



PARADIGMAS DE LINGUAGENS DE PROGRAMAÇÃO EM PYTHON

AGLOMERADOS DE DADOS EM PYTHON

Prof. Dr. Daniel Caetano

2021 - 2

Compreendendo o problema

- **Situação:** aplicação acadêmica
 - Como representar dados de um aluno de maneira coesa?

Aluno

- Nome (*str*)
- Matrícula (*int*)
- Média (*float*)

- E em linguagens sem listas genéricas?



Mentimeter

<https://www.menti.com/>

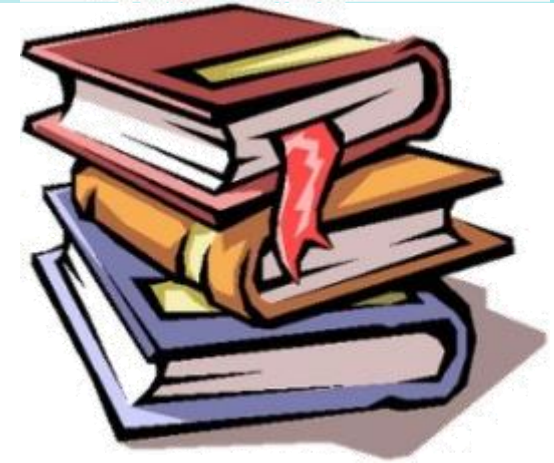
Objetivos

- Complementar os recursos de listas
- Conhecer a organização em registros
- Conhecer as tuplas e as tuplas nomeadas
- Compreender as uniões

- **Desafio Aula 07**



Bibliografia da Aula



Material**Acesso ao Material**

Apresentação

<https://www.caetano.eng.br/aulas/2021b/ara0066.php>
(Paradigmas de Programação – Aula 7)

Livro Texto

Capítulo 6, páginas 263 a 273

Aprenda Mais!

- Vídeo: Tuplas e Tuplas Nomeadas
<https://www.youtube.com/watch?v=lkoaXbLsUg8>

COMPLEMENTANDO: LISTAS EM PYTHON



Mentimeter

<https://www.menti.com/>

Prof. Dr. Daniel Caetano

Listas em Python

- Já vimos na aula passada que as listas:
 - Podem guardar dados de tipos diferentes
 - Pode ser criadas com ou sem elementos
 - Podemos adicionar elementos com *append*
 - Podemos excluir elementos com *pop* ou *remove*
 - Podemos modificar um elemento da lista
 - Podemos imprimir a lista
 - Podemos percorrer elementos da lista.
- Mas isso é tudo?
 - Podemos acrescentar mais uma ou duas coisinhas!

Filtros em Listas

- Considere a lista

```
                0      1      2      3      4  
umaLista = ["A", "B", "C", "D", "E"]
```

- Podemos filtrar um único elemento
 - Pela posição (já vimos!)

```
print(umaLista[2])  
'C'
```

Filtros em Listas

- Considere a lista

```
           0     1     2     3     4  
umaLista = ["A", "B", "C", "D", "E"]
```

- Selecionar tudo até o elemento indicado com :
 - **Exclui o último elemento!**

```
print(umaLista[:2])  
['A', 'B']
```


Filtros em Listas

- Considere a lista

```
           0     1     2     3     4  
umaLista = ["A", "B", "C", "D", "E"]
```

- Selecionar faixa, até o elemento indicado com :
 - **Exclui o último elemento!**

```
print(umaLista[1:3])  
['B', 'C']
```

Filtros em Listas

- Considere a lista

```
                0      1      2      3      4  
umaLista = ["A", "B", "C", "D", "E"]
```

- Selecionar faixa, do indicado até o fim com :
 - **Exclui o último elemento!**

```
print(umaLista[2:])  
['C', 'D', 'E']
```

Filtros em Listas

- Considere a lista

```
          0      1      2      3      4  
umaLista = ["A", "B", "C", "D", "E"]
```

- Selecionar faixa, “de tanto em tanto” com `::`

```
print(umaLista[::2])  
['A', 'C', 'E']
```

Filtros em Listas

- Considere a lista

```
           0     1     2     3     4  
umaLista = ["A", "B", "C", "D", "E"]
```

- Faixa, “de tanto em tanto” e inicial com ::

```
print(umaLista[1::3])  
['B', 'E']
```

Apagar (Elementos das) Listas

- Considere a lista

```
           0     1     2     3     4  
umaLista = ["A", "B", "C", "D", "E"]
```

- Podemos apagar uma faixa da lista com **del**

```
del umaLista[1:3]
```

- Ou apagá-la totalmente...

```
del umaLista
```

REGISTROS



Mentimeter

<https://www.menti.com/>

Prof. Dr. Daniel Caetano

Registros

- É uma forma de criar novos tipos de dados...
 - Agrupando tipos diferentes de maneira organizada
 - struct (C, C++, C#, F#, Python*...)
 - class*

Aluno

- Nome (str)
- Matrícula (int)
- Média (float)

- Forma realmente um “grupo de dados”
 - O tipo de grupo é nomeado
 - Na memória os dados ficam contíguos (**≠ listas!**).

Registros

- Exemplo (C/C++):

```
#include <iostream>
using namespace std;
struct Cliente {
    char nome[200];
    char cpf[11];
    float limite;
    int compras
};

int main() {
    struct Cliente umCliente;
    strcpy(umCliente.nome, "Fulano");
    strcpy(umCliente.cpf, "01234567890");
    umCliente.limite = 5000.0;
    umCliente.compras = 0;
}
```

Cliente

- Nome (str)
- CPF (str)
- Limite (float)
- Compras (int)

Registros

- Exemplo (Python):

```
import struct
Cliente = ('200s 11s f l')

umCliente = struct.pack(Cliente, b"Fulano",
b"01234567890", 5000.00, 0)

print(umCliente)

print(struct.unpack(Cliente, umCliente))
```

Cliente

- Nome (str)
- CPF (str)
- Limite (float)
- Compras (int)

Registros

- Registos em Python: sem nomes dos campos
 - Foco primário: trocar dados com C/C++
- Como guardar dados organizados em Python?
 - Forma mais conveniente... Tuplas nomeadas.

Tuplas?



TUPLAS



Mentimeter

<https://www.menti.com/>

Prof. Dr. Daniel Caetano

Tuplas

- Agrupamentos de dados não nomeados
 - Existe em Python, F#...
 - É como definir um registro, porém imutável
- Em Python são dados separados por vírgulas
 - Mas usualmente indicamos parênteses

```
umaTupla = (3, 5.8, "maçã")  
print (umaTupla)
```

Tuplas

- Outro exemplo

```
cliente = ("Fulano", "01234567890", 5000.0, 0)
print (cliente)
```

- O acesso é como em lista (: e :: valem!)

```
print(cliente[1])
'01234567890'
```

Tuplas

- Outro exemplo

```
cliente = ("Fulano", "01234567890", 5000.0, 0)  
print (cliente)
```

- Mas não podemos mudar um elemento!

```
cliente[1] = "1234"  
'01234567890'
```

Erro!

Tuplas

- Outro exemplo

```
cliente = ("Fulano", "01234567890", 5000.0, 0)
print (cliente)
```

- Podemos “explodir” a tupla em várias variáveis

```
nome, cpf, limite, compras = cliente
print(limite)
'5000.00'
```

Tuplas

- Outro exemplo

```
cliente = ("Fulano", "01234567890", 5000.0, 0)  
print (cliente)
```

- Podemos apagar uma tupla inteira

```
del cliente
```


Tuplas Nomeadas

- Tuplas Nomeadas: similar às structs
 - Campos ficam com nomes definidos

Cliente

- Nome
- CPF
- Limite
- Compras

```
from collections import namedtuple
Cliente = namedtuple("Cliente", "nome, cpf limite compras")

umCliente = Cliente("Fulano", "01234567890", 5000.0, 0)

print (umCliente)

print(umCliente.cpf)
```

Tuplas Nomeadas

- Tuplas Nomeadas: similar às structs
 - Campos ficam com nomes definidos
 - Os campos, porém, são imutáveis:

Cliente

- Nome
- CPF
- Limite
- Compras

```
from collections import namedtuple
```

```
Cliente = namedtuple("Cliente", "nome, cpf limite compras")
```

```
umCliente = Cliente("Fulano", "01234567890", 5000.0, 0)
```

```
umCliente.nome = "Novo Nome"
```

Erro!



UNIÕES

Uniões

- Variáveis sobrepostas na memória
 - Nomes diferentes
 - Tipos diferentes
 - Mesmas “gavetas”.

int valor
char texto[4]

0	1	2	3
?	?	?	?

- Uso limitado, porém úteis quando necessário
 - Existem em C, C++, F#...

Uniões

- Exemplo (em C/C++)

```
#include <iostream>
using namespace std;

union tipoMisto {
    int valorInt;
    float valorFloat;
};

int main() {
    union tipoMisto x;
    x.valorInt = 10;
    cout << x.valorInt << endl;
    cout << x.valorFloat << endl;
}
```



EXEMPLOS

Exemplo 1

- Mostrar Agenda com Tuplas Nomeadas

Nome: Daniel

Telefone: 11-5555-1234

Data Nasc.: 10/02/1973

Exemplo 2

- Criar Agenda com Tuplas Nomeadas

Nome: Daniel

Telefone: 11-5555-1234

Data Nasc.: 10/02/1973



ATIVIDADE

Atividade 1

- Grupo – 15 minutos
- Desenvolva um sistema de apoio a decisão que deve apresentar as perguntas de A a F e classificar o entrevistado conforme o quadro:
 - a) Nome do entrevistado?
 - b) Telefonou para a vítima?
 - c) Esteve no local do crime?
 - d) Mora perto da vítima?
 - e) Devia para a vítima?
 - f) Já trabalhou para a vítima?

Pela quantidade de respostas afirmativas:

0 ou 1: Inocente!

2: Suspeito!

3 ou 4: Cúmplice

5: Criminoso!

Atividade 2

- Grupo – 15 minutos
- Modifique o programa para que permita o cadastro de mais de um entrevistado e, ao final, liste o nome daqueles que são cúmplices ou criminosos.



ENCERRAMENTO

Resumo e Próximos Passos

- Mais operações com listas!
 - Registros: agrupamentos mistos
 - Tuplas e Tuplas Nomeadas
 - Uniões
 - **Pós Aula:** Aprenda Mais, Pós Aula e Desafio!
 - No padlet: <https://padlet.com/djcaetano/paradigmas>
-
- Tipos de dados avançados...
 - Ponteiros e referências!



PERGUNTAS?