no terminal
```

Exemplo de código – lista de **str e int**

```
#lista com 6 objetos str e int  
lista = ['Filipe', 10, 'Joao', 20, 'Ana', 30]  
print(lista) #mostra a lista no terminal
```



Índice de acesso de Listas

Para acessar um item da lista, você deve digitar o nome da variável seguido pelo índice do item entre colchetes, com a seguinte sintaxe: <variável>[<índice>].

Exemplo de código

```
lista = ['Filipe', 'Joao', 'Ana']  
print(lista[0]) #escreve Filipe  
print(lista[1]) #escreve Joao  
print(lista[2]) #escreve Ana
```

Lembre-se sempre de que o primeiro índice é zero!



Atribuições de Listas

É possível modificar o valor ou objeto contido em uma posição da lista fazendo uma atribuição com a seguinte sintaxe: `<variável>[<índice>]=<valor>`.

Exemplo de código

```
lista = ['Filipe', 'Joao', 'Ana']  
print(lista) #mostra a lista  
lista[1] = 'Matheus' #substitui Joao por Matheus  
print(lista) #mostra a lista modificada
```



Iterações de Listas

É possível fazer uma iteração com **for** para acessar cada objeto da lista de duas formas diferentes:

- **Utilizando índices** – um **for** é utilizado para iterar pelos índices da lista, e os acessos ou modificações são feitos com índices.
- **Sem utilizar índices** – um **for** é utilizado para iterar pelos itens da lista diretamente, sem usar índices.



Iterações de Listas

Utilize a função **len** para obter o tamanho da lista.

Iteração com índice

```
lista = ['Filipe', 'Joao', 'Ana']  
for i in range(len(lista)):  
    print(lista[i]) #mostra cada item da lista
```

Iteração sem índice

```
lista = ['Filipe', 'Joao', 'Ana']  
for item in lista:  
    print(item) #mostra cada item da lista
```



Inserção de elementos da Listas

É possível inserir objetos na lista em tempo de execução utilizando o método **append**, com a seguinte sintaxe:
`<variável>.append(<valor>)`

Exemplo de código

```
lista = ['Filipe', 'Joao', 'Ana']  
print(lista) #Mostra a lista  
  
#Insere 'Matheus' no final da lista  
lista.append('Matheus')  
  
print(lista) #Mostra a lista modificada
```



Remoção de elementos da Listas

É possível remover objetos da lista em tempo de execução utilizando o operador **del**, com a seguinte sintaxe: `del <variável>[<índice>]`

Exemplo de código

```
lista = ['Filipe', 'Joao', 'Ana']  
print(lista) #Mostra a lista  
del lista[1] #Remove 'Joao' da lista  
print(lista) #Mostra a lista modificada
```



Fatias da Listas

Exemplo de código – *slicing*

```
lista = [10, 24, 3, 5, 8, 29, 11] #cria lista  
fatia = lista[1:5] #obtem fatia do 24 ate o 8  
print(fatia) #mostra a fatia
```



Reversão de Listas

É possível obter uma lista “de trás para frente” com a seguinte sintaxe: `<variável>[::-1]`

Exemplo de código

```
lista = [10, 24, 3, 5, 8, 29, 11] #cria lista  
  
lista_reversa = lista[::-1]  
  
#mostra lista de tras para frente  
print(lista_reversa)
```



Ordenação de Listas

É possível ordenar uma lista de números com o método **sort**.

- Em ordem crescente: `<variável>.sort()`
- Em ordem decrescente:
`<variável>.sort(reverse=True)`

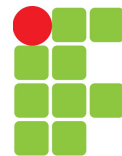
Exemplo de código

```
lista = [10, 24, 3, 5, 8, 29, 11] #cria lista
lista.sort() #ordena lista
print(lista) #mostra lista ordenada
```



Conceito de tuplas

- Tuplas podem ser vistas como **listas em Python**. No entanto, elas **não podem ser modificadas**.
- Tuplas são ideais para representar **listas de valores constantes** e para realizar **operações de empacotamento e desempacotamento de valores**.



Conceito de tuplas

As tuplas podem ser criadas de maneira semelhantes a listas

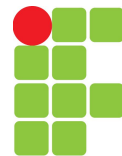
- `tupla = ()`
- `tupla = tuple()`
- `tupla = (1,2,3)`
- `lista = [1,2,3]`
- `tupla = tuple(lista)`
- `tupla = 1, 2, 3`



Conceito de tuplas

- Tuplas suportam a maior parte das operações de listas, tais como fatiamento, indexação, tamanho
 - `tupla[0]`
 - `tupla[1:]`
 - `tupla[:]`
 - `len(tupla)`
- ```
for e in tupla:
 print(e)

for i, e in enumerate(tupla):
 print(f"e[{i}]= {e}")
```



# Concatenação de tuplas

Embora não possamos alterar uma tupla, podemos concatená-la

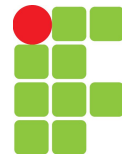
- $t1 = (1,2,3)$
- $t2 = 4, 5, 6$
- $t1+t2$
- $(1,2,3,4,5,6)$



## Concatenação de tuplas

Mas essa tupla contiver uma lista ou outro objeto que possa ser alterado, esse objeto pode ser alterado

- `t3 = (1, [2, 3, 4])`
- `len(t3)`
- `t3[1]`
- `t3[1].append(8)`
- `t3`



# empacotamento e desempacotamento de tuplas

## Empacotamento / Desempacotamento

- $a = 1, 2$
- $c, d = (3, 4)$
- $*a, b = [1, 2, 3, 4, 5]$
- $a, *b = [1, 2, 3, 4, 5]$



## Atividade

- Cada equipe deve fazer dois códigos de tuplas e listas da vídeo aula. Em seguida, enviar o print das atividades para o google sala de aula .
- Quatro equipes serão sorteadas para explicar o código durante a aula



## Atividade

- Equipe 1 - questões 76 e 77 Igor / Ian
- Equipe 2- questões 78 e 79 Pedro
- Equipe 3 - questões 80 e 81 Vinicius / Luiz Felipe
- Equipe 4 - questões 82 e 83 Isabela / Leandro
- Equipe 5 - questões 84 e 85 Débora / Isabelly





## Atividade

- Equipe 6 - questões 86 e 87 Brenda / Lane / Alana
- Equipe 7 - questões 88 e 89 Paulo
- Equipe 8 - questões 90 e 91 Luiz/ Ronaldo / Gabriel
- Equipe 9 - questões 92 e 93
- Equipe 10 - questões 94 e 95



# Atividade

- Desenvolva esses códigos em Python:

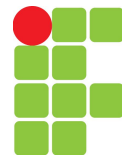
**DESAFIO 072**

💡 Crie um programa que tenha uma **tupla** totalmente preenchida com uma contagem por extenso, de **zero** até **vinte**.

Seu programa deverá ler um número pelo teclado (**entre 0 e 20**) e mostrá-lo por **extenso**.

```
cont = ('zero', 'um', 'dois', 'três', 'quatro',
 'cinco', 'seis', 'sete', 'oito', 'nove',
 'dez', 'onze', 'doze', 'treze', 'quatorze',
 'quinze', 'dezesesseis', 'dezesesete', 'dezoito',
 'dezenove', 'vinte')

while True:
 núm = int(input('Digite um número entre 0 e 20: '))
 if 0 <= núm <= 20:
 break
 print('Tente novamente. ', end='')
print(f'Você digitou o número {cont[núm]}')
```



# Atividade

- Desenvolva esses códigos em Python:

**DESAFIO 073**

💡 Crie uma **tupla** preenchida com os **20 primeiros** colocados da Tabela do **Campeonato Brasileiro de Futebol**, na ordem de colocação. Depois mostra:

- A) Apenas os **5 primeiros** colocados.
- B) Os **últimos 4** colocados da tabela.
- C) Uma lista com os times em **ordem alfabética**.
- D) Em que **posição** na tabela está o time da **Chapecoense**.

```
times = ('Corinthians', 'Palmeiras', 'Santos', 'Grêmio',
 'Cruzeiro', 'Flamengo', 'Vasco', 'Chapecoense',
 'Atlético', 'Botafogo', 'Atlético-PR', 'Bahia',
 'São Paulo', 'Fluminense', 'Sport Recife',
 'EC Vitória', 'Coritiba', 'Avaí', 'Ponte Preta',
 'Atlético-GO')

print('-=' * 15)
print(f'Lista de times do Brasileirão: {times}')
print('-=' * 15)
print(f'Os 5 primeiros são {times[0:5]}')
print('-=' * 15)
print(f'Os 4 últimos são {times[-4:]}')
print('-=' * 15)
print(f'Times em ordem alfabética: {sorted(times)}')
```



# Atividade

- Desenvolva esses códigos em Python:

## DESAFIO

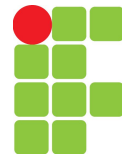
074



Crie um programa que vai gerar **cinco** números aleatórios e colocar em uma **tupla**.

Depois disso, mostre a **listagem** de números gerados e também indique o **menor** e o **maior** valor que estão na **tupla**.

```
from random import randint
numeros = (randint(1, 10), randint(1, 10), randint(1, 10),
 randint(1, 10), randint(1, 10))
print('Os valores sorteados foram: ', end='')
for n in numeros:
 print(f'{n} ', end='')
print(f'\nO maior valor sorteado foi {max(numeros)}')
print(f'O menor valor sorteado foi {min(numeros)}')
```



# Atividade

- Desenvolva esses códigos em Python:

• DESAFIO 075

💡 Desenvolva um programa que leia **quatro valores** pelo **teclado** e guarde-os em uma **tupla**. No final, mostre:

A) Quantas vezes apareceu o valor **9**.

B) Em que posição foi digitado o primeiro valor **3**.

C) Quais foram os números **par**s.

• DESAFIO 076

💡 Crie um programa que tenha uma **tupla** única com **nomes de produtos** e seus respectivos **preços**, na sequência.

No final, mostre uma listagem de preços, organizando os dados em forma **tabular**.

• DESAFIO 077

💡 Crie um programa que tenha uma **tupla** com **várias palavras** (não usar acentos). Depois disso, você deve mostrar, para cada palavra, quais são as suas **vogais**.



# Atividade

- Desenvolva esses códigos em Python:

• DESAFIO 078

💡 Faça um programa que leia 5 valores numéricos e guarde-os em uma lista.

No final, mostre qual foi o maior e o menor valor digitado e as suas respectivas posições na lista.

• DESAFIO 079

💡 Crie um programa onde o usuário possa digitar vários valores numéricos e cadastre-os em uma lista. Caso o número já exista lá dentro, ele não será adicionado. No final, serão exibidos todos os valores únicos digitados, em ordem crescente.

• DESAFIO 080

💡 Crie um programa onde o usuário possa digitar cinco valores numéricos e cadastre-os em uma lista, já na posição correta de inserção (sem usar o `sort()`). No final, mostre a lista ordenada na tela.





# Atividade

- Desenvolva esses códigos em Python:

**DESAFIO 081**

💡 Crie um programa que vai ler **vários números** e colocar em uma **lista**.

Depois disso, mostre:

A) **Quantos** números foram digitados.

B) A lista de valores, ordenada de forma **decrecente**.

C) Se o valor **5** foi digitado e está ou não na lista.

0:33 / 40:16

**DESAFIO 082**

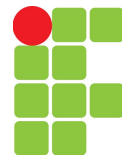
💡 Crie um programa que vai ler **vários números** e colocar em uma **lista**.

Depois disso, crie **duas listas extras** que vão contar apenas os valores **pares** e os valores **ímpares** digitados, respectivamente.

Ao final, mostre o conteúdo das **três listas** geradas.

**DESAFIO 083**

💡 Crie um programa onde o usuário digite uma **expressão** qualquer que use **parênteses**. Seu aplicativo deverá analisar se a expressão passada está com os parênteses abertos e fechados na **ordem correta**.





# Atividade

- Desenvolva esses códigos em Python:

• DESAFIO **084**

💡 Faça um programa que leia **nome** e **peso** de **várias pessoas**, guardando tudo em uma **lista**. No final, mostra:

A) **Quantas** pessoas foram cadastradas.

B) Uma listagem com as pessoas mais **pesadas**.

C) Uma listagem com as pessoas mais **leves**.

• DESAFIO **085**

💡 Crie um programa onde o usuário possa digitar **sete valores numéricos** e cadastre-os em uma **lista única** que mantenha separados os valores  **pares** e  **ímpares**. No final, mostre os valores  **pares** e  **ímpares** em ordem  **crescente**.

• DESAFIO **086**

💡 Crie um programa que crie uma **matriz** de **dimensão 3x3** e preencha com valores lidos pelo teclado.

|   |  |  |  |
|---|--|--|--|
| 0 |  |  |  |
| 1 |  |  |  |
| 2 |  |  |  |

No final, mostre a **matriz** na tela, com a **formatação correta**.



# Atividade

- Desenvolva esses códigos em Python:

• DESAFIO 087

💡 Aprimore o **desafio anterior**, mostrando no final:

A) A **soma** de todos os **valores pares** digitados.

B) A **soma** dos valores da **terceira coluna**.

C) O **maior** valor da **segunda linha**.

• DESAFIO 088

💡 Faça um programa que ajude um jogador da **MEGA SENA** a criar **palpites**. O programa vai perguntar **quantos jogos** serão gerados e vai sortear **6 números entre 1 e 60** para cada jogo, cadastrando tudo em uma **lista composta**.

• DESAFIO 089

💡 Crie um programa que leia **nome** e **duas notas** de vários alunos e guarde tudo em uma **lista composta**. No final, mostre um **boletim** contendo a **média** de cada um e permita que o usuário possa mostrar as **notas** de cada aluno individualmente.



# Atividade

- Desenvolva esses códigos em Python:

• DESAFIO 090

💡 Faça um programa que leia **nome** e **média** de um aluno, guardando também a **situação** em um **dicionário**. No final, mostre o conteúdo da estrutura na tela.

• DESAFIO 091

💡 Crie um programa onde **4 jogadores** joguem um **dado** e tenham resultados **aleatórios**. Guarde esses resultados em um **dicionário**. No final, coloque essa **dicionário** em **ordem**, sabendo que o **vencedor** tirou o **maior número** no dado.

• DESAFIO 092

💡 Crie um programa que leia **nome**, **ano de nascimento** e **carteira de trabalho** e cadastre-os (com **idade**) em um **dicionário** se por acaso a **CTPS** for diferente de **ZERO**, o dicionário receberá também o **ano de contratação** e o **salário**. Calcule a **acrescente**, além da **idade**, com quantos anos a pessoa vai se **aposentar**.



# Atividade

- Desenvolva esses códigos em Python:

• DESAFIO 093

💡 Crie um programa que gerencie o aproveitamento de um jogador de futebol. O programa vai ler o nome do jogador e quantas partidas ele jogou. Depois vai ler a quantidade de gols feitos em cada partida. No final, tudo isso será guardado em um dicionário, incluindo o total de gols feitos durante o campeonato.

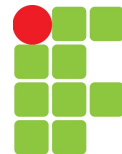
• DESAFIO 094

💡 Crie um programa que leia nome, sexo e idade de várias pessoas, guardando os dados de cada pessoa em um dicionário e todos os dicionários em uma lista. No final, mostre:

- A) Quantas pessoas foram cadastradas
- B) A média de idade do grupo.
- C) Uma lista com todas as mulheres.
- D) Uma lista com todas as pessoas com idade acima da média.

• DESAFIO 095

💡 Aprimore o DESAFIO 093 para que ele funcione com vários jogadores, incluindo um sistema de visualização de detalhes do aproveitamento de cada jogador.



# Referências

PUGA, Sandra; RISSETTI, Gerson. Lógica de Programação e Estruturas de Dados-Com Aplicações em Java. 3ª edição. 2016.

GOODRICH, Michael T.; TAMASSIA, Roberto. **Estruturas de dados & algoritmos em Java**. Bookman Editora, 2013.

CASTILHO, Marcos; SILVA, Fabiano; WEINGAERTNER, Daniel. Algoritmos e Estruturas de Dados I. 2011.

Curso de programação em Python do prof. Gustavo Guanabara. Último acesso em: 01 Set 2024.





**INSTITUTO FEDERAL DE  
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA  
PERNAMBUCO**