

# Outras Estruturas de Dados: Tuplas e Conjuntos

- Aula 12 -  
Pensamento Computacional

Prof. Me. Lucas R. C. Pessutto



# Na aula de hoje...

01

## Operações Avançadas com Listas

Concatenação  
Cópia de Listas  
Fatiamento

02

## List Comprehensions

Sintaxe  
Mapeamento  
Filtragem

03

## Aninhamento de Listas

...

# Recapitulando

| Estrutura de dados        | Uso  |
|---------------------------|--|
| <b>Listas (list)</b>      | Coleção <b>mutável</b> com itens <b>ordenados/indexáveis</b> <b>mutáveis</b> ; pode haver duplicatas   |
| <b>Strings (str)</b>      | Coleção <b>imutável</b> com itens <b>ordenados/indexáveis</b> <b>imutáveis</b> ; pode haver duplicatas   |
| <b>Dicionários (dict)</b> | Coleção <b>mutável</b> com itens <b>não-ordenados</b> com chaves <b>imutáveis</b> e valores <b>mutáveis</b> ; não pode haver chaves duplicadas |
| <b>Tuplas (tuple)</b>     | ?  |
| <b>Conjuntos (set)</b>    | ?  |

# Tuplas (tuple)



```
>>> t = ("abacaxi", "uva", "morango")
>>> type(t)
<class 'tuple'>
```

- \* É uma coleção de elementos (com possíveis duplicatas) ordenados/indexáveis, mas **imutáveis**;
- \* Muito similar a listas: comandos de acesso e fatiamento são possíveis; mas os de **atribuição falham**!
- \* Geralmente mais rápidas que listas!

# Tuplas

Sintaxe:

```
tupla = (elem1, elemem2, ..., elemN)
```

Tuplas Especiais:

Tupla Vazia:

```
tupla = ()
```

Tupla com 1 elemento:

```
tupla = (elem1, )
```

**Vírgula Obrigatória!** Caso contrário o Python interpreta como uma expressão entre parênteses.

# Acesso aos valores

O acesso aos valores de uma tupla é feito usando a notação de colchetes, da mesma maneira que as listas:

```
estudante = ("João da Silva", 22, "Ciência da Computação", 9.5)
```

```
print(f"Aluno: {estudante[0]}")
```

Qual o resultado das seguintes operações?

```
print(estudante[2])
```

Ciência da Computação

```
del estudante[3]
```

TypeError: 'tuple' object doesn't support item deletion

```
estudante.append("Pensamento")
```

AttributeError: 'tuple' object has no attribute 'append'

# Outras Operações

```
s = (10, 20, 30)
```

Qual o resultado das seguintes operações?

```
print(s[1:3])
```

```
(20, 30)
```

```
s.count(10)
```

```
1
```

```
s.index(10)
```

```
0
```

```
print(s + (40, 50))
```

```
(10, 20, 30, 40, 50)
```

# Listas x Strings x Tuplas

## Strings

```
s = "hello"  
w = "!"
```

## Listas

```
s = [1, 2, 3, 4]  
w = [5, 6]
```

## Tuplas

```
s = (1, 2, 3, 4)  
w = (5, 6)
```

Tamanho

`len(s)`

5

4

4

Seleção

`s[0]`

h

1

1

Fatiamento

`s[1:4]`  
`s[1:]`

"ell"  
"ello"

[2, 3, 4]  
[2, 3, 4]

(2, 3, 4)  
(2, 3, 4)

Contagem

`s.count('e')`  
`s.count(4)`

1

*ERRO*

0

1

0

1



# Listas x Strings x Tuplas

## Strings

```
s = "hello"  
w = "!"
```

## Listas

```
s = [1, 2, 3, 4]  
w = [5, 6]
```

## Tuplas

```
s = (1, 2, 3, 4)  
w = (5, 6)
```

Índice

```
s.index('e')  
s.index(3)
```

```
1  
--
```

```
--  
2
```

```
--  
2
```

Pertencimento

```
'h' in s
```

```
True
```

```
False
```

```
False
```

Concatenação

```
s + w
```

```
"hello!"
```

```
[1, 2, 3, 4, 5, 6]
```

```
(1, 2, 3, 4, 5, 6)
```

# Listas x Strings x Tuplas

## Strings

```
s = "hello"  
w = "!"
```

## Listas

```
s = [1, 2, 3, 4]  
w = [5, 6]
```

## Tuplas

```
s = (1, 2, 3, 4)  
w = (5, 6)
```

Valor Máximo  
`max(s)`

`'o'`

4

4

Valor Mínimo  
`min(s)`

`'e'`

1

1

Soma  
`sum(s)`

*ERRO*

10

10

# Problema 1: Horóscopo Chinês



## Enunciado de um problema:

Faça um programa para mostrar o animal do horóscopo chinês correspondente ao ano de nascimento informado pelo usuário.

Ao contrário do horóscopo ocidental, os signos são anuais, ou seja, todas as pessoas nascidas naquele ano possuem o mesmo signo.

Sabe-se que o ano de 1900 foi o ano do rato, 1901 do boi, ..., 1912 foi o ano do porco e 1913 foi o ano do rato novamente.

| Número ▾ | Signo ▾  |
|----------|----------|
| 1        | Rato     |
| 2        | Boi      |
| 3        | Tigre    |
| 4        | Coelho   |
| 5        | Dragão   |
| 6        | Serpente |
| 7        | Cavalo   |
| 8        | Cabra    |
| 9        | Macaco   |
| 10       | Galo     |
| 11       | Cão      |
| 12       | Porco    |

# Problema 1: Horóscopo Chinês

```
zodiaco = ('Rato', 'Boi', 'Tigre', 'Coelho', 'Dragão', 'Cobra', 'Cavalo', 'Cabra',  
'Macaco', 'Galo', 'Cachorro', 'Porco')  
  
print("Este programa mostra o seu signo no Horóscopo Chinês")  
  
fim = False  
while not fim:  
    ano_nasc = int(input('Informe o ano de nascimento (yyyy): '))  
  
    while ano_nasc < 1900:  
        ano_nasc = int(input('ENTRADA INCORRETA! Informe o ano de nascimento (yyyy): '))  
  
    signo =   
  
    print(f"\nSeu signo é {zodiaco[signo].upper()}\n")  
  
    fim = input("Informar outro ano [S]im ou [N]ão? ").lower() != "s"
```

# Conjuntos (set)

```
>>> c = {"abacaxi", "uva", "morango"}  
>>> type(c)  
<class 'set'>
```

- \* É uma coleção de **elementos** não-ordenados/não-indexáveis, imutáveis e sem duplicatas;
- \* A **coleção é mutável** (pode-se adicionar e remover elementos...);
- \* Essa estrutura implementa o conceito matemático de conjuntos!

# Conjuntos

Sintaxe:

```
conjunto = {elem1, elemem2, ..., elemN}
```

Conjuntos Especiais:

Conjunto Vazio:

```
conjunto = set()
```

Não é possível criar um conjunto fazendo `conjunto = {}`, pois essa sintaxe é usada para criar dicionários.

Conjunto Unitário:

```
conjunto = {elem1}
```

# Operações em Conjuntos

- \* A estrutura de dados set implementa uma série de operações de Teoria dos Conjuntos
  - Pertencimento
  - Inserção
  - Remoção
  - União
  - Intersecção
  - Diferença
  - Diferença Simétrica
  - Tamanho

# Listas x Strings x Tuplas

Python

$A = \{1, 2, 3\}$

$B = \{3, 4, 5, 6\}$

Matemática

Operador

Função

Pertencimento

$1 \in A$

`1 in A`

True

Inserção

`A.add(4)`

`{1, 2, 3, 4}`

Remoção

`A.remove(2)`

`{1, 3}`

União

$A \cup B$

`A | B`

`A.union(B)`

`{1, 2, 3, 4, 5, 6,}`



# Listas x Strings x Tuplas

Python

A = {1, 2, 3}  
B = {3, 4, 5, 6}

Matemática

Operador

Função

Intersecção

$A \cap B$

A & B

A.intersection(B)

{3}

Diferença

$A - B$

A - B

A.difference(B)

{1, 2}

Diferença Simétrica

$A \triangle B$

A ^ B A.symmetric\_difference(B)

{1, 2, 4, 5, 6}

Cardinalidade

|A|

len(A)

3

## Problema 2: Alunos



Enunciado de um problema:

Faça um programa que pede ao usuário que informe os alunos que cursam as disciplinas de “Pensamento Computacional”, “Inteligência Artificial” e “Desafios da Profissão”. Ao fim, queremos listar todos os alunos que cursam:

- \* “Inteligência Artificial” e “Desafios da Profissão”
- \* “Pensamento Computacional” ou “Inteligência Artificial”
- \* “Pensamento Computacional” mas não fazem “Desafios da Profissão”

Informe um aluno de Pensamento Computacional (ou nada para terminar): Ana  
Informe um aluno de Pensamento Computacional (ou nada para terminar): João  
Informe um aluno de Pensamento Computacional (ou nada para terminar): Maria  
Informe um aluno de Pensamento Computacional (ou nada para terminar):  
Informe um aluno de Inteligência Artificial (ou nada para terminar): Ana  
Informe um aluno de Inteligência Artificial (ou nada para terminar): José  
Informe um aluno de Inteligência Artificial (ou nada para terminar): Maria  
Informe um aluno de Inteligência Artificial (ou nada para terminar):  
Informe um aluno de Desafios da Profissão (ou nada para terminar): João  
Informe um aluno de Desafios da Profissão (ou nada para terminar): Ana  
Informe um aluno de Desafios da Profissão (ou nada para terminar): Joana  
Informe um aluno de Desafios da Profissão (ou nada para terminar):

Alunos que cursam Inteligência Artificial E Desafios da Profissão:

- Ana

Alunos que cursam Pensamento Computacional OU Inteligência Artificial:

- Maria
- João
- José
- Ana

**A** Alunos que cursam Pensamento Computacional mas NÃO Desafios da Profissão:

- Maria

```
pcomp = set()
ia = set()
des_prof = set()

nome = " "
while nome != "":
    nome = input("Informe um aluno de Pensamento Computacional (ou nada para terminar): ")
    if nome != "":
        pcomp.add(nome)

nome = " "
while nome != "":
    nome = input("Informe um aluno de Inteligência Artificial (ou nada para terminar): ")
    if nome != "":
        ia.add(nome)

nome = " "
while nome != "":
    nome = input("Informe um aluno de Desafios da Profissão (ou nada para terminar): ")
    if nome != "":
        des_prof.add(nome)
```

```
print("Alunos que cursam Inteligência Artificial E Desafios da Profissão:")
for aluno in ia.intersection(des_prof):
    print(f"    - {aluno}")

print("Alunos que cursam Pensamento Computacional OU Inteligência Artificial:")
for aluno in pcomp.union(ia):
    print(f"    - {aluno}")

print("Alunos que cursam Pensamento Computacional mas NÃO Desafios da Profissão:")
for aluno in pcomp.difference(des_prof):
    print(f"    - {aluno}")
```

# Resumo de Estruturas de Dados

| Estrutura de dados        | Uso  |
|---------------------------|--|
| <b>Listas (list)</b>      | Coleção <b>mutável</b> com itens <b>ordenados/indexáveis</b> <b>mutáveis</b> ; pode haver duplicatas   |
| <b>Strings (str)</b>      | Coleção <b>imutável</b> com itens <b>ordenados/indexáveis</b> <b>imutáveis</b> ; pode haver duplicatas   |
| <b>Dicionários (dict)</b> | Coleção <b>mutável</b> com itens <b>não-ordenados</b> com chaves <b>imutáveis</b> e valores <b>mutáveis</b> ; não pode haver chaves duplicadas |
| <b>Tuplas (tuple)</b>     | Coleção <b>imutável</b> com itens <b>ordenados/indexáveis</b> <b>imutáveis</b> ; pode haver duplicatas   |
| <b>Conjuntos (set)</b>    | Coleção <b>mutável</b> com itens <b>não-ordenados</b> <b>imutáveis</b> ; não pode haver duplicatas   |